Cohesity Cloud Scale Technology Deployment Guide Using Terraform for AWS

Release 11.2

VERITAS
now part of Cohesity

Cohesity Cloud Scale Technology deployment guide using Terraform for AWS

Last updated: 2026-05-28

Legal Notice

Copyright © 2026 Cohesity, Inc. All rights reserved.

© 2026 Cohesity, Inc. All Rights Reserved. Cohesity, the Cohesity Logo and other Cohesity Marks are trademarks of Cohesity, Inc. in the US and/or internationally. The information supplied herein is the confidential and proprietary information of Cohesity and may only be used (a) by the intended recipients and (b) in conjunction with validly licensed Cohesity software and services. Find the terms of Cohesity licenses at www.cohesity.com/agreements.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. COHESITY SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT 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 7
About Terraform 8

Chapter 2 Getting started with deployment 10

Steps for getting started with the deployment 10

Chapter 3 Prerequisities for setting up AWS environment 12

Before you start with the deployment 12
Networking requirement 13
AWS authentication and permission requirements 14

Chapter 4 Prerequisites for Terraform 19

Terraform Management Server requirements 19

Chapter 5 Deploying Cloud Scale Technology using Terraform script 21

Creating and configuring Terraform Management Server 21
Installing the packages for Terraform Management Server 22
About PreFlight checker(checklist) script 25
Stages of deploying Terraform scripts on AWS 26
Parameters for base stage 27
Parameters for addons stage 33
Parameters for deployment stage 33
PaaS based PostgreSQL deployment (DBaaS) on AWS 38
Installation instructions for deploying the Cloud Scale Technology on AWS 39
Changing database server password in PostgreSQL (AWS) 41

Chapter 6 Accessing the Cloud Scale Technology environment 43

Accessing the Cloud Scale Technology environment after deployment 43

Chapter 7 Troubleshooting and cleanup environment steps 46

Troubleshooting issues 46
Cleanup steps 48

Chapter 1

Introduction

This chapter includes the following topics:

- About this guide
- About Cloud Scale Technology
- About Terraform

About this guide

This document provide the deployment Cloud Scale Technology components in Elastic Kubernetes Services (EKS) on AWS using Terraform. The intended audience for this document includes backup administrators, cloud administrators, architects, and system administrators. 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 is 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, please 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 AWS.

Table 1-1 Important terms

Term	Description
EKS	Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service that makes it easy for you to run Kubernetes on AWS and on-premises.
EBS	Amazon Elastic Block Store (Amazon EBS) is an easy-to-use, scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2).
ECR	Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
EFS CSI	Amazon EFS Container Storage Interface (CSI) driver provides a CSI interface that allows Kubernetes clusters running on AWS to manage the lifecycle of Amazon EFS file systems.
VPC	Amazon Virtual Private Cloud (VPC) lets you create and manage virtual networks for AWS resources.

About Cloud Scale Technology

Cloud Scale Technology redefines data management for the next decade. Its service elasticity and modern web-scale technologies enables the 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. Some key points explaining the advantages of Terraform:

- **Manage any infrastructure:** Terraform uses immutable approach to infrastructure 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 that are called modules that define configurable collections of infrastructure.

Terraform supports several cloud infrastructure providers such as Amazon Web Services (AWS), Cloudflare, Microsoft Azure, 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 AWS

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

Chapter 2

Getting started with deployment

This chapter includes the following topics:

- Steps for getting started with the deployment

Steps for getting started with the 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 AWS

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 to the section See "Terraform Management Server requirements " on page 19.
2. Configure Terraform Management Server.	Refer to the section See "Creating and configuring Terraform Management Server" on page 21. Refer to the section See "Installing the packages for Terraform Management Server" on page 22.
3. Authentication with AWS.	User / role which you will be using for deployment should have minimum permissions. Refer to the section See "AWS authentication and permission requirements " on page 14.

