RING8
SCALABLE STORAGE

VEEAM SCALITY

WHITE PAPER

Using Veeam with Scality RING

VEEAM READY Object with Immutability

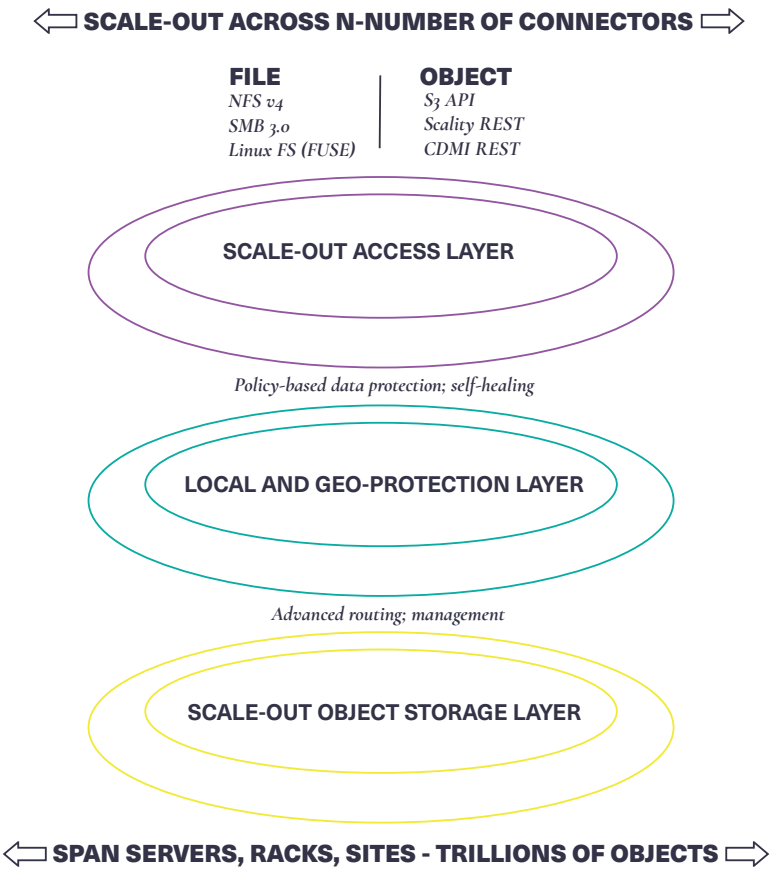
VEEAM READY Object

January 2022

I. Solution Overview

Scality RING architecture

The Scality RING software is a software-defined storage (SDS) solution that turns a pool of x86-based Linux servers into an unbounded scale-out storage system that supports object and file-based applications and a variety of use cases. The Scality RING architecture consists of three different layers: scale-out access layer, local and geo-protection layer, and the Scale-out bbject storage layer.



To scale both storage capacity and performance to massive levels, Scality RING software is designed as a distributed, parallel, scale-out architecture with a set of intelligent services for data access and presentation, data protection and systems management.

To scale both storage capacity and performance to massive levels, Scality RING software is designed as a distributed, parallel, scale-out architecture with a set of intelligent services for data access and presentation, data protection and systems management. To implement these capabilities, RING provides a set of fully abstracted software services including a top-layer of scalable access services (connectors) that provide storage protocols for applications. The middle layers

are comprised of a distributed virtual file system layer, a set of data protection mechanisms to ensure data durability and integrity, self-healing processes and a set of systems management and monitoring services. At the bottom of the stack, the system is built on a distributed storage layer comprised of virtual storage nodes and underlying IO daemons that abstract the physical storage servers and disk drive interfaces.

Scale-out access layer

Applications communicate with Scality RING via this layer. RING connectors provide multiple interfaces to access the storage, including object-based connectors such as S3, sproxyd (a native key/value REST API), and a scale-out file system connector (SOFS) that provides NFS, SMB and FUSE access. Connectors are also responsible for implementing the configured data protection storage policy (replication or erasure coding).

Local and geo-protection layer

This layer of the RING contains a set of data protection mechanisms to ensure data durability and integrity, self-healing processes, and a set of systems management and monitoring services.

