APEX File Storage for AWS

Deployment Guide with Terraform

December 2023

H19875

Deployment Guide

Abstract

This document provides instructions to deploy OneFS clusters in AWS with the Terraform module onefs. The onefs module provides an auto-deployment method to deploy AWS resources for OneFS clusters.

DELL Technologies

Copyright

Contents

Executive summary

Overview

This deployment guide provides instructions for deploying PowerScale OneFS clusters for APEX File Storage for AWS with the onefs module. It also provides guidance on post-deployment, including expanding a cluster. After you have deployed a cluster, it is recommended to refer to the APEX File Storage for AWS Getting Started Guide on the Dell Technologies Support website for more details about OneFS.

Revisions

Date	Part number/ revision	Description
December 2023	H19875	Initial release

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: Lieven Lin

Note: For links to other documentation for this topic, see PowerScale Info Hub.

Introduction

About this guide

This guide is intended as a supplement to the APEX File Storage for AWS Deployment Guide. Amazon Web Services (AWS) cloud and storage administrators can use this guide to deploy the AWS infrastructure resources automatically for deploying and configuring OneFS as an Elastic Compute Cloud (EC2) instance on AWS. To use this guide effectively, administrators should be familiar with the AWS cloud services, including EC2, Elastic Block Storage (EBS), VPC, and IAM. After you have deployed a cluster, it is recommended to refer to APEX File Storage for AWS Getting Started Guide on the Dell Technologies Support website for more details about OneFS.

The following conventions are adopted in this guide to represent the first and subsequent occurrences of frequently used terminology in a section:

Table 1. Terminology

First occurrence in the title and introductory paragraph	Usage thereafter
Dell Technologies PowerScale OneFS	OneFS
OneFS cluster	cluster
OneFS cluster node	node
OneFS cluster internal subnet	internal subnet
OneFS cluster external subnet	external subnet
OneFS cluster internal security group	internal security group
OneFS cluster external security group	external security group
OneFS cluster internal network interfaces	internal network interfaces
OneFS cluster external network interfaces	external network interfaces

Terraform and the onefs module

Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It allows users to define and provision infrastructure in a declarative configuration language. Terraform uses providers to interact with different infrastructure platforms, such as AWS, Azure, and Google Cloud. Each provider is responsible for translating the Terraform configuration into API calls for a specific platform.

onefs is an open-source Terraform module for the auto-deployment of AWS resources for a OneFS cluster. It is released and licensed under the MPL-2.0 license. You can find more details about the onefs module from the Terraform Registry.

The onefs module provides the following features to help you deploy APEX File Storage for AWS OneFS clusters in AWS:

- Provision necessary AWS resources for a single OneFS cluster, including EC2 instances, EBS volumes, placement group, and network interfaces.
- Expand cluster size by provisioning additional AWS resources, including EC2 instances, EBS volumes, and network interfaces.

Deployment overview

See the document APEX File Storage for AWS architecture for details about the architecture. APEX File Storage for AWS deployment consists of four high-level steps:

- **Plan your deployment:** APEX File Storage for AWS has a predefined supported configuration, including cluster size, cluster capacity, EC2 instance type, EBS volume type, and so on. You must know about the supported configuration before you start to deploy the cluster. For details, see the documentation Supported cluster configuration.
- **Prerequisites check list:** You must fulfill all the prerequisites before following this guide to deploy a cluster. For details, see the section Prerequisites.
- **Deploy AWS infrastructure:** After you have planned your deployment and fulfilled the prerequisites, you can use onefs module to deploy AWS resources. For details, see the section Deploy AWS resources.
- **Set up the cluster:** To add additional nodes to the cluster, connect to the first node of the cluster. For details, see the section Set up the cluster.

Plan your deployment

When setting up a new cluster, it is important to consider the type of storage configuration that is supported. Different configurations fulfill different requirements, depending on the intended use and the amount of data that needs to be stored. In general, a supported cluster configuration should be reliable, performant, and offer enough storage capacity to meet your needs.

Therefore, before starting to deploy a cluster, you should be familiar with the supported cluster configuration offerings of APEX File Storage for AWS. Based on your actual business needs and future growth, choose an SSD or HDD cluster configuration that meets your requirements.

Supported cluster configuration

For a single cluster of APEX File Storage for AWS, all nodes in the cluster must use the same configuration, including EC2 instance type and size, EBS volume type, and EBS volume size. For details, see the documentation Supported cluster configuration .

Planning cluster configuration details

After you have understood the supported cluster configurations and have analyzed your business requirements, you can start to plan your deployment details. Write down the configuration, including your OneFS cluster settings and AWS infrastructure information. The following table contains an example used in this guide for each item. For this example, we will deploy a 4-node SSD cluster using the m5dn.12xlarge instance type.

