Cohesity Cloud Scale Technology Deployment Guide Using Terraform for Microsoft Azure

Release 11.2

VERITAS
now part of Cohesity

Cohesity Cloud Scale Technology deployment guide using Terraform for Azure cloud

Last updated: 2026-05-28

Legal Notice

Cohesity Support

Reach Cohesity Support

There are several ways to create a Cohesity support case.

- Go to Cohesity Support, to search in our knowledge base; or contact us by phone - United States and Canada: 1-855-9CO-HESI (926-4374), option 2.
- Log in to the Cohesity Support Portal to create a new case.
- Click the (**?**) icon on the Cohesity UI and select Support Portal.

Support/Service Assistance

First, contact the Service Provider that you have contracted for service and support. If you work directly with Cohesity and have a product warranty/entitlement, repair pricing, or technical support-related question, see your options below:

- To find solutions to your product issues or for suggestions or best practices, visit the Cohesity Knowledge Base.
- Log in to the Cohesity Support Portal to create a new case.
- To monitor your open cases, log in to the portal and click the **Cases** tab on the home page. This page should have all the case statuses and updates. You can also view individual case status.

Cohesity Software Running on Partner Hardware

For Cohesity software running on qualified third-party hardware, the following support workflow applies:

1. The customer may contact Cohesity Support first if the issue cannot be determined as a hardware issue.

   Note: Cohesity cannot process hardware replacement requests for partner hardware.

2. Cohesity Support triages the issue. If it is a software issue, Cohesity Support continues to work on it.
3. If it is a hardware/firmware issue or is suspected to be a hardware/firmware issue, Cohesity provides information about the issue to the customer and requests that the customer open a support ticket with the appropriate partner.
4. If needed, Cohesity Support can join a three-way call with the partner and the customer.
5. The customer informs Cohesity Support on the progress of the partner's case.

Contents

Chapter 1 Introduction 6

About this guide 6
Required terminology 7
About Cloud Scale Technology on Azure cloud 7
About Terraform 8

Chapter 2 Getting started steps for deployment 10

Steps for getting started with deployment 10

Chapter 3 Prerequisites for setting up Azure environment 12

Before starting the deployment 12
Network configuration requirements 13
Azure subscription permission requirements 14
For setting up storage account for Cloud Scale deployment 16

Chapter 4 Prerequisities for Terraform scripts 18

Terraform Management Server requirements 18

Chapter 5 Deploying Cloud Scale Technology using Terraform scripts 20

Creating and configuring Terraform Management Server 20
Installing the packages for Terraform Management Server 21
About PreFlight checker (checklist) script 24
Stages of deploying Terraform scripts on Azure 25
Parameters for base stage 25
Parameters for addons stage 33
Parameters for deployment stage 33
PaaS based PostgreSQL deployment (DBaaS) on Azure 37
Installation instructions for deploying the Cloud Scale Technology on Azure 38
Change the PostgreSQL database server password 40

Chapter 6 Accessing the Cloud Scale environment 45

Accessing the Cloud Scale Technology environment after deployment 45

Chapter 7 Troubleshooting and cleanup environment steps 48

Troubleshooting issues 48
Cleanup steps 51

Chapter 1

Introduction

This chapter includes the following topics:

- About this guide
- About Cloud Scale Technology on Azure cloud
- About Terraform

About this guide

This document provides the instructions for deploying Cloud Scale Technology components in Azure Kubernetes Services (AKS) on Azure using Terraform. The intended audience for this document includes backup administrators, cloud administrators, architects, and system administrators. The purpose of this guide is to help understand the deployment of Cloud Scale Technology using Terraform scripts.

Cloud Scale Technology is a cloud native build your own form factor that uses cloud infrastructure components built on Kubernetes technology. To deploy this product, you will need the following expertise on your team in order to install and manage this environment:

- Kubernetes (also known as K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.
- Cloud native based deployments is a software approach of building, deploying, and managing modern applications in cloud computing environments. Knowledge about cloud networking, cloud commutating, and cloud storage are required to store, access, maintain, and manage data through a cloud computing provider.

Veritas also supports traditional virtual machine (VM) based IaaS deployments for Alta Data Protection. If you need further assistance on determining the best fit for your environment or have any additional questions, reach out to your local Sales team.

Required terminology

The table describes the important terms used in this guide for deploying Veritas Cloud Scale Technology on Azure.

Table 1-1 Important terms

Term	Description
Azure Virtual Network	Azure Virtual Network provides secure, private networking for your Azure and on-premises resources.
DNS	DNS translates domain names to IP addresses so browsers can load internet resources.
ACR	Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments.
AKS cluster	Azure Kubernetes Service (AKS) offers the quickest way to start developing and deploying cloud-native apps in Azure, datacenters, or at the edge with built-in code-to-cloud pipelines and guardrails.

About Cloud Scale Technology on Azure cloud

Cloud Scale Technology redefines data management for the next decade. Cloud Scale Technology's service elasticity and modern web-scale technologies enable NetBackup to operate cloud-natively within a cloud yet deliver a consistent experience across multiple clouds to improve cloud return on investment (ROI), service resiliency, and security while reducing operational complexity and costs.

The solution facilitates an orchestrated deployment of the following components on Kubernetes clusters:

- NetBackup: You can deploy NetBackup on the Kubernetes clusters of AWs or Azure for scaling the capacity of the NetBackup host to server large number of requests concurrently running on the NetBackup primary server at its peak performance capacity.
- MSDP Scaleout: In addition to the NetBackup components namely primary and media servers, the deduplication engine (1 to 16) replicas may also be deployed.
- NetBackup Snapshot Manager: You can deploy NetBackup Snapshot Manager with autoscaling capabilities for data movement.

Cloud Scale Technology is a new generation of the proven NetBackup architecture. This technology is designed to operate cloud-natively and use technologies such

as containers and microservices along with web-scale IT techniques such as service elasticity and hyper-automation. Some of the benefits of this technology are:

- A containerized, Kubernetes-based deployment model that can be used to create a new cloud-native NetBackup environment or complement an existing one that spans the data center and the cloud.
- A microservices-based architecture that provides the portability to work within multiple clouds and resiliency for service availability.
- Elastic services which autonomously grow and shrink as needed to optimize cloud resource usage and costs.
- API-driven microservices that enable cross-domain workflow automation.
- Simplified deployment directly from public cloud marketplaces and native tools.

About Terraform

Terraform is an open source "Infrastructure as Code" tool created by HashiCorp. It manages resources (such as cloud infrastructure, network appliances, Software as a Service, and Platform as a Service) with the providers.

Using Terraform, you can create and manage resources on cloud platforms and other services through their application programming interfaces (APIs). Service providers enable Terraform to work virtually with any platform or service with an accessible API.

Here are some advantages of Terraform:

- **Manage any infrastructure:** Terraform uses immutable approach which reduces the complexity of upgrading or modifying your services and infrastructure.
- **Tracks infrastructure status:** A state file keeps track of your environment and suggests changes to your infrastructure to match the configuration.
- **Standardize configurations:** Terraform supports reusable configuration components called modules that define configurable collections of infrastructure.

Terraform supports several cloud infrastructure providers such as Microsoft Azure, Amazon Web Services (AWS), Cloudflare, IBM Cloud, Google Cloud Platform, and Oracle Cloud Infrastructure.

The table describes you about the high-level steps involved in the deployment.

Table 1-2 Getting started using Terraform scripts for deploying Cloud Scale Technology on Azure

Steps
1. Ensure that the prerequisites for creating the Terraform Management Server are met.
2. Configure the Terraform Management Server.
3. Authentication with Azure
4. Execute the PreFlight checker script.
5a. Learn about the stages involved in the Terraform deployment. 5b. Installation instructions for deploying the Cloud Scale Technology.
6. Access Cloud Scale Technology UI after deployment.

Chapter 2

Getting started steps for deployment

This chapter includes the following topics:

- Steps for getting started with deployment

Steps for getting started with deployment

The topic helps you to understand the initial configuration for deploying the Cloud Scale Technology. The following table shows the steps involved in setting up the configuration.

