



Hub de automação do NetApp Console

NetApp Automation

NetApp
November 18, 2025

This PDF was generated from <https://docs.netapp.com/pt-br/netapp-automation/solutions/bac-overview.html> on November 18, 2025. Always check docs.netapp.com for the latest.

Índice

- Hub de automação do NetApp Console 1
 - Visão geral do hub de automação do NetApp Console 1
 - Amazon FSx for NetApp ONTAP 1
 - Amazon FSx for NetApp ONTAP - Explosão para a nuvem 1
 - Gerenciamento do Amazon FSx for NetApp ONTAP - Recuperação de desastres 6
 - Azure NetApp Files. 11
 - Instale o Oracle usando o Azure NetApp Files. 11
 - Cloud Volumes ONTAP para AWS 17
 - Cloud Volumes ONTAP para AWS: Explosão na nuvem 17
 - Cloud Volumes ONTAP para Azure 24
 - Cloud Volumes ONTAP para Azure - Burst para a nuvem. 24
 - Cloud Volumes ONTAP para Google Cloud. 32
 - Cloud Volumes ONTAP para Google Cloud - estourar para a nuvem 32
 - ONTAP 39
 - Dia 0/1 39

Hub de automação do NetApp Console

Visão geral do hub de automação do NetApp Console

O hub de automação do NetApp Console é uma coleção de soluções de automação disponíveis para clientes, parceiros e funcionários da NetApp . O hub de automação possui diversas funcionalidades e benefícios.

Um único local para suas necessidades de automação

Você pode acessar o "[Hub de automação do NetApp Console](#)" através da interface de usuário web do Console. Isso fornece um local único para os scripts, playbooks e módulos necessários para aprimorar a automação e a operação de seus produtos e serviços NetApp .

As soluções são criadas e testadas pela NetApp

Todas as soluções de automação e scripts foram criados e testados pela NetApp. Cada solução destina-se a um caso de uso ou solicitação específico do cliente. A maior parte do foco é a integração com os serviços de dados e arquivos do NetApp.

Documentação

Cada uma das soluções de automação inclui documentação associada para ajudar você a começar. Embora as soluções sejam acessadas pela interface web do Console, toda a documentação está disponível neste site. A documentação é organizada com base nos produtos e serviços de nuvem da NetApp .

Base sólida para o futuro

A NetApp está empenhada em ajudar nossos clientes a aprimorar e simplificar a automação de seus data centers e ambientes de nuvem. Esperamos continuar aprimorando o hub de automação do Console para atender às necessidades dos clientes, às mudanças tecnológicas e à integração contínua de produtos.

Queremos ouvir de você

A equipe de automação do escritório de experiência do Cliente (CXO) da NetApp gostaria de ouvir de você. Se você tiver algum feedback, problemas ou solicitações de recursos, envie um e-mail para [NetApp.com\[equipe de automação CXO\]](mailto:NetApp.com[equipe de automação CXO]).

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP - Explosão para a nuvem

Você pode usar esta solução de automação para provisionar o Amazon FSx for NetApp ONTAP com volumes e um FlexCache associado.



O gerenciamento do Amazon FSx for NetApp ONTAP também é chamado de **FSx para ONTAP**.

Sobre esta solução

Em alto nível, o código de automação fornecido com esta solução executa as seguintes ações:

- Provisione um sistema de arquivos FSX for ONTAP de destino
- Provisione máquinas virtuais de armazenamento (SVMs) para o sistema de arquivos

- Crie uma relação de peering de cluster entre os sistemas de origem e destino
- Crie uma relação de peering SVM entre o sistema de origem e o sistema de destino do FlexCache
- Como opção, crie volumes FlexVol usando o FSX for ONTAP
- Crie um volume FlexCache no FSX for ONTAP com a fonte apontando para armazenamento local

A automação é baseada no Docker e no Docker Compose, que deve ser instalado na máquina virtual Linux, conforme descrito abaixo.

Antes de começar

Para concluir o provisionamento e a configuração, você precisa ter o seguinte:

- Você precisa baixar o "[Amazon FSx for NetApp ONTAP - Explosão para a nuvem](#)" Solução de automação através da interface web do NetApp Console . A solução está empacotada como um arquivo. `AWS_FSxN_BTC.zip`.
- Conetividade de rede entre os sistemas de origem e destino.
- Uma VM Linux com as seguintes características:
 - Distribuição Linux baseada em Debian
 - Implantado no mesmo subconjunto VPC usado para o provisionamento do FSX for ONTAP
- Conta da AWS.

Passo 1: Instale e configure o Docker

Instale e configure o Docker em uma máquina virtual Linux baseada em Debian.

Passos

1. Prepare o 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. Instale o Docker e verifique a instalação.

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

3. Adicione o grupo Linux necessário a um usuário associado.

Primeiro verifique se o grupo **docker** existe no seu sistema Linux. Se isso não acontecer, crie o grupo e adicione o usuário. Por padrão, o usuário shell atual é adicionado ao grupo.

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

4. Ative o novo grupo e as definições de utilizador

Se você criou um novo grupo com um usuário, será necessário ativar as definições. Para fazer isso, você pode sair do Linux e depois voltar para dentro. Ou você pode executar o seguinte comando.

```
newgrp docker
```

Passo 2: Instale o Docker Compose

Instale o Docker Compose em uma máquina virtual Linux baseada em Debian.

Passos

1. Instale o 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. Verifique se a instalação foi bem-sucedida.

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

Passo 3: Prepare a imagem do Docker

Você precisa extrair e carregar a imagem Docker fornecida com a solução de automação.

Passos

1. Copie o arquivo de solução AWS_FSxN_BTC.zip para a máquina virtual onde o código de automação será executado.

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

O parâmetro de entrada `private-key.pem` é o arquivo de chave privada usado para autenticação de máquina virtual da AWS (instância EC2).

2. Navegue até a pasta correta com o arquivo de solução e descompacte o arquivo.

```
unzip AWS_FSxN_BTC.zip
```

3. Navegue até a nova pasta `AWS_FSxN_BTC` criada com a operação de descompactação e liste os arquivos. Você deve ver `aws_fsxn_flexcache_image_latest.tar.gz` arquivo.

```
ls -la
```

4. Carregue o arquivo de imagem do Docker. Normalmente, a operação de carga deve ser concluída em alguns segundos.

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

5. Confirme se a imagem do Docker está carregada.

```
docker images
```

Você deve ver a imagem do Docker `aws_fsxn_flexcache_image` com a tag `latest`.

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

Etapa 4: Criar arquivo de ambiente para credenciais da AWS

Você deve criar um arquivo de variável local para autenticação usando o acesso e a chave secreta. Em seguida, adicione o arquivo ao `.env` arquivo.

Passos

1. Crie o `awsauth.env` arquivo no seguinte local:

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

2. Adicione o seguinte conteúdo ao arquivo:

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

O formato **deve** ser exatamente como mostrado acima, sem espaços entre `key` e `value`.

3. Adicione o caminho absoluto do arquivo ao `.env` arquivo usando a `AWS_CREDS` variável. Por exemplo:

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

Passo 5: Crie um volume externo

Você precisa de um volume externo para garantir que os arquivos de estado do Terraform e outros arquivos importantes sejam persistentes. Esses arquivos devem estar disponíveis para que o Terraform execute o fluxo de trabalho e as implantações.

Passos

1. Crie um volume externo fora do Docker Compose.

Certifique-se de atualizar o nome do volume (último parâmetro) para o valor apropriado antes de executar o comando.

```
docker volume create aws_fsxn_volume
```

2. Adicione o caminho para o volume externo ao `.env` arquivo de ambiente usando o comando:

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

Lembre-se de manter o conteúdo do arquivo existente e a formatação de dois pontos. Por exemplo:

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

Em vez disso, você pode adicionar um compartilhamento NFS como o volume externo usando um comando como:

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

3. Atualize as variáveis Terraform.
 - a. Navegue até a pasta `aws_fsxn_variables`.
 - b. Confirme se existem os dois arquivos a seguir: `terraform.tfvars` E `variables.tf`.
 - c. Atualize os valores em `terraform.tfvars` conforme necessário para o seu ambiente.

Consulte "[Recurso Terraform: AWS_fsx_ONTAP_file_system](#)" para obter mais informações.

Etapa 6: provisionar o Amazon FSx for NetApp ONTAP e FlexCache

Você pode provisionar o Amazon FSx for NetApp ONTAP e FlexCache.

Passos

1. Navegue até a pasta raiz (`AWS_FSXN_BTC`) e emita o comando de provisionamento.

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

Este comando cria dois contentores. O primeiro contêiner implanta o FSX para ONTAP e o segundo contêiner cria peering de cluster, peering SVM, volume de destino e FlexCache.

2. Monitorar o processo de provisionamento.

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

Este comando fornece a saída em tempo real, mas foi configurado para capturar os logs através do arquivo `deployment.log`. Você pode alterar o nome desses arquivos de log editando o `.env` arquivo e atualizando as variáveis `DEPLOYMENT_LOGS`.

Etapa 7: Destrua o Amazon FSx for NetApp ONTAP e FlexCache

Opcionalmente, você pode excluir e remover o Amazon FSx for NetApp ONTAP e o FlexCache.

1. Defina a variável `flexcache_operation` `terraform.tfvars` no arquivo como "Destroy".
2. Navegue até a pasta raiz (`AWS_FSXN_BTC`) e emita o seguinte comando.

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

Este comando cria dois contentores. O primeiro contentor exclui FlexCache e o segundo contentor exclui o FSX for ONTAP.

3. Monitorar o processo de provisionamento.

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

Gerenciamento do Amazon FSx for NetApp ONTAP - Recuperação de desastres

Você pode usar esta solução de automação para fazer um backup de recuperação de desastres de um sistema de origem usando o Amazon FSx for NetApp ONTAP .



O gerenciamento do Amazon FSx for NetApp ONTAP também é chamado de **FSx para ONTAP**.

