

Data Analytics with SQL Server 2022 on Red Hat OpenShift and Dell ObjectScale

A Dell validated design for Data Analytics solution with SQL server 2022 on Red Hat OpenShift and Dell ObjectScale Storage

January 2024

H19851

Design Guide

Abstract

This design guide provides a solution overview and the benefits of a powerful, agile, and flexible cloud-ready infrastructure for running SQL Server 2022 data analytics solution on Red Hat OpenShift and Dell ObjectScale storage.

Dell Technologies Solutions



Copyright

The information in this publication is provided as is. Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © 2024 Dell Inc. or its subsidiaries. All Rights Reserved. Dell Technologies, Dell, and other trademarks are trademarks of Dell Inc. or its subsidiaries. Intel, the Intel logo, the Intel Inside logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries. Other trademarks may be trademarks of their respective owners. Published in the USA 01/24 Design Guide H19851.

Dell Inc. believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

Contents

Chapter 1	Introduction	5
	Introduction	6
	Business challenge	6
	Solution introduction.....	7
	Design guide introduction.....	8
	Terminology	8
Chapter 2	Solution Concepts	10
	Solution concepts.....	11
	Partner technology overview	13
Chapter 3	Solution Architecture	17
	Physical architecture	18
	Logical architecture	19
	Data access architecture.....	20
	Solution requirements	21
Chapter 4	Solution Deployment	22
	Building the OpenShift cluster	23
	Red Hat OpenShift cluster deployment	23
	Red Hat ODF deployment and configuration	28
	Building an ObjectScale Storage.....	29
	Deploying ObjectStore	30
	Apache Spark deployment	32
	SQL Server 2022 deployment.....	33
Chapter 5	Solution Execution	39
	Enable PolyBase.....	40
	SQL Server 2022 with S3 ObjectStore integration.....	41
	Create TPC-DS table schema in Pyspark	43
Chapter 6	Summary and Conclusion	50
	Summary	51
	We value your feedback.....	51
Chapter 7	References	52
	Dell Technologies documentation	53

AMD documentation.....53

Microsoft documentation53

Red Hat documentation53

Additional documentation.....53

Chapter 1 Introduction

This chapter presents the following topics:

Introduction	6
Business challenge.....	6
Solution introduction.....	7
Design guide introduction.....	8
Terminology	8

Introduction

SQL Server 2022 is the latest release of Microsoft's relational database engine. This design guide describes the deployment of Red Hat OpenShift Cluster with self-service deployment of SQL Server 2022 as pods for data analytics purposes. This guide also features some of the newly introduced features of SQL Server 2022, including PolyBase S3 compatibility, which enables querying of external datasets stored in object storage, such as Dell ObjectScale.

Data analytics is the process of examining raw, structured, and unstructured data to determine trends and answer queries. Organizations use data analytics to make informed decisions about their business.

Data analytic workloads can be CPU intensive and choosing the optimal CPUs for the data analytic servers can be challenging. There are three common CPU specifications that should be examined while choosing a CPU:

- Number of cores per socket
- Frequency of the cores
- L3 cache size

Data virtualization solutions can easily access disparate data repositories and enable organizations to quickly integrate with new data sources as they become available across the edge, core, and cloud locations. Data virtualization allows enterprises to seek flexible and cost-effective storage options for structured, unstructured, and semi-structured data.

Business challenge

Rapid growth in the volume, velocity, and variety of organizational data hinders traditional strategies for data management and analysis and agile enterprises must be able to mine value from data in a short period. In response to these challenges, enterprises invest heavily in developing data management strategies using data warehouses, data lakes, and data lakehouses to improve data management.

A solution that enables data access in its native repository has significant advantages to address these business challenges, including minimizing the latency caused by data copying and reducing expenditure for developer resources required to integrate data into data warehouses. This solution allows developers and data scientists to spend more time mining data and developing insights for the enterprise.

Conventional approaches that leverage technologies are slow in delivering innovation at the pace that business ecosystems are evolving. Red Hat OpenShift addresses a wide range of business challenges related to modern application development, deployment, and management.

Storage is another crucial factor for data analytic workloads. The most common storage for unstructured and semi-structured data is S3-compatible object storage. Procuring the right object storage for data analytic workloads can be complex and time consuming.

Solution introduction

Overview

This solution is designed with a focus on creating a cloud native data analytics platform that can be deployed across various environments such as on premise, public cloud, hybrid cloud, or at the edge with a consistent user experience.

The PolyBase feature in SQL Server facilitates data virtualization, enabling users to access and query data from various sources, which helps businesses overcome data integration and analysis challenges.

Dell ObjectScale is a software-defined enterprise-grade object storage. Built with a scale-out architecture, ObjectScale clusters expand from a few terabytes to petabytes and beyond without limits on the number of object stores, buckets, or objects. Clusters can start as small as three nodes and scale to thousands.

The underlying infrastructure for this Dell Validated Design consists of AMD EPYC 9334 processors-based Dell PowerEdge servers with internal disk drives, using Red Hat OpenShift Data Foundation as a persistent block storage.

AMD 4th Gen is part of the EPYC lineup using Zen 4 architecture with Socket SP5 and 3D V-Cache technology. This processor has up to 96 Zen 4 cores and 1152 MB of L3 cache per socket. The processor operates at 2.7 GHz by default but can boost up to 3.9 GHz depending on the workload. This 4th Gen EPYC processor is well suited for analytic workloads.

Red Hat OpenShift Container Platform provides a robust containerization platform and Kubernetes-based orchestration platform that enables organizations to build, deploy, and manage applications and databases efficiently across on-premises and multicloud environments. The Red Hat OpenShift Data Foundation (RHODF) brings persistent and scalable block storage for containerized applications and databases. By combining the two technologies, it provides an integrated, enterprise-grade storage and processing infrastructure.

This solution also highlights some of the latest features of SQL Server 2022, such as S3 compatibility of PolyBase, which enables the connection of external tables stored in Object storage, such as Dell ObjectScale.

Document purpose

The purpose of this document is to provide design guidance for deploying SQL Server 2022 containers on Red Hat OpenShift Container Platform with Dell ObjectScale for data analytics purpose. This design guide also explores integration of SQL server 2022 with Dell ObjectScale using Data virtualization.

Audience

This document is intended for developers, data engineers, data scientists, and architects. This document can also be used by enterprises that plan to run analytic workloads on SQL Server 2022 with object storage integration and use Dell PowerEdge AMD-based servers as the foundational infrastructure.

Design guide introduction

Dell validated solutions are based on established principles to provide customers with more value than creating their own solution, resulting in faster time-to-market and reduced risks. Dell systems engineers validated Microsoft SQL Server with Dell ObjectScale Storage to ensure that the solution is scalable, reliable, and efficient.

This guide outlines the major components involved in developing the solution such as data lakehouse, data virtualization, Dell ObjectScale, Red Hat OpenShift Container Platform (RHOC) and RHODF, Apache Spark, Microsoft SQL Server 2022, and other solution requirements. It also discusses the physical, logical, and data access architecture and deployment concepts. Deployment concepts described in this solution include deployment and configuration of OpenShift Cluster, MetalLB service load balancer, Red Hat OpenShift Data Foundation, Dell ObjectScale, Apache Spark, and Microsoft SQL Server 2022 with PolyBase feature.

This design guide highlights use cases such as the creation of Microsoft SQL Server 2022 container image and self-service deployment of SQL Server 2022 for data analytics purposes. It also highlights features of SQL Server 2022, including PolyBase, which introduced connecting to external tables backed by S3 compatible storage such as Dell ObjectScale.

In this design, SQL Server 2022 runs as containerized application on Red Hat OpenShift Container Platform. This solution has been validated with Red Hat OpenShift Container Platform, using Red Hat OpenShift Data Foundation as a persistent block storage.

Terminology

The following table provides definitions for some of the terms that are used in this document.

Table 1. Terminology

Term	Definition
S3	The S3 protocol, or Simple Storage Service protocol, is a data storage system that provides a simple and scalable solution for storing and retrieving data in the cloud.
Buckets	Buckets are object containers that are used to control access to objects.
EC	Erasure Coding
DC	A domain controller, or DC, is a server used in a Windows-based network that manages user authentication and authorization for accessing network resources.
DNS	Domain Name System is a service provided by Microsoft Windows server operating systems that enables domain name resolution on a local network.
CSI	Container Storage Interface (CSI) was developed as a standard for exposing arbitrary block and file storage systems to containerized workloads on Kubernetes.

Term	Definition
RHOCP	Red Hat OpenShift Container Platform
RHODF	Red Hat OpenShift Data Foundation
TPC	Transaction Processing Performance Council.
ETL	Extract, Transform and, Load is a process of combining data from multiple sources into a large, central repository like Data Warehouse.
T-SQL	Transact Structure Query Language (SQL Language which is used to interact with relational databases)
DA	Data Analytics
BI	Business Intelligence
AI	Artificial intelligence
ML	Machine learning
Pod	Pod is the smallest unit of a Kubernetes application and is a collection of one or more Linux containers.
PV	Persistent Volume
PVC	Persistent Volume Claim
EPYC	Extreme Performance Yield Computing
IoT	Internet of Things
CI/CD	Continuous integration and Continuous deployment
SVC Node	Service Node
ZTP	Zero-Touch Provisioning

Chapter 2 Solution Concepts

This chapter presents the following topics:

Solution concepts11

Partner technology overview13

Solution concepts

Overview

This section discusses the concepts involved in building the solution. The following list includes the hardware and software layers in the solution stack.

- AMD-based Dell PowerEdge servers.
- Red Hat OpenShift Container Platform.
- Red Hat OpenShift Data Foundation.
- Dell ObjectScale
- Microsoft SQL Server 2022
- Apache Spark and Delta Lake

Data Lakehouse

A traditional data lake tends to be a flexible and cost-effective implementation by storing data in its raw form typically unstructured or semi-structured. A data warehouse is a more advanced repository that stores structured data for reporting and analysis, typically cleansed and optimized for better data quality and querying.

A data lakehouse is a modern data platform that combines the flexibility low cost, and scalability of data lakes with the ACID transaction, data management, and data querying capability of the data warehouse to provide a centralized repository for all data.

A data lakehouse supports diverse datasets such as structured, semi-structured, and unstructured data, meeting the needs of both business intelligence and data science use cases. It typically supports programming languages like Python, R, and SQL.

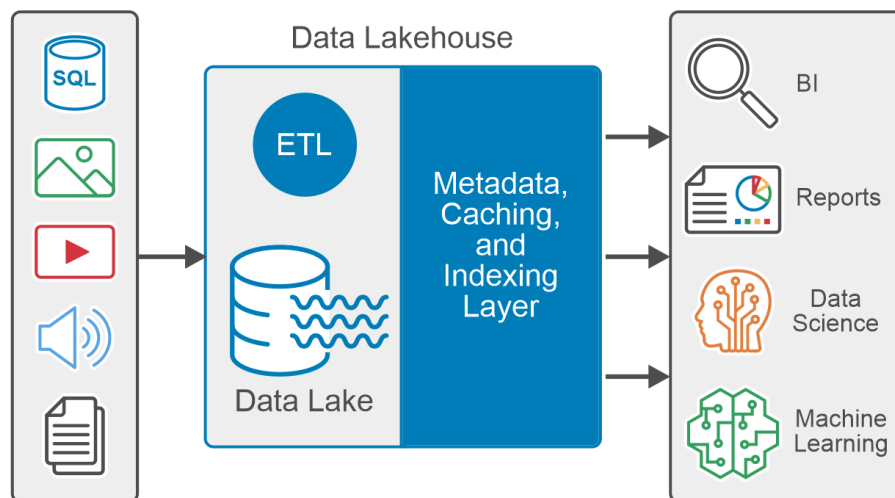


Figure 1. Data Lakehouse logical diagram

The metadata layer, like the open-source Delta Lake, is the foundation of the data lakehouse architecture. The metadata layer is a unified catalog that maintains metadata for every object in the data lake storage, which helps to organize and provide information about the data in the system. The metadata layer enables other features common in data

lakehouse, like support for ACID transactions, streaming, caching hot data, time travel, schema enforcement, evolution, and data validation.

Data virtualization

Data virtualization is a data management methodology that enables applications to access and manipulate data without requiring technical knowledge of the source data, physical location, or the format.

Data virtualization involves abstracting multiple sources through a single data access layer. This enables organizations to integrate different types of data virtually. The integration is critical for data mining, analytics, and predictive analytics tools which use machine learning (ML) and artificial intelligence (AI).

Dell ObjectScale

Dell ObjectScale is a software-defined, containerized architecture that delivers enterprise-class, high-performance object storage in a Kubernetes-native environment. ObjectScale empowers organizations with the ability to put data closer to the applications they support, reducing latency and improving the user experience. Dell ObjectScale is the foundation that provides support to store, manipulate, and analyze unstructured data on a very large scale.

Dell ObjectScale is built on a Kubernetes platform, to deliver a simplified product where operating system and hardware-level layers are managed by Kubernetes and ObjectScale manages the storage.

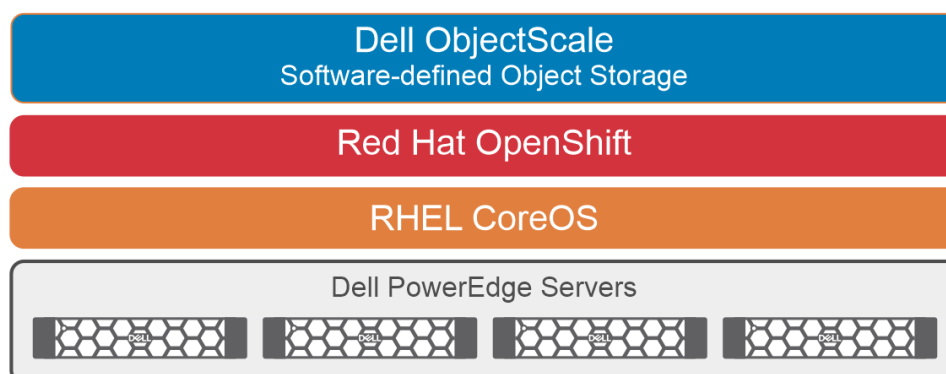


Figure 2. Dell ObjectScale physical architecture

ObjectScale is a next-generation object storage software that is lighter, faster, and deployable on existing infrastructure. It can be deployed on the Red Hat OpenShift Container Platform or SUSE Linux Enterprise Server (SLES) infrastructure. ObjectScale allows organizations to deliver scalable cloud services with the reliability and control of a private cloud infrastructure.