Table 2-1 Getting started using Terraform scripts for deploying Cloud Scale Technology on Azure

Steps	Description
1. Ensure the prerequisites for creating Terraform Management Server are met.	Ensure that the Terraform Management Server prerequisites and networking requirements are met. Refer See "Terraform Management Server requirements" on page 18.
2. Configure Terraform Management Server	Refer See "Creating and configuring Terraform Management Server" on page 20. Refer See "Installing the packages for Terraform Management Server" on page 21.
3. Authentication with Azure	User / role which you will be using for deployment should have minimum permissions. Refer See "Azure subscription permission requirements" on page 14.

Table 2-1 Getting started using Terraform scripts for deploying Cloud Scale Technology on Azure *(continued)*

Steps	Description
4. Execute the PreFlight checker script.	This checklist is executed to verify the environment readiness before deploying theCloud Scale Technology. Refer to the section See "About PreFlight checker (checklist) script" on page 24.
5a. Learn about the stages involved in the Terraform deployment	See "Stages of deploying Terraform scripts on Azure" on page 25.
5b. Installation instructions for deploying the Cloud Scale Technology	See "Installation instructions for deploying the Cloud Scale Technology on Azure" on page 38.
6. Access Cloud Scale Technology UI after deployment	See "Accessing the Cloud Scale Technology environment after deployment" on page 45.

Chapter 3

Prerequisites for setting up Azure environment

This chapter includes the following topics:

- Before starting the deployment
- Network configuration requirements
- Azure subscription permission requirements
- For setting up storage account for Cloud Scale deployment

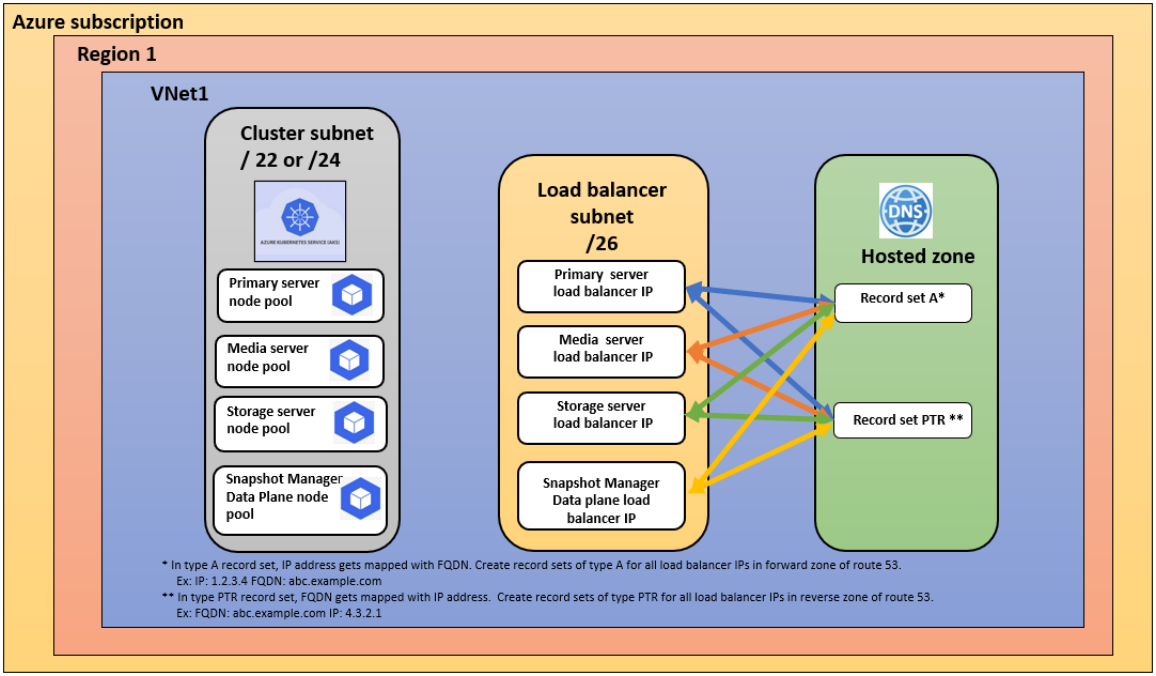
Before starting the deployment

To set up the Cloud Scale Technology deployment on the Azure environment, there are some prerequisites to be met.

- Ensure that the See "Network configuration requirements" on page 13. are met.
- Ensure that the See "Azure subscription permission requirements" on page 14. are assigned to the user before starting the deployment.

Network configuration requirements

Figure 3-1 Network configuration for managing Terraform Management Server in Azure



Ensure that the below networking requirements are met.

- VNet and subnets must be created in Azure account before the Terraform scripts are executed.
- Required address spaces:
  - **For cluster subnet:** This subnet is required with /22 or /24 subnet address space (used for node pool).
  - **Load balancer subnet:** This subnet is required with - /26 address space (This subnet needs to be empty with no virtual machines / devices installed).
- Create DNS entries in the Private Hosted Zone:
  - **Primary (1)**: primary.example.com (10.x.x.x)
  - **MSDP (1)**: msdp.example.com(10.x.x.x)
  - **Snapshot Manager (1)**: snapshotmanager.example.com(10.x.x.x)

- Outbound internet access is required from Terraform Management Server to communicate with resources, services, and the servers.
- While configuring the components or resources, avoid using prefixes like - netbackup, primary or media. The installation may fail if these keywords are used in the configuration.
- Azure reserves the first four addresses and the last address, for a total of five IP addresses within each subnet. Refer to the link for more details.
- AKS cluster nodes require internet access to install addons. Hence it is required to create route table under the resource group with subnets.
- Terraform server used to deploy Cloud Scale must be able to communicate with the cluster API server for your AKS server.

Azure subscription permission requirements

The permissions in Azure are required for the user to create clusters, deploy the Cloud Scale Technology on the Azure cloud environment, also to support backup and recovery operations. These are the minimum permission that will help user to setup the whole environment required to deploy the Cloud Scale Technology. There are two ways to assign these permission to the admin user which is used in the deployment and you will have to choose any **one** method.