The replication mechanism is used to store multiple copies of a file within the RING for durability and availability. When a file is written, RING will spread these replicas across multiple storage nodes and disk drives in order to protect them from common failures.

Erasure coding stores files durably with space efficiency (reduced overhead relative to replication) via an extended set of parity “chunks” instead of multiple copies of the original file as with replication. When an erasure coded file is broken into multiple chunks, a mathematical formula is applied to produce an additional set of parity chunks used to tolerate disk or storage node failures.

When a failure happens on the RING, such as with a disk or node, background rebuild operations are initiated to restore the missing data from its surviving replicas/erasure coding chunks. RING also has an intelligent routing mechanism that will proxy connections around component failures until they are back online to provide continuous service availability.

RING8
SCALABLE STORAGE

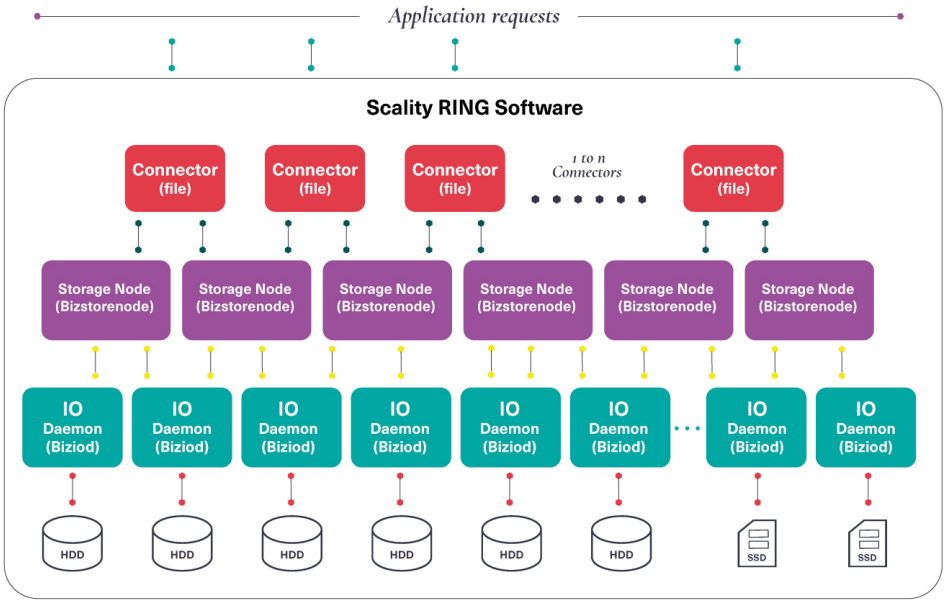
Scale-out object storage layer

At the heart of the storage layer is a scalable, distributed object key/value store based on a second generation peer-to-peer routing protocol that ensures store and lookup operations scale efficiently to large numbers of storage nodes.

RING is a fully distributed system deployed on industry standard hardware, starting with a minimum of three (3) storage servers. The system can be seamlessly scaled out to thousands of servers with hundred's of petabytes of storage capacity.

Each storage server is configured with six (6) storage node software processes. Under the storage node software processes are storage daemons. Each storage daemon is a low-level process that manages IO operations to a particular physical disk drive and maintains the mapping of object indexes to the actual object locations on disk. A typical configuration is one storage daemon per physical disk drive with support for up to hundreds of daemons per server. This allows the system to support very large and high-density storage servers.

Scality RING software components



RING software processes: RING connectores, storage and IO daemons

RING8
SCALABLE STORAGE

RING software is comprised of the following main components: RING connectors, an internal distributed NoSQL database, RING storage nodes and IO daemons, and the Supervisor web-based management portal.

RING has no single points of failure and requires no downtime during any upgrades, scaling, planned maintenance or unplanned system events. With self-healing capabilities, it continues operating normally throughout these events. To match performance to increasing capacity, RING can also independently scale-out its access layer of protocol "connectors" to enable an even match of aggregate performance to the application load.

