Cloud Volumes ONTAP for Azure

NetApp Automation

NetApp
November 18, 2025

Table of Contents

Cloud Volumes ONTAP for Azure

Cloud Volumes ONTAP for Azure - Burst to cloud

This article supports the NetApp Cloud Volumes ONTAP for Azure Automation Solution, which is available to NetApp customers from the NetApp Console automation hub.

The Cloud Volumes ONTAP for Azure Automation Solution automates the containerized deployment of Cloud Volumes ONTAP for Azure using Terraform, enabling you to deploy Cloud Volumes ONTAP for Azure rapidly, without any manual intervention.

Before you begin

- You must download the Cloud Volumes ONTAP Azure - Burst to cloud automation solution through the Console web UI. The solution is packaged as `CVO-Azure-Burst-To-Cloud.zip`.
- You must install a Linux VM on the same network as Cloud Volumes ONTAP.
- After you install the Linux VM, you must follow the steps in this solution to install the required dependencies.

Step 1: Install Docker and Docker Compose

Install Docker

The following steps use Ubuntu 20.04 Debian Linux distribution software as an example. The commands you run depend on the Linux distribution software that you are using. Refer to the specific Linux distribution software documentation for your configuration.

Steps

1. Install Docker by running the following `sudo` commands:

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

2. Verify the installation:

```
docker –version
```

3. Verify that a group named "docker" has been created on your Linux system. If necessary, create the group:

```
sudo groupadd docker
```

4. Add the user that needs to access Docker to the group:

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

5. Your changes are applied after you log out and log back in to the terminal. Alternatively, you can apply the changes immediately:

```
newgrp docker
```

Install Docker Compose

Steps

1. Install Docker Compose by running the following `sudo` commands:

```
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. Verify the installation:

```
docker-compose –version
```

Step 2: Prepare the Docker image

Steps

1. Copy the `CVO-Azure-Burst-To-Cloud.zip` folder to the Linux VM that you want to use to deploy 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` is your private key file for login without a password.
   - `azureuser` is the VM username.
   - `IP_ADDRESS_OF_VM` is the VM IP address.
   - `LOCATION_TO_BE_COPIED` is the location where the folder will be copied.

2. Extract the `CVO-Azure-Burst-To-Cloud.zip` folder. You can extract the folder in the current directory or in a custom location.

   To extract the folder in the current directory, run:

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

   To extract the folder in a custom location, run:

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

3. After you extract the content, navigate to the `CVO_Azure_Deployment` folder and run the following command to view the files:

   ```
    ls -la
   ```

   You should see a list of files, similar to the following example:

   ```
   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. Locate the `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` file. This contains the Docker image required to deploy Cloud Volumes ONTAP for Azure.
5. Untar the file:

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

6. Wait a few minutes for the Docker image to load, and then validate that the Docker image loaded successfully:

   ```
   docker images
   ```

You should see a Docker image named `cvo_azure_flexcache_ubuntu_image_latest` with the `latest` tag, as shown in the following example:

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

Step 3: Create environment variable files

At this stage, you must create two environment variable files. One file is for authentication of Azure Resource Manager APIs using service principal credentials. The second file is for setting environment variables to enable the Console Terraform modules to locate and authenticate Azure APIs.

Steps

1. Create a service principal.

   Before you can create the environment variable files, you must create a service principal by following the steps in Create an Azure Active Directory application and service principal that can access resources.

2. Assign the **Contributor** role to the newly created service principal.
3. Create a custom role.
   a. Locate the `sp_role.json` file and check for the required permissions under the actions listed.
   b. Insert these permissions and attach the custom role to the newly created service principal.
4. Navigate to **Certificates & secrets** and select **New client secret** to create the client secret.

   When you create the client secret, you must record the details from the **Value** column because you will not be able to see this value again. You must also record the following information:

   - Client ID
   - Subscription ID
   - Tenant ID

   You will need this information to create the environment variables. You can find client ID and tenant ID information in the **Overview** section of the Service Principal UI.

5. Create the environment files.
   a. Create the `azureauth.env` file in the following location:

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

      i. Add the following content to the file:

      clientId=<> clientSecret=<> subscriptionId=<> tenantId=<>

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

   b. Create the `credentials.env` file in the following location:

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

i. Add the following content to the file:

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

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

6. Add the absolute file paths to the `.env` file.

Enter the absolute path for the `azureauth.env` environment file in the `.env` file that corresponds to the `AZURE_RM_CREDS` environment variable.