- Use Azure subscription with contributor and user admin role.
- Create a custom role with following permissions attached to the user which is used for deploying the Cloud Scale Technology in Azure –

```
Microsoft.Compute/virtualMachineScaleSets/read
Microsoft.Compute/virtualMachineScaleSets/write
Microsoft.Compute/virtualMachineScaleSets/delete
Microsoft.Compute/virtualMachineScaleSets/delete/action
Microsoft.Compute/virtualMachineScaleSets/start/action
Microsoft.ContainerService/managedClusters/read
Microsoft.ContainerService/managedClusters/write
Microsoft.ContainerService/managedClusters/delete
Microsoft.ContainerService/managedClusters/start/action
Microsoft.ContainerService/managedClusters/stop/action
Microsoft.ContainerService/managedClusters/listClusterAdminCredential/action
Microsoft.ContainerService/managedClusters/listClusterUserCredential/action
Microsoft.ContainerService/managedClusters/listClusterMonitoringUserCredential/action
Microsoft.ContainerService/managedClusters/privateEndpointConnectionsApproval/action

Microsoft.ContainerService/managedClusters/runCommand/action
```

```
Microsoft.ContainerService/managedClusters/agentPools/read
Microsoft.ContainerService/managedClusters/agentPools/write
Microsoft.ContainerService/managedClusters/agentPools/delete
Microsoft.ContainerService/managedClusters/resolvePrivateLinkServiceId/action
Microsoft.ContainerService/managedClusters/agentPools/upgradeNodeImageVersion/write
Microsoft.ContainerService/managedClusters/extensionaddons/read
Microsoft.ContainerService/managedClusters/extensionaddons/write
Microsoft.ContainerService/managedClusters/privateEndpointConnections/read
Microsoft.ContainerService/managedClusters/privateEndpointConnections/write
Microsoft.ContainerService/managedClusters/privateEndpointConnections/delete

Microsoft.ContainerService/managedclustersnapshots/read
Microsoft.ContainerService/managedclustersnapshots/write
Microsoft.ContainerService/managedclustersnapshots/delete
Microsoft.Authorization/permissions/read
Microsoft.ContainerRegistry/registries/write
Microsoft.ContainerRegistry/registries/delete
Microsoft.ContainerRegistry/registries/read
Microsoft.ContainerRegistry/registries/listCredentials/action
Microsoft.ContainerRegistry/registries/operationStatuses/read
Microsoft.ContainerRegistry/registries/privateEndpointConnections/read
Microsoft.ContainerRegistry/registries/privateEndpointConnections/delete
Microsoft.ContainerRegistry/registries/privateEndpointConnections/write
Microsoft.ContainerRegistry/registries/PrivateEndpointConnectionsApproval/action
Microsoft.ContainerRegistry/registries/pull/read
Microsoft.ContainerRegistry/registries/push/write
Microsoft.Authorization/roleAssignments/read
Microsoft.Authorization/roleAssignments/write
Microsoft.Authorization/roleAssignments/delete
Microsoft.Authorization/roleDefinitions/read
Microsoft.Authorization/roleDefinitions/write
Microsoft.Authorization/roleDefinitions/delete
Microsoft.ManagedIdentity/userAssignedIdentities/assign/action
Microsoft.ManagedIdentity/userAssignedIdentities/delete
Microsoft.ManagedIdentity/userAssignedIdentities/read
Microsoft.ManagedIdentity/userAssignedIdentities/write
Microsoft.ManagedIdentity/userAssignedIdentities/listAssociatedResources/action
Microsoft.ManagedIdentity/identities/read
Microsoft.Network/privateDnsZones/write
Microsoft.Network/privateDnsZones/delete
Microsoft.Network/privateDnsZones/virtualNetworkLinks/write
Microsoft.Network/privateDnsZones/virtualNetworkLinks/delete
Microsoft.Network/privateDnsZones/join/action
```

```
Microsoft.Network/privateDnsZones/SOA/write
Microsoft.Network/privateLinkServices/privateEndpointConnections/write
Microsoft.Network/privateLinkServices/privateEndpointConnections/delete
Microsoft.Network/privateLinkServices/write
Microsoft.Network/privateLinkServices/delete
Microsoft.Network/privateEndpoints/privateDnsZoneGroups/write
Microsoft.Network/privateEndpoints/privateDnsZoneGroups/delete
Microsoft.Network/privateEndpoints/delete
Microsoft.Network/privateEndpoints/write
Microsoft.Network/*/read
Microsoft.Network/virtualNetworks/subnets/join/action
Microsoft.Network/virtualNetworks/join/action
Microsoft.Resources/subscriptions/resourcegroups/read
Microsoft.Resources/subscriptions/resourcegroups/write
```

For setting up storage account for Cloud Scale deployment

When using the existing storage account for Cloud Scale deployment, ensure that the following steps are performed before the execution of terraform scripts:

- Select **Azure Files** as the primary service.
- Select Performance as **Premium**.
- Select Redundancy as **LRS (Locally-Redundant Storage)**.
- To provide private access to storage account, integrate with private DNS zone (privatelink.core.windows.net).
- Virtual Network link must be created in **Virtual Network Links** section with VNet in `privatelink.core.windows.net` private DNS zone.
  Select Virtual Network where you are creating AKS Cluster. This is required for Cloud Scale pods to point NFS shares created in Storage Account.
- Ensure that you disable **Secure Transfer required**.

When using the existing storage account for Cloud Scale deployment, following parameters are required:

- use_existing_storage_acc
- storage_acc_id
- storage_acc_rg_name
- aks_private_dns_zone_id

For more information on the above parameters, refer to the following section:

See “Parameters for base stage ” on page 25.

Chapter 4

Prerequisities for Terraform scripts

This chapter includes the following topics:

- Terraform Management Server requirements

Terraform Management Server requirements

Terraform Management Server (also known as jump host) is required to execute the scripts. Ensure that the below server requirements are met before executing the scripts.

- Virtual machine with Linux operating system. The recommended configuration for the virtual machine is:
  - Ubuntu / RHEL
  - 2 CPUs
  - 8 GB memory
  - > = 64 GB free disk space in `/var` folder. The space is required to load the Docker images and copy the tar file on the `/var` folder.
- The following packages are required to be installed on the Terraform Management server. To install the below mentioned packages, refer to the section See “Installing the packages for Terraform Management Server” on page 21.
  - Terraform version >= 1.5.0 or later
  - Latest version of Docker
  - In case of RHEL operating system, use PODMAN.

  - kubectl (A command line tool for communicating with Kubernetes cluster's control plane. Refer the Azure documentation for more details.
  - Helm package manager
  - Azure CLI version >= 2.9.xx or later
  - BASH version >= 5.0.17 or later
  - Linux utilities like GREP, AWK, tr, PING, ENVSUBST, TAR, JQ, SED, and CUT
  - Ensure that you have enough space using the command: `~$ df -h`
- Outbound internet access is required to communicate with resources, services, and the servers.
- Copy the Veritas binary file bundle (NetBackup tar of Kubernetes.tar) file and Terraform script bundle from the Veritas Download center and copy on the Terraform Management Server which is also called as jump host. Unzip this file to access the scripts and files for deployment.

Chapter 5

Deploying Cloud Scale Technology using Terraform scripts

This chapter includes the following topics:

- Creating and configuring Terraform Management Server
- About PreFlight checker (checklist) script
- Stages of deploying Terraform scripts on Azure
- Installation instructions for deploying the Cloud Scale Technology on Azure

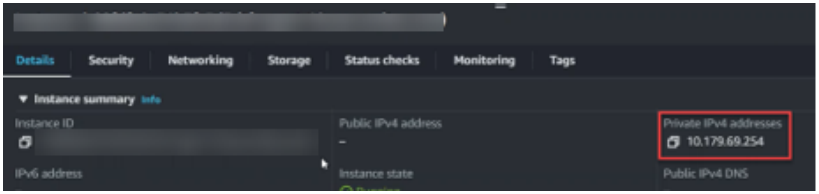
Creating and configuring Terraform Management Server

Terraform Management Server is a linux host which is required to execute terraform scripts. To deploy the Cloud Scale Technology, creating and configuring the Terraform Management Server is the first step.

The following steps describe how the Terraform Management Server is created and deployed in AWS / Azure environment.

1. Deploy an Ubuntu / RHEL version 22 virtual machine. Choose the appropriate instance type that matches these specifications:
   - 2 CPUs
   - 8 GB memory
   - >= 64 GB space on `/var` folder.

After the deployment is complete, note the IP address to connect.



2. Once the virtual machine is created, log in into the system using SSH client.

   ```
   ssh -i example.pem user@XXX.XXX.XXX.XXX
   ```

3. Ensure you have min 30 GB free space in `/var` folder. Use the below command to verify:

   ```
   ~$ df -h
   ```

4. If you are using non-root user, run the following command:

   ```
   sudo gpasswd -a "non root user" docker
   ```

   For example: `sudo gpasswd -a <user> docker`

   Restart the docker using the command:`sudo systemctl restart docker`

5. Install the listed packages from the section *Installing the packages for Terraform Management Server.*

6. Outbound internet access is required from Terraform Management Server to communicate with resources, services, and the servers.

Installing the packages for Terraform Management Server

This step is required to setup the Terraform Management Server as jump host. A jump host is an intermediary server which can be accessed beyond a firewall. It provides information needed to communicate with the target device. You can connect to the jump host a private key or username and password.

Installing packages on Terraform Management Server

1 Install Docker

Follow these steps to allow non-root user to access and leverage Docker.

a. Create and APT keyring directory using commands:

```
mkdir /etc/apt/keyrings
```

```
chmod 755 /etc/apt/keyrings
```

b. Download the `docker.gpg` file and place in the keyring folder:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
```

c. Download the Docker repository. Ensure that the below command is to be pasted as single shell. It only takes a second to run.

```
echo deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable
| tee /etc/apt/sources.list.d/docker.list
```

d. Install the Docker files using the next two commands one by one.