Sobre esta solução

Em alto nível, o código de automação fornecido com esta solução executa as seguintes ações:

- Provisione um sistema de arquivos FSX for ONTAP de destino
- Provisione máquinas virtuais de armazenamento (SVMs) para o sistema de arquivos
- Crie uma relação de peering de cluster entre os sistemas de origem e destino
- Crie uma relação de peering SVM entre o sistema de origem e o sistema de destino do SnapMirror
- Criar volumes de destino
- Crie uma relação SnapMirror entre os volumes de origem e destino
- Inicie a transferência SnapMirror entre os volumes de origem e destino

A automação é baseada no Docker e no Docker Compose, que deve ser instalado na máquina virtual Linux, conforme descrito abaixo.

Antes de começar

Para concluir o provisionamento e a configuração, você precisa ter o seguinte:

- Você precisa baixar o "[Gerenciamento do Amazon FSx for NetApp ONTAP - Recuperação de desastres](#)" Solução de automação através da interface web do NetApp Console . A solução é empacotada como FSxN_DR.zip. Este arquivo zip contém o AWS_FSxN_Bck_Prov.zip Arquivo que você usará para implantar a solução descrita neste documento.
- Conetividade de rede entre os sistemas de origem e destino.
- Uma VM Linux com as seguintes características:
 - Distribuição Linux baseada em Debian
 - Implantado no mesmo subconjunto VPC usado para o provisionamento do FSX for ONTAP
- Uma conta da AWS.

Passo 1: Instale e configure o Docker

Instale e configure o Docker em uma máquina virtual Linux baseada em Debian.

Passos

1. Prepare o 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. Instale o Docker e verifique a instalação.

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

3. Adicione o grupo Linux necessário a um usuário associado.

Primeiro verifique se o grupo **docker** existe no seu sistema Linux. Se não existir, crie o grupo e adicione o usuário. Por padrão, o usuário shell atual é adicionado ao grupo.

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

4. Ative o novo grupo e as definições de utilizador

Se você criou um novo grupo com um usuário, será necessário ativar as definições. Para fazer isso, você pode sair do Linux e depois voltar para dentro. Ou você pode executar o seguinte comando.

```
newgrp docker
```

Passo 2: Instale o Docker Compose

Instale o Docker Compose em uma máquina virtual Linux baseada em Debian.

Passos

1. Instale o 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. Verifique se a instalação foi bem-sucedida.

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

Passo 3: Prepare a imagem do Docker

Você precisa extrair e carregar a imagem Docker fornecida com a solução de automação.

Passos

1. Copie o arquivo de solução `AWS_FSxN_Bck_Prov.zip` para a máquina virtual onde o código de automação será executado.

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

O parâmetro de entrada `private-key.pem` é o arquivo de chave privada usado para autenticação de máquina virtual da AWS (instância EC2).

2. Navegue até a pasta correta com o arquivo de solução e descompacte o arquivo.

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. Navegue até a nova pasta `AWS_FSxN_Bck_Prov` criada com a operação de descompactação e liste os arquivos. Você deve ver `aws_fsxn_bck_image_latest.tar.gz` arquivo.

```
ls -la
```

4. Carregue o arquivo de imagem do Docker. Normalmente, a operação de carga deve ser concluída em alguns segundos.

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

5. Confirme se a imagem do Docker está carregada.

```
docker images
```

Você deve ver a imagem do Docker `aws_fsxn_bck_image` com a tag `latest`.

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

Etapa 4: Criar arquivo de ambiente para credenciais da AWS

Você deve criar um arquivo de variável local para autenticação usando o acesso e a chave secreta. Em seguida, adicione o arquivo ao `.env` arquivo.

Passos

1. Crie o `awsauth.env` arquivo no seguinte local:

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

2. Adicione o seguinte conteúdo ao arquivo:

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

O formato **deve** ser exatamente como mostrado acima, sem espaços entre `key` e `value`.

3. Adicione o caminho absoluto do arquivo ao `.env` arquivo usando a `AWS_CREDS` variável. Por exemplo:

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

Passo 5: Crie um volume externo

Você precisa de um volume externo para garantir que os arquivos de estado do Terraform e outros arquivos importantes sejam persistentes. Esses arquivos devem estar disponíveis para que o Terraform execute o fluxo de trabalho e as implantações.

Passos

1. Crie um volume externo fora do Docker Compose.

Certifique-se de atualizar o nome do volume (último parâmetro) para o valor apropriado antes de executar o comando.

```
docker volume create aws_fsxn_volume
```

2. Adicione o caminho para o volume externo ao `.env` arquivo de ambiente usando o comando:

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

Lembre-se de manter o conteúdo do arquivo existente e a formatação de dois pontos. Por exemplo:

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

Em vez disso, você pode adicionar um compartilhamento NFS como o volume externo usando um comando como:

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

3. Atualize as variáveis Terraform.
 - a. Navegue até a pasta `aws_fsxn_variables`.
 - b. Confirme se existem os dois arquivos a seguir: `terraform.tfvars` E `variables.tf`.
 - c. Atualize os valores em `terraform.tfvars` conforme necessário para o seu ambiente.

Consulte "[Recurso Terraform: AWS_fsx_ONTAP_file_system](#)" para obter mais informações.

Etapa 6: Implante a solução de backup

Você pode implantar e provisionar a solução de backup de recuperação de desastres.

Passos

1. Navegue até a pasta raiz (`AWS_FSxN_Bck_Prov`) e emita o comando de provisionamento.

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

Este comando cria três contentores. O primeiro contentor implanta o FSX para ONTAP. O segundo contêiner cria o peering de cluster, o peering SVM e o volume de destino. O terceiro contêiner cria a relação SnapMirror e inicia a transferência SnapMirror.

2. Monitorar o processo de provisionamento.

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

Este comando fornece a saída em tempo real, mas foi configurado para capturar os logs através do arquivo `deployment.log`. Você pode alterar o nome desses arquivos de log editando o `.env` arquivo e atualizando as variáveis `DEPLOYMENT_LOGS`.

Azure NetApp Files

Instale o Oracle usando o Azure NetApp Files

Você pode usar essa solução de automação para provisionar volumes Azure NetApp Files e instalar o Oracle em uma máquina virtual disponível. Em seguida, a Oracle usa os volumes para armazenamento de dados.

Sobre esta solução

Em alto nível, o código de automação fornecido com esta solução executa as seguintes ações:

- Configure uma conta do NetApp no Azure
- Configurar um pool de capacidade de storage no Azure
- Provisione os volumes Azure NetApp Files com base na definição
- Crie os pontos de montagem
- Monte os volumes Azure NetApp Files nos pontos de montagem
- Instale o Oracle no servidor Linux
- Crie os ouvintes e o banco de dados
- Criar os bancos de dados Pluggable (PDBs)
- Inicie o ouvinte e a instância Oracle
- Instale e configure o `azacsnap` utilitário para tirar um instantâneo

Antes de começar

Você deve ter o seguinte para concluir a instalação:

- Você precisa baixar o "[Oracle usando Azure NetApp Files](#)" Solução de automação através da interface web do NetApp Console . A solução está empacotada como um arquivo `na_oracle19c_deploy-master.zip`.
- Uma VM Linux com as seguintes características:
 - RHEL 8 (Standard_D8s_v3-RHEL-8)
 - Implantado na mesma rede virtual do Azure usada para o provisionamento do Azure NetApp Files
- Uma conta do Azure

A solução de automação é fornecida como uma imagem e executada usando Docker e Docker Compose. Você precisa instalar ambos na máquina virtual Linux conforme descrito abaixo.

Você também deve Registrar a VM com o RedHat usando o comando `sudo subscription-manager register`. O comando solicitará as credenciais da sua conta. Se necessário, você pode criar uma conta no <https://developers.redhat.com/>.

Passo 1: Instale e configure o Docker

Instale e configure o Docker em uma máquina virtual RHEL 8 Linux.

Passos

1. Instale o software Docker usando os seguintes comandos.

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

2. Inicie o Docker e exiba a versão para confirmar que a instalação foi bem-sucedida.

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

3. Adicione o grupo Linux necessário a um usuário associado.

Primeiro verifique se o grupo **docker** existe no seu sistema Linux. Se isso não acontecer, crie o grupo e adicione o usuário. Por padrão, o usuário shell atual é adicionado ao grupo.

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

4. Ative o novo grupo e as definições de utilizador

Se você criou um novo grupo com um usuário, será necessário ativar as definições. Para fazer isso, você pode sair do Linux e depois voltar para dentro. Ou você pode executar o seguinte comando.

```
newgrp docker
```

Passo 2: Instale o Docker Compose e os utilitários NFS

Instale e configure o Docker Compose juntamente com o pacote de utilitários NFS.

Passos

1. Instale o Docker Compose e exiba a versão para confirmar que a instalação foi bem-sucedida.

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

2. Instale o pacote de utilitários NFS.

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

Passo 3: Baixe os arquivos de instalação Oracle

Baixe os arquivos de instalação e patch necessários da Oracle, bem como o azacsnap utilitário.

Passos

1. Inicie sessão na sua conta Oracle, conforme necessário.
2. Transfira os seguintes ficheiros.

Ficheiro	Descrição
LINUX.X64_193000_db_home.zip	instalador base 19,3
p31281355_190000_Linux-x86-64.zip	19,8 RU patch
p6880880_190000_Linux-x86-64.zip	opatch versão 12.2.0.1.23
azacsnap_installer_v5.0.run	instalador azacsnap

3. Coloque todos os arquivos de instalação na pasta `/tmp/archive`.
4. Certifique-se de que todos os usuários do servidor de banco de dados tenham acesso total (leitura, gravação, execução) à pasta `/tmp/archive`.

Passo 4: Prepare a imagem do Docker

Você precisa extrair e carregar a imagem Docker fornecida com a solução de automação.

Passos

1. Copie o arquivo de solução `na_oracle19c_deploy-master.zip` para a máquina virtual onde o código de automação será executado.

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

O parâmetro de entrada `private-key.pem` é o arquivo de chave privada usado para autenticação de máquina virtual do Azure.

2. Navegue até a pasta correta com o arquivo de solução e descompacte o arquivo.

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

3. Navegue até a nova pasta `na_oracle19c_deploy-master` criada com a operação de descompactação e liste os arquivos. Você deve ver `ora_anf_bck_image.tar` arquivo.

```
ls -lt
```

4. Carregue o arquivo de imagem do Docker. Normalmente, a operação de carga deve ser concluída em alguns segundos.

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

5. Confirme se a imagem do Docker está carregada.

```
docker images
```

Você deve ver a imagem do Docker `ora_anf_bck_image` com a tag `latest`.

