

Automated IaC Deployment of Hitachi VSP One SDS Cloud on AWS for ROSA PostgreSQL Workloads

Reference Architecture Guide

© 2025 Hitachi Vantara LLC. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including copying and recording, or stored in a database or retrieval system for commercial purposes without the express written permission of Hitachi, Ltd., Hitachi Vantara, Ltd., or Hitachi Vantara LLC (collectively "Hitachi"). Licensee may make copies of the Materials provided that any such copy is: (i) created as an essential step in utilization of the Software as licensed and is used in no other manner; or (ii) used for archival purposes. Licensee may not make any other copies of the Materials. "Materials" mean text, data, photographs, graphics, audio, video and documents.

Hitachi reserves the right to make changes to this Material at any time without notice and assumes no responsibility for its use. The Materials contain the most current information available at the time of publication.

Some of the features described in the Materials might not be currently available. Refer to the most recent product announcement for information about feature and product availability, or contact Hitachi Vantara LLC at https://support.hitachivantara.com/en_us/contact-us.html.

Notice: Hitachi products and services can be ordered only under the terms and conditions of the applicable Hitachi agreements. The use of Hitachi products is governed by the terms of your agreements with Hitachi Vantara LLC.

By using this software, you agree that you are responsible for:

1. Acquiring the relevant consents as may be required under local privacy laws or otherwise from authorized employees and other individuals; and
2. Verifying that your data continues to be held, retrieved, deleted, or otherwise processed in accordance with relevant laws.

Notice on Export Controls. The technical data and technology inherent in this Document may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Reader agrees to comply strictly with all such regulations and acknowledges that Reader has the responsibility to obtain licenses to export, re-export, or import the Document and any Compliant Products.

Hitachi and Lumada are trademarks or registered trademarks of Hitachi, Ltd., in the United States and other countries.

AIX, DB2, DS6000, DS8000, Enterprise Storage Server, eServer, FICON, FlashCopy, GDPS, HyperSwap, IBM, IntelliMagic, IntelliMagic Vision, OS/390, PowerHA, PowerPC, S/390, System z9, System z10, Tivoli, z/OS, z9, z10, z13, z14, z15, z16, z17, z/VM, and z/VSE are registered trademarks or trademarks of International Business Machines Corporation.

Active Directory, ActiveX, Bing, Excel, Hyper-V, Internet Explorer, the Internet Explorer logo, Microsoft, Microsoft Edge, the Microsoft corporate logo, the Microsoft Edge logo, MS-DOS, Outlook, PowerPoint, SharePoint, Silverlight, SmartScreen, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, the Windows logo, Windows Azure, Windows PowerShell, Windows Server, the Windows start button, and Windows Vista are registered trademarks or trademarks of Microsoft Corporation. Microsoft product screen shots are reprinted with permission from Microsoft Corporation.

All other trademarks, service marks, and company names in this document or website are properties of their respective owners.

The open source content used in Hitachi Vantara products may be found within the Product documentation or you may request a copy of such information (including source code and/or modifications to the extent the license for any open source requires Hitachi make it available) by sending an email to OSS_licensing@hitachivantara.com.

Feedback

Hitachi Vantara welcomes your feedback. Please share your thoughts by sending an email message to Docs-Feedback@hitachivantara.com. To assist the routing of this message, use the paper number in the subject and the title of this white paper in the text.

Thank you!

Revision history

Changes	Date
Initial release	December 2025

Contents

Reference Architecture Guide.....	4
Solution overview.....	5
VSP One SDS Cloud – Automated cloud deployment and operations.....	6
End-to-end automated deployment workflow for VSP One SDS Cloud, ROSA, and PostgreSQL persistent storage integration.....	7
Solution benefits.....	8
Solution components.....	10
Infrastructure.....	10
Software.....	14
Deployment methods.....	16
Configuration procedures.....	17
Enable the ROSA service.....	19
ROSA OIDC workflow (when not creating an OIDC with Terraform).....	20
Prepare the Terraform/Ansible Dev VM.....	21
Configure the <code>terraform.tfvars</code> file.....	23
Solution validation.....	31
References.....	33

Reference Architecture Guide

Executive summary

VSP One SDS Cloud for ROSA on AWS is a co-engineered, cloud-native solution that delivers a fully automated, turnkey OpenShift experience in Amazon Web Services. Designed for modern enterprise workloads, the solution unifies AWS elasticity, Red Hat OpenShift on AWS (ROSA) orchestration, and VSP One SDS Cloud's high-performance, resilient block storage.

Built on a shared-nothing architecture, the solution ensures strict data localization, fault isolation, and independent scalability across compute, storage, and Kubernetes layers—making it ideal for regulated and data-sensitive workloads.

A key differentiator of this solution is its Terraform-driven deployment automation, enabling rapid, consistent, and repeatable provisioning of the full stack — from AWS infrastructure to SDS Cloud configuration and ROSA integration — with minimal operational overhead.

This design and configuration guide helps you deploy and operate your environment, integrating the following:

- Hitachi VSP One SDS Cloud (enterprise storage in the public cloud)
- Red Hat OpenShift Service on AWS (ROSA)
- Automated end-to-end provisioning using Terraform
- AWS-native compute, networking, and load balancing services
- Sample workload with PostgreSQL pods

Enterprises accelerating toward cloud-native architectures face growing complexity in deploying and managing Kubernetes, storage, data services, and cloud infrastructure in a consistent manner. While cloud adoption is widespread, mission-critical and data-intensive applications still demand predictable I/O performance, enterprise-grade resiliency, and operational consistency. Manual provisioning of these components is time-consuming, error-prone, and requires deep expertise across multiple domains.

Hitachi Vantara, Red Hat, and AWS jointly address these challenges by integrating VSP One SDS Cloud with ROSA into a unified DevOps-ready deployment model. The shared-nothing architecture of VSP One SDS Cloud enables localized data placement within AWS regions and availability zones, ensuring that data remains within defined geographic and regulatory boundaries while eliminating cross-node dependencies.

The Terraform automation framework significantly reduces time-to-value (TTV) and total cost of ownership (TCO) by enabling standardized infrastructure-as-code provisioning aligned with AWS and OpenShift best practices. This includes automatic creation of ROSA clusters, VSP One SDS Cloud deployment, storage pool creation, iSCSI configurations, multipath tuning, Secret/StorageClass deployment, and seamless PV provisioning for PostgreSQL and other stateful workloads.

This co-engineered solution supports the entire spectrum of OpenShift workloads — from stateless microservices to stateful enterprise databases — through a cloud-native storage architecture designed for consistency and reliability. Delivered as a pre-validated, single-stack configuration, it removes the need for multi-vendor integration and provides a streamlined operational experience. Customers gain full-stack lifecycle management, automated compliance, and integrated monitoring, backed by Hitachi's proven data management technologies.

With VSP One SDS Cloud powering persistent storage for ROSA, organizations can deploy and scale PostgreSQL pods and other stateful applications with confidence, achieving high availability, optimized capacity utilization, and predictable performance. Terraform provisioning further ensures deployments are repeatable across environments, accelerating DevOps workflows and reducing operational risk.

Supported under Hitachi Global Support and Services with a unified support model, customers receive full-stack assistance — covering AWS infrastructure, ROSA platform, VSP One SDS Cloud storage, Terraform automation, and data services — ensuring rapid issue resolution, operational continuity, and enterprise-class reliability for mission-critical cloud-native applications.

Solution overview

This solution provides a fully automated, end-to-end deployment framework for VSP One SDS Cloud on AWS using Terraform and Ansible, followed by seamless integration with Red Hat OpenShift Service on AWS (ROSA) to enable enterprise-grade persistent storage for VM and container workloads.

As an outcome, this solution delivers a production-ready ROSA environment on AWS with enterprise-grade, software-defined block storage that is fully automated, scalable, and cloud-native. Customers gain a validated deployment model where compute, networking, OpenShift, and storage are provisioned and integrated consistently using Infrastructure-as-Code, reducing operational risk and eliminating manual configuration errors.

From an AWS infrastructure perspective, VSP One SDS Cloud runs entirely on standard AWS compute, networking, and storage services, aligning with native AWS operational models. Storage services are deployed alongside ROSA worker nodes within customer VPCs and subnets, leveraging AWS elasticity while enabling independent scaling of storage performance and capacity. This architecture allows organizations to consume enterprise SDS capabilities in AWS without proprietary hardware dependencies, making it ideal for hybrid, cloud-first, and cloud-native application deployments.

A sample PostgreSQL application is deployed on the ROSA cluster to demonstrate dynamic PVC provisioning and iSCSI-based storage attachment using the Hitachi Storage Plug-in for Containers (HSPC) CSI driver.