RING provides data protection and resiliency through local or geo-distributed erasure coding and replication, with services for continuous self-healing to resolve expected failures in platform components such as servers and disk drives. RING is fundamentally built on a scale-out object storage layer that employs a second-generation peer-to-peer architecture. This approach uniquely distributes both user data and associated metadata across the underlying nodes to eliminate the typical central metadata database bottleneck. To enable file and object data in the same system, RING integrates a virtual file system layer through an internal NoSQL scale-out database system, which provides POSIX-based access semantics using standard NFS, SMB and FUSE protocols with shared access to the files as objects using the REST protocol.

RING connectors

Connectors provide the data access endpoints and protocol services for applications that use RING for data storage. As a scale-out system, RING supports any number of connectors and endpoints to support large and growing application workloads.

Connectors provide storage services for read, write, delete and lookup for objects or files stored into the RING based on either object or POSIX (file) semantics. Applications can make use of multiple connectors in parallel to scale out the number of operations per second or the aggregate throughput of the RING. A RING deployment may be designed to simultaneously provide a mix of file access and object access (over NFS and S3, for example) to support multiple application use cases.

Storage nodes and IO daemons

The heart of the RING are the storage nodes—virtual processes that own and store a range of objects associated with its portion of the RING's keyspace. Each physical storage server (host) is typically configured with six (6) storage nodes processes (termed bizstorenode). Under the storage nodes are the storage daemons (termed biziod), which are responsible for persistence of the data on disk in an underlying local standard disk file system. Each biziod instance is a low-level software process that manages the IO operations to a particular physical disk drive and maintains the mapping of object keys to the actual object locations on disk. Biziod processes are local to a given server, managing only local, direct-attached storage and communicating only with storage nodes on the same server. The typical configuration is one biziod per physical disk drive, with support for up to hundreds of daemons per server so the system can support very large, high-density storage servers.

Each biziod stores object payloads and metadata in a set of fixed size container files on the disk it is managing. With such containerization, the system can maintain high-performance access, even to small files, without any storage overhead. The bizoid deamons typically leverage low-latency flash (SSD or NVMe) devices to store the index files for faster lookup performance. The system provides data integrity assurance and validation through the use of stored checksums on the index and data container files, which are validated upon read access to the data. The use of a standard file system underneath biziod ensures that administrators can use normal operating system utilities and tools to copy, migrate, repair and maintain the disk files if required.

