Amazon FSx for NetApp ONTAP

NetApp Automation

NetApp
February 11, 2024

Table of Contents

Amazon FSx for NetApp ONTAP 1
Amazon FSx for NetApp ONTAP - Burst to cloud 1
Amazon FSx for NetApp ONTAP - Disaster recovery 5

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP - Burst to cloud

You can use this automation solution to provision Amazon FSx for NetApp ONTAP with volumes and an associated FlexCache.

Amazon FSx for NetApp ONTAP is also referred to as **FSx for ONTAP**.

About this solution

At a high level, the automation code provided with this solution performs the following actions:

- Provision a destination FSx for ONTAP file system
- Provision Storage Virtual Machines (SVMs) for the file system
- Create a cluster peering relationship between the source and destination systems
- Create an SVM peering relationship between the source system and destination system for FlexCache
- Optionally create FlexVol volumes using FSx for ONTAP
- Create a FlexCache volume in FSx for ONTAP with the source pointing to on-prem storage

The automation is based on Docker and Docker Compose which must be installed on the Linux virtual machine as described below.

Before you begin

You must have the following to complete the provisioning and configuration:

- You need to download the Amazon FSx for NetApp ONTAP - Burst to cloud automation solution through the BlueXP web UI. The solution is packaged as file `AWS_FSxN_BTC.zip`.
- Network connectivity between the source and destination systems.
- A Linux VM with the following characteristics:
  - Debian-based Linux distribution
  - Deployed on the same VPC subset used for FSx for ONTAP provisioning
- AWS account.

Step 1: Install and configure Docker

Install and configure Docker in a Debian-based Linux virtual machine.

Steps

1. Prepare the environment.

```
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. Install Docker and verify the installation.

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

3. Add the required Linux group with an associated user.

   First check if the group **docker** exists in your Linux system. If it doesn't, create the group and add the user. By default, the current shell user is added to the group.

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

4. Activate the new group and user definitions

   If you created a new group with a user, you need to activate the definitions. To do this, you can logout of Linux and then back in. Or you can run the following command.

```
newgrp docker
```

Step 2: Install Docker Compose

Install Docker Compose in a Debian-based Linux virtual machine.

Steps

1. Install 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. Verify the installation was successful.

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

Step 3: Prepare the Docker image

You need to extract and load the Docker image provided with the automation solution.

Steps

1. Copy the solution file `AWS_FSxN_BTC.zip` to the virtual machine where the automation code will run.

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

   The input parameter `private-key.pem` is your private key file used for AWS virtual machine authentication (EC2 instance).

2. Navigate to the correct folder with the solution file and unzip the file.

   ```
   unzip AWS_FSxN_BTC.zip
   ```

3. Navigate to the new folder `AWS_FSxN_BTC` created with the unzip operation and list the files. You should see file `aws_fsxn_flexcache_image_latest.tar.gz`.

   ```
   ls -la
   ```

4. Load the Docker image file. The load operation should normally complete in a few seconds.

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

5. Confirm the Docker image is loaded.

   ```
   docker images
   ```

   You should see the Docker image `aws_fsxn_flexcache_image` with the tag `latest`.

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

Step 4: Create environment file for AWS credentials

You must create a local variable file for authentication using the access and secret key. Then add the file to the `.env` file.

Steps

1. Create the `awsauth.env` file in the following location:

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

2. Add the following content to the file:

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

   The format **must** be exactly as shown above without any spaces between `key` and `value`.

3. Add the absolute file path to the `.env` file using the `AWS_CREDS` variable. For example:

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

Step 5: Create an external volume

You need an external volume to make sure the Terraform state files and other important files are persistent. These files must be available for Terraform to run the workflow and deployments.

Steps

1. Create an external volume outside of Docker Compose.

   Make sure to update the volume name (last parameter) to the appropriate value before running the command.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Add the path to the external volume to the `.env` environment file using the command:

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

   Remember to keep the existing file contents and colon formatting. For example:

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

   You can instead add an NFS share as the external volume using a command such as:

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

3. Update the Terraform variables.
   a. Navigate to the folder `aws_fsxn_variables`.
   b. Confirm the following two files exist: `terraform.tfvars` and `variables.tf`.
   c. Update the values in `terraform.tfvars` as required for your environment.

See Terraform resource: aws_fsx_ontap_file_system for more information.

Step 6: Provision Amazon FSx for NetApp ONTAP and FlexCache

You can provision Amazon FSx for NetApp ONTAP and FlexCache.

Steps

1. Navigate to the folder root (AWS_FSXN_BTC) and issue the provisioning command.

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

   This command creates two containers. The first container deploys FSx for ONTAP and the second container creates the cluster peering, SVM peering, destination volume, and FlexCache.

2. Monitor the provisioning process.

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

   This command gives you the output in real time, but has been configured to capture the logs through the file `deployment.log`. You can change the name of these log files by editing the `.env` file and updating the variables `DEPLOYMENT_LOGS`.

Step 7: Destroy Amazon FSx for NetApp ONTAP and FlexCache

You can optionally delete and remove Amazon FSx for NetApp ONTAP and FlexCache.

1. Set the variable `flexcache_operation` in the `terraform.tfvars` file to "destroy".
2. Navigate to the folder root (AWS_FSXN_BTC) and issue the following command.

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

   This command creates two containers. The first container delete FlexCache and the second container deletes FSx for ONTAP.

3. Monitor the provisioning process.

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

Amazon FSx for NetApp ONTAP - Disaster recovery

You can use this automation solution to take a disaster recovery backup of a source system using Amazon FSx for NetApp ONTAP.

Amazon FSx for NetApp ONTAP is also referred to as **FSx for ONTAP**.

About this solution

At a high level, the automation code provided with this solution performs the following actions:

- Provision a destination FSx for ONTAP file system
- Provision Storage Virtual Machines (SVMs) for the file system
- Create a cluster peering relationship between the source and destination systems
- Create an SVM peering relationship between the source system and destination system for SnapMirror
- Create destination volumes
- Create a SnapMirror relationship between the source and destination volumes
- Initiate the SnapMirror transfer between the source and destination volumes

The automation is based on Docker and Docker Compose which must be installed on the Linux virtual machine as described below.

Before you begin

You must have the following to complete the provisioning and configuration:

- You need to download the Amazon FSx for NetApp ONTAP - Disaster recovery automation solution through the BlueXP web UI. The solution is packaged as `FSxN_DR.zip`. This zip contains the `AWS_FSxN_Bck_Prov.zip` file that you will use to deploy the solution described in this document.
- Network connectivity between the source and destination systems.
- A Linux VM with the following characteristics:
  - Debian-based Linux distribution
  - Deployed on the same VPC subset used for FSx for ONTAP provisioning
- An AWS account.

Step 1: Install and configure Docker

Install and configure Docker in a Debian-based Linux virtual machine.

Steps

1. Prepare the environment.