ROSA pricing

Red Hat OpenShift Service on AWS (ROSA) provides a managed OpenShift experience integrated with AWS. The total cost of ROSA consists of two components: ROSA service fees and AWS infrastructure fees.

At the time of publication, ROSA service fees accrue on demand, at an hourly rate of \$0.171 per 4 vCPU used by worker nodes. In addition to the worker node service fee, ROSA with hosted control planes (HCP) clusters incur a \$0.25 per hour cluster fee. ROSA service fee contracts for 1-year and 3-year terms* offer savings between 33% and 55% off the on-demand service fees for worker nodes. ROSA service fees are uniform across all supported AWS standard Regions.

ROSA offers 1-year and 3-year service fee contracts that you can purchase for savings on the on-demand service fees for worker nodes. For more information, see [Purchasing a ROSA contract](#).

ROSA Worker Node Service Fees Yearly Cost	On-demand	With a 1-year contract	With a 3-year contract
Worker Node Fee Per 4 vCPU/Year	\$1500	\$1000	\$667

AWS infrastructure fees

AWS infrastructure fees apply to the underlying worker nodes, infrastructure nodes, control plane nodes, storage, and network resources hosted on AWS global infrastructure. AWS infrastructure fees vary by AWS Region.

VSP One SDS Cloud – Automated cloud deployment and operations

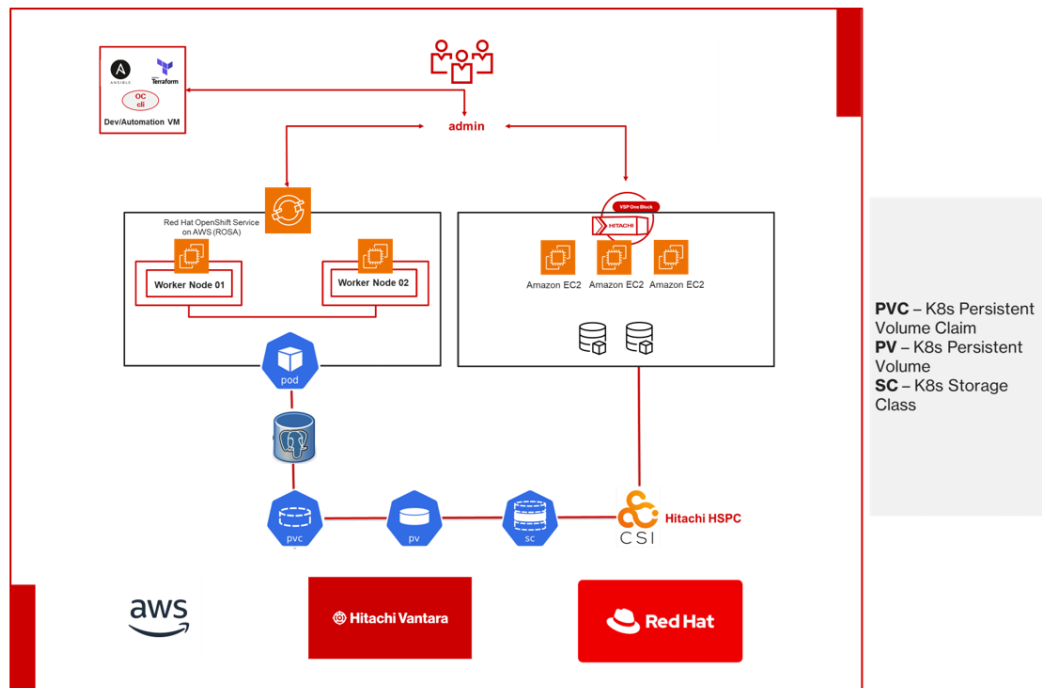
VSP One SDS Cloud on AWS enables full-stack cloud deployments in hours rather than weeks. Its automation framework accelerates time-to-value while providing consistent, repeatable deployment processes across multiple AWS regions and environments.

After deployment, the co-engineered architecture delivers policy-driven, end-to-end data services and lifecycle automation, including the following:

- Automated provisioning and scaling of ROSA clusters using Terraform
- Seamless deployment of VSP One SDS Cloud clusters, storage pools, and iSCSI access
- Unified automation for persistent storage and PostgreSQL backing volumes
- Consistent application and data management across ROSA and AWS
- Scalable platform operations without increasing operational burden
- Declarative IaC-driven lifecycle management for storage, compute, and networking
- Integrated monitoring, observability, and optimized data services

This unified cloud-native platform enables organizations to simplify Kubernetes operations, accelerate application delivery, and confidently run mission-critical stateful workloads on AWS using enterprise-grade Hitachi storage.

The following illustration shows how VSP One SDS Cloud integrates with Red Hat OpenShift Service on AWS to form a unified cloud-native data platform.



End-to-end automated deployment workflow for VSP One SDS Cloud, ROSA, and PostgreSQL persistent storage integration

The complete infrastructure can be automated, and provisions the following:

- AWS VPCs, subnets, IAM, and EC2 instances
- VSP One SDS Cloud infrastructure
- ROSA cluster resources

1. Provision VSP One SDS Cloud Cluster

SDS nodes are deployed as EC2 instances in AWS, including disks dedicated to data pools and metadata.

2. Cluster Create

Ansible or Terraform-provisioned scripts initialize the SDS nodes and form a fully functional VSP One SDS Cloud cluster.

3. Python Script to Reset Default Password

A post-deployment step resets the SDS admin password to the customer-defined secure value and prepares the SDS API for use by CSI driver authentication.

4. Ansible – Add Disks to Storage Pool

Additional EBS disks are discovered and added to the correct storage pool (performance/capacity) for application use.

5. Volume Created

SDS cluster becomes capable of provisioning block storage volumes requested by Kubernetes PVCs.

6. Provision the ROSA Cluster

Terraform provisions (or connects to) a ROSA Classic or ROSA HCP cluster.

7. CSI Driver Installation

The Hitachi Storage Plug-in for Containers (HSPC) operator and iSCSI daemonsets are deployed on ROSA.

8. StorageClass Creation

Kubernetes StorageClass is created and mapped to the VSP One SDS Cloud backend using the generated secret and API endpoint.

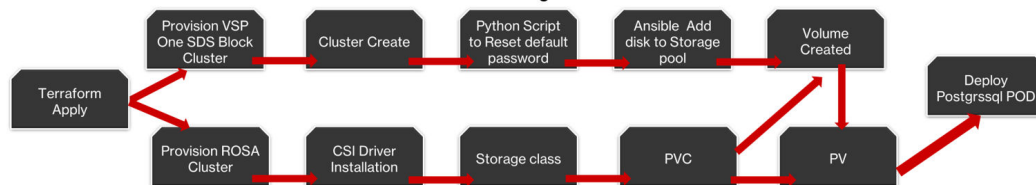
9. PVC → PV Binding

- A PersistentVolumeClaim triggers the CSI driver to create a volume on VSP One SDS Cloud.
- SDS creates a LUN
- CSI maps it to ROSA worker nodes
- A PersistentVolume object is created

10. Deploy PostgreSQL Pod

The application pod mounts the SDS-backed block device and stores its data on the VSP One SDS Cloud cluster.

The following illustration shows the workflow.



Solution benefits

VSP One SDS Cloud for ROSA on AWS delivers the following advancements and benefits:

- Lower total cost of ownership (TCO) through simplified operations, reduced integration overhead, and consistent Terraform-based lifecycle automation.
- Strengthened data security with VSP One SDS Cloud's enterprise-grade data services, multipath reliability, and encrypted data transport, combined with OpenShift and AWS security frameworks.
- Enhanced data protection via VSP One SDS Cloud snapshots, replication policies, and AWS regional resiliency options, enabling rapid recovery for stateful workloads such as containerized PostgreSQL.