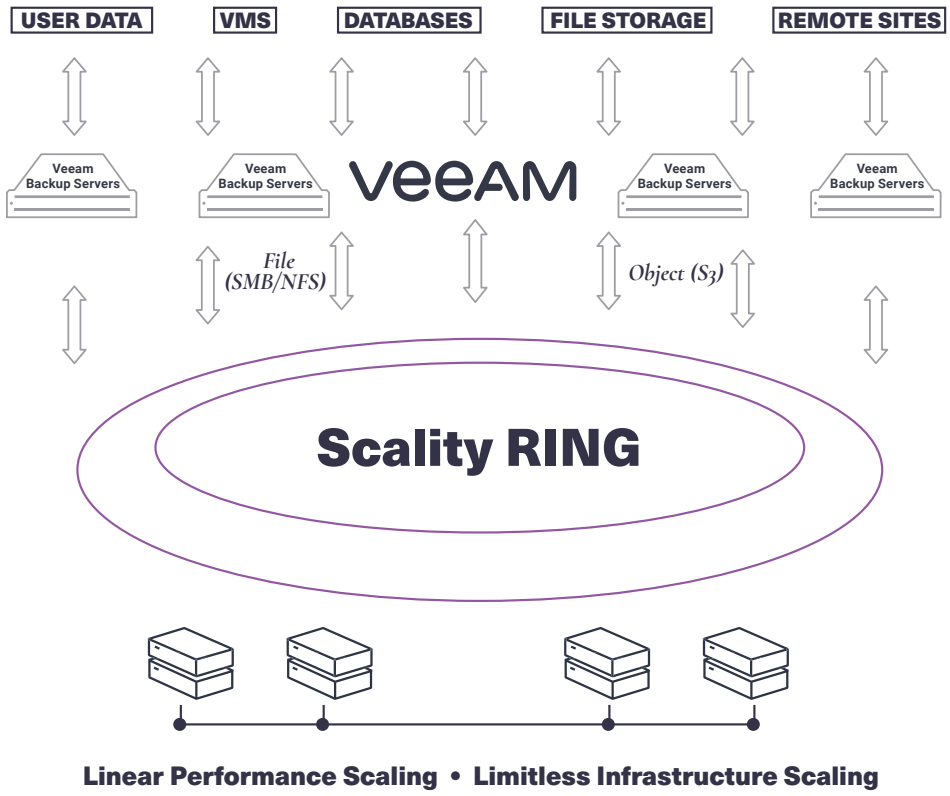
RING systems management

Managing and monitoring the RING is enabled through a cohesive suite of user interfaces built on top of a family of RESTful interfaces termed the Supervisor API ("SupAPI"). The SupAPI provides an API-based method that may be accessed from scripts, tools and frameworks for gathering statistics, metrics, health check probes and alerts, and for provisioning new services on the RING. The SupAPI is also enabled with role-based access control (RBAC), by supporting an administrator identity to provide access control privileges for super admin and monitor admin user roles.

RING8
SCALABLE STORAGE

RING provides a family of tools that use the SupAPI for accessing the same information and services. RING includes “Scality supervisor,” a browser-based portal for both systems monitoring and management of Scality components. The supervisor provides capabilities across object (S3) and file (NFS, SMB, FUSE) connectors, including integrated dashboards and key performance indicators (KPIs) with trending information such as “global health,” “performance,” “availability” and “forecast.” The supervisor also includes provisioning capabilities to add new servers in the system and a zone management module to handle customer failure domains for multi-site deployments.

II. Combined Workflow



Linear Performance Scaling • Limitless Infrastructure Scaling

III. Using Veeam with Scality S3

Scality S3 has been verified as a Veeam Ready - Object target. Verified object storage solutions have been tested with Veeam Backup and Replication Cloud Tier features.

Configuring Scality S3

S3 connector provides an AWS S3- and IAM-compatible interface to Scality RING with support for core AWS S3 bucket and object APIs, including multipart upload (MPU). Scale-out capability enables concurrent same-bucket access from multiple S3 connector instances for both read and write operations.

HTTPS

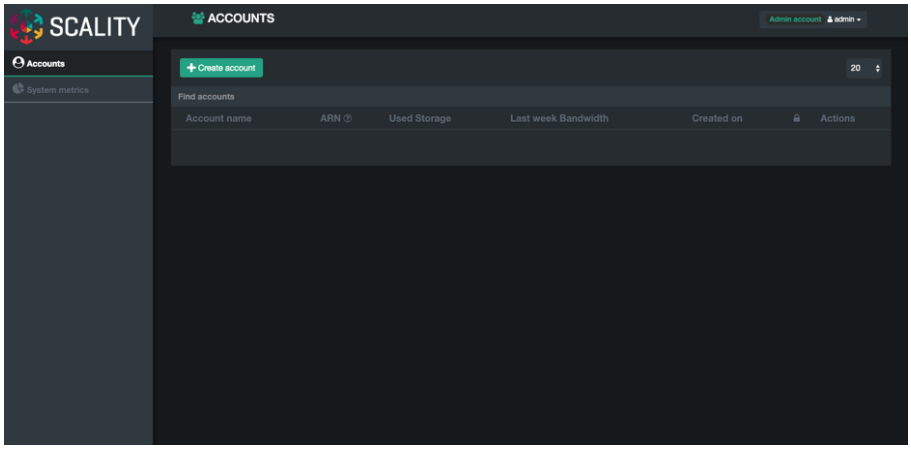
Veeam requires a TLS/HTTPS encrypted service endpoint for all S3-compatible targets. If a front load balancer is deployed, please refer to the load balancer documentation for setting up SSL termination. If end to end encryption is required or round robin DNS is being used in place of a load balancer, refer to the Operating S3 Connector manual for configuration instructions.