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

Passo 5: Crie um volume externo

Você precisa de um volume externo para garantir que os arquivos de estado do Terraform e outros arquivos importantes sejam persistentes. Esses arquivos devem estar disponíveis para que o Terraform execute o fluxo de trabalho e as implantações.

Passos

1. Crie um volume externo fora do Docker Compose.

Certifique-se de atualizar o nome do volume antes de executar o comando.

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

2. Adicione o caminho para o volume externo ao `.env` arquivo de ambiente usando o comando:

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

Lembre-se de manter o conteúdo do arquivo existente e a formatação de dois pontos. Por exemplo:

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

3. Atualize as variáveis Terraform.

- Navegue até a pasta `ora_anf_variables`.
- Confirme se existem os dois arquivos a seguir: `terraform.tfvars` E `variables.tf`.
- Atualize os valores em `terraform.tfvars` conforme necessário para o seu ambiente.

Passo 6: Instale o Oracle

Agora você pode provisionar e instalar o Oracle.

Passos

1. Instale o Oracle usando a seguinte sequência de comandos.

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

2. Recarregue suas variáveis Bash e confirme exibindo o valor para `ORACLE_HOME`.

- `cd /home/oracle`
- `source .bash_profile`
- `echo $ORACLE_HOME`

3. Você deve ser capaz de fazer login no Oracle.

```
sudo su oracle
```

Passo 7: Valide a instalação Oracle

Você deve confirmar que a instalação do Oracle foi bem-sucedida.

Passos

1. Faça login no servidor Oracle Linux e exiba uma lista dos processos Oracle. Isso confirma a instalação concluída conforme esperado e o banco de dados Oracle está em execução.

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

2. Faça login no banco de dados para examinar a configuração do banco de dados e confirmar que as PDBs foram criadas corretamente.

```
sqlplus / as sysdba
```

Você deve ver saída semelhante ao seguinte:

```
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. Execute alguns comandos SQL simples para confirmar que o banco de dados está disponível.

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

Passo 8: Instale o utilitário azacsnap e execute um backup instantâneo

Você precisa instalar e executar o utilitário para executar azacsnap um backup instantâneo.

Passos

1. Instale o recipiente.

```
docker-compose up azacsnap_install
```

2. Mude para a conta de utilizador instantâneo.

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

3. Configure um arquivo de detalhes de backup de armazenamento. Isso criará o azacsnap.json arquivo de configuração.

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

4. Faça um backup instantâneo.

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

Passo 9: Opcionalmente, migre um PDB no local para a nuvem

Opcionalmente, você pode migrar o PDB local para a nuvem.

Passos

1. Defina as variáveis nos `tfvars` arquivos conforme necessário para o seu ambiente.
2. Migrar o PDB.

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

Cloud Volumes ONTAP para AWS

Cloud Volumes ONTAP para AWS: Explosão na nuvem

Este artigo oferece suporte à solução de automação NetApp Cloud Volumes ONTAP para AWS, disponível para clientes NetApp no hub de automação do NetApp Console .

A solução de automação Cloud Volumes ONTAP para AWS automatiza a implantação em contêiner do Cloud Volumes ONTAP para AWS usando o Terraform, permitindo que você implante o Cloud Volumes ONTAP para AWS rapidamente, sem qualquer intervenção manual.

Antes de começar

- Você precisa baixar o "[Cloud Volumes ONTAP AWS: Explosão na nuvem](#)" Solução de automação através da interface web do Console. A solução é empacotada como `cvo_aws_flexcache.zip`.
- Você deve instalar uma VM Linux na mesma rede que o Cloud Volumes ONTAP.
- Depois de instalar a VM Linux, você deve seguir as etapas desta solução para instalar as dependências necessárias.

Passo 1: Instale o Docker e o Docker Compose

Instale o Docker

As etapas a seguir usam o software de distribuição Debian Linux Ubuntu 20,04 como exemplo. Os comandos que você executa dependem do software de distribuição Linux que você está usando. Consulte a documentação específica do software de distribuição Linux para sua configuração.

Passos

1. Instale o Docker executando os `sudo` seguintes comandos:

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

2. Verifique a instalação:

```
docker -version
```

3. Verifique se um grupo chamado "docker" foi criado em seu sistema Linux. Se necessário, crie o grupo:

```
sudo groupadd docker
```

4. Adicione o usuário que precisa acessar o Docker ao grupo:

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

5. As alterações são aplicadas depois de terminar sessão e voltar a iniciar sessão no terminal. Alternativamente, você pode aplicar as alterações imediatamente:

```
newgrp docker
```

Instale o Docker Compose

Passos

1. Instale o Docker Compose executando os seguintes `sudo` comandos:

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

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

2. Verifique a instalação:

```
docker-compose -version
```

Passo 2: Prepare a imagem do Docker

Passos

1. Copie a `cvo_aws_flexcache.zip` pasta para a VM Linux que você deseja usar para implantar o 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` é o seu arquivo de chave privada para login sem uma senha.
- `awsuser` É o nome de usuário da VM.
- `IP_ADDRESS_OF_VM` É o endereço IP da VM.
- `LOCATION_TO_BE_COPIED` é o local onde a pasta será copiada.

2. Extraia a `cvo_aws_flexcache.zip` pasta. Você pode extrair a pasta no diretório atual ou em um local personalizado.

Para extrair a pasta no diretório atual, execute:

```
unzip cvo_aws_flexcache.zip
```

Para extrair a pasta em um local personalizado, execute:

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

3. Depois de extrair o conteúdo, navegue até `CVO_Aws_Deployment` a pasta e execute o seguinte comando para visualizar os arquivos:

```
ls -la
```

Você deve ver uma lista de arquivos, semelhante ao seguinte exemplo:

```
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. Localize o `cvo_aws_flexcache_ubuntu_image.tar` arquivo. Isso contém a imagem do Docker necessária para implantar o Cloud Volumes ONTAP para AWS.
5. Descomprimir o ficheiro:

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

6. Aguarde alguns minutos para que a imagem do Docker seja carregada e, em seguida, valide que a imagem do Docker foi carregada com sucesso:

```
docker images
```

Você deve ver uma imagem Docker chamada `cvo_aws_flexcache_ubuntu_image` com a `latest` tag, como mostrado no exemplo a seguir:

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



Você pode alterar o nome da imagem do Docker, se necessário. Se você alterar o nome da imagem do Docker, certifique-se de atualizar o nome da imagem do Docker `docker-compose-deploy` nos arquivos e `docker-compose-destroy`.

Passo 3: Criar arquivos variáveis de ambiente

Nesta fase, você deve criar dois arquivos de variáveis de ambiente. Um arquivo é para autenticação de APIs do AWS Resource Manager usando as chaves secretas e de acesso da AWS. O segundo arquivo serve para definir variáveis de ambiente para permitir que os módulos do Console Terraform localizem e autenticem as APIs da AWS.

Passos

1. Crie o `awsauth.env` arquivo no seguinte local:

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

- a. Adicione o seguinte conteúdo ao `awsauth.env` arquivo:

não é possível aceder a uma mensagem de correio eletrónico

O formato **deve** ser exatamente como mostrado acima.

2. Adicione o caminho absoluto do arquivo ao `.env` arquivo.

Insira o caminho absoluto para o `awsauth.env` arquivo de ambiente que corresponde à `AWS_CREDS` variável de ambiente.

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

3. Navegue até a `cvo_aws_variable` pasta e atualize a chave de acesso e segredo no arquivo de credenciais.

Adicione o seguinte conteúdo ao arquivo:

```
aws_access_key_key_key_key_key_key_access_key_key_key_key_key
```

O formato **deve** ser exatamente como mostrado acima.

Etapa 4: inscreva-se no NetApp Intelligent Services

Inscreva-se no NetApp Intelligent Services por meio do seu provedor de nuvem para pagar por hora (PAYGO) ou por meio de um contrato anual. Os serviços inteligentes da NetApp incluem backup e recuperação da NetApp, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience e NetApp Disaster Recovery. A classificação de dados da NetApp está incluída na sua assinatura sem custo adicional.

Passos

1. No portal da Amazon Web Services (AWS), navegue até **SaaS** e selecione **Assinar NetApp Intelligent Services**.

Você pode usar o mesmo grupo de recursos que o Cloud Volumes ONTAP ou um grupo de recursos diferente.

2. Configure o portal do NetApp Console para importar a assinatura SaaS para o Console.

Você pode configurar isso diretamente no portal da AWS.

Você será redirecionado ao portal do Console para confirmar a configuração.

3. Confirme a configuração no portal do Console selecionando **Salvar**.

Passo 5: Crie um volume externo

Você deve criar um volume externo para manter os arquivos de estado do Terraform e outros arquivos importantes persistentes. Você deve garantir que os arquivos estejam disponíveis para o Terraform para executar o fluxo de trabalho e as implantações.

Passos

1. Criar um volume externo fora do Docker Compose:

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

Exemplo:

```
docker volume create cvo_aws_volume_dst
```

2. Use uma das seguintes opções:

- a. Adicione um caminho de volume externo ao `.env` arquivo de ambiente.

Você deve seguir o formato exato mostrado abaixo.

Formato:

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

Exemplo:

```
PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws
```

- b. Adicionar compartilhamentos NFS como volume externo.

Certifique-se de que o contentor Docker possa se comunicar com os compartilhamentos NFS e que as permissões corretas, como leitura/gravação, estejam configuradas.

- i. Adicione o caminho de compartilhamentos NFS como caminho para o volume externo no arquivo Docker Compose, como mostrado abaixo: Formato:

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

Exemplo:

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

3. Navegue até `cvo_aws_variables` a pasta.

Você deve ver o seguinte arquivo variável na pasta:

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

4. Altere os valores dentro do `terraform.tfvars` arquivo de acordo com suas necessidades.

Você deve ler a documentação de suporte específica ao modificar qualquer um dos valores de variável no `terraform.tfvars` arquivo. Os valores podem variar dependendo da região, zonas de disponibilidade e outros fatores suportados pelo Cloud Volumes ONTAP para AWS. Isso inclui licenças, tamanho de disco e tamanho de VM para nós únicos e pares de alta disponibilidade (HA).