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

Steps	Description
4. Execute the PreFlight checker script.	This checklist is executed to verify the environment readiness before deploying the Cloud Scale Technology. Refer to the section See "About PreFlight checker(checklist) script" on page 25.
5a. Learn about the stages involved in the Terraform deployment. 5b. Installation instructions for deploying the Cloud Scale Technology.	See "Stages of deploying Terraform scripts on AWS" on page 26. See "Installation instructions for deploying the Cloud Scale Technology on AWS" on page 39.
6. Access Cloud Scale Technology UI after deployment.	See "Accessing the Cloud Scale Technology environment after deployment" on page 43.

Chapter 3

Prerequisities for setting up AWS environment

This chapter includes the following topics:

- Before you start with the deployment
- Networking requirement
- AWS authentication and permission requirements

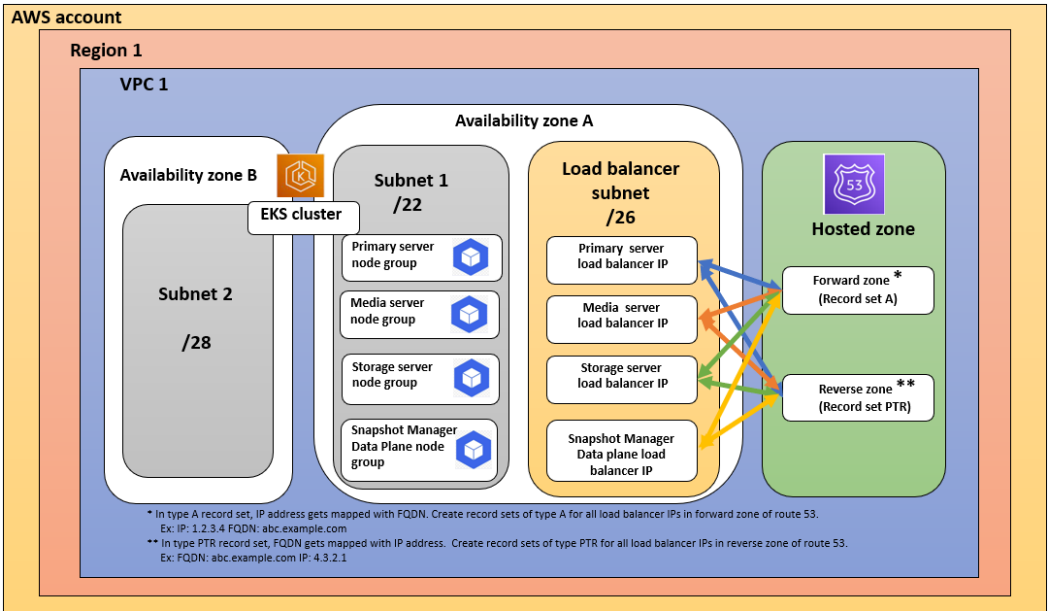
Before you start with the deployment

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

- Ensure that the See “Networking requirement” on page 13. are met.
- Ensure that the See “AWS authentication and permission requirements ” on page 14. are assigned to the user before starting the deployment.

Networking requirement

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



Ensure that the below networking requirements are met.

- VPC and subnets must be created in AWS account before the Terraform scripts are executed.
- Required address spaces for EKS - Two subnets in different availability zones (Subnet 1 and Subnet 2):
  - **For the EKS cluster subnets in Availability zone A:**There is one subnet and one load balancer.
    - Subnet 1 with /22 or /24 (used for node group).
    - Load balancer with /26. (This subnet needs to be empty with no virtual machines / devices installed)
  - **For subnet 2 in Availability zone B:** There is only one subnet:
    - Subnet 2 with /28 (for address space).
- Create DNS entries in the Route53 Hosted Zone: (Primary (1), MSDP (1), Snapshot Manager (1) before executing the Terraform scripts.

  - Define a forward lookup, private DNS zone to be used by EKS as part of the load balancing subnet.
  - Populate the forward DNS zone with the appropriate records.
  - Define a 2nd reverse lookup, private DNS zone to be used by EKS as part of the load balancing subnet.
  - Populate the reverse DNS zone with the appropriate records.
- Sufficient quota for inbound and outbound rules for a security group are required. By default, the quota for inbound and outbound rules for a security group is 60, increase it to 100. For more details refer Amazon VPC quotas.
- 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.
- Terraform server used to deploy Cloud Scale must be able to communicate with the cluster API server for your EKS server.