Credentials

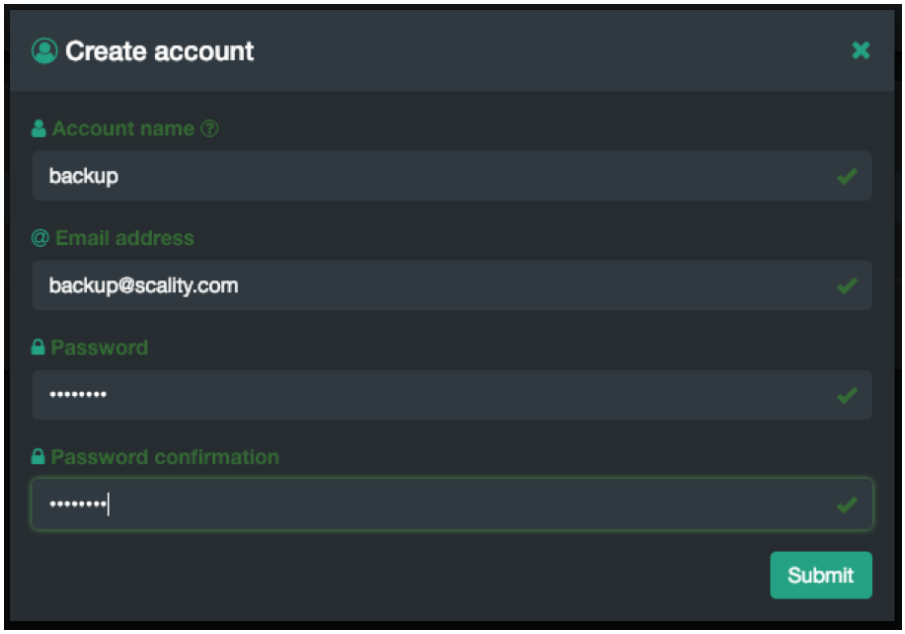
S3 accounts and users can be generated through the S3 Console.