Table 2. Cluster configuration example

Configuration items	Example used in this guide	Note
Cluster name	vonefs-cfv	
EC2 instance type	m5dn.12xlarge	
EBS type	gp3	

Configuration items	Example used in this guide	Note
Cluster size	4	If you are deploying a 4-node or 5-node cluster, and you need to expand the cluster size in the future, you must ensure that you are using a correct combination of volume count per node and single volume size, which allows you to expand to a maximum of 6 nodes. See supported cluster configuration details for all supported combinations.
EBS volume counts per node	6	
Single EBS volume sizes	1TiB	
Cluster raw capacity	24TiB	Ensure that your cluster raw capacity is within the supported range. See Supported cluster configuration for the capacity limitation.

Prerequisites

This section describes the prerequisite details that you should fulfill before using the onefs module to deploy OneFS cluster in AWS.

AWS subscription

APEX File Storage for AWS requires an AWS subscription. Visit the Amazon website for more details about AWS subscription pricing.

VPC subnet

APEX File Storage for AWS requires two different subnets:

- A dedicated internal subnet for cluster internal network interfaces. The subnet cannot be shared with other EC2 instances.
- An external subnet for cluster external network interfaces.

Note: IPv6 is not currently supported.

IAM resources

Create IAM resources by following the manual deployment guide Create IAM policy, role, and instance profile. This is a one-time activity. The IAM policy and role is reusable for additional cluster deployment.

Security group

Create an internal security group for cluster internal network interfaces only, by following the manual deployment guide Create the internal security group.

Create an external security group for cluster external network interfaces to allow specific ingress traffic from clients, by following the manual deployment guide Create the external security group.

OneFS AMI ID

See the documentation Find the OneFS AMI ID for details about OneFS AMI ID.

A host machine

A host machine with Terraform, Git, and AWS CLI installed. Ensure that the AWS CLI is configured correctly to access your AWS account. This document uses a Windows machine cmd console as an example.

Note: Do not use PowerShell for running Terraform commands.

IP range of cluster internal network interfaces

A cluster requires a dedicated internal subnet for cluster internal network interfaces. The subnet cannot be shared with other EC2 instances. For a single cluster, you must have six contiguous IPs to ensure that the cluster can expand to a maximum of six nodes.

IP range of cluster external network interfaces

Default network pool

During cluster deployment, you create a default cluster network pool named groupnet0.subnet0.pool0. Each node in the cluster is assigned one IP address from this pool. The IP addresses used in the pool groupnet0.subnet0.pool0 are the AWS primary IPv4 addresses, and cannot be moved from one node to another. Thus, the allocation type of this pool cannot be changed to dynamic.

The Externally Managed IPs feature in OneFS 9.7 introduces a limited form of DHCP for managing IP allocation in groupnet0.subnet0.pool0 and ensures that an IP in the OneFS network pool is assigned to the correct network interface of a node as the primary IP. To ensure the integrity of this process and mitigate potential security risks *of a rogue DHCP server*, it is recommended to add an inbound rule in the cluster external security group in AWS. See Table 3 for the details of the rule:

Table 3. Inbound rule for DHCP

Setting	Value
Rule name	E.g. DHCP
Type	Ingress
From port	67
To port	68
Protocol	udp
Allowed CIDR blocks	<cluster-gateway>/32

Additional network pools

After a cluster is deployed, users are allowed to create additional network pools. These new pools can use static or dynamic allocation. Any unused IPs from the cluster external subnet CIDR range can be used to create pools. The IPs from additional network pools are assigned to cluster nodes as AWS secondary IPv4 addresses.

Deploy AWS resources

Quick start

After you have fulfilled the Prerequisites, you can start to deploy AWS resources for a OneFS cluster.

This section provides instructions for deploying the required AWS infrastructure resources for APEX File Storage for AWS with Terraform, including EC2 instances, spread strategy placement group, network interfaces, and EBS volumes.