Todas as variáveis de suporte para o agente do Console e os módulos Terraform do Cloud Volumes

ONTAP já estão definidas no `variables.tf` arquivo. Você deve se referir aos nomes das variáveis no `variables.tf` arquivo antes de adicionar ao `terraform.tfvars` arquivo.

5. Dependendo dos seus requisitos, pode ativar ou desativar o FlexCache e o FlexClone definindo as seguintes opções para `true` ou `false`.

Os exemplos a seguir habilitam o FlexCache e o FlexClone:

```
° is_flexcache_required = true
° is_flexclone_required = true
```

Etapa 6: Implante o Cloud Volumes ONTAP para AWS

Siga as etapas a seguir para implantar o Cloud Volumes ONTAP para AWS.

Passos

1. Na pasta raiz, execute o seguinte comando para acionar a implantação:

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

Dois contêineres são acionados, o primeiro contêiner implanta o Cloud Volumes ONTAP e o segundo contêiner envia dados de telemetria para o AutoSupport.

O segundo recipiente aguarda até que o primeiro recipiente conclua todas as etapas com êxito.

2. Monitore o progresso do processo de implantação usando os arquivos de log:

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

Este comando fornece saída em tempo real e captura os dados nos seguintes arquivos de log:
`deployment.log`

`telemetry_asup.log`

Você pode alterar o nome desses arquivos de log editando o `.env` arquivo usando as seguintes variáveis de ambiente:

`DEPLOYMENT_LOGS`

`TELEMETRY_ASUP_LOGS`

Os exemplos a seguir mostram como alterar os nomes dos arquivos de log:

`DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

`TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Depois de terminar

Você pode usar as etapas a seguir para remover o ambiente temporário e limpar itens criados durante o

processo de implantação.

Passos

1. Se você implantou o FlexCache, defina a seguinte opção no `terraform.tfvars` arquivo variável, isso limpa os volumes do FlexCache e remove o ambiente temporário que foi criado anteriormente.

```
flexcache_operation = "destroy"
```



As opções possíveis são `deploy` e `destroy`.

2. Se você implantou o FlexClone, defina a seguinte opção no `terraform.tfvars` arquivo variável, isso limpa os volumes do FlexClone e remove o ambiente temporário que foi criado anteriormente.

```
flexclone_operation = "destroy"
```



As opções possíveis são `deploy` e `destroy`.

Cloud Volumes ONTAP para Azure

Cloud Volumes ONTAP para Azure - Burst para a nuvem

Este artigo oferece suporte à solução NetApp Cloud Volumes ONTAP para Azure Automation, disponível para clientes NetApp no hub de automação do NetApp Console .

A solução de automação do Cloud Volumes ONTAP para Azure automatiza a implantação em contêiner do Cloud Volumes ONTAP para Azure usando o Terraform, permitindo que você implante o Cloud Volumes ONTAP para Azure rapidamente, sem qualquer intervenção manual.

Antes de começar

- Você precisa baixar o "[Cloud Volumes ONTAP Azure - Burst para a nuvem](#)" Solução de automação através da interface web do Console. A solução é empacotada como `CVO-Azure-Burst-To-Cloud.zip`.
- Você deve instalar uma VM Linux na mesma rede que o Cloud Volumes ONTAP.
- Depois de instalar a VM Linux, você deve seguir as etapas desta solução para instalar as dependências necessárias.

Passo 1: Instale o Docker e o Docker Compose

Instale o Docker

As etapas a seguir usam o software de distribuição Debian Linux Ubuntu 20,04 como exemplo. Os comandos que você executa dependem do software de distribuição Linux que você está usando. Consulte a documentação específica do software de distribuição Linux para sua configuração.

Passos

1. Instale o Docker executando os `sudo` seguintes comandos:

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

2. Verifique a instalação:

```
docker -version
```

3. Verifique se um grupo chamado "docker" foi criado em seu sistema Linux. Se necessário, crie o grupo:

```
sudo groupadd docker
```

4. Adicione o usuário que precisa acessar o Docker ao grupo:

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

5. As alterações são aplicadas depois de terminar sessão e voltar a iniciar sessão no terminal. Alternativamente, você pode aplicar as alterações imediatamente:

```
newgrp docker
```

Instale o Docker Compose

Passos

1. Instale o Docker Compose executando os seguintes sudo comandos:

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

2. Verifique a instalação:

```
docker-compose -version
```

Passo 2: Prepare a imagem do Docker

Passos

1. Copie a CVO-Azure-Burst-To-Cloud.zip pasta para a VM Linux que você deseja usar para implantar o 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 é o seu arquivo de chave privada para login sem uma senha.
- azureuser É o nome de usuário da VM.
- IP_ADDRESS_OF_VM É o endereço IP da VM.
- LOCATION_TO_BE_COPIED é o local onde a pasta será copiada.

2. Extraia a CVO-Azure-Burst-To-Cloud.zip pasta. Você pode extrair a pasta no diretório atual ou em um local personalizado.

Para extrair a pasta no diretório atual, execute:

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

Para extrair a pasta em um local personalizado, execute:

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

3. Depois de extrair o conteúdo, navegue até CVO_Azure_Deployment a pasta e execute o seguinte comando para visualizar os arquivos:

```
ls -la
```

Você deve ver uma lista de arquivos, semelhante ao seguinte exemplo:

```
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. Localize o `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` arquivo. Isso contém a imagem Docker necessária para implantar o Cloud Volumes ONTAP para Azure.
5. Descomprimir o ficheiro:

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

6. Aguarde alguns minutos para que a imagem do Docker seja carregada e, em seguida, valide que a imagem do Docker foi carregada com sucesso:

```
docker images
```

Você deve ver uma imagem Docker chamada `cvo_azure_flexcache_ubuntu_image_latest` com a `latest` tag, como mostrado no exemplo a seguir:

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

Passo 3: Criar arquivos variáveis de ambiente

Nesta fase, você deve criar dois arquivos de variáveis de ambiente. Um arquivo é para autenticação de APIs do Azure Resource Manager usando credenciais de entidade de serviço. O segundo arquivo serve para definir variáveis de ambiente para permitir que os módulos do Console Terraform localizem e autenticem as APIs do Azure.

Passos

1. Crie um responsável de serviço.

Antes de criar os arquivos variáveis de ambiente, você deve criar um princípio de serviço seguindo as etapas em ["Crie um diretor de serviço e aplicativo do Azure ative Directory que possa acessar recursos"](#).

2. Atribua a função **Colaborador** ao responsável de serviço recém-criado.

3. Crie uma função personalizada.

- a. Localize o `sp_role.json` arquivo e verifique as permissões necessárias nas ações listadas.
- b. Insira essas permissões e anexe a função personalizada ao responsável de serviço recém-criado.

4. Navegue até **certificados e segredos** e selecione **segredo de novo cliente** para criar o segredo do cliente.

Quando você cria o segredo do cliente, você deve Registrar os detalhes da coluna **valor** porque você não será capaz de ver esse valor novamente. Você também deve Registrar as seguintes informações:

- ID do cliente
- ID da subscrição
- ID do inquilino

Você precisará dessas informações para criar as variáveis de ambiente. Você pode encontrar informações de ID de cliente e ID de locatário na seção **Visão geral** da IU do Serviço Principal.

5. Crie os arquivos de ambiente.

- a. Crie o `azureauth.env` arquivo no seguinte local:

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

- i. Adicione o seguinte conteúdo ao arquivo:

A Sony Computer Entertainment Europe é uma das nossas principais empresas de tecnologia de ponta

O formato **deve** ser exatamente como mostrado acima, sem espaços entre a chave e o valor.

- b. Crie o `credentials.env` arquivo no seguinte local:

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

- i. Adicione o seguinte conteúdo ao arquivo:

```
AZURE_CLIENT_ID_ID_AZURE_CLIENT_SECRET_ID_AZURE_CLIENT_ID
```

O formato **deve** ser exatamente como mostrado acima, sem espaços entre a chave e o valor.

6. Adicione os caminhos de arquivo absolutos ao `.env` arquivo.

Insira o caminho absoluto para o `azureauth.env` arquivo de ambiente no `.env` arquivo que corresponde à `AZURE_RM_CREDS` variável de ambiente.

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

Insira o caminho absoluto para o `credentials.env` arquivo de ambiente no `.env` arquivo que corresponde à `BLUEXP_TF_AZURE_CREDS` variável de ambiente.

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

Etapa 4: inscreva-se no NetApp Intelligent Services

Inscreva-se no NetApp Intelligent Services por meio do seu provedor de nuvem para pagar por hora (PAYGO) ou por meio de um contrato anual. Os serviços inteligentes da NetApp incluem backup e recuperação da NetApp, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience e NetApp Disaster Recovery. A classificação de dados da NetApp está incluída na sua assinatura sem custo adicional.

Passos

1. No portal do Azure, navegue até **SaaS** e selecione **Assinar o NetApp Intelligent Services**.
2. Selecione o plano **Cloud Manager (por Cap PYGO por hora, WORM e serviços de dados)**.

Você pode usar o mesmo grupo de recursos que o Cloud Volumes ONTAP ou um grupo de recursos diferente.

3. Configure o portal do Console para importar a assinatura SaaS para o Console.

Você pode configurar isso diretamente no portal do Azure navegando até **Detalhes do produto e do plano** e selecionando a opção **Configurar conta agora**.

Você será redirecionado ao portal do Console para confirmar a configuração.

4. Confirme a configuração no portal do Console selecionando **Salvar**.

Passo 5: Crie um volume externo

Você deve criar um volume externo para manter os arquivos de estado do Terraform e outros arquivos importantes persistentes. Você deve garantir que os arquivos estejam disponíveis para o Terraform para executar o fluxo de trabalho e as implantações.

Passos

1. Criar um volume externo fora do Docker Compose:

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

Exemplo:

```
docker volume create cvo_azure_volume_dst
```

2. Use uma das seguintes opções:

- a. Adicione um caminho de volume externo ao `.env` arquivo de ambiente.

Você deve seguir o formato exato mostrado abaixo.

Formato:

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

Exemplo:

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

- b. Adicionar compartilhamentos NFS como volume externo.

Certifique-se de que o contentor Docker possa se comunicar com os compartilhamentos NFS e que as permissões corretas, como leitura/gravação, estejam configuradas.

- i. Adicione o caminho de compartilhamentos NFS como caminho para o volume externo no arquivo Docker Compose, como mostrado abaixo: Formato:

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

Exemplo:

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

3. Navegue até `cvo_azure_variables` a pasta.

Você deve ver os seguintes arquivos variáveis na pasta:

```
terraform.tfvars
```

```
variables.tf
```

4. Altere os valores dentro do `terraform.tfvars` arquivo de acordo com suas necessidades.

Você deve ler a documentação de suporte específica ao modificar qualquer um dos valores de variável no `terraform.tfvars` arquivo. Os valores podem variar dependendo da região, zonas de disponibilidade e outros fatores suportados pelo Cloud Volumes ONTAP para Azure. Isso inclui licenças, tamanho de disco e tamanho de VM para nós únicos e pares de alta disponibilidade (HA).

Todas as variáveis de suporte para o agente do Console e os módulos Terraform do Cloud Volumes ONTAP já estão definidas no `variables.tf` arquivo. Você deve se referir aos nomes das variáveis no `variables.tf` arquivo antes de adicionar ao `terraform.tfvars` arquivo.

5. Dependendo dos seus requisitos, pode ativar ou desativar o FlexCache e o FlexClone definindo as seguintes opções para `true` ou `false`.

Os exemplos a seguir habilitam o FlexCache e o FlexClone:

```
° is_flexcache_required = true
```

```
° is_flexclone_required = true
```

6. Se necessário, você pode recuperar o valor da variável Terraform `az_service_principal_object_id` no Serviço do Azure ative Directory:

- a. Navegue até **Enterprise Applications → All Applications** (aplicações empresariais) e selecione o nome do Service Principal que criou anteriormente.

- b. Copie o ID do objeto e insira o valor da variável Terraform:

```
az_service_principal_object_id
```

Etapa 6: Implante o Cloud Volumes ONTAP para Azure

Siga as etapas a seguir para implantar o Cloud Volumes ONTAP para Azure.

Passos

1. Na pasta raiz, execute o seguinte comando para acionar a implantação:

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

Dois contêineres são acionados, o primeiro contêiner implanta o Cloud Volumes ONTAP e o segundo contêiner envia dados de telemetria para o AutoSupport.

O segundo recipiente aguarda até que o primeiro recipiente conclua todas as etapas com êxito.

2. Monitore o progresso do processo de implantação usando os arquivos de log:

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

Este comando fornece saída em tempo real e captura os dados nos seguintes arquivos de log:

```
deployment.log
```

```
telemetry_asup.log
```

Você pode alterar o nome desses arquivos de log editando o `.env` arquivo usando as seguintes variáveis de ambiente:

```
DEPLOYMENT_LOGS
```

```
TELEMETRY_ASUP_LOGS
```

Os exemplos a seguir mostram como alterar os nomes dos arquivos de log:

```
DEPLOYMENT_LOGS=<your_deployment_log_filename>.log
```

```
TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log
```

Depois de terminar

Você pode usar as etapas a seguir para remover o ambiente temporário e limpar itens criados durante o processo de implantação.

Passos

1. Se você implantou o FlexCache, defina a seguinte opção no `terraform.tfvars` arquivo, isso limpa os volumes do FlexCache e remove o ambiente temporário criado anteriormente.

```
flexcache_operation = "destroy"
```



As opções possíveis são `deploy` e `destroy`.

2. Se você implantou o FlexClone, defina a seguinte opção no `terraform.tfvars` arquivo, isso limpa os volumes do FlexClone e remove o ambiente temporário criado anteriormente.

```
flexclone_operation = "destroy"
```



As opções possíveis são `deploy` e `destroy`.

Cloud Volumes ONTAP para Google Cloud

Cloud Volumes ONTAP para Google Cloud - estourar para a nuvem

Este artigo oferece suporte à solução NetApp Cloud Volumes ONTAP para Google Cloud Automation, disponível para clientes NetApp no hub de automação do NetApp Console .

A solução de automação da nuvem do Cloud Volumes ONTAP automatiza a implantação em contêineres do Cloud Volumes ONTAP para o Google Cloud, permitindo que você implante o Cloud Volumes ONTAP rapidamente, sem qualquer intervenção manual.

Antes de começar

- Você precisa baixar o "[Cloud Volumes ONTAP para Google Cloud - estourar para a nuvem](#)" Solução de automação através da interface web do Console. A solução é empacotada como `cvo_gcp_flexcache.zip`.
- Você deve instalar uma VM Linux na mesma rede que o Cloud Volumes ONTAP.
- Depois de instalar a VM Linux, você deve seguir as etapas desta solução para instalar as dependências necessárias.

Passo 1: Instale o Docker e o Docker Compose

Instale o Docker

As etapas a seguir usam o software de distribuição Debian Linux Ubuntu 20,04 como exemplo. Os comandos que você executa dependem do software de distribuição Linux que você está usando. Consulte a documentação específica do software de distribuição Linux para sua configuração.

Passos

1. Instale o Docker executando os seguintes comandos:

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

2. Verifique a instalação:

```
docker -version
```

3. Verifique se um grupo chamado "docker" foi criado em seu sistema Linux. Se necessário, crie o grupo:

```
sudo groupadd docker
```

4. Adicione o usuário que precisa acessar o Docker ao grupo:

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

5. As alterações são aplicadas depois de terminar sessão e voltar a iniciar sessão no terminal. Alternativamente, você pode aplicar as alterações imediatamente:

```
newgrp docker
```

Instale o Docker Compose

Passos

1. Instale o Docker Compose executando os seguintes `sudo` comandos:

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

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

2. Verifique a instalação:

```
docker-compose -version
```

Passo 2: Prepare a imagem do Docker

Passos

1. Copie a `cvo_gcp_flexcache.zip` pasta para a VM Linux que você deseja usar para implantar o 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` é o seu arquivo de chave privada para login sem uma senha.
- `gcpuser` É o nome de usuário da VM.
- `IP_ADDRESS_OF_VM` É o endereço IP da VM.

◦ LOCATION_TO_BE_COPIED é o local onde a pasta será copiada.

2. Extraia a `cvo_gcp_flexcache.zip` pasta. Você pode extrair a pasta no diretório atual ou em um local personalizado.

Para extrair a pasta no diretório atual, execute:

```
unzip cvo_gcp_flexcache.zip
```

Para extrair a pasta em um local personalizado, execute:

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

3. Depois de extrair o conteúdo, execute o seguinte comando para visualizar os arquivos:

```
ls -la
```

Você deve ver uma lista de arquivos, semelhante ao seguinte exemplo:

```
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. Localize o `cvo_gcp_flexcache_ubuntu_image.tar` arquivo. Isso contém a imagem Docker necessária para implantar o Cloud Volumes ONTAP para o Google Cloud.
5. Descomprimir o ficheiro:

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

6. Aguarde alguns minutos para que a imagem do Docker seja carregada e, em seguida, valide que a imagem do Docker foi carregada com sucesso:

```
docker images
```

Você deve ver uma imagem Docker chamada `cvo_gcp_flexcache_ubuntu_image` com a `latest` tag, como mostrado no exemplo a seguir:

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



Você pode alterar o nome da imagem do Docker, se necessário. Se você alterar o nome da imagem do Docker, certifique-se de atualizar o nome da imagem do Docker `docker-compose-deploy` nos arquivos e `docker-compose-destroy`.

Passo 3: Atualize o arquivo JSON

Neste estágio, você deve atualizar o `cxo-automation-gcp.json` arquivo com uma chave de conta de serviço para autenticar o provedor do Google Cloud.

1. Crie uma conta de serviço com permissões para implantar o Cloud Volumes ONTAP e um agente do Console "[Saiba mais sobre como criar contas de serviço.](#)"
2. Transfira o ficheiro de chave para a conta e atualize o `cxo-automation-gcp.json` ficheiro com as informações do ficheiro de chave. O `cxo-automation-gcp.json` ficheiro está localizado na `cvo_gcp_variables` pasta.

Exemplo

```
{
  "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"
}
```

O formato do arquivo deve ser exatamente como mostrado acima.

Etapa 4: inscreva-se no NetApp Intelligent Services

Inscreva-se no NetApp Intelligent Services por meio do seu provedor de nuvem para pagar por hora (PAYGO) ou por meio de um contrato anual. Os serviços inteligentes da NetApp incluem backup e recuperação da

NetApp , Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience e NetApp Disaster Recovery. A classificação de dados da NetApp está incluída na sua assinatura sem custo adicional.

Passos

1. Navegue até o ["Console do Google Cloud"](#) e selecione **Assinar os Serviços Inteligentes da NetApp **.
2. Configure o portal do NetApp Console para importar a assinatura SaaS para o Console.

Você pode configurar isso diretamente do Google Cloud Platform. Você será redirecionado ao portal do Console para confirmar a configuração.

3. Confirme a configuração no portal do Console selecionando **Salvar**.

Para obter mais informações, consulte ["Gerenciar credenciais e assinaturas do Google Cloud para o NetApp Console"](#).

Etapa 5: Habilite as APIs necessárias do Google Cloud

Você deve habilitar as seguintes APIs do Google Cloud no seu projeto para implantar o Cloud Volumes ONTAP e o agente do Console.

- API do Cloud Deployment Manager V2
- API Cloud Logging
- API do Cloud Resource Manager
- API do mecanismo de computação
- API de gerenciamento de identidade e acesso (IAM)

["Saiba mais sobre como habilitar APIs"](#)

Passo 6: Crie um volume externo

Você deve criar um volume externo para manter os arquivos de estado do Terraform e outros arquivos importantes persistentes. Você deve garantir que os arquivos estejam disponíveis para o Terraform para executar o fluxo de trabalho e as implantações.

Passos

1. Criar um volume externo fora do Docker Compose:

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

Exemplo:

```
docker volume create cvo_gcp_volume_dst
```

2. Use uma das seguintes opções:
 - a. Adicione um caminho de volume externo ao `.env` arquivo de ambiente.

Você deve seguir o formato exato mostrado abaixo.