Note: If the EKS VPC endpoint is enabled for the VPC and the Terraform Management Server are in the same VPC, the deployment may fail, as the EKS OIDC service endpoint can't be accessed from inside that VPC. Consequently, operations such as creating an OIDC provider in the VPC will not work and results in a timeout when attempting to request https://oidc.eks.region.amazonaws.com. This is the limitation from AWS cloud provider. For more details, refer to the EKS VPC endpoint. link

---

AWS authentication and permission requirements

There are two ways to authenticate to AWS:

1. **Through user credentials**:

   Authenticate to AWS with user having the following permissions (mentioned in pt.2)

2. **Attach IAM role to the Terraform Management Server**:

   You have to attach IAM role to the Terraform Management Server

   Below are the required permissions to be assigned to a user or IAM role created

   - **A. Managed Policies**

```
AmazonEKSClusterPolicy
AmazonEKSWorkerNodePolicy
AmazonEC2ContainerRegistryFullAccess
AmazonEKSServicePolicy
AmazonEKS_CNI_Policy
AmazonEKSVPCResourceController
```

- **B. Create customer managed policy with below permissions and attach it to IAM role.**

```
elasticfilesystem:DescribeAccountPreferences
elasticfilesystem:DescribeBackupPolicy
elasticfilesystem:DeleteAccessPoint
elasticfilesystem:DescribeReplicationConfigurations
elasticfilesystem:UntagResource
elasticfilesystem:CreateFileSystem
elasticfilesystem:ListTagsForResource
elasticfilesystem:DeleteTags
elasticfilesystem:DescribeLifecycleConfiguration
elasticfilesystem:ClientMount
elasticfilesystem:DescribeFileSystemPolicy
elasticfilesystem:DescribeFileSystems
elasticfilesystem:DeleteMountTarget
elasticfilesystem:CreateAccessPoint
elasticfilesystem:ModifyMountTargetSecurityGroups
elasticfilesystem:DescribeMountTargets
elasticfilesystem:DescribeAccessPoints
elasticfilesystem:CreateAccessPoints
elasticfilesystem:TagResource
elasticfilesystem:CreateTags
elasticfilesystem:DescribeTags
elasticfilesystem:CreateMountTarget
elasticfilesystem:Backup
elasticfilesystem:DeleteFileSystem
elasticfilesystem:DescribeMountTargetSecurityGroups
elasticfilesystem:UpdateFileSystem
eks:UpdateClusterVersion
eks:ListTagsForResource
eks:UpdateAddon
eks:ListAddons
eks:UpdateClusterConfig
eks:DescribeAddon
eks:UpdateNodegroupVersion
```

```
eks:UpdateNodegroup
eks:AssociateEncryptionConfig
eks:ListUpdates
eks:UpdateClusterConfig
eks:DescribeAddon
eks:UpdateNodegroupVersion
eks:DescribeNodegroup
eks:AssociateEncryptionConfig
eks:DescribeAddonConfiguration
eks:UntagResource
eks:CreateNodegroup
eks:RegisterCluster
eks:DeregisterCluster
eks:DeleteCluster
eks:DescribeIdentityProviderConfig
eks:DeleteAddon
eks:DeleteNodegroup
eks:DescribeUpdate
eks:TagResource
eks:AccessKubernetesApi
eks:CreateAddon
eks:UpdateNodegroupConfig
eks:DescribeCluster
eks:ListClusters
eks:AssociateIdentityProviderConfig
iam:CreateInstanceProfile
iam:CreateServiceLinkedRole
iam:GetPolicyVersion
iam:UntagRole
iam:PutRolePermissionsBoundary
iam:TagRole
iam:UpdateOpenIDConnectProviderThumbprint
iam:RemoveRoleFromInstanceProfile
iam:DeletePolicy
iam:CreateRole
iam:AttachRolePolicy
iam:ListInstanceProfileTags
iam:PutRolePolicy
iam:DeleteRolePermissionsBoundary
iam:AddRoleToInstanceProfile
iam:ListInstanceProfilesForRole
iam:PassRole
iam:DetachRolePolicy
```

```
iam:DeleteRolePolicy
iam:ListOpenIDConnectProviderTags
iam:PutRolePolicy
iam:DeleteRolePermissionsBoundary
iam:AddRoleToInstanceProfile
iam:ListInstanceProfilesForRole
iam:PassRole
iam:DetachRolePolicy
iam:DeleteRolePolicy
iam:ListOpenIDConnectProviderTags
iam:ListPolicyTags
iam:ListRolePolicies
iam:CreatePolicyVersion
iam:DeleteOpenIDConnectProvider
iam:ListPolicies
iam:DeleteRole
iam:UpdateRoleDescription
iam:ListInstanceProfiles
iam:TagPolicy
iam:CreateOpenIDConnectProvider
iam:CreatePolicy
iam:ListPolicyVersions
iam:ListOpenIDConnectProviders
iam:GetAccountName
iam:UntagPolicy
iam:UpdateRole
iam:UntagOpenIDConnectProvider
iam:GetOpenIDConnectProvider
iam:UntagInstanceProfile
iam:TagOpenIDConnectProvider
iam:GetRolePolicy
iam:DeletePolicyVersion
iam:TagInstanceProfile
iam:ListEntitiesForPolicy
ec2:DescribeVpcs
ec2:DescribeSubnets
ec2:DescribeVpcAttribute
ec2:CreateVpcEndpoint
ec2:DescribePrefixLists
ec2:DeleteVpcEndpoints
ec2:CreateLaunchTemplate
ec2:GetLaunchTemplateData
ec2:DescribeLaunchTemplates
```

```
ec2:DescribeLaunchTemplateVersions
ec2:ModifyLaunchTemplate
ec2:DeleteLaunchTemplate
ec2:DeleteLaunchTemplateVersions
ec2:CreateLaunchTemplateVersion
ssm:ListCommands
s3:ListBucket
s3:GetObject
s3:PutObject
s3:DeleteObject
s3:CreateBucket
s3:PutBucketPolicy
s3:PutBucketAcl
s3:PutBucketLifecycleConfiguration
```