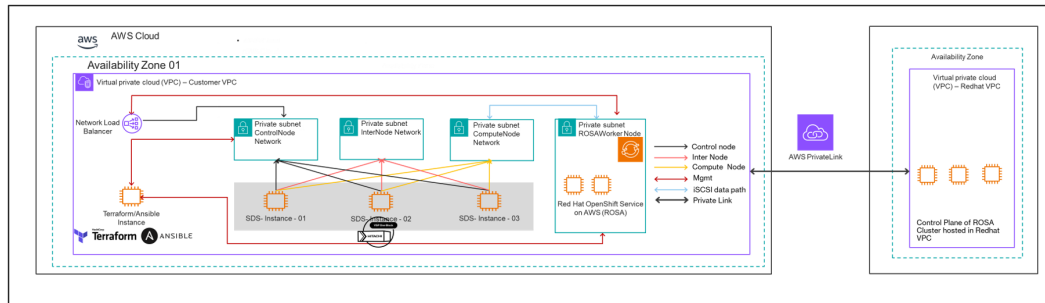
- High availability and workload resilience through VSP One SDS Cloud's distributed architecture, ROSA's managed Kubernetes control plane, and AWS Availability Zone redundancy.
- Multi-site and hybrid architecture support using AWS multi-region deployment patterns combined with VSP One SDS Cloud replication capabilities for extended data durability.
- Ransomware recovery readiness with VSP One SDS Cloud immutable snapshots, point-in-time recovery, and secure data isolation to protect Kubernetes stateful workloads.
- Future-proof platform for all cloud-native workloads, including modern application stacks, AI/ML pipelines, analytics workloads, and high-performance databases deployed inside ROSA.
- Support for persistent workloads such as PostgreSQL running as OpenShift pods using the Hitachi CSI driver for dynamic, policy-driven storage provisioning.
- Unified observability and integrated automation across AWS, ROSA, and VSP One SDS Cloud via Terraform modules and VSP One SDS Cloud APIs.
- Optimized performance and capacity utilization through VSP One SDS Cloud's highly efficient storage engine, preventing over-provisioning and reducing operational costs.
- Enterprise-class reliability delivered through Hitachi's proven storage data path, advanced multipathing, and AWS regional infrastructure.
- Scalability on demand using AWS compute elasticity combined with dynamically provisioned VSP One SDS Cloud storage pools.
- Consistent, policy-driven operations across environments with repeatable Terraform workflows and standardized OpenShift storage classes.

VSP One SDS Cloud with Terraform-based automation delivers end-to-end operational management with the following benefits:

- Rapid environment provisioning: Easily spin up and tear down isolated Test, Development, and QA environments using the same validated Infrastructure-as-Code templates, ensuring consistency across the application lifecycle.
- Unified provisioning of ROSA nodes, VSP One SDS Cloud storage pools, networking components, and Kubernetes storage objects from a single Terraform framework.
- Infrastructure-wide consistency by enforcing standardized templates, versions, and configuration baselines across regions and environments.
- Centralized management and monitoring leveraging VSP One SDS Cloud APIs, OpenShift observability, AWS CloudWatch, and Terraform state tracking.
- Integrated workflows with OpenShift operators, Kubernetes storage classes, and CSI-driven provisioning for databases and persistent workloads.
- Support for external cloud-native storage and services including AWS networking, load balancers, and service meshes for advanced application designs.

Solution components

These are the key infrastructure and software components used in the Terraform- and Ansible-based deployment of VSP One SDS Cloud on AWS, integrated with ROSA and PostgreSQL.



Infrastructure

ROSA Deployment

Node Profile	Role/Layer	AWS Instance Type (from Terraform)	Storage Backend (ROSA)	Typical Use Case
ROSA control plane (Managed)	Control plane (API, etcd)	Managed by ROSA	AWS-managed	Cluster API, scheduler, etcd
ROSA Worker – General Purpose	Application/ compute nodes	m5.xlarge (your value)	VSP One SDS Cloud via CSI	Stateless apps, microservices, web workloads

VSP One SDS Cloud AWS Deployment

Component	Role	AWS Instance Type	Drives/ Capacity	Notes
VSP One SDS Cloud Storage Node	VSP One SDS Cloud cluster node	m6i.8xlarge (from your terraform.tfvars)	6 × drives, 1579 GiB (mirroring duplication)	Forms part of the 3-node VSP One SDS Cloud storage cluster

Component	Role	AWS Instance Type	Drives/ Capacity	Notes
SDS Cluster Configuration Pattern	Redundancy Mode	Mirroring Duplication (3 Nodes)	—	Provides high availability and data protection

Terraform/Ansible VM Configuration(EC2)

Parameter	Value
OS	Amazon Linux 2023.9 (20251110)
vCPU	2 vCPUs
Memory	8 GB RAM
Storage	40 GB gp3
Instance Type	EC2 – Intel Xeon E5-2686 v4
Tools Installed	Terraform, Ansible, ROSA CLI, oc, Python
Network Access	SSH via Jump Host (X.X.X.X/XX), outbound 443
Purpose	Terraform + Ansible automation for ROSA and VSP One SDS Cloud

Terraform/Ansible VM(EC2) package details

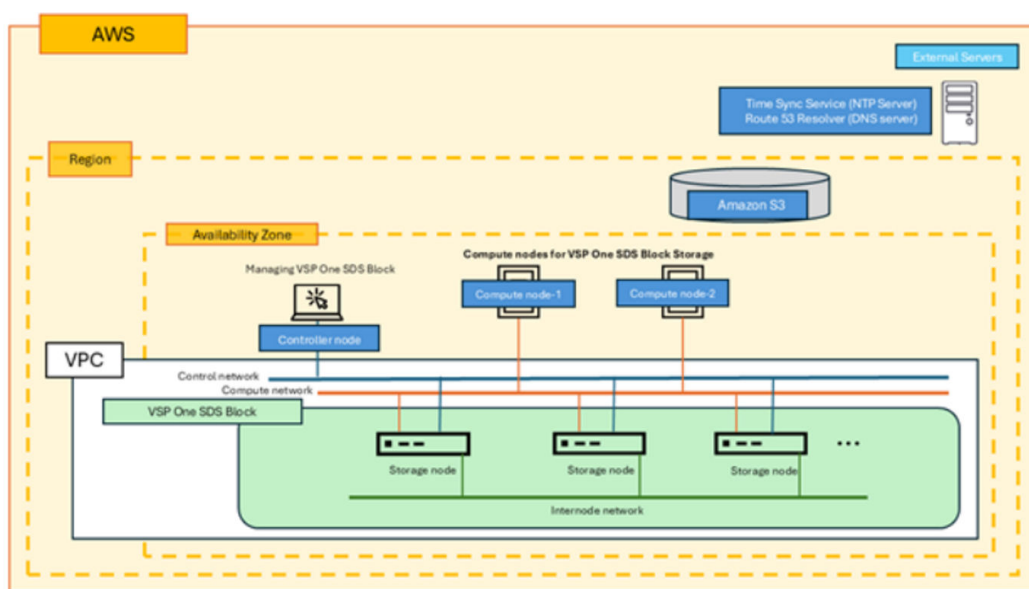
Component	Version	Function
Terraform	v1.13.5	AWS, ROSA, and VSP One SDS Cloud IaC provisioning
Ansible (Core)	2.15.3	VSP One SDS Cloud getting disk list and storage pool expansion
Python	3.9.24	Execution engine for Ansible modules
AWS CLI	v2.x (query with aws --version)	Required for Terraform/AWS operations
ROSA CLI	1.2.53	ROSA lifecycle mgmt (cluster creation, validation)
VSP One SDS Cloud	17	Offers iSCSI Persistent storage to Containerized workload in ROSA

Component	Version	Function
OpenShift CLI (oc)	4.19.13	Deployment of Kubernetes manifests in ROSA

VSP One SDS Cloud overview

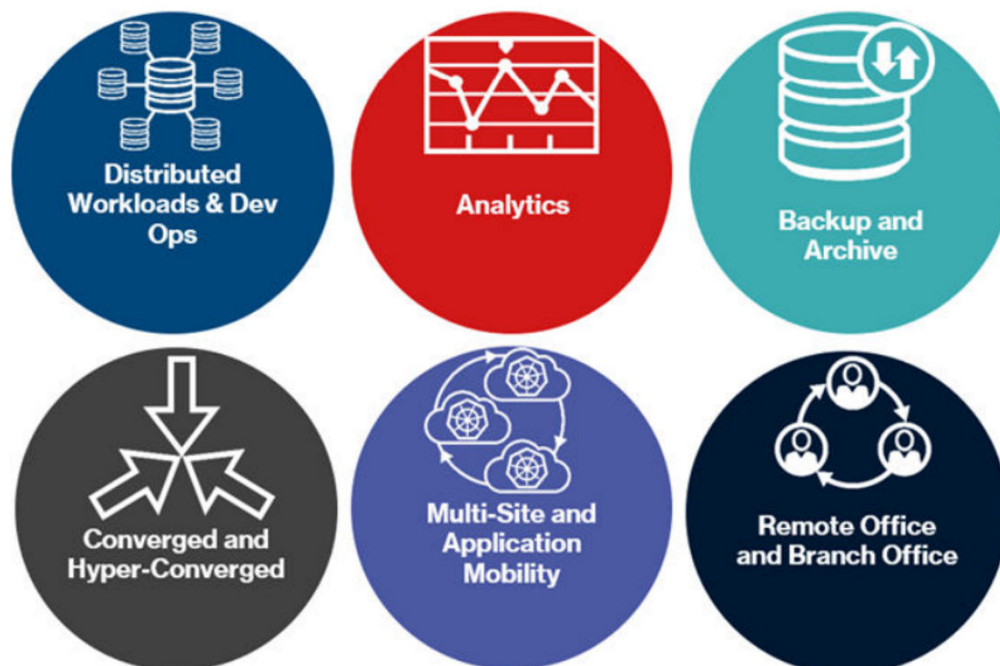
VSP One Software Defined Storage (SDS) Cloud is a storage software product that builds and sets up a virtual storage system from multiple general-purpose servers (physical or virtual). The storage system is a two-layer software-defined storage (SDS) in which storage nodes are separated from the compute nodes (disaggregated storage).