1. Get the latest version of the onefs module from the Terraform Registry. See Figure 1 for an example.

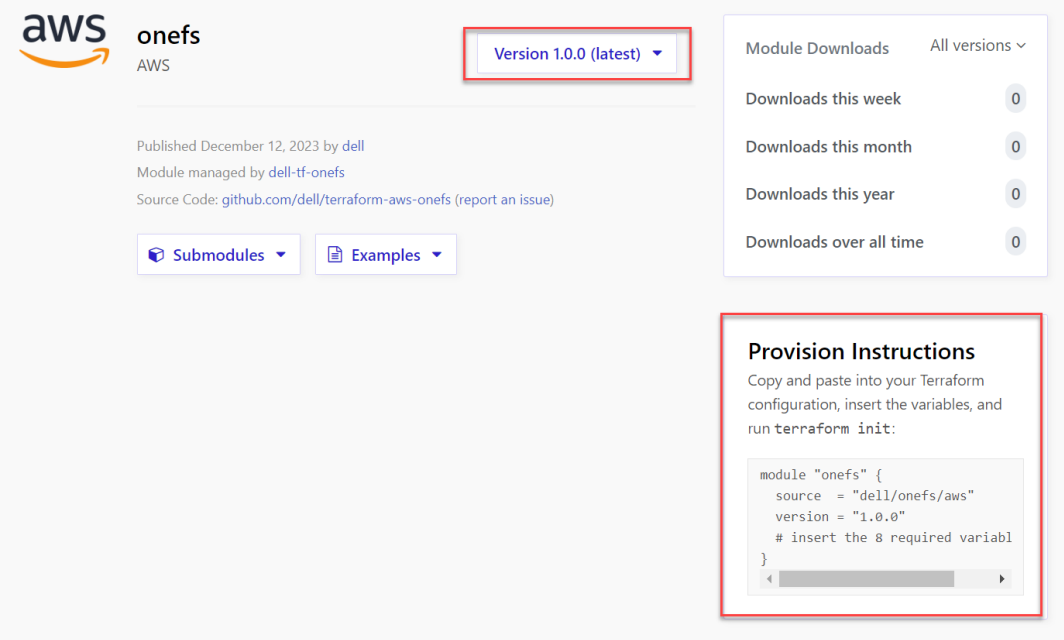

Figure 1. onefs module

2. Prepare a **main.tf** file that uses the onefs module version collected in Step 1. The onefs module requires a set of input variables. See the section Input variables for all available input variables for the onefs module. The following is an example file named **main.tf** for creating the OneFS cluster planned in Table 2. See the section Input variables for generating a hashed password.

```
module "onefs" {
  source  = "dell/onefs/aws"
  version = "1.0.0"

  region = "us-east-1"
  availability_zone = "us-east-1a"
  iam_instance_profile = "onefs-runtime-instance-profile"
  name = "vonefs-cfv"
  id = "vonefs-cfv"
  nodes = 4
  instance_type = "m5dn.12xlarge"
  data_disk_type = "gp3"
  data_disk_size = 1024
  data_disks_per_node = 6
```

```
 internal_subnet_id = "subnet-0c0106598b95ee7b6"
 external_subnet_id = "subnet-0837801239d54e245"
 contiguous_ips= true
 first_external_node_hostnum = 5
 internal_sg_id = "sg-0ee87249a52397219"
 security_group_external_id = "sg-0635f298c9cb764da"
 image_id = "ami-0f1a267119a34361c"
 credentials_hashed = true
 hashed_root_passphrase =
"$5$9874f5d2c724b8ca$IFZZ5e9yfUVqNKVL82s.iFLIktr4WLavFhUVa8A"
 hashed_admin_passphrase =
"$5$9874f5d2c724b8ca$IFZZ5e9yfUVqNKVL82s.iFLIktr4WLavFhUVa8A"
 dns_servers = ["169.254.169.253"]
 timezone = "Greenwich Mean Time"
}

output "onefs-outputs" {
  value = module.onefs
  sensitive = true
}
```

3. Change your current working directory to the **main.tf** directory.
4. Initialize the module's root directory by installing the required providers and modules for the deployment. In the following example, the onefs module is downloaded automatically from the Terraform Registry.

```
# terraform init

Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/dell/onefs/aws 1.0.0 for
onefs...
- onefs in .terraform\modules\onefs
- onefs.onefsbase in .terraform\modules\onefs\modules\base
- onefs.onefsbase.machineid in
.terraform\modules\onefs\modules\machineid

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.30.0...
- Installed hashicorp/aws v5.30.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to
record the provider
selections it made above. Include this file in your version
control repository
so that Terraform can guarantee to make the same selections
by default when
you run "terraform init" in the future.
```

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running
"terraform plan" to see
any changes that are required for your infrastructure. All
Terraform commands
should now work.

If you ever set or change modules or backend configuration
for Terraform,
rerun this command to reinitialize your working directory. If
you forget, other
commands will detect it and remind you to do so if necessary.
```

5. Verify the configuration files in the onefs directory.

   ```
   # terraform validate
   ```

6. Apply the configurations by running the following command.

   ```
   # terraform apply
   ```

7. Enter “yes” after you have previewed and confirmed the changes.

   ```
   Do you want to perform these actions?
     Terraform will perform the actions described above.
     Only 'yes' will be accepted to approve.

     Enter a value: yes
   ```

8. Wait for the AWS resources to be provisioned. The output displays all the cluster information. If the deployment fails, re-run the `terraform apply` command to deploy.

   ```
   Apply complete! Resources: 13 added, 0 changed, 0 destroyed.