```
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. Install Docker and verify the installation.

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

3. Add the required Linux group with an associated user.

   First check if the group **docker** exists in your Linux system. If it doesn't exist, create the group and add the user. By default, the current shell user is added to the group.

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

4. Activate the new group and user definitions

   If you created a new group with a user, you need to activate the definitions. To do this, you can logout of Linux and then back in. Or you can run the following command.

```
newgrp docker
```

Step 2: Install Docker Compose

Install Docker Compose in a Debian-based Linux virtual machine.

Steps

1. Install 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. Verify the installation was successful.

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

Step 3: Prepare the Docker image

You need to extract and load the Docker image provided with the automation solution.

Steps

1. Copy the solution file `AWS_FSxN_Bck_Prov.zip` to the virtual machine where the automation code will run.

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

The input parameter `private-key.pem` is your private key file used for AWS virtual machine authentication (EC2 instance).

2. Navigate to the correct folder with the solution file and unzip the file.

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. Navigate to the new folder `AWS_FSxN_Bck_Prov` created with the unzip operation and list the files. You should see file `aws_fsxn_bck_image_latest.tar.gz`.

```
ls -la
```

4. Load the Docker image file. The load operation should normally complete in a few seconds.

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

5. Confirm the Docker image is loaded.

```
docker images
```

You should see the Docker image `aws_fsxn_bck_image` with the tag `latest`.

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

Step 4: Create environment file for AWS credentials

You must create a local variable file for authentication using the access and secret key. Then add the file to the `.env` file.

Steps

1. Create the `awsauth.env` file in the following location:

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

2. Add the following content to the file:

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

The format **must** be exactly as shown above without any spaces between `key` and `value`.

3. Add the absolute file path to the `.env` file using the `AWS_CREDS` variable. For example:

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

Step 5: Create an external volume

You need an external volume to make sure the Terraform state files and other important files are persistent. These files must be available for Terraform to run the workflow and deployments.

Steps

1. Create an external volume outside of Docker Compose.

   Make sure to update the volume name (last parameter) to the appropriate value before running the command.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Add the path to the external volume to the `.env` environment file using the command:

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

   Remember to keep the existing file contents and colon formatting. For example:

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

   You can instead add an NFS share as the external volume using a command such as:

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

3. Update the Terraform variables.
   a. Navigate to the folder `aws_fsxn_variables`.
   b. Confirm the following two files exist: `terraform.tfvars` and `variables.tf`.
   c. Update the values in `terraform.tfvars` as required for your environment.

      See Terraform resource: aws_fsx_ontap_file_system for more information.

Step 6: Deploy the backup solution

You can deploy and provision the disaster recovery backup solution.

Steps

1. Navigate to the folder root (AWS_FSxN_Bck_Prov) and issue the provisioning command.

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

   This command creates three containers. The first container deploys FSx for ONTAP. The second container creates the cluster peering, SVM peering, and destination volume. The third container creates the SnapMirror relationship and initiates the SnapMirror transfer.

2. Monitor the provisioning process.

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

   This command gives you the output in real time, but has been configured to capture the logs through the file `deployment.log`. You can change the name of these log files by editing the `.env` file and updating the variables `DEPLOYMENT_LOGS`.

Copyright information

Trademark information