The following figure shows the configuration of the VSP One SDS Cloud solution in AWS using EC2 VMs.



Hitachi VSP One SDS Cloud delivers enterprise-grade, software-defined block storage optimized for cloud environments. By leveraging standard cloud infrastructure, organizations can deploy and scale storage independently of compute, seamlessly integrating with Kubernetes and cloud-native platforms. This approach enables efficient workload consolidation, predictable performance, and long-term scalability while preserving the flexibility and automation benefits of the cloud.

The following figure shows some of the use cases for this solution.



Red Hat OpenShift overview

Red Hat OpenShift Container Platform (OCP) provides a single platform to build, deploy, and manage applications consistently across on-premises and hybrid cloud deployments. OCP provides the control plane and data plane within the same interface. OCP provides administrator views to deploy operators, monitor container resources, manage container health, manage users, work with operators, manage pods and deployment configurations, as well as to define storage resources.

Scalability

Applications running on Red Hat OpenShift can scale to thousands of instances across hundreds of nodes in seconds.

Flexibility

Red Hat OpenShift simplifies deployment and management of a hybrid infrastructure, giving you the flexibility to have a self-managed or fully managed service, running on-premises or in cloud and hybrid environments.

Open-source standards

Red Hat OpenShift incorporates Open Container Initiative (OCI) containers and Cloud Native Computing Foundation-certified Kubernetes for container orchestration, in addition to other open-source technologies.

Container portability

Container images built on the OCI industry standard ensure portability between developer workstations and Red Hat OpenShift production environments.

Enhanced developer experience

Red Hat OpenShift offers a comprehensive set of developer tools, multilanguage support, command-line and integrated development environment (IDE) integrations. Features include continuous integration/continuous delivery (CI/CD) pipelines based on Tekton and third-party CI/CD solutions, service mesh, serverless capabilities, and monitoring and logging capabilities.

Automated installation and upgrades

Automated installation and over-the-air platform upgrades are supported in the cloud with Amazon Web Services, Google Cloud Platform, IBM Cloud, and Microsoft Azure, and on premises using vSphere, Red Hat OpenStack® Platform, Red Hat Virtualization, or bare metal. Services used from the OperatorHub can be deployed fully configured and are upgradable with one click.

Automation

Streamlined and automated container and app builds, deployments, scaling, health management, and more are included.

Edge architecture support

Red Hat OpenShift enhances support for smaller footprint topologies in edge scenarios, such as 3-node clusters, single-node OpenShift, and remote worker nodes, which better map to varying physical size, connectivity, and availability requirements of different edge sites. The edge use cases are further enhanced with support for Red Hat OpenShift clusters on ARM architecture, commonly used for low-power-consumption devices.

Multicenter management

Red Hat OpenShift with Red Hat Advanced Cluster Management for Kubernetes can easily deploy apps, manage multiple clusters, and enforce policies across clusters at scale.

Advanced security and compliance

Red Hat OpenShift offers core security capabilities such as access controls, networking, and enterprise registry with a built-in scanner. Red Hat Advanced Cluster Security for Kubernetes enhances this with security capabilities such as runtime threat detection, full lifecycle vulnerability management, and risk profiling.

Persistent storage

Red Hat OpenShift supports a broad spectrum of enterprise storage solutions, including Red Hat OpenShift Data Foundation, for running both stateful and stateless apps.

Software

HashiCorp Terraform

HashiCorp Terraform is an industry-leading Infrastructure-as-Code (IaC) automation tool used to provision and manage cloud infrastructure in a predictable, repeatable, and version-controlled manner. Terraform enables teams to define infrastructure using a declarative configuration language, allowing automated creation, modification, and teardown of resources across multiple providers, including AWS.

In this solution, Terraform is used to do the following:

- Automate deployment of AWS networking and foundational infrastructure (VPC, subnets, security groups, IAM roles).
- Provision ROSA (Red Hat OpenShift Service on AWS) clusters, including HCP components, OIDC, operator roles, and cluster configuration.
- Deploy and initialize VSP One SDS Cloud clusters, including EC2 storage nodes, storage pools, access groups, and S3 logging integrations.
- Configure OpenShift storage integration by generating Kubernetes manifests (Secrets, StorageClasses, DaemonSets).
- Enable fully automated, end-to-end environment creation with minimal manual intervention.

Terraform ensures every deployment is consistent, reproducible, secure, and scalable, making it the recommended method for production-grade implementations.

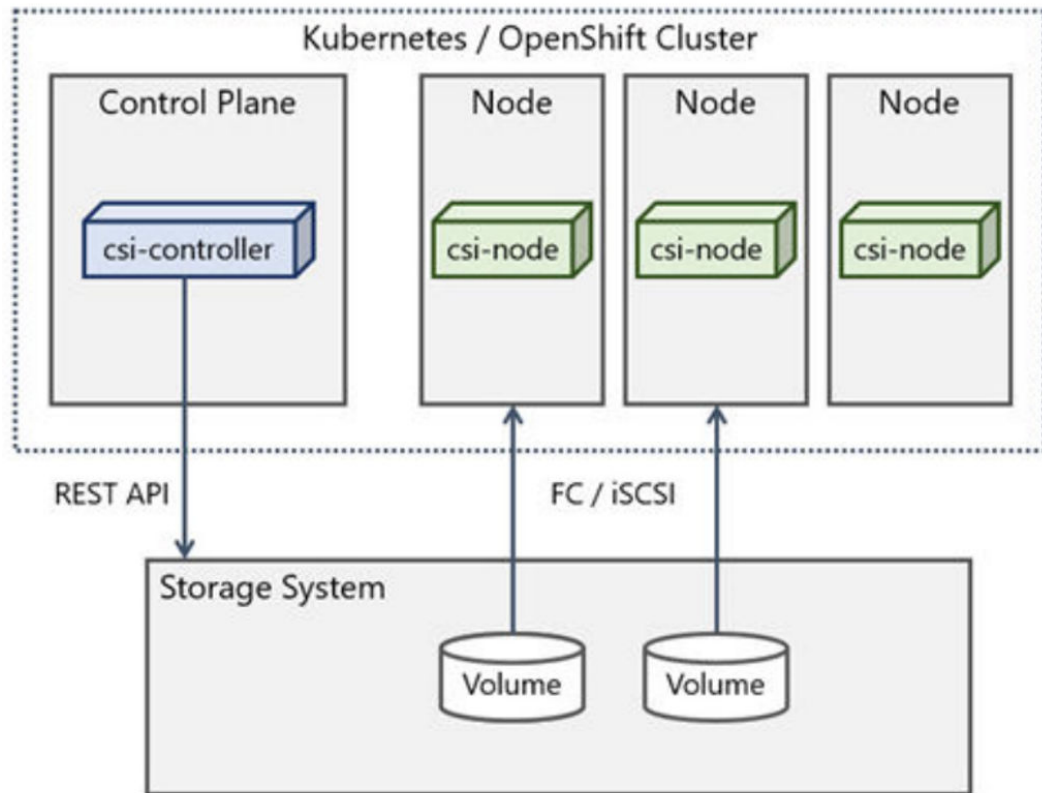
Hitachi Storage Plug-in for Containers (Hitachi CSI)

Hitachi Storage Plug-in for Containers, or Hitachi CSI, refers to the CSI plugin and associated components by Hitachi Vantara to integrate automation of VSP One storage platforms into various Kubernetes distributions such as Red Hat OpenShift/Virtualization and SUSE Rancher/Virtualization and other container orchestration environments.

The storage platforms include Hitachi VSP One Block, VSP, and VSP One SDS Cloud in Azure, AWS and GCP public clouds — The CSI enables platform operators to dynamically or statically provision and manage volumes from reliable performant Hitachi Storage for use by VM pods and container workloads.

- It includes capabilities such as snapshots, cloning, expansion, raw block mode and supports RWX, RWO etc.
- It includes capabilities to create both standard PVs but also stretched PVs for high availability across metro distances.
- It enables presentation of volumes over FC, iSCSI, NVMe-of including NVMe/TCP.
- It includes a HRPC component to automate asynchronous and synchronous replication of the volumes between clusters for migration and disaster recovery use cases.
- For monitoring and telemetry, it includes a HSPP component to expose metrics to Prometheus and surfaced by Grafana.
- A NAS CSI is also available.