Chapter 4

Prerequisites for Terraform

This chapter includes the following topics:

- Terraform Management Server requirements

Terraform Management Server requirements

Terraform Management Server 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 operating system.
  - 2 CPUs
  - 8 GB memory
  - > = 64 GB space on `/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 22.
  - Terraform version > = Latest version
  - 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 Amazon documentation for more details.
  - Helm package manager

  - AWS CLI version > = Latest version
  - BASH > = 5.0.17
  - Install ssm-agent - For security purpose of Cloud Strike purpose or create and instance on cloud. Install SSM Agent on RHEL 8.x and 9.x
  - Linux utilities like GREP, AWK, tr, PING, ENVSUBST, TAR, JQ, SED, and CUT
- Ensure you have enough space using the command:

  ```
  ~$ df -h
  ```
- Outbound internet access is required from Terraform Management Server to communicate with resources, services, and the servers.
- Copy the Veritas binary file bundle (NetBackup tar of Kubernetes file) and Terraform script bundle from the Veritas Download center and copy them 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 script

This chapter includes the following topics:

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

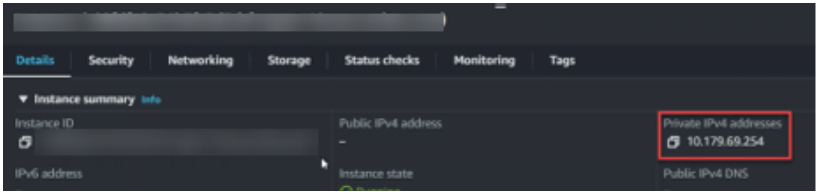
Creating and configuring Terraform Management Server

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.