Key features of ObjectScale include:

- Kubernetes native, customer-deployable
- Rich S3 compatibility
- Self-service APIs help in the quick spin-up of object storage containers
- Scalability
- Built as a scaled-out, software-defined architecture, following the microservices principle of cloud applications

- Data protection
- Geo-replication
- Superior performance irrespective of the size of an object

The following list describes some of the benefits that Kubernetes provides for ObjectScale:

- Dynamic scaling of resources
- Deployment using minimal required resources
- Highly portable
- Self-healing: Auto placement, auto restart, and auto replication

For more information, see the [Dell ObjectScale webpage](#).

Partner technology overview

Red Hat OpenShift Container Platform

RHOCP is a consistent hybrid cloud foundation for containerized applications, powered by Kubernetes. Using RHOCP Developers and DevOps engineers can quickly build, modernize, deploy, run, and manage applications anywhere, securely, and at scale.

RHOCP is powered by open-source technologies and offers flexible deployment options ranging from physical, virtual, private, and public cloud, and at the Edge. RHOCP cluster consists of one or more control-plane nodes and a set of worker nodes.

For more information about RHOCP, see the [Red Hat webpage](#).

Red Hat OpenShift Data Foundation

RHODF is a highly available software-defined, persistent storage and cluster data management solution integrated with and optimized for the Red Hat OpenShift Container Platform. RHODF allows the provision and management of file, block, and object storage for containerized workloads, from the same underlying raw block storage.

RHODF runs anywhere OpenShift runs such as on-premises, private, or public cloud environments, and at the edge, and provides a consistent experience, irrespective of the underlying infrastructure. Organizations can extend and federate data across multiple infrastructures using multicloud data management capabilities.

RHODF is deployed, consumed, and managed through the OpenShift administrator console. This platform is built on Red Hat Ceph storage, offering tightly integrated persistent data services for OpenShift environments. RHODF supports a broad range of diverse workloads and applications including analytics, and AI/ML.

For more information, see the [Red Hat webpage](#).

AMD processor

The 4th Generation AMD EPYC processors have been engineered to provide optimization capabilities across various market segments and applications. By doing so, businesses can effectively allocate data center resources to facilitate additional workload processing and expedite output.

For the SQL Server 2022 database instances, we opted for AMD EPYC 9334 processors in this exercise. This decision was made based on the requirement for fast response time

when running T-SQL queries for data analytics. The AMD EPYC 9334 processors are equipped with 32 cores per socket, operating at a default speed of 2.7 GHz, and offer an L3 cache size of up to 1152 MB per socket. These specifications provide the necessary horsepower to handle any data analytic workloads.

For more information about AMD EPYC, see the [AMD webpage](#).

Microsoft SQL Server 2022

Microsoft SQL Server 2022 is a robust database platform that includes many new features that enhance its capabilities as a leading relational database management system.

The analytics feature of Microsoft SQL Server 2022 is a game-changer for organizations seeking to extract valuable insights from disparate data sources. This feature introduces object storage integration to the data platform, enabling users to seamlessly integrate SQL Server with S3-compatible object storage.

PolyBase technology has been enhanced in SQL Server 2022. This technology significantly reduces data transfer and transformation processes such as extract, transform, and load (ETL) by allowing data to remain in its source location and format. It empowers businesses to run T-SQL queries across relational databases, big data systems, and cloud-based data sources.

For more information about Microsoft SQL Server 2022 and the PolyBase feature, see the [Microsoft webpage](#).

Apache Spark

Apache Spark is an open-source distributed data-processing engine used for big data. It can handle both batch and real-time analytics and data processing workloads. It is designed to deliver the computational speed, scalability, and programmability required for big data applications.

One of the main features of Apache Spark is its in-memory computing which increases the processing speed for any application. Apache Spark is designed to cover a wide range of workloads such as batch processing, interactive queries, and streaming. Apart from supporting all these workloads in a system, it also reduces the management burden of maintaining multiple tools.

For more information about Apache Spark, see the [Spark webpage](#).

Delta Lake

A Delta Lake is an open-source storage layer that brings reliability, security, and performance to the data lake. Delta Lake runs on top of your existing data lake and supports ACID transactions, scalable metadata handling, and unified streaming and batch data processing.

It enables building a Lakehouse architecture with compute engines including Spark, PrestoDB, Flink, Trino, and Hive and APIs for Scala, Java, Rust, and Python.

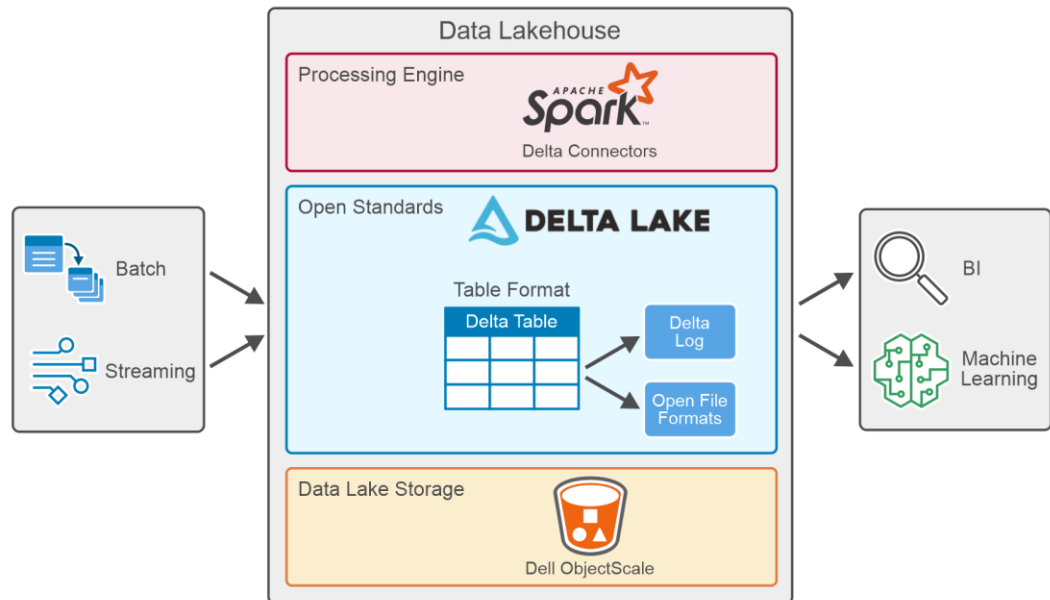


Figure 3. Data Lakehouse data processing

A table in Delta Lake (Delta Table) is stored in open formats such as Parquet files along with a file-based transaction log stored in JSON format, for ACID transactions and scalable metadata handling.

A delta connector is a library that allows external applications to read from and write to Delta tables. Delta connectors are available for various popular big data frameworks, including Apache Spark, Apache Hive, and Apache Flink.

Key features defining Delta Lake include:

- **ACID transactions:** ACID properties ensure that data never falls into a bad data quality because of a transaction that failed or partially completed and provides data reliability and integrity. Delta Lake keeps track of all the commits made to the delta table in a transaction log to implement ACID Transactions.
- **Scalable metadata:** Handles terabytes or even petabytes of table data with ease. Metadata is stored just like any other data and handled by Delta Lake.
- **Time travel (data versioning):** Delta Lake allows users to access and analyze previous versions of data using time travel capabilities for querying, auditing, and rollback. It also enables data exploration and analysis at different points in time, making it easier to identify trends, track changes and perform historical analysis.
- **Unified batch and stream processing:** Every table in a Delta Lake is a batch and streaming sink. Additionally, with efficient metadata handling, ease of scale, and ACID transactions, near-real-time analytics become possible without using a more complicated two-tiered data architecture.
- **Schema enforcement:** Delta Lake automatically handles schema variations to prevent insertion of bad records during data ingestion.

- **Schema evolution:** Delta Lake supports schema evolution, allowing users to evolve the schema of their data over time.
- **Audit history:** Delta Lake logs all changes to the transaction log and provides historical audit trails.
- **DML operations:** Delta Lakes supports DML operations like updates, deletes, and merges, which play a crucial role in complex data operations.
- **Open source:** Community-driven, open standards, open protocol, open file format, and open discussions.

For more information about Delta Lake, see the [Delta IO webpage](#).

Chapter 3 Solution Architecture

This chapter presents the following topics:

Physical architecture18

Logical architecture19

Data access architecture20

Solution requirements21

Physical architecture

Validating a solution architecture for running Microsoft SQL Server 2022 on the RHOCF, RHODF, and Dell ObjectScale object storage is a critical step to ensure a robust and scalable deployment.

Figure 4 provides an overview of the physical architecture that was used in this validation.

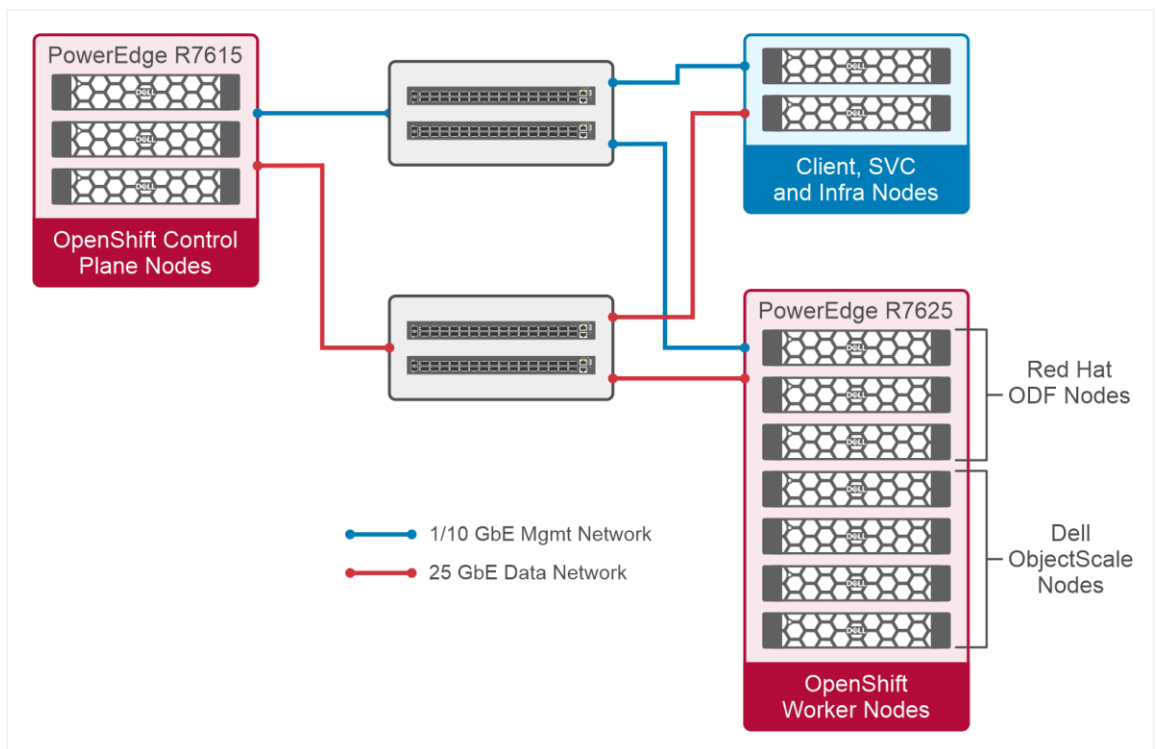


Figure 4. Red Hat OpenShift high level physical architecture

OpenShift Control Plane Nodes

Control plane nodes run the components that are responsible for managing the OpenShift cluster. We used three Dell PowerEdge R7615 Servers as control plane nodes.

Dell PowerEdge R7615 Server is a single socket rack-based server model and powered by AMD EPYC 9124 processor with 16 physical cores. Dell PowerEdge R7615 is designed to be the most economical investment per dollar for the data center and provides better price performance ratio and flexible low-latency storage options. Target workloads include software-defined storage (SDS), virtualization, data analytics.

OpenShift Worker Nodes

Worker nodes run the pods that contain the application and provides compute and storage resources to the applications. We used seven Dell PowerEdge R7625 Servers as worker nodes. Storage of four worker nodes is dedicated for the ObjectScale instance and three worker nodes are dedicated for RHODF.

These worker nodes have enough CPU and memory to handle the compute resources required for SQL analytic workloads. They also provide capacity for both persistent block storage (RHODF) and object storage (Dell ObjectScale).

Dell PowerEdge R7625 is a dual-socket server model and powered by AMD EPYC 9334 processor with 32 physical cores, provides breakthrough performance. Dell PowerEdge R7625 is designed to be the backbone of the data center. These extremely powerful servers provide ample performance and flexible low-latency storage options. Target workloads include High Performance Computing (HPC), Virtual Desktop Infrastructure (VDI), virtualization, and data analytics.

Networking

We used two Dell PowerSwitch S5248F-ON top of the rack (ToR) switches for the data path. The Dell PowerSwitch S5248F-ON provides state-of-the-art, high-density 25 GbE ports and a broad range of functionality to meet the growing demands of today's data center environment. These switches offer flexible options for changing network configurations and adapt to the deployment in a containerized environment. Additionally, we have used two 1GbE/10 GbE management switches for out-of-band connectivity.

Client, SVC, and Infra node

These nodes hosts VMs which run client applications and tools required to manage the Red Hat OpenShift cluster, ObjectScale, Apache Spark and SQL Server. Also, infra VMs host different services such as AD, DNS, NTP.

Logical architecture

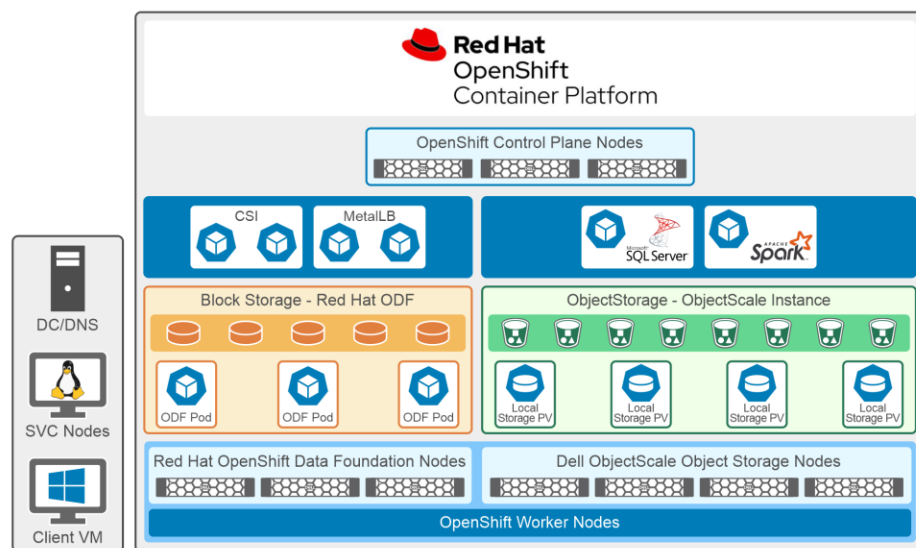


Figure 5. Overview of Red Hat OpenShift logical architecture

Red Hat OpenShift is the foundation of this solution because one of the major focus areas during the development of this solution was to create a cloud-native data analytics solution that could be location agnostic and provide a consistent user experience across on-premises, public, and hybrid cloud, or edge infrastructure.