Formato:

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

Exemplo:

```
PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp
```

b. Adicionar compartilhamentos NFS como volume externo.

Certifique-se de que o contentor Docker possa se comunicar com os compartilhamentos NFS e que as permissões corretas, como leitura/gravação, estejam configuradas.

- i. Adicione o caminho de compartilhamentos NFS como caminho para o volume externo no arquivo Docker Compose, como mostrado abaixo: Formato:

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

Exemplo:

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

3. Navegue até `cvo_gcp_variables` a pasta.

Você deve ver os seguintes arquivos na pasta:

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

4. Altere os valores dentro do `terraform.tfvars` arquivo de acordo com suas necessidades.

Você deve ler a documentação de suporte específica ao modificar qualquer um dos valores de variável no `terraform.tfvars` arquivo. Os valores podem variar dependendo da região, zonas de disponibilidade e outros fatores suportados pelo Cloud Volumes ONTAP para Google Cloud. Isso inclui licenças, tamanho de disco e tamanho de VM para nós únicos e pares de alta disponibilidade (HA).

Todas as variáveis de suporte para o agente do Console e os módulos Terraform do Cloud Volumes ONTAP já estão definidas no `variables.tf` arquivo. Você deve se referir aos nomes das variáveis no `variables.tf` arquivo antes de adicionar ao `terraform.tfvars` arquivo.

5. Dependendo dos seus requisitos, pode ativar ou desativar o FlexCache e o FlexClone definindo as seguintes opções para `true` ou `false`.

Os exemplos a seguir habilitam o FlexCache e o FlexClone:

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

Etapas 7: Implante o Cloud Volumes ONTAP para o Google Cloud

Siga as etapas a seguir para implantar o Cloud Volumes ONTAP para o Google Cloud.

Passos

1. Na pasta raiz, execute o seguinte comando para acionar a implantação:

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

Dois contêineres são acionados, o primeiro contêiner implanta o Cloud Volumes ONTAP e o segundo contêiner envia dados de telemetria para o AutoSupport.

O segundo recipiente aguarda até que o primeiro recipiente conclua todas as etapas com êxito.

2. Monitore o progresso do processo de implantação usando os arquivos de log:

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

Este comando fornece saída em tempo real e captura os dados nos seguintes arquivos de log:
deployment.log

telemetry_asup.log

Você pode alterar o nome desses arquivos de log editando o .env arquivo usando as seguintes variáveis de ambiente:

DEPLOYMENT_LOGS

TELEMETRY_ASUP_LOGS

Os exemplos a seguir mostram como alterar os nomes dos arquivos de log:

DEPLOYMENT_LOGS=<your_deployment_log_filename>.log

TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log

Depois de terminar

Você pode usar as etapas a seguir para remover o ambiente temporário e limpar itens criados durante o processo de implantação.

Passos

1. Se você implantou o FlexCache, defina a seguinte opção no terraform.tfvars arquivo, isso limpa os volumes do FlexCache e remove o ambiente temporário criado anteriormente.

```
flexcache_operation = "destroy"
```



As opções possíveis são deploy e destroy.

2. Se você implantou o FlexClone, defina a seguinte opção no terraform.tfvars arquivo, isso limpa os volumes do FlexClone e remove o ambiente temporário criado anteriormente.

```
flexclone_operation = "destroy"
```



As opções possíveis são deploy e destroy.

ONTAP

Dia 0/1

Visão geral da solução ONTAP Day 0/1

Você pode usar a solução de automação ONTAP day 0/1 para implantar e configurar um cluster ONTAP usando o Ansible. A solução está disponível em "[Hub de automação do NetApp Console](#)".

Opções flexíveis de implantação do ONTAP

Dependendo dos seus requisitos, use o hardware no local ou simule o ONTAP para implantar e configurar um cluster do ONTAP com o Ansible.

Hardware no local

É possível implantar essa solução usando hardware local executando ONTAP, como um sistema FAS ou AFF. Use uma máquina virtual do Linux para implantar e configurar o cluster do ONTAP usando o Ansible.

Simular ONTAP

Para implantar essa solução usando um simulador ONTAP, você deve baixar a versão mais recente do Simulate ONTAP no site de suporte da NetApp. Simule ONTAP é um simulador virtual para o software ONTAP. Simule a execução do ONTAP em um hypervisor VMware em um sistema Windows, Linux ou Mac. Para hosts Windows e Linux, você deve usar o hipervisor VMware Workstation para executar essa solução. Se você tiver um Mac os, use o hypervisor do VMware Fusion.

Design em camadas

A estrutura do Ansible simplifica o desenvolvimento e a reutilização de tarefas lógicas e de execução de automação. A estrutura faz uma distinção entre as tarefas de tomada de decisão (camada lógica) e as etapas de execução (camada de execução) na automação. Entender como essas camadas funcionam permite que você personalize a configuração.

Um "manual de estratégia" do Ansible executa uma série de tarefas do início ao fim. O `site.yml` manual de estratégia contém o `logic.yml` manual de estratégia e `execution.yml` o manual de estratégia.

Quando uma solicitação é executada, o `site.yml` manual de estratégia faz uma chamada primeiro para o `logic.yml` manual de estratégia e, em seguida, chama o `execution.yml` manual de estratégia para executar a solicitação de serviço.

Você não é obrigado a usar a camada lógica da estrutura. A camada lógica fornece opções para expandir a capacidade da estrutura além dos valores codificados para execução. Isso permite que você personalize os recursos da estrutura, se necessário.

Camada lógica

A camada lógica consiste no seguinte:

- O `logic.yml` manual de estratégia
- Arquivos de tarefa lógica dentro do `logic-tasks` diretório

A camada lógica fornece a capacidade para tomada de decisões complexas sem a necessidade de integração personalizada significativa (por exemplo, conectando-se ao ServiceNOW). A camada lógica é configurável e

fornece a entrada para microservices.

A capacidade de ignorar a camada lógica também é fornecida. Se você quiser ignorar a camada lógica, não defina a `logic_operation` variável. A invocação direta `logic.yml` do manual de estratégia fornece a capacidade de fazer algum nível de depuração sem execução. Você pode usar uma instrução "debug" para verificar se o valor do `raw_service_request` está correto.

Considerações importantes:

- O `logic.yml` manual de estratégia procura a `logic_operation` variável. Se a variável for definida na solicitação, ela carregará um arquivo de tarefa do `logic-tasks` diretório. O arquivo de tarefa deve ser um arquivo `.yml`. Se não houver nenhum arquivo de tarefa correspondente e a `logic_operation` variável for definida, a camada lógica falhará.
- O valor padrão `logic_operation` da variável é `no-op`. Se a variável não for definida explicitamente, ela será padrão para `no-op`, que não executa nenhuma operação.
- Se a `raw_service_request` variável já estiver definida, a execução prossegue para a camada de execução. Se a variável não estiver definida, a camada lógica falhará.

Camada de execução

A camada de execução consiste no seguinte:

- O `execution.yml` manual de estratégia

A camada de execução faz as chamadas de API para configurar um cluster ONTAP. O `execution.yml` manual de estratégia requer que a `raw_service_request` variável seja definida quando executada.

Suporte para personalização

Você pode personalizar esta solução de várias maneiras, dependendo de suas necessidades.

As opções de personalização incluem:

- Modificação de playbooks do Ansible
- Adicionar funções

Personalizar arquivos do Ansible

A tabela a seguir descreve os arquivos Ansible personalizáveis contidos nesta solução.

Localização	Descrição
<code>playbooks/inventory/hosts</code>	Contém um único arquivo com uma lista de hosts e grupos.
<code>playbooks/group_vars/all/*</code>	O Ansible fornece uma maneira conveniente de aplicar variáveis a vários hosts de uma só vez. Pode modificar qualquer ou todos os ficheiros desta pasta, incluindo <code>cfg.yml</code> , <code>clusters.yml</code> , <code>defaults.yml</code> , <code>services.yml</code> , <code>standards.yml</code> e <code>vault.yml</code> .
<code>playbooks/logic-tasks</code>	Dá suporte a tarefas de tomada de decisão no Ansible e mantém a separação entre lógica e execução. Pode adicionar ficheiros a esta pasta que correspondam ao serviço relevante.

Localização	Descrição
playbooks/vars/*	Valores dinâmicos usados nos playbooks e funções do Ansible para permitir a personalização, flexibilidade e reutilização de configurações. Se necessário, você pode modificar qualquer ou todos os arquivos nesta pasta.

Personalizar funções

Também é possível personalizar a solução adicionando ou alterando as funções do Ansible, também chamadas de microsserviços. Para obter mais detalhes, ["Personalizar"](#) consulte .

Prepare-se para usar a solução ONTAP Day 0/1

Antes de implantar a solução de automação, você precisa preparar o ambiente ONTAP e instalar e configurar o Ansible.

Considerações iniciais de Planejamento

Você deve analisar os requisitos e considerações a seguir antes de usar essa solução para implantar um cluster do ONTAP.

Requisitos básicos

Para usar essa solução, você precisa atender aos seguintes requisitos básicos:

- Você deve ter acesso ao software ONTAP, seja no local ou por meio de um simulador ONTAP.
- Você deve saber como usar o software ONTAP.
- Você precisa saber como usar as ferramentas de software de automação do Ansible.

Considerações de Planejamento

Antes de implantar essa solução de automação, você deve decidir:

- O local onde você executará o nó de controle do Ansible.
- O sistema ONTAP, seja hardware no local ou um simulador ONTAP.
- Se você vai ou não precisar de personalização.

Prepare o sistema ONTAP

Não importa se você está usando um sistema ONTAP no local ou simule o ONTAP, prepare o ambiente antes de implantar a solução de automação.

Opcionalmente, instale e configure o Simulate ONTAP

Se você quiser implantar essa solução através de um simulador ONTAP, você deve baixar e executar o Simulate ONTAP.

Antes de começar

- É necessário fazer o download e instalar o hypervisor VMware que você vai usar para executar o Simulate ONTAP.
 - Se você tiver um sistema operacional Windows ou Linux, use o VMware Workstation.
 - Se você tiver um Mac os, use o VMware Fusion.



Se você estiver usando um Mac os, você deve ter um processador Intel.