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 that Docker is installed correctly.

```
docker run hello-world
```

3 Install Terraform package

a. Install the unzip using the command:`sudo apt-get install unzip`

b. Confirm the latest version on the Terraform website: `https://www.terraform.io/downloads.html`

c. Download the latest version of the Terraform. Substitute the new version number if needed:

```
wget
https://releases.hashicorp.com/terraform/<latest_version>/terraform_<latest_version>_linux_amd64.zip
```

d. Extract the downloaded file archive:

```
unzip terraform_<latest_version>_linux_amd64.zip
```

e. Move the executable into a directory using the command: `sudo mv terraform /usr/local/bin/`

f. Execute the Terraform using the command: `terraform --version`

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
```

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 AWS command line interface**

a. Download the latest version of AWS CLI bundle

```
curl -sSL
https://awscli.amazonaws.com/awscli-exe-linux-x86_64-<latest_version>.zip
-o awscli-exe-linux-x86_64-<latest_version>.zip
```

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

```
unzip awscli-exe-linux-x86_64-<latest_version>.zip
```

c. Execute the installation script:

```
./aws/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:

- Ubuntu: `/home/ubuntu`
- RHEL: `/home/ec2-user`

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 AWS” on page 39.

Stages of deploying Terraform scripts on AWS

There are 3 stages to AWS 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 Elastic Kubernetes Service (EKS) cluster.
   - Creates node groups.
   - Creates IAM role for EKS clusters and nodes if not using the existing one.
2. **Addons stage**
   - Creates IAM roles with required permissions needed for the addons.
   - Installs EBS driver, EFS driver and load balancer controller.
   - Installs Cert Manager
   - Installs Cluster Autoscaler
   - Creates EFS file systems.
   - Installs trust-manager
3. **Deployment stage**
   - Loads the Cloud Scale container images to local repository.
   - Tags and push the container images and Helm chart to ECR.
   - Deploys Cloud Scale Technology using Helm chart.

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

The actual installation of Cloud Scale Technology are provided in the section See "Installation instructions for deploying the Cloud Scale Technology on AWS" on page 39..

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 to the `sample.tfvars` file present in the base directory which has a format for passing the input parameters.

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

Table 5-1 Parameters for base stage

Parameters	Description
Required variables for base stage	
cloudscale_instance_id	A unique identifier used in tags and names to identify the cloud scale resources specific to this deployment. The cloudscale_instance_id should be between 3 to 24 characters long, can contain only lowercase letters, numbers, and hyphens. It must start with an alphabet.
region	The AWS region to provision the cloud scale resources.
vpc_id	The VPC ID of an existing VPC to provision the cloud scale resources.
eks_nodes_zone_01_subnet_id	The subnet ID of an existing subnet in the first availability zone. This subnet is where cloud scale resources will be provisioned. It is also used to create node groups.
eks_nodes_zone_02_subnet_id	The subnet ID of an existing subnet in the second availability zone.
eks_security_group_ids	List of security group ids to associate with the cloud scale cluster. This security group enables EKS cluster to access other resources.
Optional ebs private endpoint create parameters	

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

Parameters	Description
ebs_endpoint_create	Default value is false. Specifies whether to create EBS - VPC endpoint with EBS service configuration.
Default variables required for base stage	
kubernetes_version	The version of the Kubernetes to provision the cloud scale EKS cluster. The supported Kubernetes cluster version by default is 1.33.
use_existing_iam	Option to use existing IAM role or provision new IAM role. If set to `false`, provisioning will automatically create the IAM role required for the EKS cluster and node groups. If set to `true`, provide the `iam_cluster_role_name` value. Default is set to `false`. Refer to See "Permissions attached to iam_custer_role" on page 32. in case if you the Terraform to create an new iam_cluster_role.
iam_cluster_role_name	Must be set if the `use_existing_iam` parameter is set to `true`. This value is the IAM role Name of an existing IAM role that allows the Kubernetes control plane to manage AWS resources. The property cannot be changed after the cluster is created.
eks_enable_private_access	To verify whether the Amazon EKS private API server endpoint is enabled. Default value is `true`.
eks_enable_public_access	To verify whether the Amazon EKS public API server endpoint is enabled. Default value is `false`.