```
apt update
```

2
```
apt install -y docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

e. Confirm the Docker is installed correctly.

```
docker run hello-world
```

3 Install Terraform package using root user

a. Download GPG key and place in same keyrings directory created during Docker install (Step1a).

```
curl -sSL https://apt.releases.hashicorp.com/gpg | gpg --dearmor
-o /etc/apt/keyrings/hashicorp-archive-keyring.gpg
```

b. Download and install Terraform repository .Ensure that the below command is to be pasted as single shell.

```
echo deb
[signed-by=/etc/apt/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main | tee
/etc/apt/sources.list.d/hashicorp.list
```

c. Install Terraform 1.5.0 package using command:

```
apt update
```

```
apt install -y terraform=1.5.0
```

4 Install Kubectl using root user

a. Download the kubectl binary.

```
curl -LO https://dl.k8s.io/release/v1.25.0/bin/linux/amd64/kubectl
```

b. Install the kubectl binary into `/usr/local/bin`

```
install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

5 Install Helm package manager

a. Download the binary file:

```
curl -sSL https://get.helm.sh/helm-vx.xx.x-linux-amd64.tar.gz -o
helm-vx.xx.x-linux-amd64.tar.gz
```

Example: `curl -sSL https://get.helm.sh/helm-v3.15.2-linux-amd64.tar.gz -o helm-v3.15.2-linux-amd64.tar.gz`

b. Unarchive the Helm binary file.

```
tar xvf helm-vx.xx.x-linux-amd64.tar.gz
```

c. Copy the binary into /usr/local/bin

```
cp linux-amd64/helm /usr/local/bin/helm
```

```
chmod 775 /usr/local/bin/helm
```

6 **Install the Azure command line interface:**

a. Download the Azure CLI bundle, version 2.9.xx

```
curl -sSL
https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt
-o azurecli-exe-linux-x86_64-2.9.xx.zip
```

b. Unzip the bundle (LOTS of files in this unzip)

```
unzip azurecli-exe-linux-x86_64-2.9.xx.zip
```

c. Execute the installation script:

```
./azure/install
```

7 Copy over the Veritas binary file bundle and Terraform script bundle. This is a large file which may take sometime.

8 Unzip the file downloaded on the location : `/var/terraform` folder.

Configuring Terraform on RHEL

Using the following commands you can configure Terraform for RHEL operating system:

- 1. To configure the Terraform, use the following command: `sudo dnf config-manager --add-repo`.
- 2. To install the Terraform on RHEL, use the command: `sudo dnf install -y dnf-plugins-core`

About PreFlight checker (checklist) script

The initial most important check with your environment readiness to deploy the Cloud Scale Technology. The PreFlight checklist enables you to understand the prerequisites in very detailed manner and also helps in readiness and troubleshooting the environment, if any.

The script checks for all the necessary permissions, IP address availability, network infrastructure to ensure that the Cloud Scale Technology is deployed using Terraform script flawlessly.

Refer to the following section for more information on how to execute the PreFlight checker script:

See “Installation instructions for deploying the Cloud Scale Technology on Azure” on page 38.

Stages of deploying Terraform scripts on Azure

There are 3 stages to Azure provisioning and deployment. Each stage is executed separately in each phase of their respective subdirectories.

- **Stage 1: Base stage**
- **Stage 2: Addons stage**
- **Stage 3: Deployment stage**

The below mentioned points lets you know what actions are taken in each deployment stage.

1. **Base stage**
   - Creates Azure Kubernetes Service (AKS) cluster.
   - Creates container registry.
   - Creates roles.
2. **Addons stage**
   - Installs Cert Manager
   - Installs Trust Manager
3. **Deployment stage**
   - Loads the Cloud Scale container images to local repository.
   - Tag and push the container images and Helm chart to ACR.
   - Deploys Cloud Scale Technology using Helm chart.

Note: Terraform also supports single-node deployment, using `single_node.tfvars` as the input file.

Parameters for base stage

Refer to the following tables and provide the configuration details depending on the type of installation you want to perform.

Note: Refer the `sample.tfvars` file which is placed in the base directory which has a format for passing the input parameters. Ensure you follow the parameter order.

Note: Cloud Scale Technology deployment is supported on hybrid DNS environment.

Table 5-1 Parameters for base stage

Parameters	Description
Networking parameters that already exists	
vnet_rg_name	Resource group name that contains the Virtual Network.
vnet_name	Virtual Network name where to provision cloud scale resources.
subnet_name_cluster	Subnet name where to provision cloud scale in AKS cluster.
subnet_name_loadbalancer	Subnet name where to provision cloud scale AKS load balancer.
Cloud Scale resources created by Terraform	
cpdata_node_pool_scaling	Scaling configuration block for the cpdata pool nodes. See default value for example. `desired_size`: Desired number of nodes in cpdata pool. `max_size`: Maximum number of nodes in the cpdata pool when autoscaling is enabled. `min_size`: Minimum number of nodes in the cpdata pool when autoscaling is enabled.
custom_tags	Additional tags to be added to resources.
zone	Specify an availability zone in which AKS cluster should be located.
cloudscale_instance_id	A unique identifier to be used in tags and names to identify the Cloud Scale Technology resources specific to this deployment.
new_rg_name	Name of new resource group to be provisioned.
new_user_identity_name	Name of new User Managed Identity to be provisioned.
location	Location region to provision resources.
aks_name	Name of the Cloud Scale Technology AKS cluster.

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
enable_role_base d_access_control_for_aks	false
new_acr_name	Name of new container registry to be provisioned.
use_existing_private_dns_zone	Option to use an existing private DNS zone.
private_dns_zone_rg_name	The resource group name where the private DNS zone resides.
dns_to_vnet_link_name	Name for the DNS zone virtual network link used for Cloud Scale Technology cluster. Required if creating a new private DNS zone.
use_existing_nbsm_role	Option to use existing role or to create new NetBackup Snapshot Manager (NBSM) role. Default is set to `false` If this role is set to `false`, provisioning will automatically create the required roles needed for deployment. If this role is set to `true`, provide nbsm_role_name values. Refer to See "Permissions attached to nbsm_role" on page 32. in case if you use the Terraform to create an new nbsm_role.
nbsm_role_name	This role is set if the `use_existing_nbsm_role` is set to true. This value is the name of an existing role to be used for NetBackup Snapshot Manager to work with Azure assets. This property cannot be changed after the cluster is created.
aks_network_profile	The network profile for the cluster. See the `modules/cloudscale-aks/REAMDE.md` for defaults. See documentation for options.
use_existing_storage_acc	Set it to true only if you want to use existing storage account to create NFS PVC.
storage_acc_id	This is required in case of use_existing_storage_acc is set to true. Mention resource id of storage account name

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
storage_acc_rg_name	This is required in case of use_existing_storage_acc is set to true. Mention storage account resource group name
aks_private_dns_zone_id	Mention resource id of custom private DNS zone
ACR private DNS zone values created by Terraform When the `use_existing_acr_private_dns_zone = false`	