In this setup, the Microsoft SQL Server 2022 pod will be scheduled by RHOCP in one of the worker nodes. Apache Spark is also deployed as a pod in RHOCP and used in this solution to convert CSV datasets to delta lake format and to store it in Dell ObjectScale. The PolyBase feature in SQL Server facilitates data virtualization and enables SQL Server instances to query data from Dell ObjectScale storage.

SQL instances use RHODF block storage to store Database files. RHODF delivers persistent block storage for Microsoft SQL Server 2022 and supports database availability

requirements. RHODF also provides fault tolerant storage to sustain disk and node level failures.

Dell ObjectScale is a S3 compatible object storage, used in this solution to store and access external delta tables. ObjectScale uses various Erasure Coding (EC) schemes for data protection. EC is a method of data protection in which data is broken into fragments, expanded, and then encoded with redundant data pieces. The data pieces are stored across different locations or storage media. The objective of EC is to enable data that becomes corrupted at some point in the disk storage process to be reconstructed by using information about the data that is stored elsewhere in ObjectScale instance.

MetalLB is a layer 2 load balancer and can be deployed as an Operator in OpenShift, when a service of Type Load Balancer is added to the cluster, MetalLB adds a fault-tolerant external IP address for that service.

Data access architecture

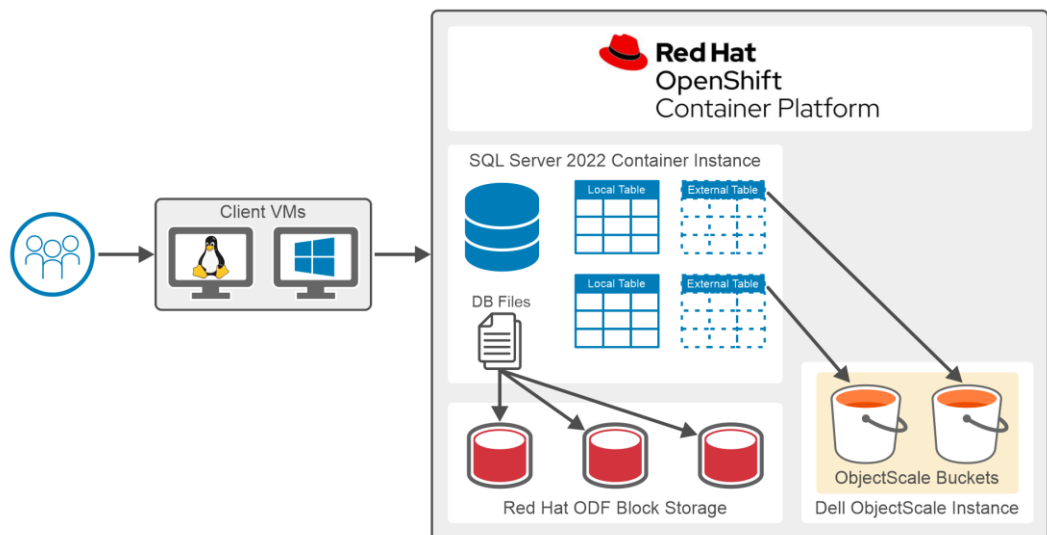


Figure 6. Data access architecture

In this setup, database files for the SQL Server 2022 container instance are stored in RHODF Storage. External Table objects are stored in ObjectScale storage buckets, and PolyBase provides SQL Server the ability to access external tables.

Solution requirements

Hardware Requirements

The following table describes the solution hardware requirements.

Components	3 X Control plane Nodes	7 X Worker Nodes
Chassis	PowerEdge 2U Standard Bezel R7615 with chassis up to 16 SAS4/SATA Drives	PowerEdge 2U Standard Bezel R7625 with chassis up to 16 SAS4/SATA Drives
Processor	1 x AMD EPYC 9124 16C @3000 MT/s	2 x AMD EPYC 9334 32C @2700 MT/s
Memory	4 x 64 GB DDR-5 @4800 MT/s	16 x 64 GB DDR-5 @4800 MT/s
Disk Controller	1 x BOSS-N1 Monolithic (Embedded)	1 x BOSS-N1 Monolithic (Embedded)
	1 x PERC H355 Front (Embedded)	1 x PERC H965i Front (Embedded)
Disks (OS)	2 x 447.13 GB NVMe SSDs	2 x 447.13 GB NVMe SSDs
Disks (Data)		6 X 3576.38 GB SAS SSDs
Nic Card	2 x Broadcom Adv. Dual 25Gb Ethernet	2 x Broadcom Adv. Dual 25Gb Ethernet
Power Supply	Dual, Hot-Plug, Power Supply 2400W Redundant, D Mixed Mode	Dual, Hot-Plug, Power Supply 2400W Redundant, D Mixed Mode
TPM	Trusted Platform Module 2.0 V3	Trusted Platform Module 2.0 V3
iDRAC	iDRAC9, Enterprise 16G	iDRAC9, Enterprise 16G

Software requirements

The following table lists the solution software requirements.

Component	Version details
BIOS	1.3.11
Linux Kernel OS	Red Hat CoreOS 4.12
Red Hat Enterprise Linux Server	9.2
Clustering	Red Hat OpenShift 4.12
Block Storage	Red Hat ODF 4.12
Object Storage	Dell ObjectScale 1.2.1
ObjectScale OS	1.2.0-98.f3c9add
BareMetal CSI	1.2.0-629.b170400
CSI Operator	1.2.0-116.57df4c8
Delta Lake	Delta Core 2.1
Database	SQL Server 2022
Dataset	TPC-DS

Chapter 4 Solution Deployment

This chapter presents the following topics:

Building the OpenShift cluster	23
Red Hat OpenShift cluster deployment	23
Red Hat ODF deployment and configuration	28
Building an ObjectScale Storage	29
Deploying ObjectStore.....	30
Apache Spark deployment	32
SQL Server 2022 deployment.....	33

Building the OpenShift cluster

Overview

Deploying Microsoft SQL Server on the Red Hat OpenShift cluster is a strategic decision for any organization aiming to enhance and modernize their data management infrastructure and database solution.

To ensure proper deployment, it is important to consider the following factors:

- The utilization of an appropriate deployment sequence
- The application of customary best practices
- The connection of solution components with due consideration to network requirements
- The validation of the overall deployment to ensure its expected functionality

This chapter outlines the required procedures for implementing the solution. The deployment process is categorized based on the technologies used in the environment.

Initial setup

There are several components of the solution that must be set up initially.

- **iDRAC setup:** The Integrated Dell Remote Access Controller (iDRAC) is designed to give system administrators increased productivity and to improve the availability of Dell servers. iDRAC reduces the need for physical access to Dell systems by allowing remote management and provides alerts for system issues. Additional information is available in [Dell iDRAC setup webpage](#).
- **BIOS setup:** BIOS versions are available from the [Dell Support Site](#). This solution uses the latest available BIOS; however, it may be necessary to select a different BIOS depending on the environment and workloads an enterprise plans on deploying.
- **Firmware setup:** Firmware requirements will vary based on the components used within physical servers such as the Dell PowerEdge R7615 & R7625. It is important to deploy the appropriate firmware and to [upgrade the firmware](#) to its latest releases.

Dell provides documentation explaining the process of setting up iDRAC and deploying the BIOS and other firmware on Dell servers. See the [Dell PowerEdge R7615](#) and [R7625](#) installation and service manual for detailed instructions.

Red Hat OpenShift cluster deployment

The architectural design outlines the utilization of Red Hat OpenShift 4.12 to establish an environment for hosting containerized applications. This section describes the infrastructure that is required for deploying and operating the Red Hat OpenShift Container Platform. This section also provides steps for deploying a production-ready OpenShift Container Platform on Dell 16G AMD-based PowerEdge servers.

The OpenShift cluster is built using ten 16G Dell PowerEdge Servers. This OpenShift Cluster manages SQL Server and Spark Containers, RHODF, and ObjectScale storage.

OpenShift Container Platform installation

There are four types of installation procedures for the OpenShift Container Platform.

- **Interactive:** Using the web-based Assisted Installer, a cluster deployment can be achieved easily for networks connected to the Internet. The Assisted Installer streamlines the installation process of the OpenShift Container Platform by providing intelligent defaults and conducting pre-flight validations.
- **Local Agent-based:** For environments that are disconnected or have restricted networks, the [Agent-based Installer](#) can be used to set up a local cluster. This installer provides many of the same benefits as the Assisted Installer, but it is necessary to download and configure it beforehand. Configuration is performed using a command-line interface, making it an ideal solution for disconnected environments.
- **Automated:** Clusters can be deployed on infrastructure provisioned by the installer, using the baseboard management controller (BMC) of each host for provisioning. Deployment of clusters is feasible in both connected and disconnected environments.
- **Full control:** Deploying a cluster on self-prepared and maintained infrastructure offers the flexibility in customization. Clusters can be deployed in both connected and disconnected environments.

In our validation, we used an agent-based installation. The “oc install agent create image” subcommand generates a bootable ISO image with all the information required to deploy an OpenShift Container Platform.

We have methods to provide inputs about our environment and required cluster configurations in the form of yaml manifests. We used `install-config.yaml` and `agent-config.yaml` during our deployment. There are some optional ZTP manifests that can be used to provide inputs. For more information about ZTP and installation workflow, see the [Agent-based installer webpage](#).

To deploy Red Hat OpenShift cluster, download [installation and configuration](#) files for bare metal servers.

The following list describes the high-level deployment steps required to set up Red Hat OpenShift cluster:

1. Add entry of your hosts and cluster into your DNS.
2. Download Custom ISO and pull secret from Red Hat for bare-metal deployment.
3. Set and validate node specific networking.
4. Set node and cluster specific configurations with the help of `install-config.yaml` and `agent-config.yaml`.
5. Download and install the client CLI on your service node.
6. Bring all the dependent container images to your local container registry.
7. Deploy Red Hat OpenShift Cluster Platform using CLI.

Red Hat provides a detailed step by step [guideline](#) for the Agent-based installation.

Once Red Hat OpenShift Cluster deployment completes, we can log in to the Red Hat OpenShift cluster dashboard using the temporary “kubeadmin” user credentials.

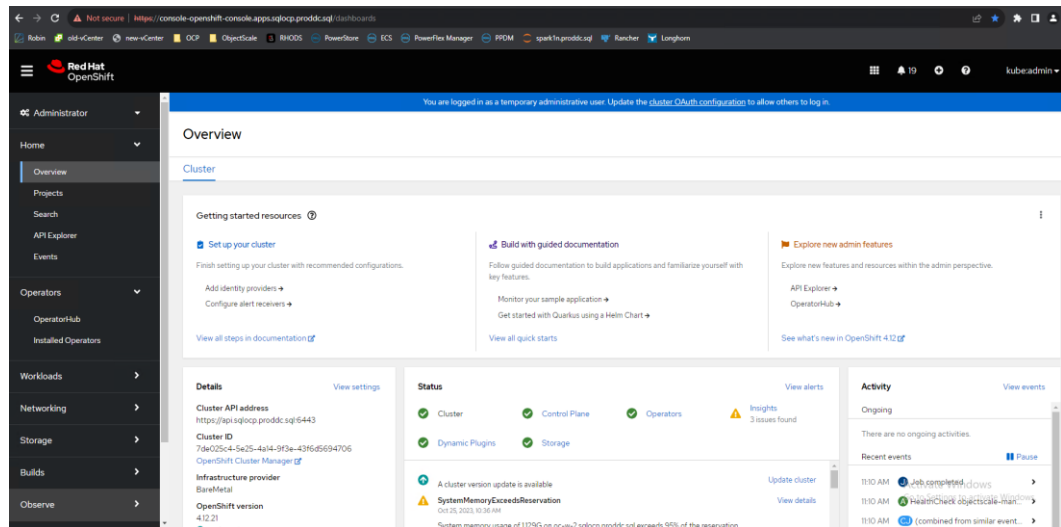


Figure 7. Red Hat OpenShift web console

MetalLB

MetalLB is a LoadBalancer solution designed for bare metal containerized clusters, leveraging widely adopted routing protocols for fault-tolerant external IP address.

Pre-requisites to install MetalLB Operator:

To MetalLB operator in Red Hat OpenShift Cluster using the dashboard, use the following procedure:

1. Log in as a user with **cluster-admin** privileges.
2. In the OpenShift Container Platform web console, go to **Operators** → **OperatorHub**.
3. Type a keyword into the Filter by keyword box or scroll to find the Operator you want. For example, type **metalb** to find the MetalLB Operator.