Check the latest release for updated capabilities as some capabilities are not available in the Cloud option at the time of publication.



Ansible

Ansible is a powerful, agentless configuration management and automation platform used to execute operational tasks, orchestrate workflows, and apply configuration changes across multiple nodes with ease. It uses simple YAML-based playbooks, making it easy to automate repetitive tasks while maintaining accuracy and compliance.

In this solution, Ansible is used to do the following:

- Perform storage expansion tasks, such as detecting and validating newly attached EBS disks.
- Add new disks to existing VSP One SDS Cloud storage pools to increase usable capacity.

Ansible enhances operational efficiency by enabling consistent, repeatable configuration management, reducing manual effort and improving reliability across all environments.

Deployment methods

There are three different deployment options available with a decreasing level of automation depending on customer circumstances.

1. Validated Deployment: Fully automated deployment method (recommended)

The fully automated deployment method uses an end-to-end Terraform/Ansible-driven Infrastructure-as-Code (IaC) workflow to provision the complete environment required for running Red Hat OpenShift Service on AWS (ROSA) with VSP One SDS Cloud and

containerized PostgreSQL workloads. This method provides a consistent, repeatable, and error-free deployment experience, significantly accelerating time-to-value and ensuring cloud-native operational consistency across environments. With a single Terraform execution, the following components are deployed and configured automatically:

- AWS foundational infrastructure, Instances, security groups, IAM roles, and S3 logging.
- ROSA (HCP) cluster provisioning, including cluster creation, operator roles, OIDC (uses the existing one), and administrative access.
- VSP One SDS Cloud cluster deployment, including EC2 storage nodes, networking, storage pool creation, and cluster initialization.
- Integration of VSP One SDS Cloud with OpenShift, including CSI driver configuration, secrets, StorageClasses, and persistent volume provisioning.
- Enablement of PostgreSQL workloads, including dynamic PV provisioning, iSCSI multipath tuning, and node-level storage enablement.
- End-to-end automation for Day-1 setup, reducing the need for manual intervention and ensuring consistent configuration across all environments.

This fully automated approach eliminates variability between deployments, improves reliability, and allows infrastructure teams to scale environments rapidly using standardized, version-controlled Terraform modules. It is the recommended deployment method for production, multi-site, and large-scale OpenShift and VSP One SDS Cloud environments.

2. Semi-Automated Deployment (Terraform + manual enhancements)

Terraform deploys the core infrastructure and VSP One SDS Cloud, while selected post-configuration steps (such as custom ROSA tuning or PostgreSQL setup) are performed manually or using Ansible.

3. Manual Deployment (Console + CLI driven)

All AWS, ROSA, and VSP One SDS Cloud components are created manually using the AWS Console, ROSA CLI, and SDS UI. This method is only suitable for PoC or lab environments.

Configuration procedures

The following table lists VPC and subnet network architecture CIDR details (single-AZ, single-VPC deployments). Use this information when configuring this solution.

Component	Subnet Purpose	AWS Resource ID	CIDR Cloud	Description
VPC	Primary VPC for ROSA + VSP One SDS Cloud C + Terraform/ Ansible VM	vpc-xxxxxxx	Not specified	Dedicated isolated network boundary
Availability Zone	Deployment AZ	us-west-2a	—	All resources deployed in a single AZ
Control Subnet	VSP One SDS Cloud control-plane + Terraform/ Ansible Dev VM	subnet-xxxxx01	X.X.X.X/27 (from ControlNetworkCidrCloud)	Hosts SDS Storage Node control ENI endpoints
Internode Subnet	VSP One SDS Cloud inter-node replication	subnet-xxxxx02	-	Used for SDS internal communication & replication
Compute Subnet	VSP One SDS Cloud storage nodes (m6i.8xlarge)	subnet-xxxxx03	-	VSP One SDS Cloud storage/ data plane
ROSA Worker Subnet	ROSA Worker Nodes/Infra VM	subnet-xxxxx04	X.X.X.X/24 (from machine_cidr)	Hosts Terraform VM/Ansible VM ROSA workers, PostgreSQL pods, workloads



Note: For ROSA Cluster the subnet mask/CIDR must be between /16 and /24.

Enable the ROSA service

Before you begin

Log in to your [AWS account](#) and click Get started to use ROSA.

Containers

Red Hat OpenShift Service on AWS (ROSA)

Fully managed Red Hat® OpenShift® service on AWS

Red Hat OpenShift Service on AWS allows you to deploy fully operational and managed Red Hat OpenShift clusters while leveraging the full breadth and depth of AWS.

Get started

ROSA service fee pricing (US)

Control plane	\$0.03 per hour*
Worker nodes (hourly)	\$0.17/4 vCPU per hour*
Worker nodes (annually)	\$1,000/4 vCPU per year*
Worker nodes (3-year)	\$667/4 vCPU per year*

*EC2 Pricing is additional

Getting started

How it works

- Configure permissions**
Set permissions to ensure successful cluster creation and support by Red Hat Site Reliability Engineers
- Download CLI**
Download the command line tool to create and manage your OpenShift clusters
- Provision cluster**
Specify your cluster requirements in the CLI, and your OpenShift clusters are automatically created in minutes
- Deploy your applications**
Deploy your OpenShift applications to your Amazon Red Hat OpenShift clusters

Procedure

1. Click the checkbox to agree to the terms, and then click **Enable ROSA**.

ROSA > Get Started

Verify ROSA prerequisites [Info](#)

This page verifies if your account meets the prerequisites to create a Red Hat OpenShift Service on AWS (ROSA) cluster.

ROSA enablement [Info](#)

ROSA is jointly managed by AWS and Red Hat. To get started, enable ROSA to create a connection with Red Hat. This connection is required for metering and billing.

I agree to share my contact information with Red Hat.

Enable ROSA

2. Log in to your Red Hat account.
 - a. Enter `rosa login` in a terminal.
 - b. You will be prompted to open a web browser and go to <https://console.redhat.com/openshift/token/rosa>.
 - c. Log in with your Red Hat account credentials.

3. Pass the following values into Terraform:

```
create_oidc      = false
oidc_endpoint_url = "<issuer_url_from_rosa>"
oidc_config_id   = "<oidc_config_id_from_rosa>"
```

Prepare the Terraform/Ansible Dev VM

This section describes the prerequisites and step-by-step procedure for preparing the unified Terraform/Ansible Dev VM used to deploy and manage the AWS infrastructure, ROSA cluster, VSP One SDS Cloud cluster, and Kubernetes storage integration components.

Procedure

1. Create an EC2 instance with the following specifications:

- AMI: Amazon Linux 2023
- Instance Type: t3.large (2 vCPU with 8 GB RAM minimum)
- Root Disk: 40 GB gp3
- Network: Deploy in the ROSA Worker node Subnet
- Security Group: Allow SSH from Jump Host
- IAM Role Permissions:
 - ec2:*, iam:*, s3:*, sts:*, cloudformation:*
 - ROSA-required IAM permissions
 - (AdministratorAccess is optional for PoC)

2. Update System Packages.

```
sudo dnf update -y
sudo dnf install unzip tar wget git -y
```

3. Install Terraform.

```
wget https://releases.hashicorp.com/terraform/1.13.5/
terraform_1.13.5_linux_amd64.zip
unzip terraform_1.13.5_linux_amd64.zip
sudo mv terraform /usr/local/bin
terraform version
```

4. Install Ansible.

```
sudo dnf install ansible -y
ansible --version
```

5. The Terraform/Ansible VM must have the required Ansible collections installed before execution.

The `hitachivantara.vspone_block` Ansible collection is used to manage and configure VSP One SDS Cloud resources.

Install the collection using the following command:

```
ansible-galaxy collection install hitachivantara.vspone_block
```

This step is mandatory for storage pool expansion and VSP One SDS Cloud configuration workflows.

6. Install Python 3 and required libraries.

```
sudo dnf install python3 python3-pip -y
python3 --version
```

7. Install the AWS CLI.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

If the IAM Role is not attached, attach it to an instance by running the `aws configure` command and providing the following:

- AWS Access Key
- AWS Secret Key
- Region (for example, us-west-2)
- Output format: json

8. Install the ROSA CLI.

```
wget
https://mirror.openshift.com/pub/openshift-v4/clients/rosa/latest/rosa-
linux.tar.gz
tar -xvf rosa-linux.tar.gz
sudo mv rosa /usr/local/bin/
sudo chmod +x /usr/local/bin/rosa
rosa version
```

9. Install the OpenShift oc CLI.

```
curl -LO https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest-4.19/
openshift-client-linux.tar.gz
tar -xvf openshift-client-linux.tar.gz
sudo mv oc kubect1 /usr/local/bin/
oc version
```

10. Run the following command on the Terraform/Ansible VM.

```
rosa login
```

Provide the generated Red Hat offline access token from **Verify authentication**:

```
rosa whoami
```

Successful authentication is required for all ROSA cluster discovery and lifecycle operations.