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

Parameters	Description

node_group_scaling_primary_pool	Scaling configuration block for the primary pool nodes. See the default value for example.
	`desired_size`: Desired number of worker nodes.
	`max_size`: Maximum number of worker nodes.
	`min_size`: Minimum number of worker nodes.
	The default values for primary pool are:
	`{"desired_size": 1,`
	`"max_size": 2,`
	`"min_size": 1}`

node_group_scaling_media_pool	Scaling configuration block for the media pool nodes. See the default value for example.
	`desired_size`: Desired number of worker nodes.
	`max_size`: Maximum number of worker nodes.
	`min_size`: Minimum number of worker nodes.
	The default values for media pool are:
	`{"desired_size": 1,`
	`"max_size": 1,`
	`"min_size": 1}`

node_group_scaling_storage_pool	Scaling configuration block for the storage pool nodes. See the default value for example.
	`desired_size`: Desired number of worker nodes.
	`max_size`: Maximum number of worker nodes.
	`min_size`: Minimum number of worker nodes.
	The default values for storage pool are:
	`{"desired_size": 1,`
	`"max_size": 1,`
	`"min_size": 1}`

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

Parameters	Description

node_group_scaling_data_plane_pool	Scaling configuration block for the data plane pool nodes. See the default value for example.
	`desired_size`: Desired number of worker nodes.
	`max_size`: Maximum number of worker nodes.
	`min_size`: Minimum number of worker nodes.
	The default values for data plane pool are:
	`{"desired_size": 1,`
	`"max_size": 1,`
	`"min_size": 1}`

node_disk_size_primary_pool	Disk size in GiB for worker nodes in the primary pool. The default disk size is 200.
node_instance_types_primary_pool	List of instance types associated with the nodes in the primary pool. Set the default value as `r5.xlarge`.
node_disk_size_media_pool	Disk size in GiB for worker nodes in the media pool. The default disk size is 100.
node_instance_types_media_pool	List of instance types associated with the nodes in the media pool. Set the default value as `t3.xlarge`.
node_disk_size_storage_pool	Disk size in GiB for worker nodes in the storage pool. The default disk size is 100.
node_instance_types_storage_pool	List of instance types associated with the nodes in the storage pool. Set the default value as `t3.xlarge`.
node_disk_size_data_plane_pool	Disk size in GiB for worker nodes in the data plane pool. The default disk size is 100.
node_instance_types_data_plane_pool	List of instance types associated with the nodes in the data plane pool. Set the default value as `t3.xlarge`.
efs_throughput_mode	A throughput mode for the file system. The values that are allowed are - 'bursting' and 'provisioned'. The default throughput mode is ' bursting'.
efs_provisioned_throughput_in_mibps	This is required only if the EFS throughput mode is 'provisioned'. The default value is 256.
Optional fields for encryption using CMK	

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

Parameters	Description
arn_of_cmk_for_efs	ARN of CMK for encrypting the EFS. This is an optional field. If not provided EFS will be encrypted with AWS managed key.
arn_of_cmk_for_ebs	ARN of CMK for encrypting EBS volumes. This is an optional field.
arn_of_cmk_for_ecr	ARN of CMK for encrypting ECR. This is an optional field
Optional Postgres db parameters. These parameters are required to be set only if the 'db_create' is set to 'true'.	
db_create	The default value is false. Specifies whether to create RDS PostgreSQL database
db_use_rds_proxy	The default value is true. Specifies whether to use RDS Proxy. 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_instance_class	db.t3.medium The instance type of the RDS instance. This parameter is optional and only required if db_create is set to true.
db_nodes_zone_01_subnet_id	The subnet ID of an existing subnet in the first availability zone. This subnet is where PostgreSQL resources will be provisioned to use. This parameter is optional and only required if db_create is set to true.
db_nodes_zone_02_subnet_id	The subnet id of an existing subnet in the second availability zone. This subnet is where PostgreSQL resources will be provisioned to use. This parameter is optional and only required if db_create is set to true.
db_allocated_storage	The default value is 30. Specifies the value for Storage Autoscaling. This parameter is optional and only required if db_create is set to true.
db_max_allocated_storage	The default value is 100. The allocated storage is in GB. This parameter is optional and is only required if `db_create` is set to `true`.