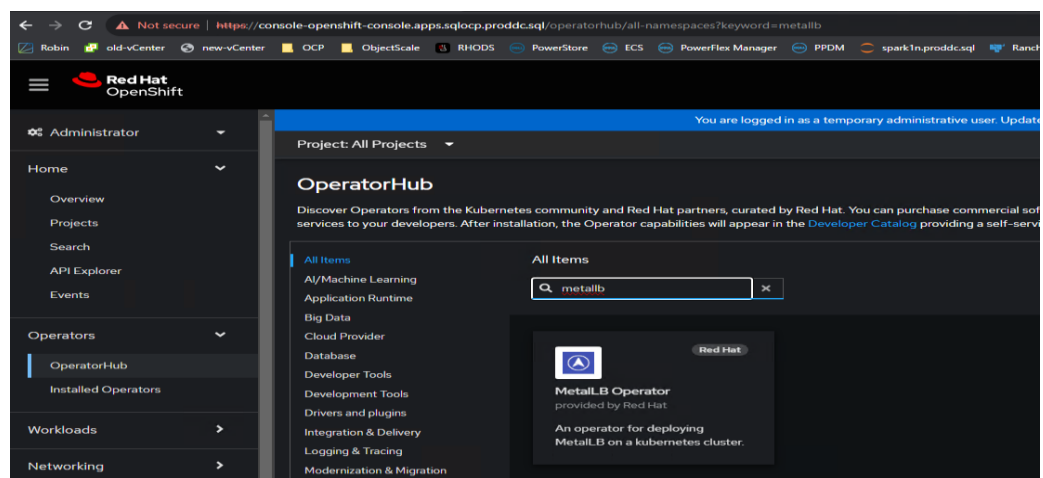


Figure 8. Filtering MetalLB operator in Red Hat OpenShift Cluster using the dashboard

4. On the Install Operator page, accept the defaults, and click Install.

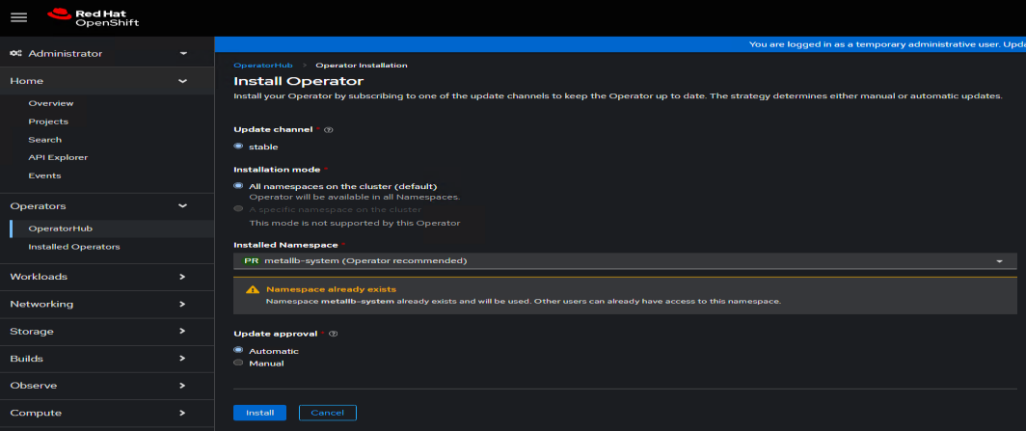


Figure 9. Installing MetalLB operator in Red Hat OpenShift Cluster dashboard

5. Once the operator is installed, you can see the following screen.

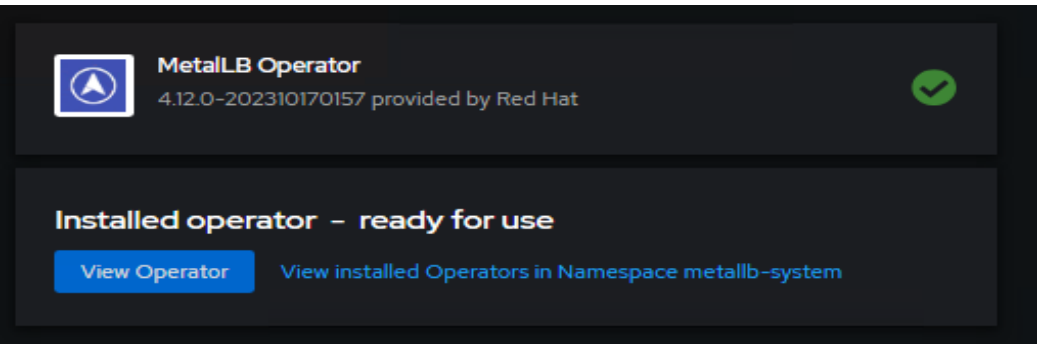


Figure 10. Successful installation of MetalLB operator

6. Create a single instance of a MetalLB custom resource using the Create Instance link for MetalLB as shown below.

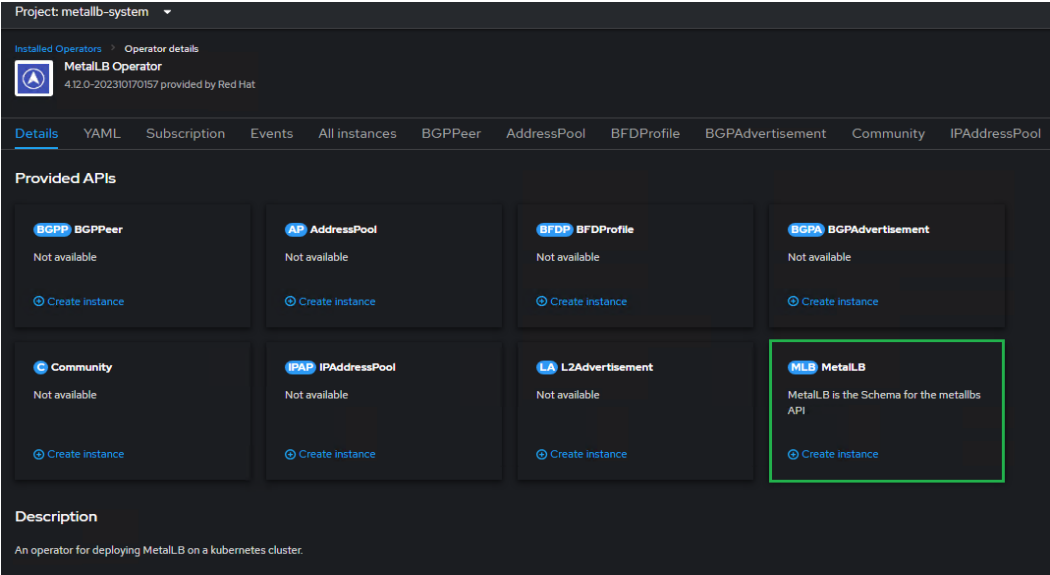


Figure 11. Create MetalLB custom resource

- On the MetalLB create instance page, accept the defaults, and click Install.

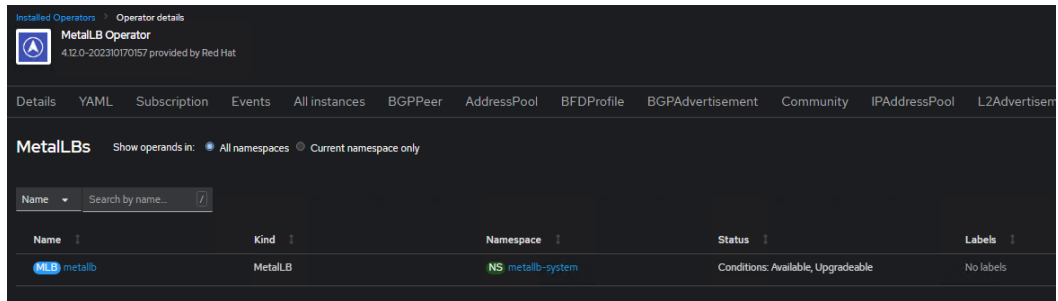


Figure 12. MetalLB create instance

- Create AddressPool by going to the Create Instance link for the addresspool and specifying the yaml manifests as shown below.

Note: Change the IP address range as per your environment.

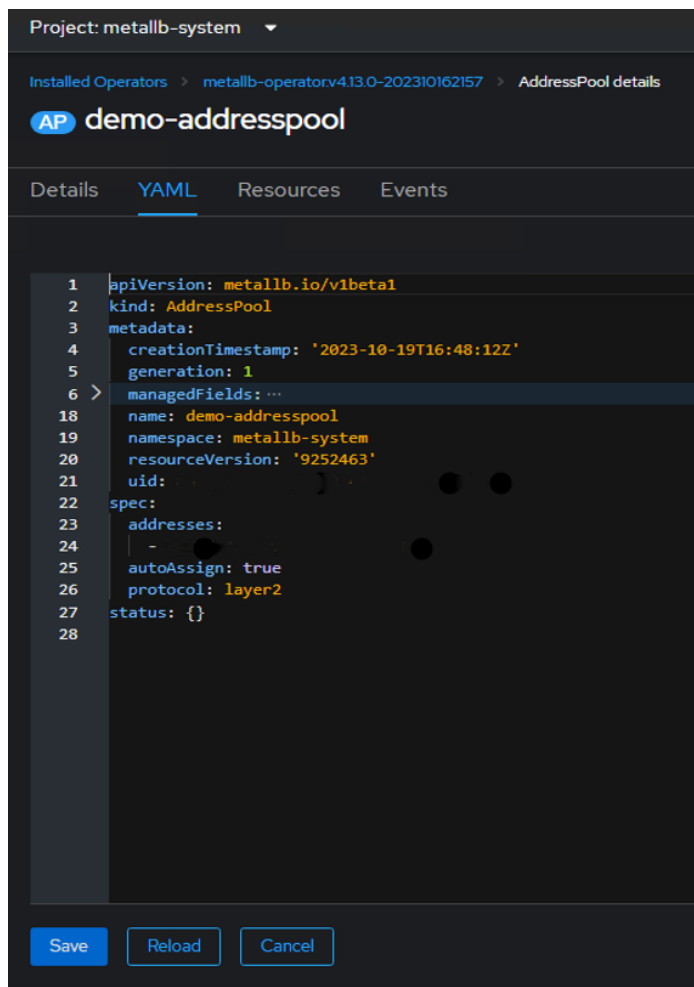


Figure 13. Create addresspool .yaml file

- Finally the MetalLB setup is complete and ready to serve IP from its IP pool whenever we request LoadBalancer Service type.

For more information about MetalLB operator installation, see the [Red Hat OpenShift website](#).

Red Hat ODF deployment and configuration

Overview

The following section provides an overview of RHODF deployment on bare metal infrastructure with an existing installation of RHOCP.

See [Preparing to deploy OpenShift Data Foundation webpage](#) to view the requirements for installing RHODF.

Complete the following steps to successfully deploy RHODF.

1. Install the Local Storage Operator.
2. Install the Red Hat OpenShift Data Foundation Operator from OperatorHub.

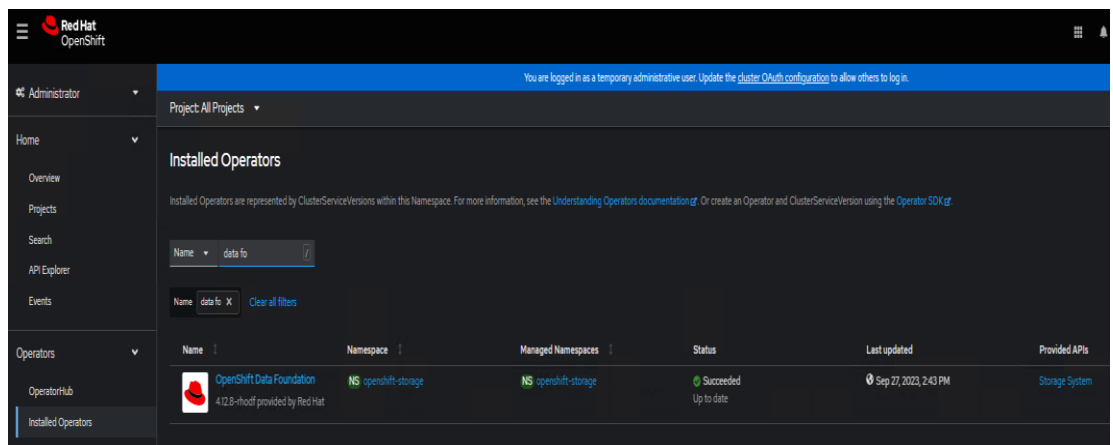


Figure 14. RHODF operator

3. Go to create an OpenShift Data Foundation cluster on bare metal.

The step-by-step procedure to install RHODF on bare metal infrastructure is available on [RHODF webpage](#).

4. Once the RHODF is configured, we can validate using data foundation dashboard, as shown in Figure 15.

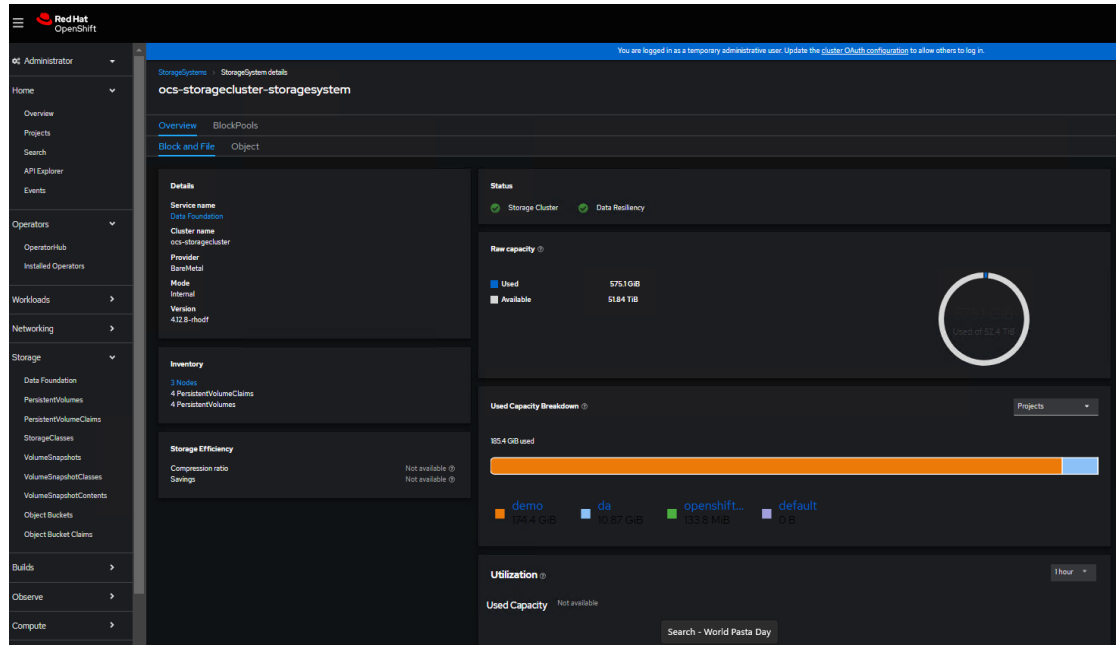


Figure 15. Data foundation dashboard

Verification process for the OpenShift Data Foundation deployment

After the successful installation of RHODF on bare metal, it is essential to verify the provided environment. Verify the following:

- Verify the state of the pods
- Verify that the OpenShift Data Foundation cluster is healthy
- Verify that the Multicloud Object Gateway is healthy
- Verify that the OpenShift Data Foundation specific storage classes exist
- Verify the Multus networking

Building an ObjectScale Storage

Deployment pre-requisites

To successful installation of ObjectScale, it is important to verify that all essential pre-requisites have been fulfilled. This includes:

- Ensure that you have an active account on the [Dell Support](#)
- Run the ObjectScale qualification tool to ensure that the target cluster meets the requirements.
- Linux Orchestration Host requirements host that must have connectivity to the cluster and at least have 150 GB free space available.
- Hardware requirements for the cluster
- Operating system requirements for the cluster
- Other installation considerations like ensuring deployment nodes are networked and accessible to all the target nodes

ObjectScale deployment

For more information about the [ObjectScale qualification tool](#), hardware, and operating system requirements for the cluster, see the [Dell portal](#).