Passos

Use o seguinte procedimento para instalar dois simuladores ONTAP em seu ambiente local:

1. Faça o download do Simulate ONTAP no "[Site de suporte da NetApp](#)".



Embora você instale dois simuladores ONTAP, você só precisa baixar uma cópia do software.

2. Se ele ainda não estiver em execução, inicie seu aplicativo VMware.
3. Localize o arquivo do simulador que foi baixado e clique com o botão direito do Mouse para abri-lo com o aplicativo VMware.
4. Defina o nome da primeira instância do ONTAP.
5. Aguarde a inicialização do simulador e siga as instruções para criar um cluster de nó único.

Repita as etapas para a segunda instância do ONTAP.

6. Opcionalmente, adicione um complemento de disco completo.

Em cada cluster, execute os seguintes comandos:

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

Estado do sistema ONTAP

Você deve verificar o estado inicial do sistema ONTAP, seja no local ou em execução através de um simulador ONTAP.

Verifique se os seguintes requisitos do sistema ONTAP são atendidos:

- O ONTAP está instalado e em execução sem cluster definido ainda.
- O ONTAP é inicializado e exibe o endereço IP para acessar o cluster.
- A rede é acessível.
- Você tem credenciais de administrador.
- O banner mensagem do dia (MOTD) é exibido com o endereço de gerenciamento.

Instale o software de automação necessário

Esta seção fornece informações sobre como instalar o Ansible e preparar a solução de automação para implantação.

Instalar o Ansible

O Ansible pode ser instalado em sistemas Linux ou Windows.

O método de comunicação padrão usado pelo Ansible para se comunicar com um cluster ONTAP é SSH.

Consulte a ["Primeiros passos com o NetApp e o Ansible: Instale o Ansible"](#) instalação do Ansible.



O Ansible precisa ser instalado no nó de controle do sistema.

Baixe e prepare a solução de automação

Você pode usar as etapas a seguir para baixar e preparar a solução de automação para implantação.

1. Baixe o ["ONTAP - dia 0/1 verificações de estado"](#) Solução de automação através da interface web do Console. A solução é empacotada como `ONTAP_DAY0_DAY1.zip`.
2. Extraia a pasta zip e copie os arquivos para o local desejado no nó de controle em seu ambiente Ansible.

Configuração inicial da estrutura do Ansible

Execute a configuração inicial da estrutura do Ansible:

1. Navegue até `playbooks/inventory/group_vars/all`.
2. Descriptar o `vault.yml` ficheiro:

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

Quando for solicitada a senha do cofre, digite a seguinte senha temporária:

NetApp123!



"NetApp123!" é uma senha temporária para descriptografar o `vault.yml` arquivo e a senha do cofre correspondente. Após o primeiro uso, você **deve** criptografar o arquivo usando sua própria senha.

3. Modifique os seguintes arquivos do Ansible:

- `clusters.yml` - Modifique os valores neste arquivo para se adequar ao seu ambiente.
- `vault.yml` - Depois de descriptografar o arquivo, modifique os valores de cluster, nome de usuário e senha do ONTAP para se adequar ao seu ambiente.
- `cfg.yml` - Defina o caminho do arquivo `log2file` e defina `show_request_cfg` como `True` para exibir o `raw_service_request`.

A `raw_service_request` variável é exibida nos arquivos de log e durante a execução.



Cada arquivo listado contém comentários com instruções sobre como modificá-lo de acordo com suas necessidades.

4. Recriptografe o `vault.yml` arquivo:

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



Você será solicitado a escolher uma nova senha para o cofre após a criptografia.

5. Navegue `playbooks/inventory/hosts` e defina um interpretador Python válido.

6. Implantar o `framework_test` serviço:

O comando a seguir executa o `na_ontap_info` módulo com um `gather_subset` valor `cluster_identity_info` de `.` Isso valida que a configuração básica está correta e verifica se você pode se comunicar com o cluster.

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

Execute o comando para cada cluster.

Se for bem-sucedido, você verá uma saída semelhante ao seguinte exemplo:

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

Implante o cluster do ONTAP usando a solução

Após concluir a preparação e o Planejamento, você estará pronto para usar a solução ONTAP day 0/1 para configurar rapidamente um cluster do ONTAP usando o Ansible.

A qualquer momento durante as etapas desta seção, você pode optar por testar uma solicitação em vez de executá-la. Para testar uma solicitação, altere o `site.yml` manual na linha de comando para `logic.yml`.



A `docs/tutorial-requests.txt` localização contém a versão final de todos os pedidos de assistência utilizados durante este procedimento. Se tiver dificuldade em executar uma solicitação de serviço, você pode copiar a solicitação relevante do `tutorial-requests.txt` arquivo para `playbooks/inventory/group_vars/all/tutorial-requests.yml` o local e modificar os valores codificados conforme necessário (endereço IP, nomes agregados, etc.). Em seguida, você deve ser capaz de executar com sucesso a solicitação.

Antes de começar

- Tenha o Ansible instalado.
- Você precisa ter baixado a solução do dia 0/1 do ONTAP e extraído a pasta para o local desejado no nó de controle do Ansible.
- O estado do sistema ONTAP deve atender aos requisitos e você precisa ter as credenciais necessárias.

- Você deve ter concluído todas as tarefas necessárias descritas na "[Prepare-se](#)" seção.



Os exemplos dessa solução usam "Cluster_01" e "Cluster_02" como os nomes dos dois clusters. É necessário substituir esses valores pelos nomes dos clusters no ambiente.

Etapa 1: Configuração inicial do cluster

Neste estágio, você deve executar algumas etapas iniciais de configuração do cluster.

Passos

1. Navegue até o `playbooks/inventory/group_vars/all/tutorial-requests.yml` local e reveja a `cluster_initial` solicitação no arquivo. Faça as alterações necessárias para o seu ambiente.
2. Crie um arquivo `logic-tasks` na pasta para a solicitação de serviço. Por exemplo, crie um arquivo `cluster_initial.yml` chamado .

Copie as seguintes linhas para o novo arquivo:

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

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

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

3. Defina a `raw_service_request` variável.

Você pode usar uma das seguintes opções para definir a `raw_service_request` variável no `cluster_initial.yml` arquivo que você criou na `logic-tasks` pasta:

- **Opção 1:** Defina manualmente a `raw_service_request` variável.

Abra o `tutorial-requests.yml` arquivo usando um editor e copie o conteúdo da linha 11 para a linha 165. Cole o conteúdo sob a `raw service request` variável no novo `cluster_initial.yml` arquivo, como mostrado nos exemplos a seguir:

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

Mostrar exemplo

Ficheiro de exemplo 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
    - vservers-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:
            - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

[illegible]

```
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 }}"
    vservers:              "{{ 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 }}"
    vservers:              "{{ 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 }}"

```

- **Opção 2:** Use um modelo Jinja para definir a solicitação:

Você também pode usar o seguinte formato de modelo Jinja para obter o `raw_service_request` valor.

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

4. Execute a configuração inicial do cluster para o primeiro cluster:

```

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

```

Verifique se não existem erros antes de prosseguir.

5. Repita o comando para o segundo cluster:

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

Verifique se não há erros para o segundo cluster.

Ao rolar para cima em direção ao início da saída do Ansible, você verá a solicitação que foi enviada para a estrutura, como mostrado no exemplo a seguir:

[illegible]

```

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

6. Faça login em cada instância do ONTAP e verifique se a solicitação foi bem-sucedida.

Etapa 2: Configurar os LIFs entre clusters

Agora você pode configurar as LIFs entre clusters adicionando as definições de LIF à `cluster_initial` solicitação e definindo o `ontap_interface microservice`.

A definição do serviço e a solicitação trabalham em conjunto para determinar a ação:

- Se você fornecer uma solicitação de serviço para um microservice que não esteja nas definições de serviço, a solicitação não será executada.
- Se você fornecer uma solicitação de serviço com um ou mais microsserviços definidos nas definições de serviço, mas omitido da solicitação, a solicitação não será executada.

O `execution.yml` manual de estratégia avalia a definição do serviço digitalizando a lista de microsserviços na ordem listada:

- Se houver uma entrada na solicitação com uma chave de dicionário que corresponda à `args` entrada contida nas definições de microservices, a solicitação será executada.
- Se não houver nenhuma entrada correspondente na solicitação de serviço, a solicitação será ignorada sem erro.

Passos

1. Navegue até o `cluster_initial.yml` arquivo que você criou anteriormente e modifique a solicitação adicionando as seguintes linhas às definições da solicitação:

```

ontap_interface:
- hostname:                "{{ cluster_name }}"
  vservers:                "{{ 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 }}"
  vservers:                "{{ 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 }}"
  vservers:                "{{ 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 }}"
  vservers:                "{{ 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. Execute o comando:

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

3. Faça login em cada instância para verificar se os LIFs foram adicionados ao cluster:

Mostrar exemplo

```
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.
```

A saída mostra que os LIFs foram **not** adicionados. Isso ocorre porque o `ontap_interface` microservice ainda precisa ser definido no `services.yml` arquivo.

4. Verifique se os LIFs foram adicionados à `raw_service_request` variável.

Mostrar exemplo

O exemplo a seguir mostra que os LIFs foram adicionados à solicitação:

```
"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",  
    "vservers": "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",  
    "vservers": "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",  
    "vservers": "Cluster_02"  
  },  
  {  
    "address": "10.0.0.126",  
    "home_node": "Cluster_02-01",  
    "home_port": "e0c",  
    "hostname": "Cluster_02",
```

```

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

```

5. Defina o `ontap_interface` microservice em `cluster_initial` no `services.yml` arquivo.

Copie as seguintes linhas para o arquivo para definir o microservice:

```

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

```

6. Agora que o `ontap_interface` microservice foi definido na solicitação e no `services.yml` arquivo, execute a solicitação novamente:

```

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

```

7. Faça login em cada instância do ONTAP e verifique se os LIFs foram adicionados.

Etapa 3: Opcionalmente, configure vários clusters

Se necessário, você pode configurar vários clusters na mesma solicitação. Você deve fornecer nomes de variáveis para cada cluster quando definir a solicitação.

Passos

1. Adicione uma entrada para o segundo cluster `cluster_initial.yml` no arquivo para configurar ambos os clusters na mesma solicitação.