use_existing_acr_private_dns_zone	Option to use an existing private DNS zone for ACR
acr_dns_to_vnet_link_name	This parameter is required only while creating new Azure Container Registry (ACR) private DNS zone. The virtual network link name is used to link the ACR private DNS to the ACR virtual network.
acr_private_dns_zone_rg_name	**For existing private DNS zone:** This is the resource group name where it exists. **For creating a new private DNS zone:** This is the resource group name where to provision the zone.
When the `use_existing_acr_private_dns_zone = true`	
acr_private_endpoint_name	Name of the private endpoint when provisioning a private ACR.
acr_rg_name	The Resource Group name where the ACR resides.
acr_public_network_access_enabled	Option to enable public access on a new ACR.
acr_private_service_connection_name	Name of the private service connection when provisioning a private ACR.
cloud_environment	Specify which cloud environment to use. Default value is public and possible values are public, usgovernment, german, and china.
kubernetes_version	Specify which Kubernetes version to use. The default used is the latest Kubernetes version available in the region
acr_private_dns_zone_group_name	Name of the private DNS zone group when provisioning a private ACR.
node_instance_size_primary_pool	The node virtual machine size of the primary nodes. Use Azure virtual machine skus

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
node_instance_size_media_pool	The node virtual machine size of the media nodes. Use Azure virtual machine skus.
node_instance_size_msdp_pool	The node virtual machine size of the msdp nodes. Use Azure virtual machine skus.
node_instance_size_cpdata_pool	The node virtual machine size of the cpdata nodes. Use Azure virtual machine skus.
primary_node_pool_scaling	Scaling configuration block for the Primary pool nodes. See default value for example. `desired_size: 1` Desired number of nodes in primary pool. `max_size: 2` Maximum number of nodes in the primary pool when autoscaling is enabled. `min_size: 1` Minimum number of nodes in the primary pool when autoscaling is enabled.
msdp_node_pool_scaling	Scaling configuration block for the storage pool nodes. See default value for example. `desired_size:1` Desired number of nodes in msdp pool. `max_size: 1` Maximum number of nodes in the msdp pool when autoscaling is enabled. `min_size: 1` Minimum number of nodes in the msdp pool when autoscaling is enabled.
media_node_pool_scaling	Scaling configuration block for the media pool nodes. See default value for example. `desired_size:1` Desired number of nodes in media pool. `max_size:1` Maximum number of nodes in the media pool when autoscaling is enabled. `min_size:1` Minimum number of nodes in the media pool when autoscaling is enabled.

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
cpdata_node_pool_scaling	Scaling configuration block for the cpdata pool nodes. See default value for example. `desired_size:1` Desired number of nodes in cpdata pool. `max_size:1` Maximum number of nodes in the cpdata pool when autoscaling is enabled. `min_size:1` Minimum number of nodes in the cpdata pool when autoscaling is enabled.
private_dns_zone_name	The name of the private DNS Zone resource used for cloud scale. Needs to be a unique name in the Azure Subscription. This is not required in case of Hybrid DNS environment.
DBaaS configuration. The db_* variables are not required while using internal database (db_create = false)	
db_create	Specifies whether to create Azure Flexible Server PostgreSQL
db_subnet_name	The name of the subnet to create the PostgreSQL Flexible Server. (Should not have any resource deployed in) This parameter is optional and only required if `db_create` is set to true.
db_username	Username for the master DB user. This parameter is optional and only required if `db_create` is set to true.
db_compute_tier	Tier for PostgreSQL Flexible server sku : Compute and storage options in Azure database. Possible values are: GeneralPurpose, Burstable, MemoryOptimized. This parameter is optional and only required if `db_create` is set to true.
db_compute_size	Size for PostgreSQL Flexible server sku : Compute and storage options in Azure database for PostgreSQL This parameter is optional and only required if db_create is set to true.
db_zone	Specify availability-zone for PostgreSQL flexible main server. This parameter is optional and only required if `db_create` is set to true.

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
db_standby_zone	Specify availability-zone to enable high_availability and create standby PostgreSQL Flexible Server. (Null to disable high-availability) This parameter is optional and only required if `db_create` is set to true.
db_backup_retention_days	The days to retain backups for. Must be between 1 and 35. This parameter is optional and only required if db_create is set to true.
db_maintenance_day	The day of week for maintenance window. i.e. Sunday = 0, Monday = 1. Defaults to 0. This parameter is optional and only required if `db_create` is set to true.
db_maintenance_hour	The start hour for maintenance window. Defaults to 0. This parameter is optional and only required if `db_create` is set to true.
db_maintenance_minute	The start minute for maintenance window. Defaults to 0. This parameter is optional and only required if `db_create` is set to true.
db_parameters	PostgreSQL configurations to enable. This parameter is optional and only required if db_create is set to true.
db_geo_redundant_backup_enabled	Enable Geo Redundant Backup for the PostgreSQL Flexible Server. This parameter is optional and only required if db_create is set to true.
db_postgresql_version	Version of PostgreSQL Flexible Server. Possible values are: Version of PostgreSQL Flexible Server This parameter is optional and only required if db_create is set to true.
db_server_name	The name of PostgreSQL Flexible Server instance. This parameter is optional and only required if db_create is set to true.
db_standby_zone	Specify availability-zone to enable high_availability and create standby PostgreSQL Flexible Server. This parameter is optional and only required if db_create is set to true.

Table 5-1 Parameters for base stage *(continued)*

Parameters	Description
db_storage_mb	Storage allowed for PostgresSQL Flexible server. Possible values : azurerm_postgresql_flexible_server This parameter is optional and only required if db_create is set to true.
Optional parameters to support external container registry **Note:** Applicable only when ext_container_registry is set to true.	
ext_container_registry_url	Specifies the URL for the external container registry.
ext_container_registry_secret_name	Name of the secret containing credentials for the external container registry.
ext_container_registry_username	Username to authenticate with the external container registry.
ext_container_registry_password	Password to authenticate with the external container registry.
Note the following: ■ If the external container registry parameters are not provided, the deployment will default to the cloud-specific container registry. ■ If the optional external container registry parameters are not provided or set, the existing functionality is not affected.	

Parameters for addons stage

There are no parameters required for addons stage.

Permissions attached to nbsm_role

While deploying the **Base** stage, Terraform creates a nbsm_role if the `use_existing_nbsm_role` is set to `false`. By default, the Terraform assigns permissions required for below features:

```
Backup from snapshot
Creating backup from snapshot
Restore from backup from snapshot
Protection of Virtual Machines
Restore disks from snapshots/restore point
Rollback restore/Cleanup in restore
Restore disk
```

```
Cleanup
List Resources
Discovery
Snapshots and Restores
Snapshot
List restore points
List snapshots
List disk snapshots
Write snapshots
Snapshot cleanup
Create restore point collections
Restore VM
Get cluster information
Scale-in/Scale-out
High availability
```

Refer to the section *Configuring permissions on Microsoft Azure* from the guide *NetBackup™ Snapshot Manager Install and Upgrade Guide* to get more details about permissions for the listed features and add new permissions in case you want to use features which are not listed here.

Parameters for addons stage

There are no parameters required for addons stage.

Parameters for deployment stage

Refer to the following tables and provide the configuration details depending on the type of installation you want to perform.

Table 5-2 Parameters for deployment stage

Parameters	Description
tar_file_location	NetBackup Cloud Scale tar location.
tar_file_name	Name of the NetBackup Cloud Scale tar.
media_server_replica_count	Provide the number of replicas for media server. The desired size of the media server pool and the replica count should be same. The `media_server_replica_count`must be between 1-16. The default is 1.

Table 5-2 Parameters for deployment stage *(continued)*

Parameters	Description
storage_server_replica_count	Provide the number of replicas for Storage Server. The desired size of the storage server node pool and the replica count should be same. The `storage_server_replica_count` must be between 1-16. The default is 1.
primary_server_ip_fqdn_mapping	Provide IP hostname mapping for NetBackup primary server. The primary username must be of 1-32 characters long and must start with a lowercase letter and can only contain alphanumeric characters, hyphens, and underscores.
storage_server_ip_fqdn_mapping	Provide hostname of NetBackup storage server. Storage server IP FQDN entries must be equal to storage server replica count. You can add multiple entries and it can be provided as comma separated objects like [{},{}].
snapshot_manager_ip_fqdn_mapping	Provide mapping of NetBackup Snapshot Manager Server.
primary_username	Provide username to configure primary server. The `primary_username` must be of 1-32 characters long and must start with a lowercase letter and can only contain alphanumeric characters, hyphens, and underscores. It is used to login into NetBackup web UI.
primary_password	Provide password for the user to configure the primary server The `primary_password` must be at least 8 characters long and must have at least a number, a lowercase, uppercase, and a special character (@$%!*?&.).
host_master_key_id	Provide the Host Master Key ID. The `host_master_key_id` must be of 1-32 characters long, must contain only lowercase alphanumeric characters, hyphens, and underscores.
host_master_key_passphrase	Provide the Host Master Key passphrase. The `host_master_key_passphrase` must be at least 12 characters long and must have at least a number, a lower case, an upper case and a special character (@$%!*?&.).

Table 5-2 Parameters for deployment stage *(continued)*

Parameters	Description
key_protection_key_id	Provide the Key Protection Key ID. The `key_protection_key_id` must be of 1-32 characters long, must contain only lowercase alphanumeric characters, hyphens, and underscores.
key_protection_key_passphrase	Provide the Key Protection Key passphrase. The `key_protection_key_passphrase` must be at least 12 characters long and must have at least a number, a lowercase,an uppercase and a specialcharacter(@$%!*?&.)
storage_server_kms_key_group	Provide the name of KMS Key Group for storage server. The `storage_server_kms_key_group` must be of 1-64 characters long with at least one lowercase alphabet, other characters include alphanumeric characters and hyphens.
storage_server_kms_ key_secret_name	Provide the KMS key name for storage server. The `storage_server_kms_key_secret_name` must be of 1-32 characters long, must contain only lowercase alphanumeric characters, hyphens or underscores.
storage_server_kms_ key_secret_password	Provide the KMS key password for storage server. The `storage_server_kms_key_secret_password` must be at least 12 characters long and must have at least a number, a lower case, an upper case and a special character (@$%!*?&.).
storage_server_kms_ key_secret_username	Provide the KMS key username for storage server.
storage_server_credential_ secret_name	Provide the credential name for storage server.
storage_server_credential_ secret_username	Provide the username for storage server credentials. The `storage_server_credential_secret_username` must be of 1-62 characters long, must be in the printable ASCII range (0x20-0x7E) except for spaces, leading/trailing quotes and the special characters ('*', '\', '/', '^', '(', ')',"", '<', '>', '&', '[', ']', '%', '@', '#).

Table 5-2 Parameters for deployment stage *(continued)*

Parameters	Description
storage_server_credential_ secret_password	Provide the password for storage server credentials. The `storage_server_credential_secret_password` must be of 8-62 characters long, must be in the printable ASCII range (0x20-0x7E) except for spaces, leading/trailing quotes and the special characters ('*', '\', '/', '^', '(', ')',"", '<', '>', '&', '[', ']', '%', '@', '#').
primary_server_catalog_size_in_gi	Provide the size for primary server catalog volume. It must be at least 100 Gi.
primary_server_log_size_in_gi	Provide the size for primary server log volume. It must be at least 30 Gi.
primary_server_data_size_in_gi	Provide the size for primary server data volume. It must be at least 30 Gi.
media_server_log_size_in_gi	Provide the size for media server log volume. It must be at least 30 Gi.
media_server_data_size_in_gi	Provide the size for media server data volume. It must be at least 50 Gi.
storage_server_log_size_in_gi	Provide the size for storage server log volume. It must be at least 5 Gi.
storage_server_data_size_in_gi	Provide the size for storage server data volume. It must be at least 5 Gi.
snapshot_manager_log_size_in_gi	Provide the size for snapshot manager log volume. It must be at least 5 Gi.
snapshot_manager_data_size_in_gi	Provide the size for snapshot manager data volume. It must be at least 30 Gi.
fluentbit_log_collector_size_in_gi	Provide the size of the fluentbit log collector. It must be at least 100 Gi.
snapshot_manager_vx_http_proxy	Provide the value to be used as the HTTP proxy for all connections for Snapshot Manager. This is optional field.
snapshot_manager_vx_https_proxy	Provide the value to be used as the HTTPS proxy for all connections for Snapshot Manager. This is optional field.

Table 5-2 Parameters for deployment stage *(continued)*

Parameters	Description
snapshot_manager_vx_no_proxy	Provide the addresses that are allowed to bypass the proxy server. You can specify host names, IP addresses, and domain names in this parameter as comma separated. This is optional field. While providing multiple values please escape commas and dots in urls if any with \\. For example "localhost\\,mycompany\\.com\\,1.2.3.4"
dr_info_secret_name	Name of secret to pass DR information. This is an optional field.
dr_info_secret_passphrase	Details of DR passphrase. This is an optional field.
dr_info_secret_email_address	Details of DR email address. This is an optional field.
email_server_configmap_name	Name of the `configmap` that will contain all required information to configure email server. This is an optional field.
email_server_configmap_details	Details required to configure email server. This is an optional field. Provide all the required fields comma separated. Please escape commas with \\ while providing values. For example: `email_server_configmap_details="smtp=smtpServerName:port\\,ssl-verify=ignore\\,smtp-use-starttls"`
global_timezone	Provide timezone for Primary server pods. This is an optional field. For example: primary_server_timezone="/usr/share/zoneinfo/Asia/Kolkata"
enable_sidecars_logging	Enable or disable the sidecar logging. Default value is *false*. For example, `enable_sidecars_logging = false`

PaaS based PostgreSQL deployment (DBaaS) on Azure

PostgreSQL is a service which adds support to use external database instead of the internal one to use with Cloud Scale Technology services. Using the external PostgreSQL database which manages the database, it improves the resiliency,

allows the database to be scaled up as needed, and reduces the maintenance requirements for NetBackup's database services in AKS.

DBaaS deployment is selected by setting `db_create = true` in the Terraform deployment input file at the base step.

Ffor more details on the stages of deployment, refer to the following section:

See "Stages of deploying Terraform scripts on Azure" on page 25.

See "Troubleshooting issues" on page 48.

To reset the password for PostgreSQL database, refer to the following section:

See "Change the PostgreSQL database server password" on page 40.

For maintenance purpose after deployment, refer to the **Managing PostrgreSQL DBaaS** section in the *Cohesity Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters*.

Installation instructions for deploying the Cloud Scale Technology on Azure

Following steps are required to build the infrastructure for deploying the Cloud Scale Technology environment.

Note: Terraform stores the state about your managed infrastructure and configuration. This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and improve performance for large infrastructures. This state is stored by default in a local file named `terraform.tfstate` in 3 respective directories. Terraform uses state to determine what changes to make to your infrastructure. Hence, the `terraform.tfstate` is very crucial and we recommend taking backup of whole terraform source code along with `terraform.tfstate` files by creating zip file and uploading it into the storage account after completing the deployment successfully.

Before proceeding to execute the scripts, you need to execute the PreFlight checker script twice. To know about the PreFlight checker, refer to the section See "About PreFlight checker (checklist) script" on page 24.

Deploying the Cloud Scale Technology on Azure

1. Locate and execute the PreFlight checker script from the repository and execute it before the **Base** step using the following command:

   ```
   ./cloudscale-preflight-check.sh -p azure -t preInfra
   ```

2. Execute the **Base** stage instructions:

   - Log in and authenticate the Azure account using Azure CLI.
   - Change the directory using the command:

     ```
     cd azure/base
     ```

   - Create a new `.tfvars` based on the sample .tfvars with the appropriate values and execute the commands below:

     ```
     terraform init
     terraform plan -var-file <vars-file>.tfvars
     terraform apply -var-file <vars-file>.tfvars
     ```

3. Execute the **Addons** steps instruction given in the next procedure.

   - Change the directory using the command:

     ```
     cd azure/addons
     terraform init
     terraform plan
     terraform apply
     ```

4. Again execute the PreFlight script after the **Addons** step using the command:

   ```
   ./cloudscale-preflight-check.sh -p azure -t postInfra
   ```

   You will have to provide the Base input .tfvars file and Deployment input .tfvars file path for validation. Before running the `postInfra` script you will have to modify the `deployment.tfvars` file.

5. Execute the **Deployment** steps given in the next procedure.

   - Change the directory using the command:

     ```
     cd azure/deployment
     ```

   - Create new `.tfvars` file based on the sample .tfvars with the appropriate values.
     - `terraform init`
     - `terraform plan -var-file <vars-file>.tfvars`
     - `terraform apply -var-file <vars-file>.tfvars`

   Verify the deployment status, using the following command:

```
kubectl get environment -n netbackup
```

The status is displayed as follows:

```
$ kubectl get environment -n netbackup -w
NAME       READY   AGE   STATUS
assdbii    4/4     18h   Success
```

6 Execute the steps mentioned in the following section:

See "Change the PostgreSQL database server password" on page 40.

Terraform stores input values in the state file and to improve the security reset the database password after deployment.

Change the PostgreSQL database server password

This section describes on how to change the database password. Using the Azure CLI, you have to first create the Azure CLI container, run the reset password command from that container and then change the password.

Note: When setting the PostgreSQL password in DBaaS, ensure that the password does not contain the following special characters: equal (=), double quote ("), single quote ('), percentage (%), at sign (@), ampersand (&), question mark (?), underscore (_), and hash (#)

Steps to change password

1 Launch an Azure CLI pod into the AKS cluster using the following command:

```
$ kubectl run az-cli --image=mcr.microsoft.com/azure-cli:2.53.0
--command sleep infinity
```

Note: Access to Azure Key Vault is restricted to specific subnets. Passwords that are stored in Azure Key Vault can be easily updated from a pod running in AKS.

---

2 Using `exec`, log in into the Azure CLI pod:

```
$ kubectl exec -it az-cli -- /bin/ash
```

3 From the Azure CLI pod, log in into the Azure account:

```
$ az login --scope https://graph.microsoft.com//.default
```

4 (Optional step) Create a Key Vault policy to allow the current user to retrieve the database credential. Keep a note of your resource group, key vault, and ID of the current user by using the following respective commands:

- **Resource group name:**

```
$ RESOURCE_GROUP=<resource_group_name>
```

- **Key Vault name:**

```
$ KEY_VAULT_NAME=$(az keyvault list --resource-group
$RESOURCE_GROUP --resource-type vault | jq -r '.[].name')
```

- **Current user ID name:**

```
$ USER_ID=$(az account show | jq -r '.user.name')
```

- **Create a Key Vault access policy:**

```
$ az keyvault set-policy -n $KEY_VAULT_NAME --upn $USER_ID
--resource-group $RESOURCE_GROUP --secret-permissions all
```

5 Note the log in name for the key vault (DBADMINUSER):

```
$ DBADMINUSER=$(az keyvault secret show --vault-name
$KEY_VAULT_NAME --name dbadminlogin | jq -r .value)
```

6 Note the password for the Key Vault (OLD_DBADMINPASSWORD):

```
$ OLD_DBADMINPASSWORD=$(az keyvault secret show --vault-name
$KEY_VAULT_NAME --name dbadminpassword | jq -r .value)
```

7 Note the server name (DBSERVER):

```
DBSERVER=$(az postgres flexible-server list --resource-group
$RESOURCE_GROUP | jq -r '.[].name')
```

8 To set a new password, follow the steps:

```
NEW_DBADMINPASSWORD="<new_password>" az postgres flexible-server
execute -p $OLD_DBADMINPASSWORD -u $DBADMINUSER -n $DBSERVER -d
postgres -q "ALTER USER\"nbdbadmin\" WITH PASSWORD
'$NEW_DBADMINPASSWORD';"
```

To re-encrypt the current password without changing it

```
az postgres flexible-server execute -p $OLD_DBADMINPASSWORD -u
$DBADMINUSER -n $DBSERVER -d postgres -q "ALTER USER\"nbdbadmin\"
WITH PASSWORD '$OLD_DBADMINPASSWORD';"
```

Note: You also have an option to reset the flexible server password using the command. `az postgres flexible-server update -g $RESOURCE_GROUP -n $DBSERVER --admin-password <password>` This command can be run outside of the Azure CLI (az-cli) container.

---

9 To verify if the password uses the correct encryption method (SCRAM-SHA-256), run the command:

```
$ az postgres flexible-server execute -p "<new_password>" -u
$DBADMINUSER -n $DBSERVER -d postgres -q "SELECT * from
azure_roles_authtype();"
```

```
+---------------------------+-----------+
| rolename                  | authtype  |
|---------------------------+-----------|
| azuresu                   | NOLOGIN   |
| pg_database_owner         | NOLOGIN   |
| pg_read_all_data          | NOLOGIN   |
| pg_write_all_data         | NOLOGIN   |
| pg_monitor                | NOLOGIN   |
| pg_read_all_settings      | NOLOGIN   |
| pg_read_all_stats         | NOLOGIN   |
| pg_stat_scan_tables       | NOLOGIN   |
| pg_read_server_files      | NOLOGIN   |
| pg_write_server_files     | NOLOGIN   |
| pg_execute_server_program | NOLOGIN   |
| pg_signal_backend         | NOLOGIN   |
| azure_pg_admin            | NOLOGIN   |
| replication               | NOLOGIN   |
| nbdbadmin                 | SCRAM-256 |
+---------------------------+-----------+
SELECT 15
Time: 0.009s
```

10 To store the updated password in the key vault using the command:

```
$ az keyvault secret set --vault-name $KEY_VAULT_NAME --name
dbadminpassword --value "<new_password>"
```

11 (Optional step) To delete the Key Vault access policy that is created in step 4:

```
$ az keyvault delete-policy -n $KEYVAULT --upn $USER_ID
```

12 To exit from the Azure CLI pod using the command:

```
$ exit
```

13 To delete the Azure CLI pod using the command:

```
$ kubectl delete pod az-cli
```

14 To restart the primary pod using the command *Applicable only for an existing Veritas Cloud Scale deployment*:

```
$ kubectl rollout restart "statefulset/${PRIMARY}" --namespace
"${NAMESPACE}"
```

In the command:

- NAMESPACE is the namespace containing your NetBackup deployment.
- PRIMARY is the name of the primary pod's stateful set.

To obtain the NAMESPACE and PRIMARY, use the command:

```
$ kubectl get --namespace "${NAMESPACE}" primaryserver -o
jsonpath='{.items[0].status.attributes.resourceName}'
```

For resetting the password for a containerized PostgreSQL database, refer to the **hanging database server password in DBaaS** section in the *Cohesity Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters*.

Chapter 6

Accessing the Cloud Scale environment

This chapter includes the following topics:

- Accessing the Cloud Scale Technology environment after deployment

Accessing the Cloud Scale Technology environment after deployment

Once the operators are created successfully, the Terraform scripts display deployment as successful. To verify the product deployment status, execute the below commands from the same Terraform Management Server.

1. Login to Azure environment and execute the command:

   ```
   kubectl get namespaces
   ```

   After executing the above command, you will get list of namespaces created. You can also view by navigating through UI under Kubernetes resources.

2. To view the Cloud Scale Technology deployment environment, execute the below command and refer the table for output:

   ```
   kubectl get --namespace netbackup
   all,environments,primaryservers,mediaservers,msdpscaleouts,cpservers
   ```

3. **Output:**

   ```
   NAME                                                READY   STATUS
   RESTARTS  AGE
   pod/10-244-117-22.aks-nbux-medium-cfg-te-15902.internal   2/2
     Running   0          11m
   pod/dedupe1-uss-agent-54j9t                                1/1
   ```

```
  Running   0          11m
pod/dedupe1-uss-agent-6jnff                                  1/1
  Running   0          11m
pod/dedupe1-uss-agent-bbsmn                                  1/1
  Running   0          11m
pod/dedupe1-uss-agent-lrktl                                  1/1
  Running   0          11m
pod/dedupe1-uss-controller-0                                 1/1
  Running   0          11m
pod/dedupe1-uss-mds-1                                        1/1
  Running   0          12m
pod/flexsnap-agent-59fb7f957b-5t5vj                          1/1
  Running   0          2m20s
pod/flexsnap-api-gateway-7b89c8957d-vlj5j                    1/1
  Running   0          2m21s
pod/flexsnap-certauth-65944c6797-vvspm                       1/1
  Running   0          3m45s
pod/flexsnap-coordinator-84ccfd95c5-59ztr                    1/1
  Running   0          2m20s
pod/flexsnap-fluentd-9b22l                                   1/1
  Running   0          3m8s
pod/flexsnap-fluentd-collector-85fbc6677b-k2b56              1/1
  Running   0          3m7s
pod/flexsnap-fluentd-rqqkd                                   1/1
  Running   0          3m8s
pod/flexsnap-listener-8654fb56d9-4ltrs                       1/1
  Running   0          2m18s
pod/flexsnap-nginx-787878dfb6-j6m6r                          1/1
  Running   2          2m21s
pod/flexsnap-notification-548bf5fdb6-tdwm6                   1/1
  Running   0          2m19s
pod/flexsnap-rabbitmq-0                                      1/1
  Running   0          2m57s
pod/flexsnap-scheduler-578d4646fd-z8fcv                      1/1
  Running   0          2m19s
pod/flexsnap-workflow-general-1709012159-12c95675-tpnqw      1/1
  Running   0          78s
pod/media1-media-0                                           1/1
  Running   0          6m58s
pod/nb-postgresql-0                                          1/1
  Running   0          39m
pod/nucleus-env-primary-0                                    1/1
  Running   0          34m
```