Paste the generated offline token when prompted (<https://cloud.redhat.com/openshift/token>).

11. Clone the Deployment Repository. After downloading the code from the repository, navigate to the Terraform Config root directory. Contact the Hitachi Vantara team if you have issues with deployment repository access by sending an email to redhathelp@hitachivantara.com with the subject line *Request Terraform Automation for Hitachi SDS in Public Cloud*.

```
git clone <terraform-ansible-repository-url>
cd <repository-directory>
```

Result

The Terraform/Ansible Dev VM is now prepared to run `terraform init`, `terraform plan`, and `terraform apply`.

Configure the `terraform.tfvars` file

The `terraform.tfvars` file is used to provide environment-specific values for the Terraform SDSC + ROSA + PostgreSQL deployment. This file controls AWS settings, VSP One SDS Cloud configuration, ROSA cluster parameters, and security-related values.

Follow these steps to prepare and update the `terraform.tfvars` file.

Procedure

1. After downloading the code from the repository, navigate to the Terraform Config root directory. Contact the Hitachi Vantara team if you have issues with deployment repository access by sending an email to redhathelp@hitachivantara.com with the subject line *Request Terraform Automation for Hitachi SDS in Public Cloud*.

```
cd hv-iac-rosa-sdsc-postgressql
```

2. Configure AWS environment details.
Open `terraform.tfvars` in an editor:

```
vi terraform.tfvars
```

```

# AWS Environment Details
region                = "us-west-2"
cluster_name         = "VSPOneSDSBlock-PI-8"
availability_zone    = "us-west-2a"
ControlNetworkCidrBlock = "X.X.X.X/X"

# AWS VPC and Subnet Details
vpc_id               = "vpc-xxxxx"
control_subnet_id   = "subnet-01-xxxx"
internode_subnet_id = "subnet-02-xxxx"
compute_subnet_id   = "subnet-03-xxxx"

# Hitachi VSP One SDS Block Configuration Details
storage_node_instance_type = "m6i.8xlarge"
configuration_pattern      = "Mirroring Duplication ( 3 Nodes )"
drive_count                = "6"
physical_drive_capacity    = "Mirroring Duplication 1579 GiB"
ebs_volume_encryption     = "disable"
ebs_volume_kms_key_id     = "None"

time_zone                = "UTC"
billing_code             = "xxxx"
s3_uri                   = "s3://<bucket_name>/CF_Log/"
iam_role_name            = "xxxx"

#Provide your Jump host IP/Subnet (replace example below), uncomment and modify if you want to hardcode the value
#Otherwise pass it during "terraform apply".

jmp_host_ip             = "X.X.X.X/X"

#Allow ROSA workers iSCSI access to SDS storage (TCP 3260)
rosa_subnet_cidr        = "XX.XX.XX.XX/X"

#SDS Credentials
sds_admin_username      = "admin"
sds_new_password        = "NTest@123"
sds_default_password    = "hds-admin"

```

3. Update the AWS environment section with the correct values:
 - region – AWS region where the solution is deployed.
 - cluster_name – Logical name for the VSP One SDS Cloud deployment.
 - availability_zone – Single AZ used for all components.
 - ControlNetworkCidrCloud – CIDR used by the control subnet (Terraform VM and SDS APIs).
 - vpc_id – Primary VPC for ROSA and VSP One SDS Cloud.
 - control_subnet_id – Subnet for control-plane and Terraform/Ansible Dev VM.
 - internode_subnet_id – Subnet used for VSP One SDS Cloud internode traffic.
 - compute_subnet_id – Subnet for VSP One SDS Cloud storage/data-plane nodes.
 - storage_node_instance_type – EC2 instance type for VSP One SDS Cloud nodes.
 - configuration_pattern – SDS cluster layout (3-node mirroring).
 - drive_count/physical_drive_capacity – Disk configuration per node.
 - ebs_volume_encryption/ebs_volume_kms_key_id – EBS encryption settings.
 - time_zone, billing_code – Operational metadata.
 - s3_uri – S3 bucket path for logs.
 - iam_role_name – IAM role used by SDS deployment.
 - jmp_host_ip – CIDR or IP range allowed to SSH into the Terraform/Ansible VM.
 - rosa_subnet_cidr – CIDR for ROSA worker nodes allowed to access SDS iSCSI (TCP/3260).

- `sds_admin_username = "admin"`
- `sds_new_password = "NTest@123"`
- `sds_default_password = "hsds-admin"`

4. Update the ROSA environment section with the correct values:

```
## ROSA Variables
token                = "xxxxxx"
aws_region           = "us-west-2" # Replace with your AWS region
rosacluster_name     = "pi-my-rosa-cluster"
openshift_version    = "4.19.13" # Verify with `rosa list versions`
account_role_prefix  = "pi-my-rosa-account"
operator_role_prefix = "pi-my-rosa-hcp"
aws_subnet_ids       = ["subnet-04-xxxx"] # Replace with your private subnet IDs
private              = true
admin_password       = "Testrosa@123456789" # Replace with a secure password
pod_cidr              = "10.128.0.0/14"
service_cidr         = "172.30.0.0/16"
machine_cidr         = "X.X.X.X/X"

compute_nodes        = 2
compute_machine_type = "m5.xlarge"
oidc_endpoint_url    = "oidc.opl.openshiftapps.com/xxxxxxx"
oidc_config_id       = "xxxxxxx"
jump_server_cidr     = "X.X.X.X/X"
```

- `token` – ROSA offline token from cloud.redhat.com (sensitive).
- `rosacluster_name` – Name of the ROSA cluster.
- `openshift_version` – Target OpenShift version.
- `account_role_prefix/operator_role_prefix` – Prefixes for ROSA IAM roles.
- `aws_subnet_ids` – Private subnet(s) used by ROSA workers.
- `private` – Indicates private ROSA cluster.
- `admin_password` – OpenShift admin password.
- `pod_cidr`, `service_cidr`, `machine_cidr` – Cluster networking CIDRs.
- `compute_nodes`, `compute_machine_type` – Number and type of ROSA worker nodes.
- `oidc_endpoint_url`, `oidc_config_id` – OIDC configuration for ROSA HCP.

Get the open from the Step: ROSA OIDC Workflow (When NOT letting Terraform create OIDC)

```
create_oidc = false
oidc_endpoint_url = "https://my-oidc-bucket.s3.us-west-2.amazonaws.com"
oidc_config_id = "6fb290493a2adxxxxxxx"
```

- `jump_server_cidr` – CIDR allowed to access the cluster and Terraform/Ansible Dev VM.
5. Carefully validate each variable and its corresponding value to ensure correctness for your deployment environment. After confirming all entries, save the `terraform.tfvars` file to finalize the configuration.
 6. Update the `sds.tf` file.

Edit `sds.tf` and navigate to the `null_resource "run_ansible_expand_pool"` block at the end of the file. In the `--extra-vars` string, replace the existing `storage_password` value with your new SDSC administrator password:

```
vi sds.tf
```

```
resource "null_resource" "run_ansible_expand_pool" {
  depends_on = [
    null_resource.run_passwordreset_local
  ]

  provisioner "local-exec" {
    command = <<EOT

    echo "Running Ansible playbook to expand storage pool..."
    ansible-playbook ./ansible/storage_pool.yml \
      -i localhost, \
      --extra-vars "storage_address=${data.aws_lb.lb.dns_name} storage_username=admin storage_password=NTest@123"
    EOT
  }
}
```

After making the changes save the file.

7. Run `terraform init`.

After the `terraform.tfvars` file is configured, initialize the Terraform working directory. This step downloads all required provider plugins and prepares the backend for the deployment.