   Outputs:

   onefs-outputs = <sensitive>
   ```

9. Get the cluster details information by running the following command.

   ```
   # terraform output --json
   ```

   The following example output is truncated.

   ```
   additional_nodes = 3
   cluster_id = "vonefs-cfv"
   control_ip_address = "10.0.32.5"
   external_ip_addresses = [
     "10.0.32.5",
     "10.0.32.6",
     "10.0.32.7",
     "10.0.32.8",
   ```

```
]
gateway_hostnum = 1
instance_id = [
  "i-0eead1ee1dd67da6e",
  "i-054efe96f6e605009",
  "i-06e0b1ce06bad42a1",
  "i-0e463c742974641d7",
]
internal_ip_addresses = [
  "10.0.16.5",
  "10.0.16.6",
  "10.0.16.7",
  "10.0.16.8",
]
internal_network_high_ip = "10.0.16.8"
internal_network_low_ip = "10.0.16.5"
mgmt_ip_addresses = []
node_configs = {
  "0" = {
    "external_interface_id" = "eni-09ddea1fd79f0d0ab"
    "external_ips" = [
      "10.0.32.5",
    ]
    "internal_interface_id" = "eni-0caeee71581a8c429"
    "internal_ips" = [
      "10.0.16.5",
    ]
    "mgmt_interface_id" = null
    "mgmt_ips" = null /* tuple */
    "serial_number" = "SV200-930073-0000"
  }
  "1" = {
    "external_interface_id" = "eni-00869c96a27c20c93"
    "external_ips" = [
      "10.0.32.6",
    ]
    "internal_interface_id" = "eni-0471bbba5a7f6596d"
    "internal_ips" = [
      "10.0.16.6",
    ]
    "mgmt_interface_id" = null
    "mgmt_ips" = null /* tuple */
    "serial_number" = "SV200-930073-0001"
  }
  "2" = {
    "external_interface_id" = "eni-0dac5052668bd3a4f"
    "external_ips" = [
      "10.0.32.7",
    ]
    "internal_interface_id" = "eni-09d35ffa61b3dcd60"
```

```
    "internal_ips" = [
      "10.0.16.7",
    ]
    "mgmt_interface_id" = null
    "mgmt_ips" = null /* tuple */
    "serial_number" = "SV200-930073-0002"
  }
  "3" = {
    "external_interface_id" = "eni-028d211ef2d5b577c"
    "external_ips" = [
      "10.0.32.8",
    ]
    "internal_interface_id" = "eni-02a99febea713f2d1"
    "internal_ips" = [
      "10.0.16.8",
    ]
    "mgmt_interface_id" = null
    "mgmt_ips" = null /* tuple */
    "serial_number" = "SV200-930073-0003"
  }
}
region = "us-east-1"
```

10. Write down the following output variables for setting up a cluster in the section Set up the cluster.
    - **control_ip_address**: The external IP address of the cluster's first node
    - **external_ip_addresses**: The external IP addresses of all provisioned cluster nodes
    - **internal_ip_addresses**: The internal IP addresses of all provisioned cluster nodes
    - **internal_network_high_ip**: The highest internal IP address assigned
    - **internal_network_low_ip**: The lowest internal IP address assigned
    - **instance_id**: The EC2 instance IDs of the cluster nodes

All AWS resources are now provisioned. After the cluster's first node starts, it will form a single node cluster. We use the cluster's first node to add additional nodes to the cluster.

Input variables

Terraform input variables let you customize aspects of Terraform modules without altering the module's own source code. The onefs module supports the input variables listed in Table 4.

Table 4. Onefs input variables

Variable Name	Type	Description
region	string	(Required) The AWS region of OneFS cluster nodes.
availability_zone	string	(Required) The AWS availability zone of OneFS cluster nodes.

Variable Name	Type	Description
iam_instance_profile	string	(Required) The AWS instance profile name of OneFS cluster nodes. For more details, see the AWS documentation Instance profiles.
name	string	(Required) The OneFS cluster name. Cluster names must begin with a letter and can contain only numbers, letters, and hyphens. If the cluster is joined to an Active Directory domain, the cluster name must be 11 characters or fewer.
id	string	(Required) The ID of the OneFS cluster. The onefs module uses the ID to add tags to the AWS resources. It is recommended to set the ID to your cluster name.
nodes	number	(Required) The number of OneFS cluster nodes: it should be 4, 5, or 6.
instance_type	string	(Required) The EC2 instance type of OneFS cluster nodes. All nodes in a cluster must have the same instance size. The supported instance sizes are: • EC2 m5dn instances: m5dn.8xlarge, m5dn.12xlarge, m5dn.16xlarge, m5dn.24xlarge • EC2 m6idn instances: m6idn.8xlarge, m6idn.12xlarge, m6idn.16xlarge, m6idn.24xlarge • EC2 m5d instances: m5d.24xlarge • EC2 i3en instances: i3en.12xlarge **Note**: You must run PoC if you intend to use m5d.24xlarge or i3en.12xlarge EC2 instance types. For details, contact your Dell account team.
data_disk_type	string	(Required) The EBS volume type for the cluster, gp3 or st1.
data_disk_size	number	(Required) The single EBS volume size in GiB. Consider the Supported cluster configuration, it should be 1024 to 16384 for gp3, 4096 or 10240 for st1.
data_disks_per_node	number	(Required) The number of EBS volumes per node. Consider the Supported cluster configuration, it should be 5, 6, 10, 12, 15, 18, or 20 for gp3, 5 or 6 for st1.
internal_subnet_id	string	(Required) The AWS subnet ID for the cluster internal network interfaces.
external_subnet_id	string	(Required) The AWS subnet ID for the cluster external network interfaces.
contiguous_ips	bool	(Required) A boolean flag to indicate whether to allocate contiguous IPv4 addresses to the cluster nodes' external network interfaces. It is recommended to set to true.

Variable Name	Type	Description
first_external_node_hostnum	number	(Required if contiguous_ips=true) The host number of the first node's external IP address in the given AWS subnet. Default is set to 5, The first four IP addresses in an AWS subnet are reserved by AWS, so the onefs module will allocate the fifth IP address to the cluster's first node. If the IP is in use, the module will fail. Therefore, when setting contiguous_ips=true, ensure that you set a correct host number that has sufficient contiguous IPs for your cluster. Refer to Terraform cidrhost Function for more details about host number.
internal_sg_id	string	(Required) The AWS security group ID for the cluster internal network interfaces.
security_group_external_id	string	(Required) The AWS security group ID for the cluster external network interfaces.
image_id	string	(Required) The OneFS AMI ID described in Find the OneFS AMI ID.
credentials_hashed	bool	(Required) A boolean flag to indicate whether the credentials are hashed or in plain text.
hashed_root_passphrase	string	(Required if credentials_hashed=true) The hashed root password for the OneFS cluster
hashed_admin_passphrase	string	(Required if credentials_hashed=true) The hashed admin password for the OneFS cluster
root_password	string	(Required if credentials_hashed=false) The root password for the OneFS cluster
admin_password	string	(Required if credentials_hashed=false) The admin password for the OneFS cluster
dns_servers	list(string)	(Optional) The cluster DNS server, default is set to ["169.254.169.253"], which is the AWS Route 53 Resolver. For details, see Amazon DNS server.
dns_domains	list(string)	(Optional) The cluster DNS domain, default is set to ["*<region>*.compute.internal"]
timezone	string	(Optional) The cluster time zone, default is set to "Greenwich Mean Time". Several available options are: Greenwich Mean Time, Eastern Time Zone, Central Time Zone, Mountain Time Zone, Pacific Time Zone. You can change the time zone after the cluster is deployed by following the steps in the section OneFS documentation – Set the cluster date and time.
resource_tags	map(string)	(Optional) The tags that will be attached to provisioned AWS resources. For example, resource_tags={"project": "onefs-poc", "tester": "bob"}.

When creating the EC2 instances, the Terraform module passes user data to each node for forming the OneFS cluster. The user data can only be accessed from within the node through AWS Instance Meta Data Service (IMDS). However, anyone who has direct access to the instance, and potentially any software running on the instance, can view this

data. To protect sensitive data such as passwords, we recommend that cluster administrators grant ISI_PRIV_LOGIN_SSH to as few accounts as possible. To protect passwords, we also recommend using one of the following methods:

- Change the password when the cluster deployment is complete. See the OneFS documentation for resetting a user password.
- Use a hashed password generated using openssl (sha256) and set the input variable "credentials_hashed" to true. The following example makes a sha256 hashed value for plaintext. Replace <password> to your settings.