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

Parameters	Description
db_maintenance_window	Mon:00:00-Mon:03:00 The window to perform maintenance in. Syntax: 'ddd:hh24:mi-ddd:hh24:mi'. This parameter is optional and only required if db_create is set to true.
db_skip_final_snapshot	The default value is true. Determines whether a final database snapshot is created before the database instance is deleted. If true is specified, no database snapshot is created. If false is specified, a database snapshot is created before the database instance is deleted. RSD snapshots may incur additional cost. This parameter is optional and only required if db_create is set to true.
db_enhanced_monitoring_interval	The default value is 0. The interval, in seconds, between points when Enhanced Monitoring metrics are collected for the DB instance. To disable collecting Enhanced Monitoring metrics, specify 0. The default is 0. Valid Values: 0, 1, 5, 10, 15, 30, 60. This parameter is optional and only required if db_create is set to true.

Permissions attached to iam_custer_role

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

```
KMS
Protection of RDS resources
Recovery of RDS resources
Backup of EC2 resources
Recovery of EC2 resources
Backup from snapshot
Restore from backup copy
Identity management and authorization
Provider managed consistent snapshots
Permissions on workload VM
EKS
High availability
Deployment
```

Refer to the section *AWS permissions required by NetBackup Snapshot Manager* from the guide *AWS permissions required by NetBackup Snapshot Manager* 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.

Note: Refer the `sample.tfvars` file present in the deployment directory which has a format for passing the input parameters.

Table 5-2 Parameters for deployment stage

Parameters	Description
tar_file_location	Cloud Scale Technology tar location.
tar_file_name	Name of the Cloud Scale Technology tar.
load_balancer_subnet	Provide load balancer subnet id. The subnet ID should be from where the load balancer service would provision IP address to the cloud scale services. `EKS_NODES_ZONE_01_SUBNET_ID and LOAD_BALANCER_SUBNET` must be in the same availability zone.
load_balancer_security_group_id	The values allowed are: ID of an already created security group or **eks-managed**. If set to **eks-managed** load balancer uses the AWS managed security group. **Note:** Ensure that all NetBackup load balancer ports exist. For a detail list of all the ports, refer to the 'Default ports used in the Load Balancer service' section of the *Cohesity Cloud Scale Technology Manual Deployment Guide for Kubernetes Clusters*.

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

Parameters	Description
media_server_replica_count	Provide the number of replicas for the 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 value is 1.
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 value is 1.
primary_server_ip_fqdn_mapping	Provide IP hostname mapping for NetBackup primary server. The hostname 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 mappings 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 hostname of NetBackup mappings 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 lower case, an upper case, and a special character (@$%!*?&.).

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

Parameters	Description
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 (@$%!*?&.).
key_protection_key_id	Provide the Key Protection Key ID. The `key_protection_key_id` must be of 1-32 characters long, must contain only lowercase alphanumeric characters, hyphens, and underscores.
key_protection_key_passphrase	Provide the Key Protection Key Passphrase. The `key_protection_key_passphrase` must be at least 12 characters long and must have at least a number, a lower case, an upper case and a special character (@$%!*?&.).
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 (@$%!*?&.).

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

Parameters	Description
storage_server_kms_key_secret_username	Provide the KMS key username for storage erver. The `storage_server_kms_key_secret_username` must be of 1-32 characters long, must contain only lowercase alphanumeric characters, hyphens or underscores.
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 ('*', '\', '/', '^', '(', ')',"", '<', '>', '&', '[', ']', '%', '@', '#).
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.

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

Parameters	Description
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 for Fluentbit log collector. It must be at least 100 Gi.
log_collection_namespaces	Provide the namespaces for the logging daemonsets to collect pod `stdout` logs.
enable_sidecars_logging	Enable or disable the sidecar logging. Default value is *true*. For example, `enable_sidecars_logging = true`
Optional timezone input for NB servers	
global_timezone	Provide value like global_timezone="/usr/share/zoneinfo/Asia/Kolkata. Keep the timezone as `blank` value.
Optional fields	
snapshot_manager_vx_http_proxy	Provide the value to be used as the HTTP proxy for all connections for Snapshot Manager.
snapshot_manager_vx_https_proxy	Provide the value to be used as the HTTPS proxy for all connections for Snapshot Manager.
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. While providing multiple values please escape commas and dots in urls if any with `\\ e.g "localhost\\,mycompany\\.com\\,1.2.3.4"`
dr_info_secret_name	Name of secret to pass the DR information.
dr_info_secret_passphrase	Details of DR passphrase.