Run the following command from the Terraform root directory:

```
terraform init
```

A successful initialization confirms that Terraform is ready to proceed with the deployment.

```
$ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/time from the dependency lock file
- Reusing previous version of hashicorp/null from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of terraform-redhat/rhcs from the dependency lock file
e
- Using previously-installed hashicorp/local v2.6.1
- Using previously-installed hashicorp/time v0.13.1
- Using previously-installed hashicorp/null v3.2.4
- Using previously-installed hashicorp/random v3.7.2
- Using previously-installed hashicorp/aws v6.21.0
- Using previously-installed terraform-redhat/rhcs v1.7.2

Terraform has been successfully initialized!

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

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

8. Run `terraform plan`.

After initializing the Terraform environment, the next step is to generate an execution plan.

The `terraform plan` command validates all variables, evaluates resource dependencies, and displays the actions Terraform will take during deployment—without making any actual changes.

Run the following command from the Terraform root directory:

```
terraform plan
```

```
[ec2-user@ip-10-77-72-9 sds]$ terraform plan
```

A successful plan output confirms that your configuration is valid and the environment is ready for deployment.

Review the plan carefully before proceeding to the `apply` phase.

9. Run `terraform apply`.

After reviewing the execution plan, proceed with the actual deployment by running:

```
terraform apply
```

```
[ec2-user@ip-10-77-72-9 sds]$ terraform apply
```

Terraform will display the proposed changes again and prompt for confirmation.

10. Type `yes` to begin the automated provisioning of all AWS, ROSA, and SDS resources.

A successful `apply` indicates that the environment has been fully deployed and is ready for further configuration and integration tasks.

During the `terraform apply` phase, the automation performs the complete end-to-end provisioning workflow. This includes the following:

- Deploying the VSP One SDS Cloud cluster in AWS
- Resetting the default VSP One SDS Cloud administrator password
- Adding the attached disks to the storage pool to prepare for volume creation
- Provisioning the ROSA (OpenShift) cluster
- Installing and configuring the Hitachi SDS CSI driver and related components
- Deploying the PostgreSQL pod, creating the PVC, and mounting the SDS-backed persistent volume to the pod

This stage fully builds the infrastructure and application stack in a single automated execution.

After the Terraform execution is complete, allow an additional 15 to 20 minutes for the VSP One SDS Cloud cluster to transition into a fully ready and stable state.

11. After the deployment is successful, log in to the VSP One SDS Cloud cluster and navigate to the **Events** section to review the sequence of cluster initialization and storage provisioning activities.

log in to **VSP One SDS Cloud Cluster > Event**

EVENT	MONITOR	PROTECTION DOMAINS	STORAGE NODES	DRIVES	COMPUTE NODES	VOLUMES	STORAGE POOLS	PORTS
12/01/2025 23:06:51	Metadata allocation processing co...	Info	Drive	KARS0...	Allocation of metadata capacity for the write back mode with c...	e...		
12/01/2025 23:06:51	Prepare for capacity allocation pro...	Info	Drive	KARS0...	Prepare for capacity allocation processing of the storage contr...	2...		
12/01/2025 23:06:26	Failure to complete job	4	Info	Service	KARS1...	The job has ended abnormally. (Operation = VOLUME_CREATE, ...	f...	
12/01/2025 23:06:26	Start of job	Info	Service	KARS1...	The job has started. (Operation = VOLUME_CREATE, Job ID = 49...	3...		
12/01/2025 23:06:16	Prepare for capacity allocation pro...	Info	Drive	KARS0...	Prepare for capacity allocation processing of the storage contr...	d...		
12/01/2025 23:06:16	Metadata allocation processing start	Info	Drive	KARS0...	Allocation of metadata capacity for the write back mode with c...	4...		
12/01/2025 23:06:13	Prepare for capacity allocation pro...	Info	Drive	KARS0...	Prepare for capacity allocation processing of the storage contr...	a...		
12/01/2025 23:06:13	Metadata allocation processing start	Info	Drive	KARS0...	Allocation of metadata capacity for the write back mode with c...	c...		
12/01/2025 23:06:10	Prepare for capacity allocation pro...	3	Info	Drive	KARS0...	Prepare for capacity allocation processing of the storage contr...	c...	
12/01/2025 23:06:10	Metadata allocation processing start	Info	Drive	KARS0...	Allocation of metadata capacity for the write back mode with c...	2...		
12/01/2025 23:06:07	Successful completion of job	2	Info	Service	KARS1...	The job has completed successfully. (Operation = POOL_EXPAN...	8...	
12/01/2025 23:05:25	User information reflection to the ...	Info	Storag...	KARS2...	User information was reflected to the console interface. (Stora...	4...		
12/01/2025 23:05:24	User information reflection to the ...	Info	Storag...	KARS2...	User information was reflected to the console interface. (Stora...	9...		
12/01/2025 23:05:18	User information reflection to the ...	Info	Storag...	KARS2...	User information was reflected to the console interface. (Stora...	c...		
12/01/2025 23:05:11	Start of job	1	Info	Service	KARS1...	The job has started. (Operation = POOL_EXPAND, Job ID = fadce...	a...	
12/01/2025 22:42:23	User authentication setting reflecti...	Info	Storag...	KARS2...	User authentication settings were reflected to the console inter...	2...		
12/01/2025 22:42:22	User authentication setting reflecti...	Info	Storag...	KARS2...	User authentication settings were reflected to the console inter...	f...		

- Event 1: Pool expansion is initiated by Ansible.
- Event 2: Pool expansion completes successfully.
- Event 3: The storage controller begins processing capacity allocation.

- Event 4: PVC-triggered volume creation attempts fail because capacity allocation (Event 3) is still in progress.

At this stage The PVC attachment failed and as a result, the pod did not transition to the running state.

Check the current pod status using the `oc get pod` command. You will see the pod in a **Pending** state, which is expected while the VSP One SDS Cloud on AWS is still becoming fully ready.

```
[ec2-user@ip-10-77-72-9 sds]$ oc get pod
NAME                READY   STATUS    RESTARTS   AGE
hspc-csi-controller-7fd765b9fd-8kwcz  6/6    Running   0           46m
hspc-csi-node-jhg75    2/2    Running   0           46m
hspc-csi-node-lqz7h    2/2    Running   0           46m
postgres-786cc4d9d8-zcc7c  0/1    Pending   0           46m
[ec2-user@ip-10-77-72-9 sds]$
```

Run `oc get pvc` to view the PVC status. You will see the PVC in a **Pending** state, which is also expected at this stage.

```
ec2-user@ip-10-77-72-9 sds]$ oc get pvc
NAME                STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS          VOLUMEATTRIBUTESCLASS  AGE
ostgres-pvc        Pending                10Gi         RWO          sc-sample-vsp-one-sds-block  <unset>                47m
ec2-user@ip-10-77-72-9 sds]$
```

After a few minutes, the storage becomes fully ready, as shown.

DATE	EVENT NAME	SEVERITY...	CATEGOR...	MESSAGE...	MESSAGE	ID...
12/01/2025 23:28:07	Successful completion of job	6	Service	KARS1...	The job has completed successfully. (Operation = VOLUME_SER...	4...
12/01/2025 23:28:04	Start of job	Info	Service	KARS1...	The job has started. (Operation = VOLUME_SERVER_CONNECTI...	d...
12/01/2025 23:28:02	Successful completion of job	Info	Service	KARS1...	The job has completed successfully. (Operation = PATH_CREATE...	d...
12/01/2025 23:28:02	Start of job	Info	Service	KARS1...	The job has started. (Operation = PATH_CREATE, Job ID = 0d9d1...	5...
12/01/2025 23:28:02	Successful completion of job	Info	Service	KARS1...	The job has completed successfully. (Operation = HBA_CREATE,...	d...
12/01/2025 23:28:02	Start of job	Info	Service	KARS1...	The job has started. (Operation = HBA_CREATE, Job ID = 5b2021...	b...
12/01/2025 23:28:02	Successful completion of job	Info	Service	KARS1...	The job has completed successfully. (Operation = SERVER_CREA...	b...
12/01/2025 23:28:02	Start of job	Info	Service	KARS1...	The job has started. (Operation = SERVER_CREATE, Job ID = 924...	a...
12/01/2025 23:28:01	Initial expansion of the storage con...	Info	Storag...	KARS1...	Capacity allocation processing (initial expansion) of the storage...	6...
12/01/2025 23:27:59	Failure to complete job	Info	Service	KARS1...	The job has ended abnormally. (Operation = SERVER_CREATE, J...	f...
12/01/2025 23:26:51	Start of job	Info	Service	KARS1...	The job has started. (Operation = SERVER_CREATE, Job ID = 4a3...	b...
12/01/2025 23:26:47	Initial expansion of the storage con...	Info	Storag...	KARS1...	Capacity allocation processing (initial expansion) of the storage...	c...
12/01/2025 23:26:39	Successful completion of job	5	Service	KARS1...	The job has completed successfully. (Operation = VOLUME_CRE...	5...
12/01/2025 23:26:33	Start of job	Info	Service	KARS1...	The job has started. (Operation = VOLUME_CREATE, Job ID = 64...	0...
12/01/2025 23:25:22	Initial expansion of the storage con...	Info	Storag...	KARS1...	Capacity allocation processing (initial expansion) of the storage...	2...

- Event 5: After approximately 10–20 minutes, the system completes capacity allocation, and you will see that volume creation has succeeded.
- Event 6: The volume server connection is successfully established, indicating that the volume is now ready for use by the ROSA workload.

12. Log in to the ROSA cluster at this stage to observe the status of the PersistentVolumeClaim (PVC).

Run the following command to verify:

```
oc get pvc
```

You should now see that the PVC has been created successfully.