1. Navigate to **https://<S3 endpoint URL>/_/console** and login with your administrator username and password.

2. In the top right of the account management page, click on **Create Account.**

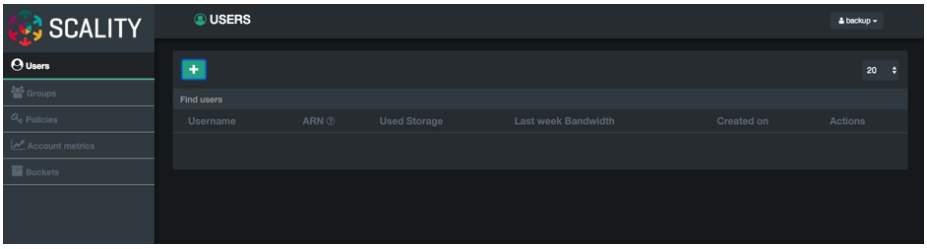


3. Create a new backup account. Provide an account name, email address and password then click **Submit.**

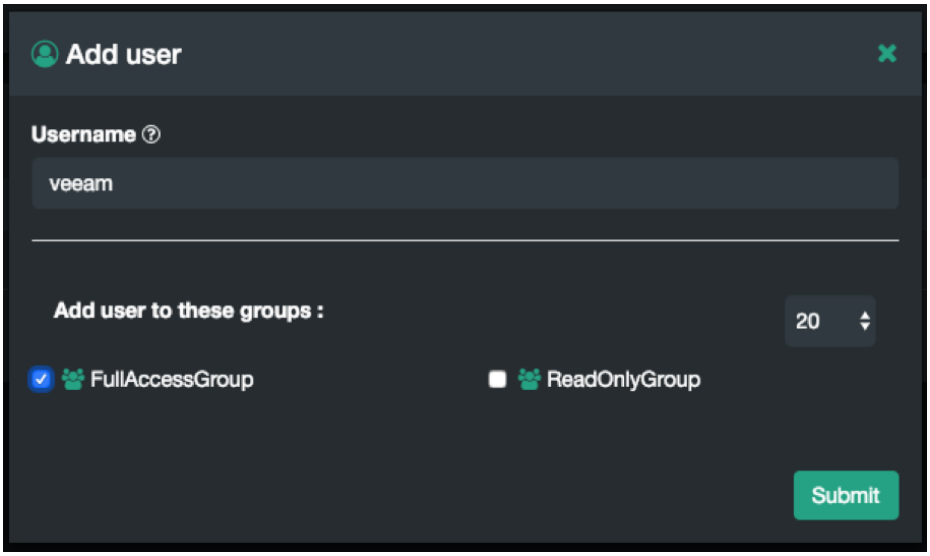


4. Log out of the S3 Console and log back in using the backup account name and password.

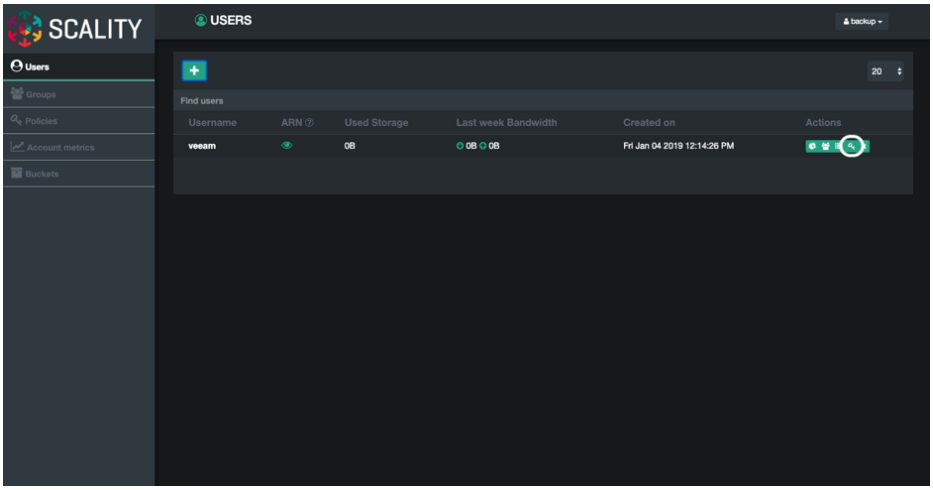
5. In the top right-hand corner, click the + button to create a new user.



6. Provide a username and add the user to the FullAccessGroup, then click **Submit.**

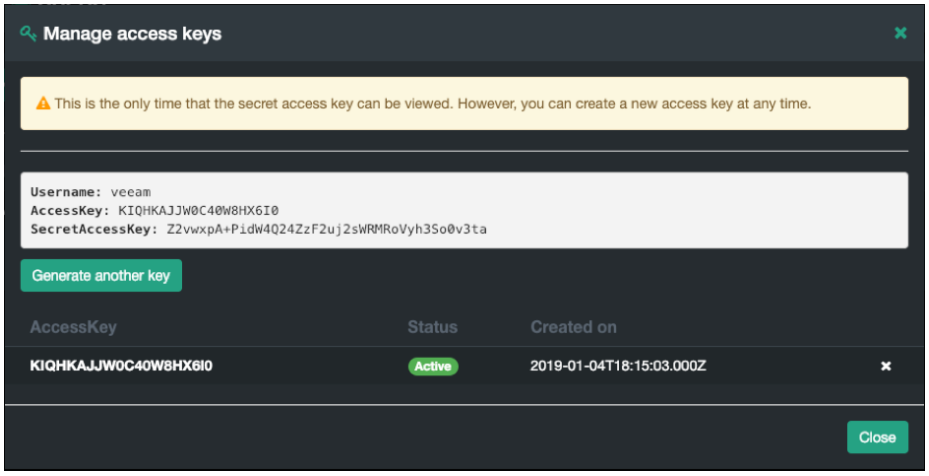


7. To generate the secret and access key for the Veeam user, click the **key** button next to the user name.



8. Click the **Generate a new key** button and follow the prompts to complete the request.

9. This is the only time that the secret key can be viewed. Be sure to make a copy of the key in a safe location.



10. Provision an Object Lock enabled bucket:

a) Create aws cli profile

```
[centos@gateway ~]$ aws configure --profile veeam

AWS Access Key ID [None]: H60L05DSR0WO3C4QTYSV

AWS Secret Access Key [None]:
**************************************
Default region name [None]:

Default output format [None]:
```

b) Create bucket

```
[centos@gateway ~]$ aws --profile veeam --endpoint-url
https://s3.isv.scality.com s3api create-bucket --bucket
veeam-bucket --object-lock-enabled-for-bucket

{
“Location”: “/veeam-bucket”
}
```

Configuring Veeam

Capacity tier respository

A new repository type is available for Backup & Replication V10, which is called the Capacity Tier Repository. The Capacity Tier Repository is an extension of Veeam's Scale-Out Backup Repository (SOBR) feature. The SOBR we are about to create will include the primary backup repository and the Scality S3 connector repository.