  ```
  # openssl passwd -5 -salt `head -c 8 /dev/random | xxd -p` <password>
  ```

Set up the cluster

Now that you have completed the AWS infrastructure deployment, you need to access the command line of the cluster's first node to complete the cluster formation. This can be accomplished in different ways, including:

- Use the AWS EC2 serial console of the first node
- SSH to the first node with its external network interface IP

This guide uses the AWS EC2 serial console of the first node to set up the cluster. For more details, see the AWS documentation Connect to the EC2 Serial Console. After you connect to the cluster's first node, perform the following steps to complete the cluster setup.

Note: Cluster nodes can take several minutes to start up for the first time. If you get an error when connecting to and authenticating with OneFS, wait a few more minutes and then try again.

---

1. When you can connect to the cluster's first node, log in to the node as root.
2. When you are connected, check the cluster health by running the following command until the cluster and node health report a status of OK. Note that the command may fail in various ways as the cluster is booting. If this occurs, wait several minutes, and then try again.

   ```
   # isi status
   ```

```
vonefs-cfv-1# isi stat -q
Cluster Name: vonefs-cfv
Cluster Health:     [  OK ]
Data Reduction:     1.00 : 1
Storage Efficiency: 1.00 : 1
Cluster Storage:  HDD                 SSD Storage
Size:             0 (0 Raw)           5.9T (5.9T Raw)
VHS Size:         0
Used:             0 (n/a)             444.0M (< 1%)
Avail:            0 (n/a)             5.9T (> 99%)