There are two deployment alternatives available for ObjectScale. The first option involves installing ObjectScale as a software bundle on bare metal, using SUSE Linux Enterprise Server. Alternatively, ObjectScale can be deployed onto an existing Red Hat OpenShift Cluster environment. The installation procedures for both options include a qualification check to verify that the underlying platform meets the necessary requirements. For more information, see the installation guide available on [Dell Support](#).

A detailed step by step installation procedure for ObjectScale deployment on the Red Hat OpenShift environment is available on [Dell ObjectScale Application 1.2.x Installation Guide for Red Hat OpenShift](#).

Once the ObjectScale is deployed successfully, we can access the ObjectScale dashboard as shown below.

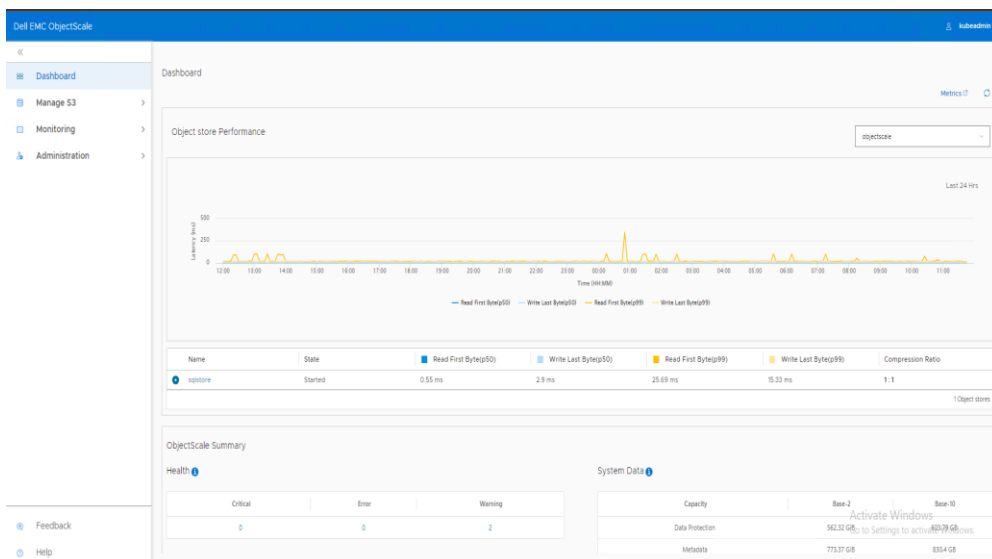


Figure 16. ObjectScale dashboard

Adding ObjectScale License

After ObjectScale deployment completes, it is necessary to apply a valid ObjectScale license to create an object store. To obtain a valid ObjectScale license from Dell, please refer to the [Get Dell ObjectScale license page](#).

To successfully apply and activate the ObjectScale license for your ObjectScale instance, see the [ObjectScale activation steps](#).

Deploying ObjectStore

Overview

The ObjectStore is a conceptual framework that has been implemented in ObjectScale to offer distinct data services to each object store within the ObjectScale. An instance of ObjectScale can accommodate multiple object stores. These stores can be created, updated, and deleted independently from all other objects that are managed by ObjectScale.

While implementing object stores in ObjectScale, it is important to follow the specific naming convention [guidelines](#) for the creation of object stores and group labels.

For detailed instructions to create a new object store, see the steps outlined in the [Create object store](#).

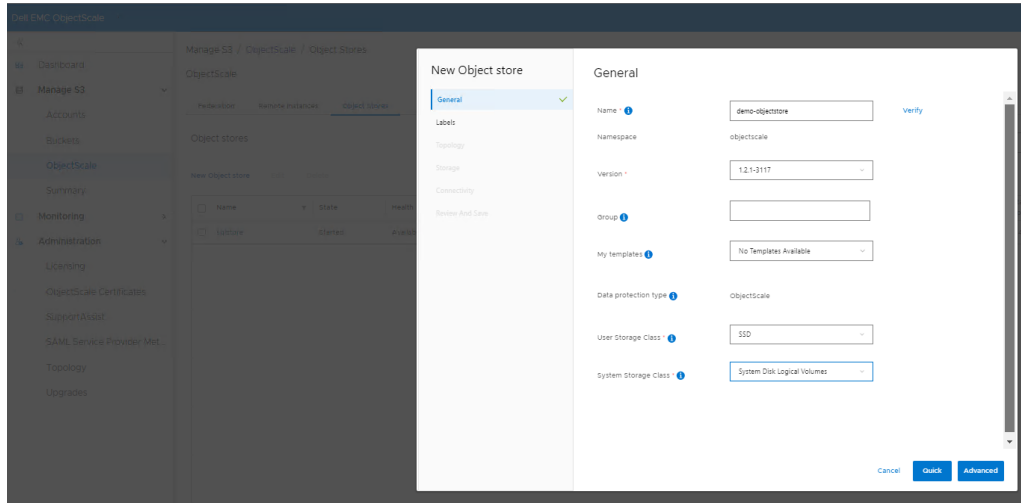


Figure 17. ObjectScale Object storage creation

To enable users within an IAM account for managing various aspects of an object store, it is imperative to associate the IAM account with the object store. Once this association has been established, IAM accounts that are linked to the object store can create new buckets within the store. Additionally, specific compliance settings and retention policies can be configured to meet the wanted requirements.

Buckets are object containers that are used to control access to the objects. For more information, see the [Create a bucket](#) page.

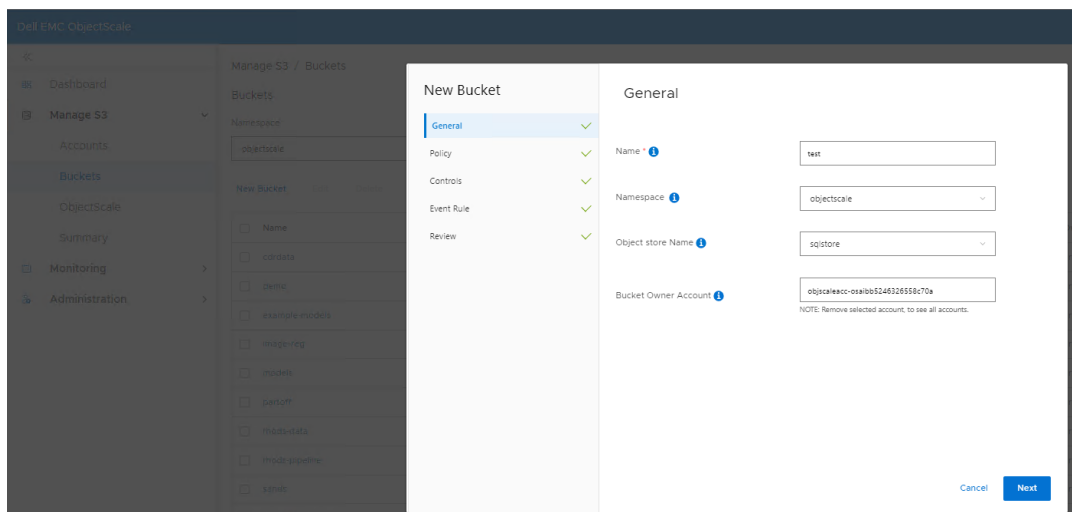


Figure 18. ObjectScale bucket creation

Apache Spark deployment

Overview

To convert data files to the Delta Lake format, we must create an Apache Spark image with Delta Lake drivers.

1. The deployment of a Spark pod in OpenShift is facilitated through the utilization of a yaml manifest file. In this scenario, a custom Spark image is used, which is stored within a private image registry.

```
[root@csah-sec ~]# cat spark.yaml
apiVersion: v1
kind: Pod
metadata:
  name: spark
  labels:
    app: spark
  namespace: objs
spec:
  containers:
    - name: spark
      image: '10.230.87.64:5000/sranjan1/da/spark-cert5:v3.2.1'
      ports:
        - containerPort: 8080
      command: ["/bin/bash", "-c", "sleep 2000000000"]
[root@csah-sec ~]#
```

Figure 19. Spark pod yaml manifest

2. After the pod has been deployed, it can be verified by using the following oc command.

```
[root@csah-sec ~]# oc get pods -n objs
NAME      READY   STATUS    RESTARTS   AGE
spark     1/1     Running   0           10d
[root@csah-sec ~]#
```

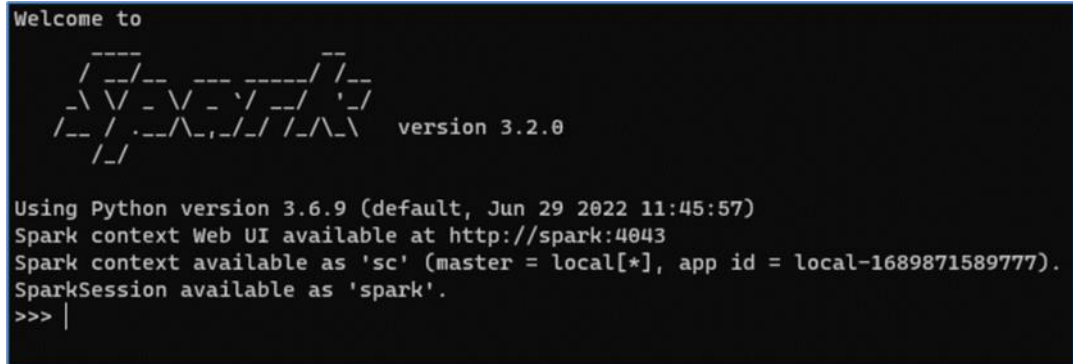
Figure 20. Spark pod status

3. We can now establish a connection to the Spark pod and commence a PySpark shell by running the pyspark command. In this process, we provide the relevant ObjectScale S3 API endpoints, as well as the access and secret keys.

```
./bin/pyspark --packages io.delta:delta-core_2.12:1.1.0,org.apache.hadoop:hadoop-aws:3.2.3 \
--conf spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDriverLogStore \
--conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
--conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog" \
--conf spark.hadoop.fs.s3a.endpoint= \
--conf spark.hadoop.fs.s3a.access.key=OKIA7E4E1A4A7BC31413 \
--conf spark.hadoop.fs.s3a.secret.key=3e+APPUbZSDtAJX4h1WHiE/pWcIXTd0gy68qoeW1 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem \
--conf spark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider
```

Figure 21. Spark running pyspark commands

When the command is successfully run, the Spark login screen will appear, as depicted in the following image.



```

Welcome to
  ____
 /  _ \  /  _ \  /  _ \  /  _ \  /  _ \  /  _ \  /  _ \  /  _ \
/_/_/\_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_
version 3.2.0

Using Python version 3.6.9 (default, Jun 29 2022 11:45:57)
Spark context Web UI available at http://spark:4043
Spark context available as 'sc' (master = local[*], app id = local-1689871589777).
SparkSession available as 'spark'.
>>> |

```

Figure 22. Spark shell

SQL Server 2022 deployment

Overview

The process of implementing SQL Server 2022 within the Red Hat OpenShift cluster is simple and straightforward. The SQL Server 2022 image has been customized to incorporate PolyBase, which facilitates the validation of potential use cases. PolyBase is not enabled by default and has been included in this custom image for data virtualization.

We have developed a customized image that includes the installation of PolyBase along with Microsoft SQL Server 2022 as shown below.

```

FROM ubuntu:20.04
#Create file layout for SQL and set permissions
RUN useradd -M -s /bin/bash -u 10001 -g 0 mssql
RUN mkdir -p -m 770 /var/opt/mssql/security/ca-certificates && chgrp -R 0
/var/opt/mssql/security/ca-certificates
# Installing system utilities
RUN apt-get update && \
    apt-get install -y apt-transport-https curl gnupg2 && \
    curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - && \
    curl https://packages.microsoft.com/config/ubuntu/20.04/mssql-server-preview.list >
/etc/apt/sources.list.d/mssql-server-preview.list
# Installing SQL Server drivers and tools
RUN apt-get update && \
    apt-get install -y mssql-server-PolyBase && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists
RUN /opt/mssql/bin/mssql-conf traceflag 13702 on
# Run SQL Server process as non-root
USER mssql
CMD /opt/mssql/bin/sqlservr

```

Use the following command to build the container image.

```
docker build -t sql-custom-2022:latest
```

Change the image artifacts using the tag command.

```
docker tag sql-custom-2022:latest docker.io/username/sql-custom-2022:latest
```

Then push the newly created container image to container registry.

```
docker push username/sql-custom-2022:latest
```

In this command, replace username with your dockerhub.io public container registry username. This container image can be pulled to run SQL Server 2022 with PolyBase pre-installed on any setup.

Mapping storage certificate with Openshift

The certificate must be mapped so that the Kubernetes cluster can communicate with the Elastic Cloud Storage deployment. This can be done using the Kubernetes ConfigMap, which maps the certificate to a SQL container.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rootca-cert
  namespace: demo
data:
  rootCA.crt: |
    -----BEGIN CERTIFICATE-----
    MIID9zCCAt+gAwIBAgIUKeay3nw4sQe8QHbIIPr8q4vK7AMwDQYJKoZIhvcNAQEL
    BQAwgYoxCzAJBgNVBAYTAIVTMQswCQYDVQQIDAJNQTESMBAGA1UEBwwJSG9wa2
    lu
    dG9uMQ0wCwYDVQQKDAREZWxsMQ8wDQYDVQQLEDAZCaXpBcHAXGDAWBgNVBAM
    MD0hB
    .
    --- certificate content ---
    .
    4/XHKa9DXk8g9pBvSpF7HC1DJRee7ZnJ6p4Vme9LyXjmA+OBzaovOU4i54iKwvkl
    qXXlevx+E0KLy1QbHL9n49F2zi0SZRsjJ5+A5+gCz9CKhLL9b7VvsYqg/Ok40ZT8
    to7ahH31PTtnrzM=
    -----END CERTIFICATE-----
```

The certificate must be mapped so that the OpenShift cluster can communicate with the Dell ObjectScale storage deployment. This can be done using the OpenShift ConfigMap, which maps the certificate to a SQL container.

```
oc create ConfigMap ca-s3lb --from-file=rootCA.crt
```

Persistent volume claim creation

A PersistentVolumeClaim (PVC) is a request for storage by a user. It is like a Pod, as Pods consume node resources and PVCs consume PV resources. Pods can request specific resource amounts (CPU and Memory). Claims can request specific amounts of storage and can be mounted with ReadWriteOnce, ReadOnlyMany or ReadWriteMany access modes.

To provide persistent storage to a container or pod, a persistent volume claim is created using the RHODF storage class "ocs-storagecluster-cephfs". The following script provides an example of how to create a PVC that will be used by the SQL Server pods. This pvc yaml script creates a 500 GB volume in ReadWriteOnce access mode.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: new-sql-pvc-31445
  namespace: demo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Gi
  storageClassName: ocs-storagecluster-cephfs
```

SQL Server 2022 secret creation

A Secret is an object that contains a small amount of sensitive data like a password, token, or key which might otherwise be put in a Pod specification or container image.