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

Enter the absolute path for the `credentials.env` environment file in the `.env` file that corresponds to the `BLUEXP_TF_AZURE_CREDS` environment variable.

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

Step 4: Sign up for NetApp Intelligent Services

Sign up for NetApp Intelligent Services through your cloud provider to pay hourly (PAYGO) or through an annual contract. NetApp Intelligent Services include NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience and NetApp Disaster Recovery. NetApp Data Classification is included with your subscription at no additional cost

Steps

1. From the Azure portal, navigate to **SaaS** and select **Subscribe to NetApp Intelligent Services**.
2. Select the **Cloud Manager (by Cap PYGO by Hour, WORM and data services)** plan.

   You can either use the same resource group as Cloud Volumes ONTAP or a different resource group.

3. Configure the Console portal to import the SaaS subscription to the Console.

   You can configure this directly from the Azure portal by navigating to **Product and plan details** and selecting the **Configure account now** option.

   You will then be redirected to the the Console portal to confirm the configuration.

4. Confirm the configuration in the the Console portal by selecting **Save**.

Step 5: Create an external volume

You should create an external volume to keep the Terraform state files, and other important files persistent. You must make sure that the files are available for Terraform to run the workflow and deployments.

Steps

1. Create an external volume outside of Docker Compose:

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

Example:

```
docker volume create cvo_azure_volume_dst
```

2. Use one of the following options:
   a. Add an external volume path to the `.env` environment file.

      You must follow the exact format shown below.

      Format:

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

      Example:
      `PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure`

   b. Add NFS shares as an external volume.

      Make sure that the Docker container can communicate with the NFS shares and that the correct permissions, such as read/write, are configured.

      i. Add the NFS shares path as the path to the external volume in the Docker Compose file, as shown below:
      Format:

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

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

3. Navigate to the `cvo_azure_variables` folder.

   You should see the following variable files in the folder:

   `terraform.tfvars`

   `variables.tf`

4. Change the values inside the `terraform.tfvars` file according to your requirements.

   You must read the specific supporting documentation when modifying any of the variable values in the `terraform.tfvars` file. The values can vary depending on region, availability zones and other factors supported by Cloud Volumes ONTAP for Azure. This includes licenses, disk size, and VM size for single nodes and high availability (HA) pairs.

   All supporting variables for the Console agent and Cloud Volumes ONTAP Terraform modules are already defined in the `variables.tf` file. You must refer to the variable names in the `variables.tf` file before adding to the `terraform.tfvars` file.

5. Depending on your requirements, you can enable or disable FlexCache and FlexClone by setting the following options to `true` or `false`.

The following examples enable FlexCache and FlexClone:

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

6. If necessary, you can retrieve the value for the Terraform `az_service_principal_object_id` variable from the Azure Active Directory Service:
   a. Navigate to **Enterprise Applications –> All Applications** and select the name of the Service Principal you created earlier.
   b. Copy the object ID and insert the value for the Terraform variable:

      `az_service_principal_object_id`

Step 6: Deploy Cloud Volumes ONTAP for Azure

Use the following steps to deploy Cloud Volumes ONTAP for Azure.

Steps

1. From the root folder, run the following command to trigger deployment:

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

   Two containers are triggered, the first container deploys Cloud Volumes ONTAP and the second container sends telemetry data to AutoSupport.

   The second container waits until the first container completes all of the steps successfully.

2. Monitor progress of the deployment process using the log files:

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

   This command provides output in real time and captures the data in the following log files:

   `deployment.log`

   `telemetry_asup.log`

   You can change the name of these log files by editing the `.env` file using the following environment variables:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   The following examples show how to change the log file names:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

After you finish

You can use the following steps to remove the temporary environment and clean up items that were created during the deployment process.

Steps

1. If you deployed FlexCache, set the following option in the `terraform.tfvars` file, this cleans up FlexCache volumes and removes the temporary environment that was created earlier.

   ```
   flexcache_operation = "destroy"
   ```

   The possible options are `deploy` and `destroy`.

2. If you deployed FlexClone, set the following option in the `terraform.tfvars` file, this cleans up FlexClone volumes and removes the temporary environment that was created earlier.

   ```
   flexclone_operation = "destroy"
   ```

   The possible options are `deploy` and `destroy`.

Copyright information

Trademark information