                  Health Ext  Throughput (bps)  HDD Storage      SSD Storage
ID |IP Address     |DASR |C/N|  In   Out  Total| Used / Size     |Used / Size
---+---------------+-----+---+-----+-----+-----+-----------------+-----------------
  1|10.0.32.5      | OK  | C |448.0| 751k| 751k|(No Storage HDDs)| 444M/ 5.9T(< 1%)
---+---------------+-----+---+-----+-----+-----+-----------------+-----------------
Cluster Totals:              |448.0| 751k| 751k|(No Storage HDDs)| 444M/ 5.9T(< 1%)

    Health Fields: D = Down, A = Attention, S = Smartfailed, R = Read-Only
          External Network Fields: C = Connected, N = Not Connected
```

Figure 2. First node status

3. Check that the authorization system is up by running the following command. Note that the command may fail in various ways as the cluster is booting. If this occurs, wait several minutes, and try again.

   ```
   # isi auth user list
   ```

```
vonefs-cfv-1# isi auth user list
Name
-------------
Guest
root
admin
compadmin
remotesupport
ese
ftp
insightiq
www
ntpd
_ypldap
tests
_lldpd
nobody
git_daemon
isdmgmt
-------------
Total: 16
```

Figure 3. Authentication service status

4. If you forget to write down the output of the module, you can get the output again by specifying the state file. The following is an example for this guide.

   ```
   # terraform output -state=vonefs-cfv.tfstate
   ```

5. Modify the cluster internal network IP range to the **internal_ip_addresses** output variable mentioned in Step 10 in the section Deploy AWS resources

6. Quick start.

```
# isi cluster internal-networks modify --int-a-ip-addresses
<internal_network_low_ip>-<internal_network_high_ip>
```

The following is an example for this guide.

```
# isi cluster internal-networks modify --int-a-ip-addresses
10.0.16.5-10.0.16.8
```

7. Add additional nodes' external IP addresses **external_ip_addresses** to the default OneFS IP pool groupnet0.subnet0.pool0. If you do not use contiguous IPs for cluster nodes' external network interfaces, you have to add all the IPs one by one.

   The following is an example for this guide.

   ```
   # isi network pools modify groupnet0.subnet0.pool0 --add-
   ranges 10.0.32.5-10.0.32.8 --force
   ```

8. Wait for additional nodes to boot. Check that the additional nodes are ready to join by running the following command until they appear as available.

   ```
   # isi devices node list
   ```

```
vonefs-cfv-1# isi devices node list
Serial Number     Product                                                    Version      Status
-------------------------------------------------------------------------------------------------
SV200-930073-0001 AWS-m5dn.12xlarge-Cloud-Single-192GB-1x1GE-6144GB SSD B_MAIN_3854R available
SV200-930073-0002 AWS-m5dn.12xlarge-Cloud-Single-192GB-1x1GE-6144GB SSD B_MAIN_3854R available
SV200-930073-0003 AWS-m5dn.12xlarge-Cloud-Single-192GB-1x1GE-6144GB SSD B_MAIN_3854R available
-------------------------------------------------------------------------------------------------
Total: 3
```

Figure 4. Additional nodes status

9. Add the additional nodes into the cluster one by one, starting from node 2 whose serial number is SV200-930073-0001. Run the following command to add the second node:

   ```
   # isi devices node add SV200-930073-0001 --async
   ```

   Check the cluster status by running the following command and wait for the cluster and node health to report a status of OK before you proceed.

   ```
   # isi status
   ```

10. Run the following command to add the third node:

    ```
    # isi devices node add SV200-930073-0002 --async
    ```

    Check the cluster status by running the following command and wait for the cluster and node health to report a status of OK before proceeding.

    ```
    # isi status
    ```

11. Run the following command to add the fourth node:

    ```
    # isi devices node add SV200-930073-0003 --async
    ```

    Check the cluster status by running the following command and wait for the cluster and node health to report a status of OK before proceeding.

    ```
    # isi status
    ```

12. Then wait for the node to join by watching the cluster and node status. Wait for the nodes to appear in the output and report a status of OK, and the cluster to return a status of OK.

    ```
    # isi status
    ```

```
vonefs-cfv-1# isi stat -q
Cluster Name: vonefs-cfv
Cluster Health:     [  OK ]
Data Reduction:     1.02 : 1
Storage Efficiency: 0.25 : 1
Cluster Storage:  HDD                 SSD Storage
Size:             0 (0 Raw)           21.6T (23.8T Raw)
VHS Size:         2.2T
Used:             0 (n/a)             1.8G (< 1%)
Avail:            0 (n/a)             21.6T (> 99%)

                  Health Ext  Throughput (bps)  HDD Storage      SSD Storage
ID |IP Address     |DASR |C/N|  In   Out  Total| Used / Size     |Used / Size
---+---------------+-----+---+-----+-----+-----+----------------+----------------
  1|10.0.32.5      | OK  | C |80.6k|39.7k| 120k|(No Storage HDDs)| 482M/ 5.4T(< 1%)
  2|10.0.32.6      | OK  | C |    0|25.6k|25.6k|(No Storage HDDs)| 459M/ 5.4T(< 1%)
  3|10.0.32.7      | OK  | C |    0|16.8k|16.8k|(No Storage HDDs)| 471M/ 5.4T(< 1%)
  4|10.0.32.8      | OK  | C |    0|    0|    0|(No Storage HDDs)| 471M/ 5.4T(< 1%)
---+---------------+-----+---+-----+-----+-----+----------------+----------------
Cluster Totals:              |80.6k|82.1k| 163k|(No Storage HDDs)| 1.8G/21.6T(< 1%)