O exemplo a seguir exibe o `ontap_agggr` campo depois que a segunda entrada é adicionada.

```

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

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

```

2. Aplique as alterações para todos os outros itens em `cluster_initial`.
3. Adicione peering de cluster à solicitação copiando as seguintes linhas para o arquivo:

```

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. Execute a solicitação do 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>

```

Etapa 4: Configuração inicial da SVM

Nesta etapa do procedimento, você configura os SVMs no cluster.

Passos

1. Atualize a `svm_initial` solicitação no `tutorial-requests.yml` arquivo para configurar um relacionamento de pares SVM e SVM.

Você deve configurar o seguinte:

- O SVM

- O relacionamento entre pares SVM
 - A interface SVM para cada SVM
2. Atualize as definições de variáveis nas `svm_initial` definições de solicitação. Você deve modificar as seguintes definições de variáveis:

- `cluster_name`
- `vserver_name`
- `peer_cluster_name`
- `peer_vserver`

Para atualizar as definições, remova o `*` depois `req_details` para a `svm_initial` definição e adicione a definição correta.

3. Crie um arquivo `logic-tasks` na pasta para a solicitação de serviço. Por exemplo, crie um arquivo `svm_initial.yml` chamado .

Copie as seguintes linhas para o arquivo:

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

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

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

4. Defina a `raw_service_request` variável.

Pode utilizar uma das seguintes opções para definir a `raw_service_request` variável `svm_initial` `logic-tasks` na pasta:

- **Opção 1:** Defina manualmente a `raw_service_request` variável.

Abra o `tutorial-requests.yml` arquivo usando um editor e copie o conteúdo da linha 179 para a

linha 222. Cole o conteúdo sob a `raw service request` variável no novo `svm_initial.yml` arquivo, como mostrado nos exemplos a seguir:

```
177
178   svm_initial:
179     service:          svm_initial
180     request_type:     create
181     std_name:         none
182     req_details:
183
184     ontap_vserver:
185       - hostname:      "{{ cluster_name }}"
186         name:          "{{ vserver_name }}"
187         root_volume_aggregate: n01_aggr1
188
189       - hostname:      "{{ peer_cluster_name }}"
190         name:          "{{ peer_vserver }}"
191         root_volume_aggregate: n01_aggr1
192
```

Mostrar exemplo

Ficheiro de exemplo `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
    - vservers-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:          "{{ vservers_name }}"
          root_volume_aggregate:  n01_aggr1

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

      ontap_vserver_peer:
        - hostname:      "{{ cluster_name }}"
          vservers:      "{{ vservers_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

```

- **Opção 2:** Use um modelo Jinja para definir a solicitação:

Você também pode usar o seguinte formato de modelo Jinja para obter o `raw_service_request` valor.

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

5. Execute a solicitação:

```

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. Faça login em cada instância do ONTAP e valide a configuração.

7. Adicione as interfaces SVM.

Defina `ontap_interface` o serviço em `svm_initial` `services.yml` no arquivo e execute a solicitação novamente:

```
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. Faça login em cada instância do ONTAP e verifique se as interfaces SVM foram configuradas.

Etapa 5: Opcionalmente, defina uma solicitação de serviço dinamicamente

Nas etapas anteriores, a `raw_service_request` variável é codificada por hardware. Isso é útil para aprendizado, desenvolvimento e teste. Você também pode gerar dinamicamente uma solicitação de serviço.

A seção a seguir fornece uma opção para produzir dinamicamente o necessário `raw_service_request` se você não quiser integrá-lo com sistemas de nível superior.



- Se a `logic_operation` variável não estiver definida no comando, o `logic.yml` arquivo não importa nenhum arquivo da `logic-tasks` pasta. Isso significa que o `raw_service_request` precisa ser definido fora do Ansible e fornecido à estrutura em execução.
- Um nome de arquivo de tarefa `logic-tasks` na pasta deve corresponder ao valor da `logic_operation` variável sem a extensão `.yml`.
- Os arquivos de tarefa na `logic-tasks` pasta definem dinamicamente um `raw_service_request`. o único requisito é que um válido `raw_service_request` seja definido como a última tarefa no arquivo relevante.

Como definir dinamicamente uma solicitação de serviço

Há várias maneiras de aplicar uma tarefa lógica para definir dinamicamente uma solicitação de serviço. Algumas destas opções estão listadas abaixo:

- Usando um arquivo de tarefa Ansible `logic-tasks` da pasta
- Invocando uma função personalizada que retorna dados adequados para converter para um `raw_service_request` variable.
- Invocando outra ferramenta fora do ambiente Ansible para fornecer os dados necessários. Por exemplo, uma chamada de API REST para o Active IQ Unified Manager.

Os comandos de exemplo a seguir definem dinamicamente uma solicitação de serviço para cada cluster usando o `tutorial-requests.yml` arquivo:

```
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
```

Etapa 6: Implante a solução ONTAP Day 0/1

Nesta fase, você já deve ter completado o seguinte:

- Revisou e modificou todos os arquivos `playbooks/inventory/group_vars/all` de acordo com suas necessidades. Há comentários detalhados em cada arquivo para ajudá-lo a fazer as alterações.
- Adicionado todos os arquivos de tarefa necessários ao `logic-tasks` diretório.
- Adicionado todos os arquivos de dados necessários ao `playbook/vars` diretório.

Use os comandos a seguir para implantar a solução ONTAP day 0/1 e verificar a integridade da implantação:



Nesta fase, você já deve ter descriptografado e modificado o `vault.yml` arquivo e ele deve ser criptografado com sua nova senha.

- Execute o serviço 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>
```

- Execute o serviço ONTAP Day 1:

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

- Aplicar definições de largura do 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>
```

- Executar verificações de integridade:

```
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>
```

Personalize a solução ONTAP Day 0/1

Para personalizar a solução do dia 0/1 do ONTAP de acordo com seus requisitos, você pode adicionar ou alterar as funções do Ansible.

As funções representam os microsserviços na estrutura do Ansible. Cada microservice executa uma

operação. Por exemplo, o ONTAP Day 0 é um serviço que contém vários microsserviços.

Adicione funções do Ansible

Você pode adicionar funções do Ansible para personalizar a solução para seu ambiente. As funções necessárias são definidas pelas definições de serviço na estrutura do Ansible.

Uma função deve atender aos seguintes requisitos para ser usada como microsserviço:

- Aceite uma lista de argumentos na `args` variável.
- Use a estrutura "bloco, resgate, sempre" do Ansible com certos requisitos para cada bloco.
- Use um único módulo do Ansible e defina uma única tarefa dentro do bloco.
- Implemente todos os parâmetros do módulo disponíveis de acordo com os requisitos detalhados nesta seção.

Estrutura de microsserviço necessária

Cada função deve suportar as seguintes variáveis:

- `mode`: Se o modo estiver definido para `test` a função tenta importar o `test.yml` que mostra o que a função faz sem realmente executá-la.



Nem sempre é possível implementar isso por causa de certas interdependências.

- `status`: O status geral da execução do playbook. Se o valor não estiver definido para `success` a função não será executado.
- `args`: Uma lista de dicionários específicos da função com chaves que correspondem aos nomes dos parâmetros da função.
- `global_log_messages`: Reúne mensagens de log durante a execução do manual de estratégia. Há uma entrada gerada cada vez que a função é executada.
- `log_name`: O nome usado para se referir à função dentro das `global_log_messages` entradas.
- `task_descr`: Uma breve descrição do que o papel faz.
- `service_start_time`: O carimbo de data/hora usado para rastrear a hora em que cada função é executada.
- `playbook_status`: O status do manual de estratégia do Ansible.
- `role_result`: A variável que contém a saída de função e é incluída em cada mensagem dentro das `global_log_messages` entradas.

Exemplo de estrutura de função

O exemplo a seguir fornece a estrutura básica de uma função que implementa um microservice. Você deve alterar as variáveis neste exemplo para sua configuração.

Mostrar exemplo

Estrutura básica da função:

```
- 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 ] }}"

```

Variáveis usadas na função de exemplo:

- `<NAME>`: Um valor substituível que deve ser fornecido para cada microserviço.
- `<LOG_NAME>`: O nome do formulário curto da função usada para fins de Registro. Por exemplo, `ONTAP_VOLUME`.
- `<TASK_DESCRIPTION>`: Uma breve descrição do que o microservice faz.
- `<MODULE_NAME>`: O nome do módulo Ansible para a tarefa.



O manual de estratégia de nível superior `execute.yml` especifica a `netapp.ontap` coleção. Se o módulo faz parte da `netapp.ontap` coleção, não há necessidade de especificar completamente o nome do módulo.

- `<MODULE_SPECIFIC_PARAMETERS>`: Parâmetros do módulo Ansible que são específicos do módulo usado para implementar o microservice. A lista a seguir descreve os tipos de parâmetros e como eles devem ser agrupados.
 - Parâmetros necessários: Todos os parâmetros necessários são especificados sem valor padrão.
 - Parâmetros que têm um valor padrão específico para o microservice (não o mesmo que um valor padrão especificado pela documentação do módulo).
 - Todos os parâmetros restantes usam `default(omit)` como valor padrão.

Usando dicionários de vários níveis como parâmetros do módulo

Alguns módulos do NetApp fornecem o Ansible que usam dicionários de vários níveis para parâmetros do módulo (por exemplo, grupos de políticas de QoS fixos e adaptáveis).

Usar `default(omit)` sozinho não funciona quando esses dicionários são usados, especialmente quando há mais de um e eles são mutuamente exclusivos.

Se você precisa usar dicionários de vários níveis como parâmetros de módulo, você deve dividir a funcionalidade em vários microserviços (funções) para que cada um tenha a garantia de fornecer pelo menos um valor de dicionário de segundo nível para o dicionário relevante.

Os exemplos a seguir mostram grupos de políticas de QoS fixos e adaptáveis divididos em dois microserviços.

O primeiro microservice contém valores fixos de grupo de políticas de QoS:

```

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)
}}}"

```

O segundo microservice contém os valores do grupo de políticas de QoS adaptáveis:

```

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) }}}"

```

Informações sobre direitos autorais

Copyright © 2025 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTES DOCUMENTOS. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALIENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTES SOFTWARES, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em <http://www.netapp.com/TM> são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.