```
NAME                                          READY   AGE
statefulset.apps/dedupe1-uss-controller   1/1     11m
statefulset.apps/flexsnap-rabbitmq        1/1     2m58s
statefulset.apps/media1-media             1/1     6m59s
statefulset.apps/nb-postgresql            1/1     39m
statefulset.apps/nucleus-env-primary      1/1     34m

NAME
COMPLETIONS   DURATION   AGE
job.batch/flexsnap-workflow-general-1709012159-12c95675  0/1
       79s        79s

NAME                                                    TAG    AGE
   STATUS
primaryserver.netbackup.veritas.com/nucleus-env       10.4   38m
   Success

NAME                                      AGE   TAG    SIZE   READY
msdpscaleout.msdp.veritas.com/dedupe1    12m   20.4   1      1

NAME                                        TAG   AGE     PRIMARY
 SERVER                                         STATUS
mediaserver.netbackup.veritas.com/media1   10.4   7m59s
<buildnumber>.aks-nbux-medium-cfg-te-15902.internal   Success

NAME                                        TAG    AGE     STATUS
cpserver.netbackup.veritas.com/cpserver-1   10.4   3m56s   Success
```

4. Access theCloud Scale TechnologyWeb UI using the https://%3Cprimaryserver%3E/webui/login.

   The primaryserver is the host name or IP address of the NetBackup primary server that you want to sign in to.

Terraform scripts helps to quickly and easily build the infrastructure and deploy Veritas Cloud Scale Technology on desired cloud environment.

Chapter 7

Troubleshooting and cleanup environment steps

This chapter includes the following topics:

- Troubleshooting issues
- Cleanup steps

Troubleshooting issues

The following table lists some of the issues that you may come across while deploying Terraform on Azure.

Table 7-1 List of troubleshooting issues

SrNo	Issue	Description / Resolution
1	Error: "psql: error: connection to server at "sveritas1-postgres.postgres.database.azure.com" (10.119.74.36), port 5432 failed: FATAL: no pg_hba.conf entry for host "10.119.72.151", user "nbdbadmin", database "postgres", no encryption"	The deployment scripts request the db password to encrypt the password using SCRAM_SHA-256 method but Azure encrypts it using MD5. **Resolution:** You may have to re-encrypt the passwords after deploying the AKS and DBaaS infrastructure.

Table 7-1 List of troubleshooting issues *(continued)*

SrNo	Issue	Description / Resolution
2	The Terraform supports the podman-based Cloud Scale Technology deployments which will not support the docker implemented nbbuilder script for engineering binary installations.	**Resolution:** The Podman does not support engineering binary installation as the nbbuilder script supports only docker installation.
3	Even after executing the `destroy` command, execute the manual steps provided in the resolution if there are any folders that are not removed from the en4vironment.	If you want to delete the entire infrastructure, using the Azure Portal: ■ Delete resource group For the clean deployment next time, ensure that you have also deleted the following: ■ .tfstate ■ .tfstate.backup ■ .terraform.lock.hcl file ■ terraform folder from base, addons, and deployment

4	**Error: Azure API returned the following**
	`Error: updating Flexible Server (Subscription: "1afb8748-7dc0-4ddc-8faf-e453dcdb7ca3"`
	`| Resource Group Name: "rg-ananmainldbrem" | Flexible Server Name: "anshadbrem-postgres"): polling after Update: polling failed: the Azure API returned the following error: | | Status: "Failed" | Code: "Failed" | Message: "Server anshadbrem-postgres is busy with other operations. Please try later" |`
		Resolution: Retry the base deployment.

Table 7-1 List of troubleshooting issues *(continued)*

SrNo	Issue	Description / Resolution
5	Before executing the `terraform destroy` command, execute the following command: `"TOKEN=$(az acr login --name acr_name --expose-token --output tsv --query accessToken);helm registry login acr_name --username 00000000-0000-0000-0000-000000000000 --password $TOKEN""`	**Reason:**As per official documentation from Microsoft, it is recommended to run the `az acr login` command before executing any docker command as the `acr login` expires after 3 hours.

6	Following warning messages are displayed during the addon deployment:
	`W1002 14:51:24.301599 27385 warnings.go:70] spec.privateKey.rotationPolicy: In cert-manager >= v1.18.0, the default value changed from `Never` to `Always`.`
	`null_resource.install_trust_manager (local-exec): :warning: WARNING: Consider increasing the Helm value `replicaCount` to 2 if you require high availability.`
	`null_resource.install_trust_manager (local-exec): :warning: WARNING: Consider setting the Helm value `podDisruptionBudget.enabled` to true if you require high availability.`
		Ignore these warnings and proceed further.

Table 7-1 List of troubleshooting issues *(continued)*

SrNo	Issue	Description / Resolution
7	Following error message is displayed when deploying with RG/MC_group creation time: `Status 400 (400 Bad Request) with response: { "code": "InvalidParameter", "details": null, "message": "The length of the node resource group name is too long. The maximum length is 80 and the length of the value provided is 84. Please see https://aka.ms/aks-naming-rules for more details.", "subcode": "", "target": "name" } with module.cloudscale.azurerm_kubernetes_cluster.cloudscale, on modules/cloudscale-aks/main.tf line 11, in resource "azurerm_kubernetes_cluster" "cloudscale": 11: resource "azurerm_kubernetes_cluster" "cloudscale" {`	To avoid errors, ensure that you follow the guidelines listed in the following documentation while creating RG/MC_group: Naming restrictions for AKS resources and parameters

Cleanup steps

These steps are to be followed if you wish to cleanup the resource which are created during the deployment including infrastructure and product deployment.

Terraform destroy command can be used to destroy the resources created during the deployment. The destroy operation is performed in reverse order from that of creation. It is used instead of deleting the assets individually.

Note: Before executing the `terraform destroy` command, execute the following command:

```
"TOKEN=$(az acr login --name acr_name --expose-token --output tsv
--query accessToken);helm registry login acr_name --username
00000000-0000-0000-0000-000000000000 --password $TOKEN""
```

Sequence to cleanup the deployment infrastructure

Pass the input variable files (`.tfvars`) which were used during creation. Navigate to the respective directories and execute the following commands:

1. **Deployment**:

   ```
   cd azure/deployment
   ```

   ```
   terraform destroy -var-file <vars-file>.tfvars
   ```

   You may need to run the `destroy` command twice to cleanup the environment.

   Note: It may happen that even after executing the `destroy` command, the environment is not cleaned. Execute the manual steps to cleanup the remains. Refer to the pt.3 from theSee "Troubleshooting issues" on page 48.

2. **Addons**:

   ```
   cd azure/addons
   ```

   ```
   terraform destroy
   ```

3. **Base**:

   ```
   cd azure/base
   ```

   ```
   terraform destroy -var-file <vars-file>.tfvars
   ```

Sequence to cleanup the values from Deployment and Addons

1. Deploy base
2. Deploy addons
3. Deploy deployment
4. Destroy deployment
5. Execute the following commands:

   ```
   kubectl delete pvc --all -n netbackup
   ```

   ```
   kubectl delete pvc --all -n netbackup-operator-system
   ```

```
kubectl delete pv --all
```

6. Destroy addons