Secrets can be created independently of the Pods that use them, reducing the risk of the secrets data being exposed during the workflow of creating, viewing, and editing Pods.

To provide the SA password within the deployment file for the containers, a secret was created in OpenShift's cluster environment using a base 64 encryption for the password.

```
apiVersion: v1
kind: Secret
metadata:
  name: mssql-secret
  namespace: demo
data:
  MSSQL_SA_PASSWORD: *****
type: Opaque
```

SQL Server 2022 deployment

The deployment section provides a declarative update pattern for pods and replica sets. A preferred state is described, and the deployment controller changes the actual state to match the preferred state at a controlled rate. This allows high availability to be maintained, and it is possible to specify when resources should be destroyed, allowing cluster updates to occur without any loss of service in some cases.

This yaml script is used to deploy SQL Server 2022 on a OpenShift cluster.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: new-sql-deployment
  namespace: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mssql-31445
  template:
    metadata:
      labels:
        app: mssql-31445
    spec:
      terminationGracePeriodSeconds: 30
      hostname: mssqlinst
      securityContext:
        fsGroup: 10001
      containers:
        - name: mssql
          image: docker.io/username/sql-custom-2022:latest
          resources:
            requests:
              memory: "128G"
              cpu: "12000m"
            limits:
              memory: "128G"
              cpu: "12000m"
          ports:
            - containerPort: 31445
          env:
            - name: MSSQL_PID
              value: "Developer"
            - name: ACCEPT_EULA
              value: "Y"
            - name: MSSQL_SA_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mssql-secret
                  key: MSSQL_SA_PASSWORD
          volumeMounts:
            - name: mssqldb
              mountPath: /var/opt/mssql
            - name: rootca-cert
              mountPath: /var/opt/mssql/security/ca-certificates/rootCA.crt
              subPath: rootCA.crt

```

```

readOnly: true
volumes:
- name: mssqldb
  persistentVolumeClaim:
    claimName: new-sql-pvc-31445
- name: rootca-cert
  ConfigMap:
    name: rootca-cert

```

Service creation

Services are an abstract way to expose an application running on a set of Pods as a network service. With OpenShift there is no need to modify the application to use an unfamiliar service discovery mechanism. Pods are given IP addresses by OpenShift. Pods can be load balanced and can share a single DNS.

A Service is an abstraction which defines a logical set of Pods and their access policy. The set of Pods targeted by a Service is usually determined by a selector (in this case it is "mssql-31445"). Specific ports can be specified for container port mapping.

```

kind: Service
apiVersion: v1
metadata:
  name: mssql-service-31445
  namespace: demo
spec:
  type: NodePort
  ports:
    - port: 1433
      targetPort: 1433
      nodePort: 31445
  selector:
    app: mssql-31445

```

There are several types of services that can be created such as ClusterIP, NodePort, LoadBalancer, and ExternalName. This example creates a service of type NodePort. More details on the service and service types can be found on the [OpenShift website](#).

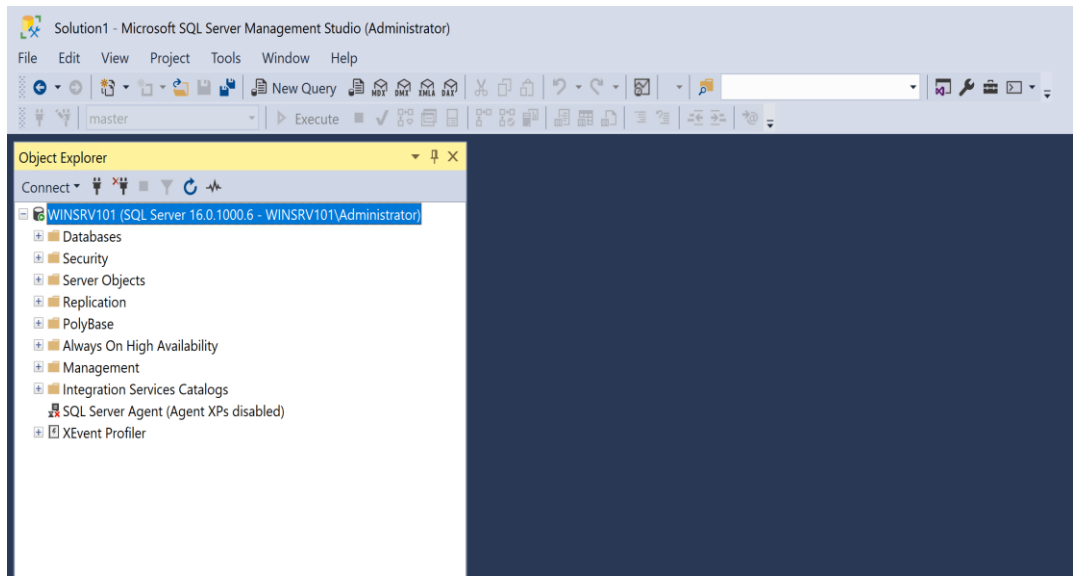


Figure 23. SQL Server SSMS

Once the SQL Server pods are running successfully, these SQL Server pods can be connected to just like any SQL Server instance. The above screenshot shows a connection to a SQL Server 2022 instance running in an OpenShift cluster environment.

Many different tools can be used to connect to a SQL Server instance that is running in a pod. SQL Server Management Studio, PowerShell, Visual Studio, third-party SQL monitoring or development tools and other platforms will connect to the container instance as intended.

Chapter 5 Solution Execution

This chapter presents the following topics:

Enable PolyBase	40
SQL Server 2022 with S3 Object store integration	41
Create TPC-DS table schema in Pyspark	43

Enable PolyBase

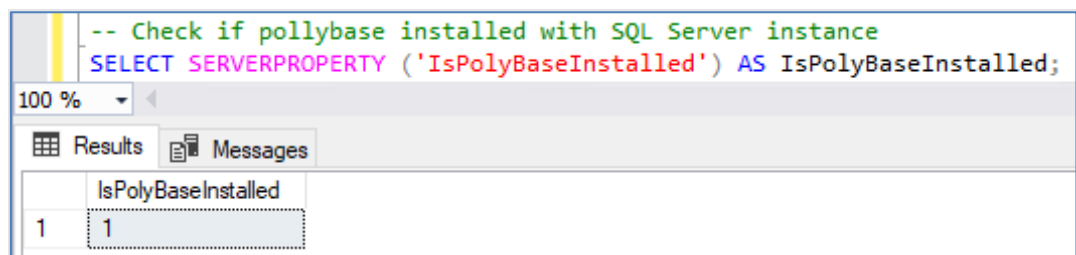
The PolyBase feature of SQL Server 2022 facilitates data virtualization by allowing a SQL Server instance to directly query data using T-SQL from various sources, including SQL Server itself. To ensure seamless functioning of this feature, it is important to install and enable PolyBase on the respective SQL Server instance.

Figure 24 shows how to verify that PolyBase is installed and enabled.

Query:

```
--Check if PolyBase installed with SQL Server instance
SELECT SERVERPROPERTY ('IsPolybaseInstalled') AS IsPolybaseInstalled;
```

Results:



IsPolyBaseInstalled
1

Figure 24. T-SQL Query to verify PolyBase feature installation status

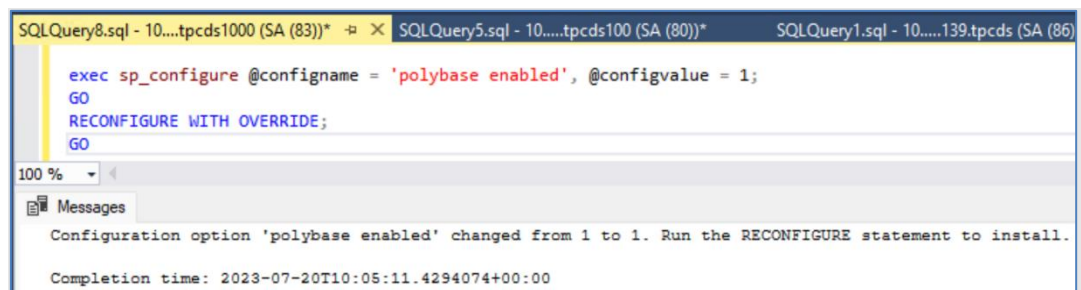
PolyBase feature installation SQL Server instance level verification

Enable the PolyBase functionality at the instance level of the SQL Server, if it is not enabled already.

Query:

```
sp_configure @configname = 'polybase enabled', @configvalue = 1;
RECONFIGURE;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

Results:



```
exec sp_configure @configname = 'polybase enabled', @configvalue = 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

Configuration option 'polybase enabled' changed from 1 to 1. Run the RECONFIGURE statement to install.

Completion time: 2023-07-20T10:05:11.4294074+00:00

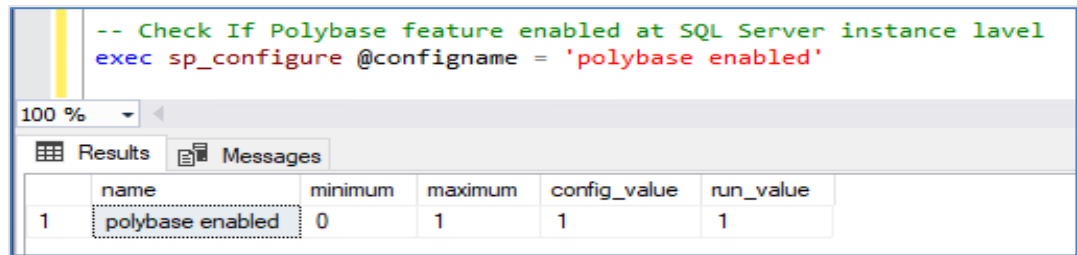
Figure 25. T-SQL Commands to enable PolyBase feature

Verify that PolyBase was enabled successfully.

Query:

```
--Check if PolyBase feature enabled at SQL Server instance level
exec sp_configure @configname = 'polybase enabled';
```

Results:



	name	minimum	maximum	config_value	run_value
1	polybase enabled	0	1	1	1

Figure 26. PolyBase feature verification

SQL Server 2022 with S3 ObjectStore integration

TPC-DS

To validate the results, we used the TPC-DS Decision Support Benchmark, which is widely recognized as an industry standard. This benchmark standard is designed to model various relevant aspects of a decision support system, such as data maintenance and queries. The benchmark offers a comprehensive assessment of performance as a versatile decision support system. It encompasses the measurement of query response time in a single user environment, query throughput in a multiuser environment, and data maintenance performance. These measurements are conducted under controlled conditions, involving a complex, multi-user decision support workload. The benchmark is designed to evaluate the performance of a specific hardware, operating system, and data processing system configuration.

For additional information regarding TPC-DS benchmarking, including table structure, query templates, and specifications, see the [TPC-Benchmark DS – Standard Specifications](#).

We can generate the TPC-DS dataset by using the dsdgen.exe cli tool provided by TPC.org. Figure 27 shows an example of a sample datafile.



```
warehouse_1_126 - Notepad
File Edit Format View Help
1|AAAAAAAAAAAAAAAA|Conventional childr|977787|651|6th |Parkway|Suite 470|Shiloh|San Miguel County|NM|89275|United States|-7|
2|AAAAAAAAAAAAAAAA|Important issues liv|138504|600|View First|Avenue|Suite P|Fairview|Ziebach County|SD|55709|United States|-6|
3|AAAAAAAAAAAAAAAA|Doors canno|294242|534|Ash Laurel|Dr.|Suite 0|Five Points|Ziebach County|SD|56098|United States|-6|
4|AAAAAAAAAAAAAAAA|Bad cards must make.|621234|368|Wilson Elm|Drive|Suite 80|Five Points|Richland County|OH|46098|United States|-5|
5|AAAAAAAAAAAAAAAA||||||Riverside|Fairfield County|SC|29231|United States|
6|AAAAAAAAAAAAAAAA|Local, mass universi|838797|957|Lincoln Adams|Dr.|Suite X|Five Points|Oglethorpe County|GA|36098|United States|-5|
7|AAAAAAAAAAAAAAAA|Quite effectiv|662475|69|7th Sunset|Ct.|Suite 350|Pine Grove|Davie County|MO|64593|United States|-6|
8|AAAAAAAAAAAAAAAA|Plain, reluctant|514427|410|3rd |ST|Suite 370|Shiloh|Ziebach County|SD|59275|United States|-6|
9|AAAAAAAAAAAAAAAA|Rooms cook |73065|420|Spring |Wy|Suite I|Oak Grove|Franklin Parish|LA|78370|United States|-6|
10|AAAAAAAAAAAAAAAA|National, comple|914242|269|Elm Madison|Street|Suite 90|Liberty|Davie County|MO|63451|United States|-6|
11|AAAAAAAAAAAAAAAA|Friendly, suitable |863277|15|7th Park|Ave|Suite 90|Salem|Oglethorpe County|GA|38048|United States|-5|
12|AAAAAAAAAAAAAAAA|New proceedings fo|926078|305|Oak Ninth|Ln|Suite R|Greenwood|Wadena County|MN|58828|United States|-6|
13|AAAAAAAAAAAAAAAA|New, net purposes|758375|728|Johnson |Avenue|Suite 480|Fairview|Williamson County|TN|35709|United States|-5|
14|AAAAAAAAAAAAAAAA|Important dem|185634|288|Hickory Laurel|Parkway|Suite 430|Pleasant Hill|Walker County|AL|33604|United States|-6|
15|AAAAAAAAAAAAAAAA|Please central m|418158|779|Ash Center|Ct.|Suite H|Pleasant Valley|Marshall County|IN|42477|United States|-5|
16|AAAAAAAAAAAAAAAA|Governments shoul|621938|332|Miller Broadway|Way|Suite P|Riverside|Bronx County|NY|19231|United States|-5|
17|AAAAAAAAAAAAAAAA|Months woul|733897|15|Center |RD|Suite 260|Pleasant Grove|Franklin Parish|LA|74136|United States|-6|
18|AAAAAAAAAAAAAAAA|Surprising emot|520093|176|Washington 7th|Wy|Suite F|Salem|Mobile County|AL|38048|United States|-6|
19|AAAAAAAAAAAAAAAA|Empty, middle women |198212|624|Ridge |RD|Suite 160|Pine Grove|Huron County|MI|44593|United States|-5|
20|AAAAAAAAAAAAAAAA|Therefore urg|572104|846|North |Road|Suite Q|Liberty|Davie County|MO|63451|United States|-6|
```

Figure 27. Sample TPC-DS datafile in .dat files with pipe (|) as column separator

Once the datafiles were generated for all tables, they were transferred to the ObjectScale bucket. The aws cli tool was used for file copying process following