     Health Fields: D = Down, A = Attention, S = Smartfailed, R = Read-Only
          External Network Fields: C = Connected, N = Not Connected
```

Figure 5. Cluster status

You have now completed all steps to deploy an APEX File Storage for AWS cluster using Terraform. It is highly recommended to refer to the **APEX File Storage for AWS Getting Started Guide** on the Dell Technologies Support website for more details about OneFS.

Expand a OneFS cluster

You can scale-out an existing cluster at any time to meet your business needs. This section shows how to add a node to an existing cluster. We will use the 4-node cluster we created in the section Set up the cluster as an example.

1. Navigate to the directory that contains the **main.tf** file.
2. Open the **main.tf** file, then modify the **nodes** input variable to a desired number. In this example, we are changing the number of nodes from 4 to 5.
3. Apply the new configurations again. The following is an example for this guide.

   ```
   # terraform apply
   ```

4. Enter “yes” after you have previewed and confirmed the changes.

   ```
   Do you want to perform these actions?
     Terraform will perform the actions described above.
     Only 'yes' will be accepted to approve.

     Enter a value: yes
   ```

5. Wait for the AWS resources to be provisioned. The output will display all the cluster information.

```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

onefs-outputs = <sensitive>
```

6. Get the cluster details information by running the following command.

```
# terraform output --json
```

The following example output is truncated.

```
additional_nodes = 4
cluster_id = "vonefs"
control_ip_address = "10.0.32.5"
external_ip_addresses = [
  "10.0.32.5",
  "10.0.32.6",
  "10.0.32.7",
  "10.0.32.8",
  "10.0.32.9",
]
gateway_hostnum = 1
instance_id = [
  "i-0f4a7d403f303faa7",
  "i-0ff1380791ff7b17f",
  "i-015195c2ad7de84ea",
  "i-0341b2a4894f14607",
  "i-0afe679289220dc67",
]
internal_ip_addresses = [
  "10.0.16.5",
  "10.0.16.6",
  "10.0.16.7",
  "10.0.16.8",
  "10.0.16.9",
]
internal_network_high_ip = "10.0.16.9"
internal_network_low_ip = "10.0.16.5"
mgmt_ip_addresses = []
node_configs = {
  "0" = {
    "external_interface_id" = "eni-0a26e3ed7bd215b9f"
    "external_ips" = [
      "10.0.32.5",
    ]
    "internal_interface_id" = "eni-03d2e42a6ccaa3a1c"
    "internal_ips" = [
      "10.0.16.5",
    ]
    "mgmt_interface_id" = null
```

```
  "mgmt_ips" = null /* tuple */
  "serial_number" = "SV200-930073-0000"
}
"1" = {
  "external_interface_id" = "eni-0fc2debd6aa39cd7b"
  "external_ips" = [
    "10.0.32.6",
  ]
  "internal_interface_id" = "eni-0c44ca2d7be2f6f15"
  "internal_ips" = [
    "10.0.16.6",
  ]
  "mgmt_interface_id" = null
  "mgmt_ips" = null /* tuple */
  "serial_number" = "SV200-930073-0001"
}
"2" = {
  "external_interface_id" = "eni-0f39640f8e6718279"
  "external_ips" = [
    "10.0.32.7",
  ]
  "internal_interface_id" = "eni-001893c923b7f78bf"
  "internal_ips" = [
    "10.0.16.7",
  ]
  "mgmt_interface_id" = null
  "mgmt_ips" = null /* tuple */
  "serial_number" = "SV200-930073-0002"
}
"3" = {
  "external_interface_id" = "eni-0ff1dce6cf7a77c8f"
  "external_ips" = [
    "10.0.32.8",
  ]
  "internal_interface_id" = "eni-00f0e0035bb4e9823"
  "internal_ips" = [
    "10.0.16.8",
  ]
  "mgmt_interface_id" = null
  "mgmt_ips" = null /* tuple */
  "serial_number" = "SV200-930073-0003"
}
"4" = {
  "external_interface_id" = "eni-033d18f7950b05a4d"
  "external_ips" = [
    "10.0.32.9",
  ]
  "internal_interface_id" = "eni-041dfb09c27d3fd71"
  "internal_ips" = [
    "10.0.16.9",
```