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

Parameters	Description
dr_info_secret_email_address	Details of DR email address.
email_server_configmap_name	Name of the config map that contains all the required information to configured email server.
email_server_configmap_details	Details required to configure email server. Provide all the required fields comma separated. Escape commas with \\ while providing values. For example: email_server_configmap_details="smtp_smtp.server.com:port\\,ssl-verify:no\\,smtp_use_starttls"
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.	

PaaS based PostgreSQL deployment (DBaaS) on AWS

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 cloud provider, it improves the resiliency, allows the database to be scaled up as needed, and reduces the maintenance requirements for NetBackup's database services in EKS.

DBaaS deployment is selected by setting `db_create = false` in the Terraform deployment input file at the base step. For more details on the stages of deployment,

refer to the See "Stages of deploying Terraform scripts on AWS" on page 26. section.

See "Troubleshooting issues" on page 46.

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

See "Changing database server password in PostgreSQL (AWS)" on page 41.

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 AWS

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 S3 bucket after completing the deployment successfully.

Before proceeding to execute the deployment 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 25.

Deploying the Cloud Scale Technology on AWS

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 aws -t preInfra
   ```

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

   - Log in to AWS account and authenticate your access. Refer to Configure the AWS CLI for more details.
   - Change the directory using the below command:

     ```
     cd aws/base
     ```

- Create new `.tfvars` based on 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
  ```

  See "Parameters for base stage" on page 27.

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

- Change the directory from base to addons using the below command:

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

Note: No input .tfvar file is required at this step.

---

4 Before executing this step, modify the values from the `deployment.tfvars` file and then execute the below command.

Execute the PreFlight script using the following command:

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

You will have to provide the `Base input .tfvars file` and `Deployment input .tfvars file` path for validation.

5 Execute the steps mentioned in the following section to reset the database password:

See "Changing database server password in PostgreSQL (AWS)" on page 41.

6 Execute the **Deployment** step.

- Change the directory from addons to deployment using the below command:

  ```
  cd aws/deployment
  ```

- Create new `.tfvars` file based on 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
```

See "Parameters for deployment stage" on page 33.

Changing database server password in PostgreSQL (AWS)

Using the LAMBDA_ARN function, you can change the database password. This can be obtained from the lambda function on the AWS console. The ARN stands for Amazon Resource Name.

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 (#)

To change the database server password

1. To change the database server password, use the lambda function LAMBDA_ARN.

   ```
   $ aws lambda invoke --function-name $LAMBDA_ARN \
   --cli-binary-format raw-in-base64-out --payload
   '{"password":"$NEW_PASSWORD"}' \ response_file
   ```

   Note: NEW_PASSWORD is the new password to be used.

   ---

2. To obtain the POSTGRESQL_ID (database identifier) of your RDS Postgres database from the RDS database page of the AWS console, use the following command and wait for the database to be available.

   ```
   $ aws rds wait db-instance-available --db-instance-identifier
   $POSTGRESQL_ID
   ```

3. Restart the primary pod using the command:

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

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

In the above command:

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

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

Chapter 6

Accessing the Cloud Scale Technology 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 the AWS 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 the Cloud Scale Technology Web UI using the **https://<primaryServer>/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 AWS.

Table 7-1 List of troubleshooting issues while deployment

Sr.No.	Issue	Description
1	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.

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

Sr.No.	Issue	Description
2	Even after executing the destroy command, if there are any folders that are not removed from the environment.	**Resolution:** On the AWS console, do the following: ■ Delete EKS cluster ■ Delete IAM roles and Policies ■ Delete EFS ■ Delete Load balancers ■ Delete Volumes/Snapshots ■ Delete RDS Target Groups ■ Delete OIDC Provider If you are using the PostgreSQL database server, do the following: ■ Delete Postgres RDS instance ■ Delete Lambda function ■ Delete Secret Manager 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

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

Sr.No.	Issue	Description

3	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.

Cleanup steps

These steps are to be followed if you wish to cleanup the resources 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.

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 aws/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.2 from the See “Troubleshooting issues” on page 46.

---

2. **Addons**

```
cd aws/addons
```

```
terraform destroy
```

3. **Base**

```
cd aws/base
```

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