```
[ec2-user@ip-10-77-72-9 sds]$ oc get pvc
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS          VOLUMEATTRIBUTESCLASS  AGE
postgres-pvc        Bound    pvc-71b429fe-c2be-461f-8546-d30499e248ed  10Gi         RWO          sc-sample-vsp-one-sds-block  <unset>                7h44m
```

13. After confirming that the PVC has been created successfully, validate that the corresponding Persistent Volume (PV) has also been provisioned.

Run the following command:

```
oc get pv
```

This will display the PV associated with your PVC and confirm that storage has been successfully allocated from the VSP One SDS Cloud backend.

```
[ec2-user@ip-10-77-72-9 sds]$ oc get pv
NAME                                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM                                STORAGECLASS
pvc-59b63355-e935-45e5-bcf2-1988ea65a3a8  100Gi     RWO           Delete          Bound   openshift-monitoring/prometheus-data-prometheus-k8s-0  gp3-cs
<unset>
pvc-71b429fe-c2be-461f-8546-d30499e248ed  10Gi      RWO           Delete          Bound   default/postgres-pvc                                sc-sam
p1e-vsp-one-sds-block <unset>
pvc-7e1c9ba2-167d-45de-a46c-7cd3a82554c3  100Gi     RWO           Delete          Bound   openshift-monitoring/prometheus-data-prometheus-k8s-1  gp3-cs
<unset>
```

- After confirming the PV and PVC, verify that the PostgreSQL pod is running successfully on the ROSA cluster.

Run the following command:

```
oc get pod
```

```
[ec2-user@ip-10-77-72-9 sds]$ oc get pod
NAME                                READY  STATUS   RESTARTS  AGE
hspc-csi-controller-7fd765b9fd-8kwcz  6/6    Running  0          7h44m
hspc-csi-node-jhg75                    2/2    Running  0          7h44m
hspc-csi-node-lqz7h                    2/2    Running  0          7h44m
postgres-786cc4d9d8-zcc7c             1/1    Running  0          7h45m
```

- To review detailed information about the PostgreSQL pod—including events, volume mounts, node assignment, and container status—run the `oc describe` command.

```
oc describe pod <pod name>
```

```
[ec2-user@ip-10-77-72-9 sds]$ oc get pod
NAME                                READY  STATUS   RESTARTS  AGE
hspc-csi-controller-7fd765b9fd-8kwcz  6/6    Running  0          7h44m
hspc-csi-node-jhg75                    2/2    Running  0          7h44m
hspc-csi-node-lqz7h                    2/2    Running  0          7h44m
postgres-786cc4d9d8-zcc7c             1/1    Running  0          7h45m
[ec2-user@ip-10-77-72-9 sds]$ oc describe pod postgres-786cc4d9d8-zcc7c
```

Output:

```

Status:          Running
IP:             10.129.0.44
IPs:
  IP:          10.129.0.44
Controlled By: ReplicaSet/postgres-786cc4d9d8
Containers:
  postgres:
    Container ID:  cri-o://afc5f97b08a9e82776197d0fd7ec9a83c1aa2acla439c241f5a75cab94ae0026
    Image:         postgres:15
    Image ID:     docker.io/library/postgres@sha256:24d6c206bba8c0440bceb24a8d4bf642f60bf7aea94887051ea5761d29c22323
    Port:        5432/TCP
    Host Port:   0/TCP
    State:       Running
      Started:   Wed, 03 Dec 2025 18:41:13 +0000
    Ready:       True
    Restart Count: 0
    Environment:
      POSTGRES_USER:  admin
      POSTGRES_PASSWORD:  admin123
      POSTGRES_DB:    hitachi_db
      PGDATA:         /var/lib/postgresql/data/pgdata
    Mounts:
      /var/lib/postgresql/data from postgres-storage (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jclsm (ro)
Conditions:
  Type                    Status
  PodReadyToStartContainers  True
  Initialized                True
  Ready                      True
  ContainersReady            True
  PodScheduled                True
Volumes:
  postgres-storage:
    Type:      PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName: postgres-pvc
    ReadOnly:  false
  kube-api-access-jclsm:
    Type:      Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
    ConfigMapName: openshift-service-ca.crt
    ConfigMapOptional: <nil>
QoS Class:           BestEffort
Node-Selectors:      <none>
Tolerations:         node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:

```

This command opens an interactive shell inside the specified pod, allowing you to run commands directly inside the container.

```
oc exec -it <pod name> -- /bin/bash
```

`df -h` helps identify the PVC mount point inside the container by showing all mounted filesystems and their usage.

```
df -h
```

```

[ec2-user@ip-10-77-72-9 sds]$ oc exec -it postgres-786cc4d9d8-zcc7c -- /bin/bash
root@postgres-786cc4d9d8-zcc7c:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         300G  21G  279G   7% /
tmpfs           64M   0   64M   0% /dev
shm            64M  1.1M   63M   2% /dev/shm
tmpfs          3.1G  13M   3.1G   1% /etc/hostname
/dev/nvme0n1p4 300G  21G  279G   7% /etc/hosts
/dev/mapper/mpatha 9.8G  46M   9.2G   1% /var/lib/postgresql/data
tmpfs         15G  16K   15G   1% /run/secrets/kubernetes.io/serviceaccount
devtmpfs      4.0M   0   4.0M   0% /proc/keys

```

Solution validation

The following table lists the test scenarios completed to validate the solution.



Note: Testing of this configuration was in a lab environment. Many factors affect production environments beyond prediction or duplication in a lab environment. Follow the recommended practice of conducting proof-of-concept testing for acceptable results in a non-production, isolated test environment that otherwise matches your production environment before your production implementation of this solution.

Test Scenario	Terraform/ Ansible Deployment	ROSA Cluster	VSP One SDS Cloud	Observed Behavior/ Validation Result
Provision AWS Infrastructure (VPC, Subnets, EC2, LB, IAM) using Terraform	Yes	N/A	N/A	Successfully provisioned all required AWS components using Terraform in a fully automated manner.
Deploy VSP One SDS Cloud cluster using Terraform + Ansible	Yes	N/A	Yes	VSP One SDS Cloud cluster created successfully; nodes joined; storage services initialized.
Reset default SDS admin password using automated Python script	Yes	N/A	Yes	Password reset executed successfully; SDS API prepared for CSI driver authentication.
Add disks to storage pool using Ansible	Yes	N/A	Yes	Disks discovered and added to storage pool without manual intervention.
Provision ROSA HCP Cluster	Yes	Yes	N/A	ROSA cluster provisioned and reachable; worker nodes ready for CSI installation.
Install Hitachi HSPC CSI Driver on ROSA	N/A	Yes	Yes	CSI operator and iSCSI daemonsets deployed successfully; cluster ready for SDS-backed storage.
Auto-render Kubernetes secret using Terraform templatefile()	Yes	Yes	Yes	Secret.yaml created automatically with SDS endpoint, user, password encoded in base64.
Create StorageClass referencing SDS backend	N/A	Yes	Yes	StorageClass created successfully; validated using <code>oc get sc</code> .

Test Scenario	Terraform/ Ansible Deployment	ROSA Cluster	VSP One SDS Cloud	Observed Behavior/ Validation Result
Create PVC and validate dynamic PV creation	N/A	Yes	Yes	PVC triggered SDS to create a volume; CSI bound PV correctly; seen via <code>oc get pvc/pv</code> .
Deploy PostgreSQL Pod with SDS-backed Persistent Volume	N/A	Yes	Yes	Pod deployed successfully; SDS LUN attached through iSCSI; verified using <code>oc rsh + df -h</code> .

References

See the following documents for more information:

- [Getting started with Red Hat OpenShift Service on AWS \(ROSA\)](#)
- [Red Hat ROSA Account Setup](#)
- [Terraform Installation guide](#)
- [Hitachi Storage Plug-in for Containers](#)
- [Automate the Deployment of VSP One SDS Block Cluster Provisioning video](#)
- [ROSA Pricing](#)
- [ROSA Terraform Registry](#)

See the following for the latest updated VSP One SDS Cloud/ROSA compatibility, deployment, and support details:

- [VSP One SDS Cloud – Product Documentation](#)
- [Red Hat OpenShift Service on AWS \(ROSA\) Docs](#)
- [Hitachi Storage Plug-in for Containers](#)
- [ROSA FAQ](#)

Hitachi Vantara



Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA

HitachiVantara.com/contact