```
    ]
    "mgmt_interface_id" = null
    "mgmt_ips" = null /* tuple */
    "serial_number" = "SV200-930073-0004"
  }
}
region = "us-east-1"
```

7. Write down the following output variables for adding the new node into the cluster later.
   - **control_ip_address**: The external IP address of the cluster's first node
   - **external_ip_addresses**: The external IP addresses of all provisioned cluster nodes
   - **internal_ip_addresses**: The internal IP addresses of all provisioned cluster nodes
   - **internal_network_high_ip**: The highest internal IP address assigned
   - **internal_network_low_ip**: The lowest internal IP address assigned
   - **instance_id**: The EC2 instance IDs of the cluster nodes
8. In the first node's CLI, modify the cluster internal network IP range to the **internal_ip_addresses** output variable mentioned in the previous step.

   ```
   # isi cluster internal-networks modify --int-a-ip-addresses
   <internal_network_low_ip>-<internal_network_high_ip>
   ```

   The following is an example for this guide.

   ```
   # isi cluster internal-networks modify --int-a-ip-addresses
   10.0.16.5-10.0.16.9
   ```

9. Add the new node's external IP addresses **external_ip_addresses** to the default OneFS IP pool groupnet0.subnet0.pool0.

   The following is an example for this guide.

   ```
   # isi network pools modify groupnet0.subnet0.pool0 --add-
   ranges 10.0.32.9-10.0.32.9 --force
   ```

10. Wait for the new node to boot. Check that the new node is ready to join by running the following command until it appears as available.

```
vonefs-cfv-1# isi devices node list
Serial Number     Product                                                    Version      Status
-------------------------------------------------------------------------------------------------
SV200-930073-0004 AWS-m5dn.12xlarge-Cloud-Single-192GB-1x1GE-6144GB SSD B_MAIN_3854R available
-------------------------------------------------------------------------------------------------
Total: 1
```

Figure 6. New node status

11. Add the new node to the cluster. Run the following command to add the fifth node:

    ```
    # isi devices node add SV200-930073-0004 –async
    ```

12. Then wait for the node to join by watching the cluster and node status. Wait for the node to appear in the output and report a status of OK, and the cluster to return to a status of OK.

    ```
    # isi status
    ```

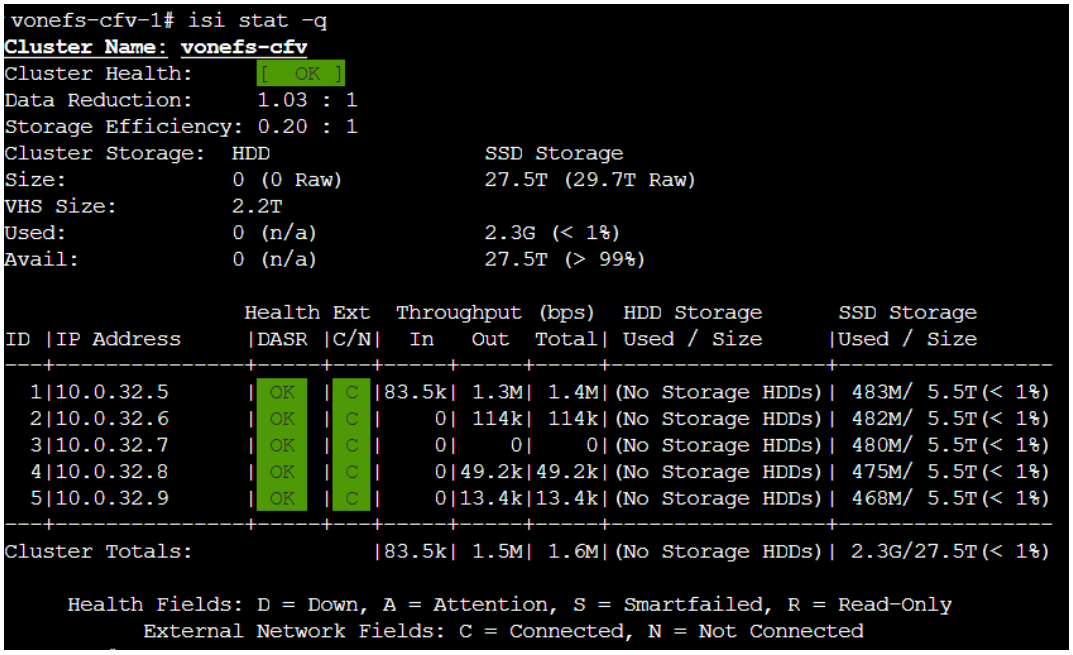

Figure 7. Expanded cluster status

You have now completed all steps to scale out an existing cluster of APEX file Storage for AWS with Terraform.

Destroy a OneFS cluster

To destroy a cluster provisioned with Terraform, do the following:

1. Navigate to the directory that contains the main.tf file of your cluster.
2. Run the `terraform destroy` command, to tear down all AWS resources created with Terraform:

   ```
   # terraform destroy
   ```

More cluster management

See the following resources to learn more about cluster management:

- Create an interface endpoint for OneFS
- OneFS SmartConnect
- Replace a volume for a cluster node
- Replace a cluster node
- Maintenance events from AWS
- Stopping OneFS node instances