```
aws s3 cp C:\tpcds\s1000\data\Final\ s3://tpcds/sf1000/dat/ --recursive --endpoint-url https://1X.XXX.X6.XXX --no-verify-ssl
```

For more information about aws_cli, see the installation or update instructions for the latest version of the AWS CLI.

Figure 28 shows the ObjectScale bucket “tpcds” (under directory /sf1000/dat) using S3 Browser GUI tool. See the [S3 Browser webpage](#) for additional information about the S3 Browser GUI tool.

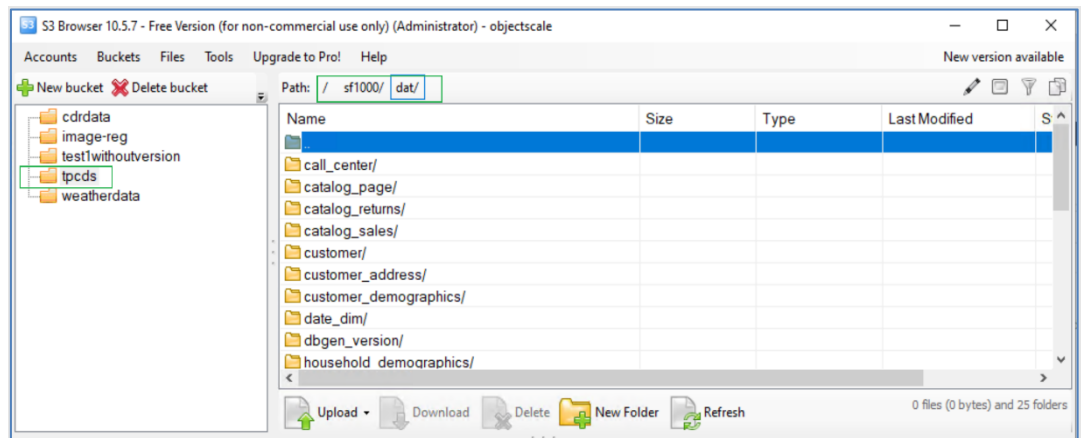


Figure 28. Data present at ObjectScale Bucket

Create TPC-DS table schema in Pyspark

Use the following steps to create a TPC-DS table schema in Pyspark.

1. Define a schema based on the data file, as shown in the following sample script:

```
import sys
import pyspark
from delta import *
from pyspark.sql import functions as F
from pyspark.sql.functions import udf, concat, col, lit
from pyspark.sql.types import StructType, StructField
from pyspark.sql.types import IntegerType, ArrayType
from pyspark.sql.types import StringType, DoubleType
from pyspark.sql.types import DateType, TimestampType

schema_customer_demographics = StructType([
    StructField('cd_demo_sk', IntegerType(), True),
    StructField('cd_gender', StringType(), True),
    StructField('cd_marital_status', StringType(), True),
    StructField('cd_education_status', StringType(), True),
    StructField('cd_purchase_estimate', IntegerType(), True),
    StructField('cd_credit_rating', StringType(), True),
    StructField('cd_dep_count', IntegerType(), True),
    StructField('cd_dep_employed_count', IntegerType(), True),
    StructField('cd_dep_college_count', IntegerType(), True)
])
```

2. After the schema is created, load the data into PySpark data frame as shown in the following script:

```
df_customer_demographics=spark.read.format('csv')
    .option('header', False)
    .option("delimiter", "|")
    .schema(schema_customer_demographics)
    .load("s3a://tpcds/sf10/dat/customer_demographics/*.*)")
```

3. Save this data frame as Delta Lake format on ObjectScale bucket, as shown in the following script:

```
df_customer_demographics.write.mode('overwrite')
    .format('delta')
    .save("s3a://tpcds/sf10/delta/customer_demographics")
```

4. Verify the data in Delta Lake format by utilizing S3 Browser, as illustrated in Figure 29.

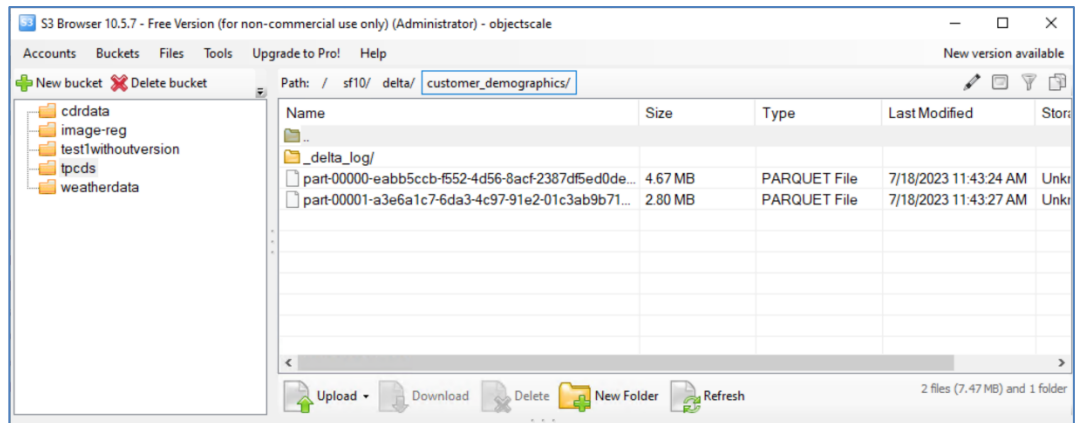


Figure 29. S3 browser bucket view

All the .dat files for each of the 25 tables must be converted in the TPC-DS data set and then ingest them into the ObjectScale bucket.

5. After installing and enabling PolyBase, it is recommended to create an external data source. In this exercise, the external source was established on a Dell ObjectScale object storage. To ensure secure communication between the SQL Server instance and the external data source, an encryption key must be provided. Figure 30 provides an example that demonstrates the process of generating an encryption key and validation of the connection between the SQL Server and the Dell ObjectScale instance.
6. Create an Encryption key with a password in the desired user database.

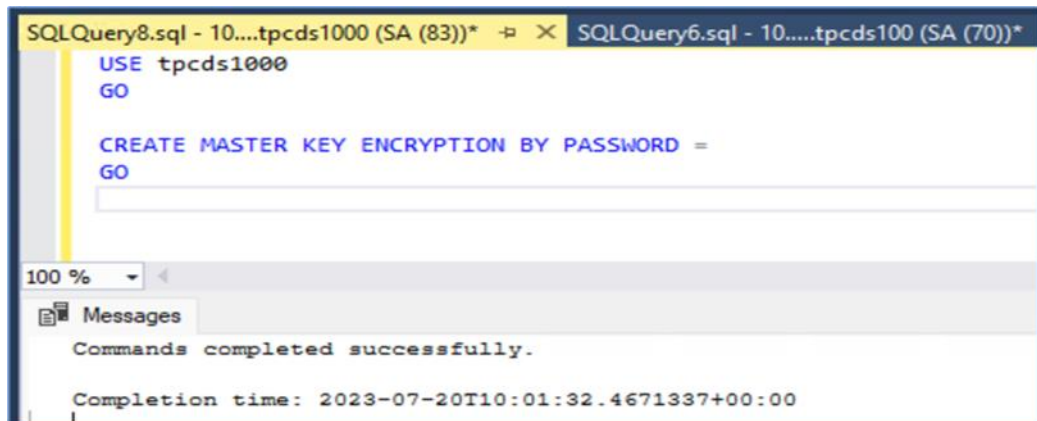


Figure 30. Create encryption key

7. Create the database scope credentials within the wanted user database.

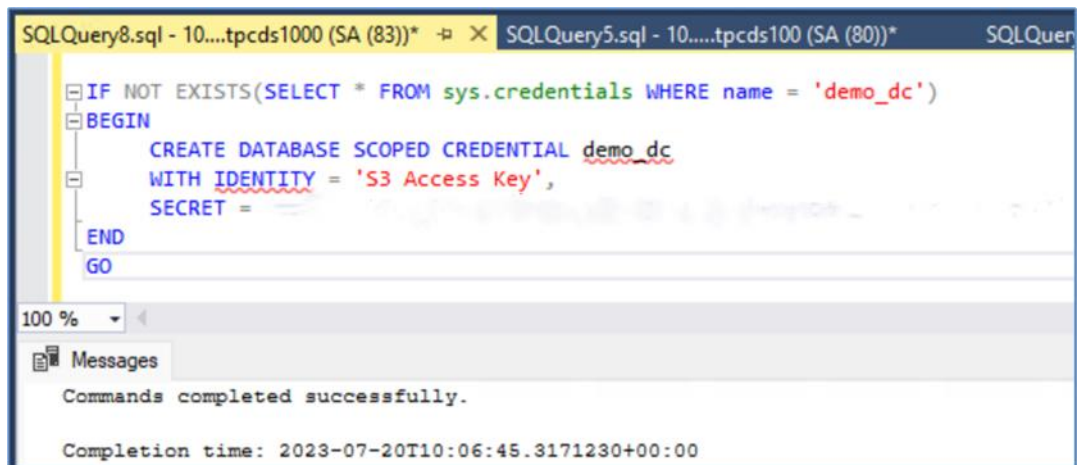


Figure 31. Create database scoped credentials

To establish an external data source, it is necessary to specify the S3 storage URL and the credentials associated with the database.

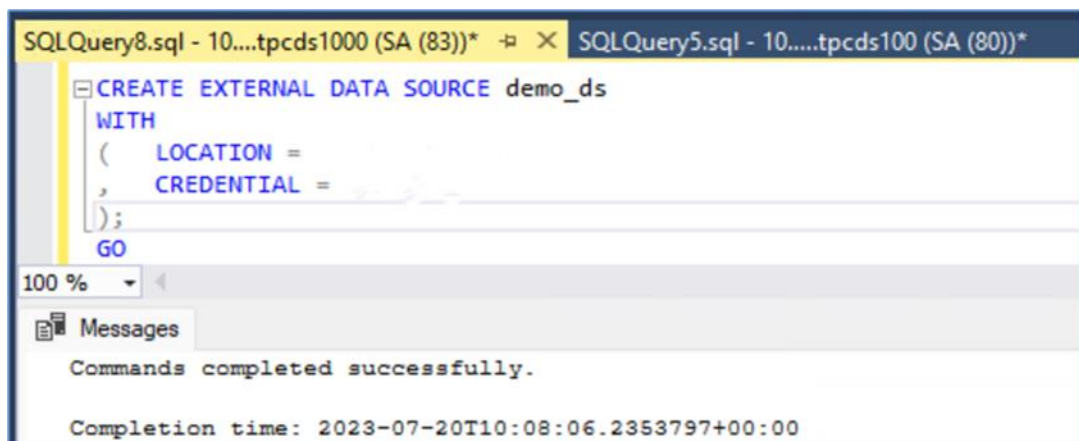


Figure 32. Create external data source

8. Validate data accessibility in Delta Lake table format within the ObjectScale storage using OPENROWSET.

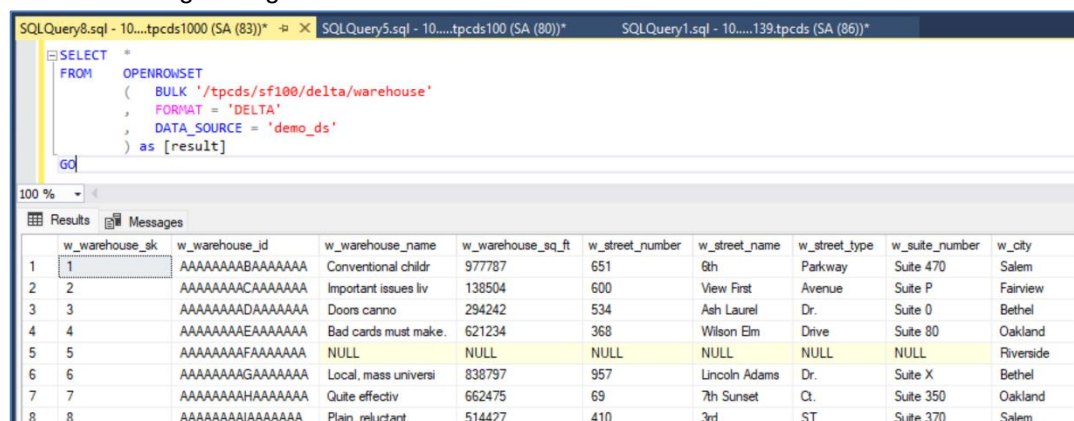


Figure 33. Configure and validate an external data source

SQL Server External Table

The utilization of SQL Server external table streamlines the procedure of handling data beyond the confines of SQL Server. By leveraging PolyBase, the external table can effortlessly retrieve data that is stored externally to SQL Server, such as on ObjectScale object storage in our specific scenario.

Before creating the external table, perform the following steps:

- Create an external file format
- Create an external data source
- Specify the location of the external files

The following figures demonstrate the creation of the external file in Delta Lake file format.

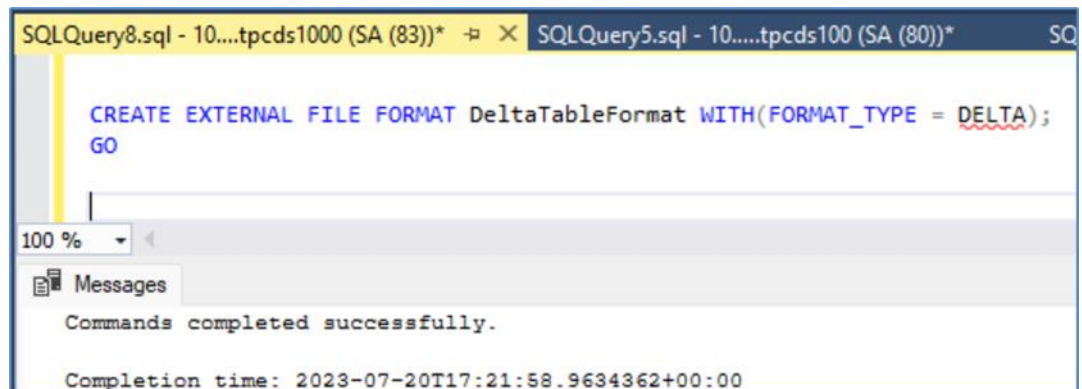


Figure 34. Creating external Delta File Format

After creating the external file in Delta Lake format (DeltaTableFormat), the next step is to create the SQL Server External table. This can be achieved by utilizing the "LOCATION" and "DATA_SOURCE" parameters.

```

SQLQuery8.sql - 10....tpcds1000 (SA (83))*
SQLQuery5.sql - 10....tpcds100 (SA (80))*

create EXTERNAL table customer_address
(
    ca_address_sk          integer          not null,
    ca_address_id          char(16)         not null,
    ca_street_number       char(10)         ,
    ca_street_name         varchar(60)      ,
    ca_street_type         char(15)         ,
    ca_suite_number        char(10)         ,
    ca_city                varchar(60)      ,
    ca_county              varchar(30)      ,
    ca_state               char(2)          ,
    ca_zip                 char(10)         ,
    ca_country             varchar(20)      ,
    ca_gmt_offset          float            ,
    ca_location_type       char(20)         ,
    primary key (ca_address_sk)
)WITH
(
    LOCATION = '/tpcds/sf1000/delta/customer_address'
, FILE_FORMAT = DeltaTableFormat
, DATA_SOURCE = demo_ds
)
GO

```

100 %

Messages

Commands completed successfully.

Completion time: 2023-07-20T17:21:58.9634362+00:00

Figure 35. Creating a SQL Server external table

After the successful creation of the external table, a standard SQL Server "select" statement can be executed like any other tables within the SQL Server database. An illustration of running a "select" statement on the "customer_address" table is depicted in Figure 36

```

SQLQuery8.sql - 10....tpcds100 (SA (83))*
SQLQuery5.sql - 10....tpcds100 (SA (80))*
SQLQuery1.sql - 10....139.tpcds (SA (86))*

SELECT * FROM customer_address
GO

```

100 %

Results Messages

	ca_address_sk	ca_address_id	ca_street_number	ca_street_name	ca_street_type	ca_suite_number	ca_city	ca_county	ca_state	ca_zip	ca_country
1	479176	AAAAAAAAIMPEHAAA	177	Oak	Ct.	Suite 50	Woodlawn	Starke County	IN	44098	Unit
2	479177	AAAAAAAAAJMPEHAAA	764	Jackson	Blvd	Suite 50	Kingston	Scioto County	OH	44975	Unit
3	479178	AAAAAAAAAKMPEHAAA	743	8th Ash	Way	Suite X	Bridgeport	Walsh County	ND	55817	Unit
4	479179	AAAAAAAAALMPEHAAA	958	Spruce Chestnut	Ave	Suite S	Liberty	Otsego County	MI	43451	Unit
5	479180	AAAAAAAAAMMPEHAAA	813	Johnson	WY	Suite 290	Providence	Hopewell city	VA	26614	Unit
6	479181	AAAAAAAAANMPEHAAA	334	Mill	Avenue	Suite G	Woodland	Pottawatomie County	OK	74854	Unit
7	479182	AAAAAAAAAOMPEHAAA	299	Cedar	Drive	Suite E	Oak Grove	Custer County	NE	68370	Unit
8	479183	AAAAAAAAAPMPEHAAA	387	Hill Ash	Lane	Suite 470	Pine Valley	Terrebonne Parish	LA	78209	Unit
9	479184	AAAAAAAAANPEHAAA	879	Poplar 8th	Street	Suite U	Lakeside	Saline County	IL	69532	Unit
10	479185	AAAAAAAAABNPEHAAA	310	Green	Street	Suite 200	NULL	Henry County	GA	32808	Unit
11	479186	AAAAAAAAACNPEHAAA	515	Tenth Meadow	Boulevard	Suite B	Five Forks	Cobb County	GA	32293	Unit
12	479187	AAAAAAAAADNPEHAAA	133	Lakeview 8th	Ln	Suite A	Salem	Carlisle County	KY	48048	Unit
13	479188	AAAAAAAAAENPEHAAA	627	Oak 9th	Parkway	Suite 220	Crossroads	Bryan County	GA	30534	Unit
14	479189	AAAAAAAAAGNPEHAAA	389	Cedar	Drive	Suite J	Berea	Adams County	CO	83804	Unit
15	479190	AAAAAAAAAGNPEHAAA	941	Cedar	Way	Suite 430	Springdale	Orange County	NY	18883	Unit
16	479191	AAAAAAAAAHNPEHAAA	97	East	Avenue	Suite C	Glendale	Marion County	TX	73951	Unit
17	479192	AAAAAAAAAINPEHAAA	182	7th	Dr.	Suite H	Highland ...	McCook County	SD	56534	Unit
18	479193	AAAAAAAAAJNPEHAAA	735	Sunset Washi...	Court	Suite X	Springdale	Genesee County	MI	48883	Unit
19	479194	AAAAAAAAAKNPEHAAA	403	College 1st	Ave	Suite D	Sunnyside	Piscataquis County	ME	02552	Unit
20	479195	AAAAAAAAALNPEHAAA	425	Spring	Cr.	Suite J	Walnut G...	Brazos County	TX	77752	Unit
21	479196	AAAAAAAAAMNPEHAAA	909	Fifth Hickory	Ave	Suite 170	Brownsville	Eau Claire County	WI	53910	Unit
22	479197	AAAAAAAAANNPEHAAA	142	Jackson 10th	Ave	Suite 30	Oak Grove	Clark County	AR	78370	Unit
23	479198	AAAAAAAAAONPEHAAA	585	5th	Road	Suite A	Salem	Custer County	SD	58048	Unit
24	479199	AAAAAAAAAFNPEHAAA	200	Willow Adams	NULL	NULL	NULL	NULL	NULL	NULL	Unit
25	479200	AAAAAAAAAOPEHAAA	851	Green 1st	Boulevard	Suite U	Adrian	Warren County	NY	13301	Unit
26	479201	AAAAAAAAAOPPEHAAA	686	1st	Ct.	Suite 150	Midway	Cheshire County	NH	02504	Unit

Query executed successfully. 10.230.96.139 (16.0 RC0) | SA (83) | tpcds100 | 00:00:14 | 1,000,000 rows

Figure 36. Select data from external table backed by delta table present on ObjectScale

All 25 TPC-DS tables have been established as External tables within SQL Server. Consequently, the TPC-DS data is now stored externally to SQL Server, with solely the metadata for these tables remaining within the SQL Server instance. The next step is to run all 99 standard TPC-DS queries and conducting the necessary data analytics in accordance with the TPC-DS specifications.

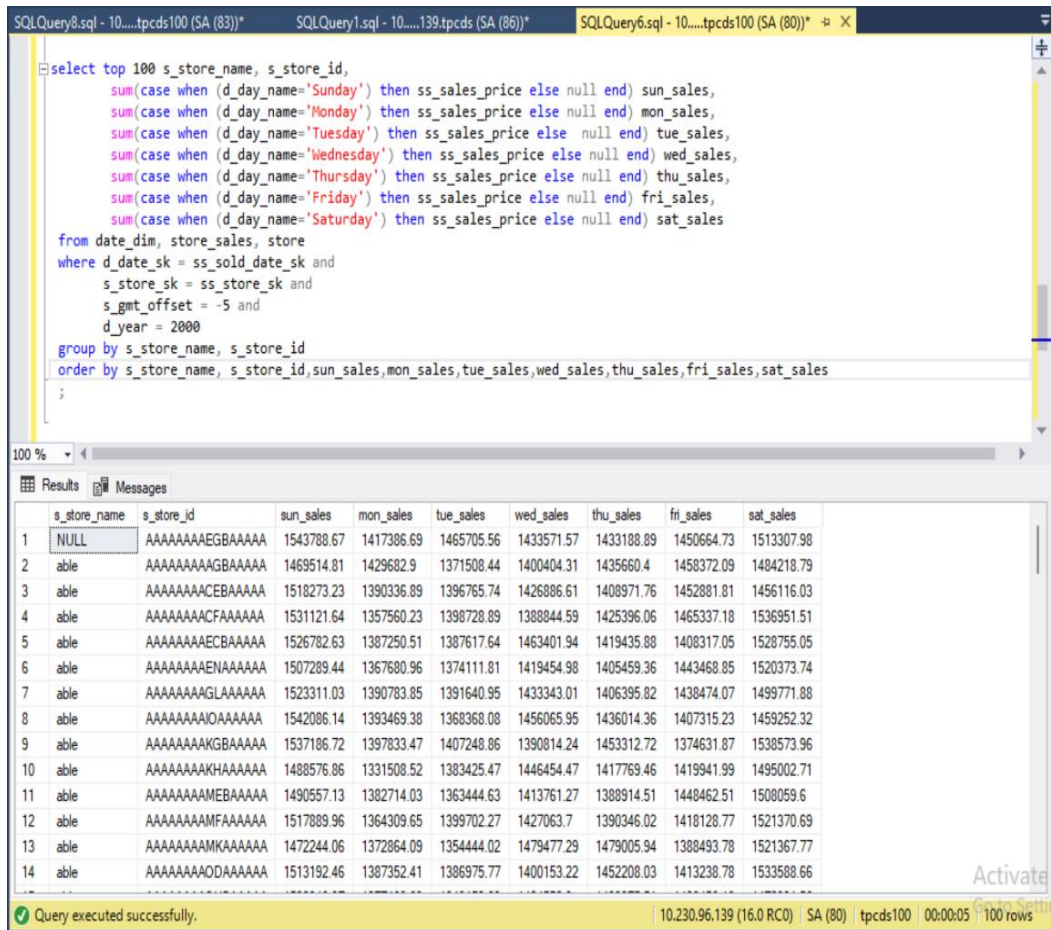


Figure 37. Sample SQL Query

Figure 37 illustrates one of the standard TPC-DS queries, among the 99 queries, along with its corresponding result. The query in question utilizes the date_dim (consisting of 73,049 records), store_sales (comprising 28,79,97,024 records), and store (containing 402 records) tables, all of which are located on ObjectScale.

The execution plan for this query demonstrates that plan includes a "remote scan" operator showcasing actual data coming from outside SQL Server at run time, as indicated in Figure 38.

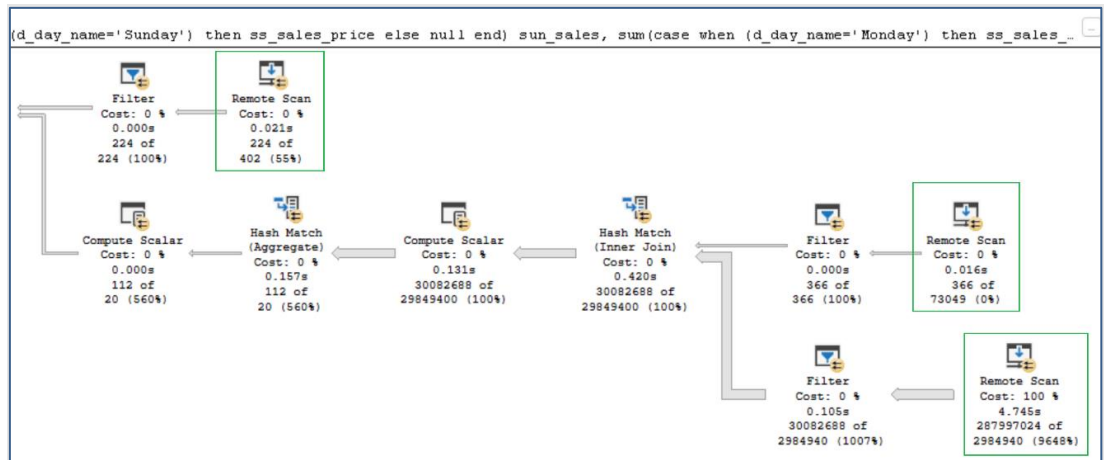


Figure 38. SQL execution plan

Chapter 6 Summary and Conclusion

This chapter presents the following topics:

Summary51

We value your feedback51

Summary

The Dell Validated Design for Data Analytics with SQL Server 2022 on Red Hat OpenShift and Dell ObjectScale has been developed to address the needs of organizations that have to develop and run cloud-native data analytics solutions using SQL Server 2022.

Both large and small organizations can benefit from the new features introduced in SQL Server 2022. Leveraging Red Hat OCP and ODF for containerized SQL Server 2022 databases with PolyBase features and with Dell ObjectScale storage brings numerous benefits to organizations seeking to optimize their data storage and analytics processes.

Combining the power of SQL Server's advanced analytics capabilities and the scalability and durability of Dell ObjectScale storage, organizations can store and process vast amounts of data directly on object storage without the limitations of traditional file systems.

In addition, embracing the powerful 4th Gen AMD EPYC 9334 processors on Dell PowerEdge servers gives organizations a competitive edge in their analytic workloads. The combination of these products not only delivers deeper insights quicker for business stakeholders, but also provides a platform that is cloud-ready, highly available, flexible, and scalable.

Building, deploying, and managing cloud-native applications can be complex. In this validated design we have provided design guidance on deploying a containerized SQL Server 2022 data analytics solution on Red Hat OpenShift and Dell ObjectScale.

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by [email](#).

Author: Sanjeev Ranjan, Sunil P, Balakrishnan R

Contributors: Ava English

Note: For links to additional documentation for this solution, see the [Dell Technologies Solutions Info Hub for SQL Server](#).

Chapter 7 References

This chapter presents the following topics:

Dell Technologies documentation	53
AMD documentation	53
Microsoft documentation	53
Red Hat documentation	53
Additional documentation	53

Dell Technologies documentation

The following Dell Technologies documentation provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell Technologies representative.

- [Dell ObjectScale Solution Overview](#)
- [Dell ObjectScale: Best Practices on OpenShift](#)
- [Dell EMC ObjectScale Application 1.2.x Installation Guide for Red Hat OpenShift](#)
- [Dell ObjectScale 1.2.x Administration Guide](#)
- [Dell ECS and Microsoft SQL 2022 S3 Object Storage](#)
- [PowerEdge](#)

AMD documentation

The following AMD documentation provides additional information about AMD EPYC 9334 Processors.

- [AMD EPYC 9334 Processor](#)

Microsoft documentation

The following Microsoft documentation provides additional and relevant information:

- [Data Virtualization with PolyBase for SQL Server 2022](#)
- [Microsoft SQL Server 2022](#)

Red Hat documentation

The following Red Hat documentation provides additional information about RHOCP and RHODF:

- [Red Hat OpenShift Container Platform](#)
- [Red Hat OpenShift Data Foundation](#)

Additional documentation

The following documentation provides additional information about Delta Lake, TCP-DS, and AWS CLI.

- [TPC-DS](#)
- [Delta Lake](#)
- [AWS CLI](#)
- [S3 Browser](#)