1. Under **Backup Infrastructure** in Navigation Pane, click on **Add Repository** and for the Scality S3 connector, choose **Object Storage.**

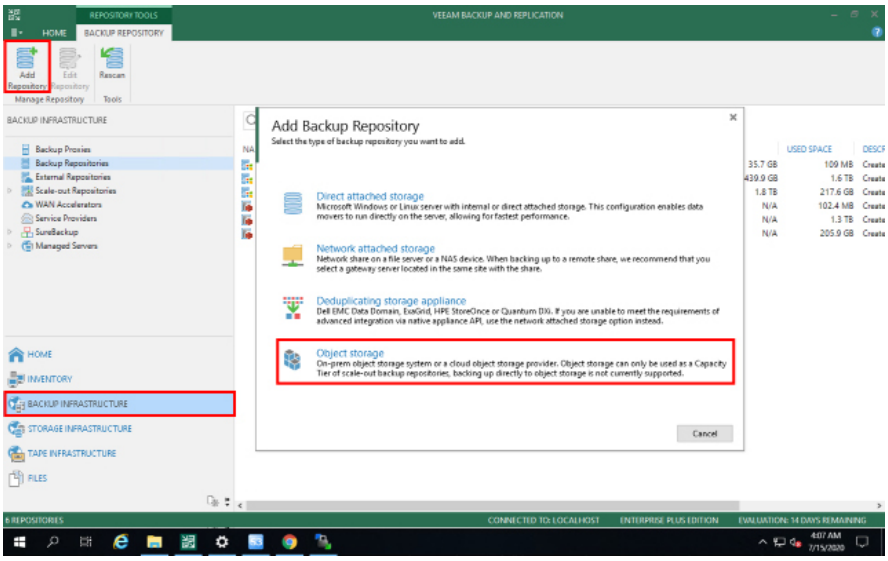


Figure 1 - Create Object Repository

2. From the **Object Storage** screen, select **S3 Compatible.**

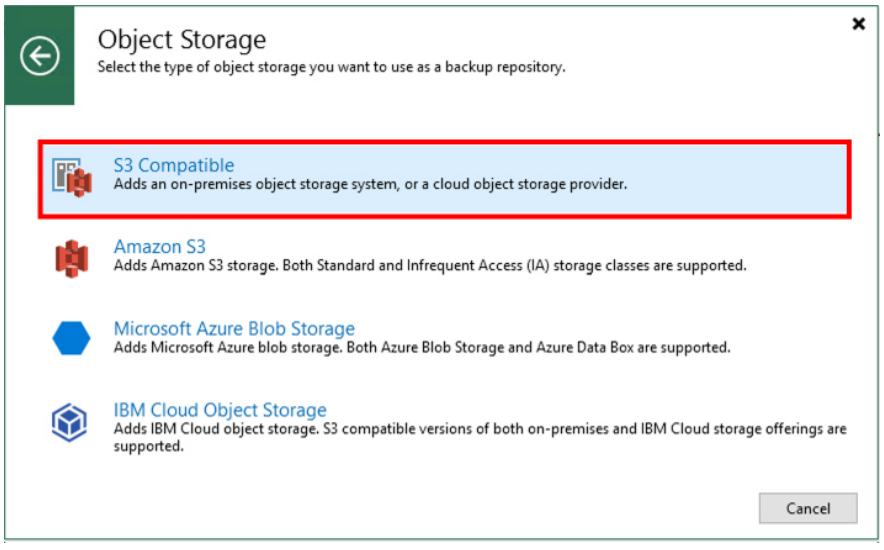


Figure 2 - Object Storage Type

3. Enter a name and description of the new Object Storage Repository. Click the **Next** button to continue.

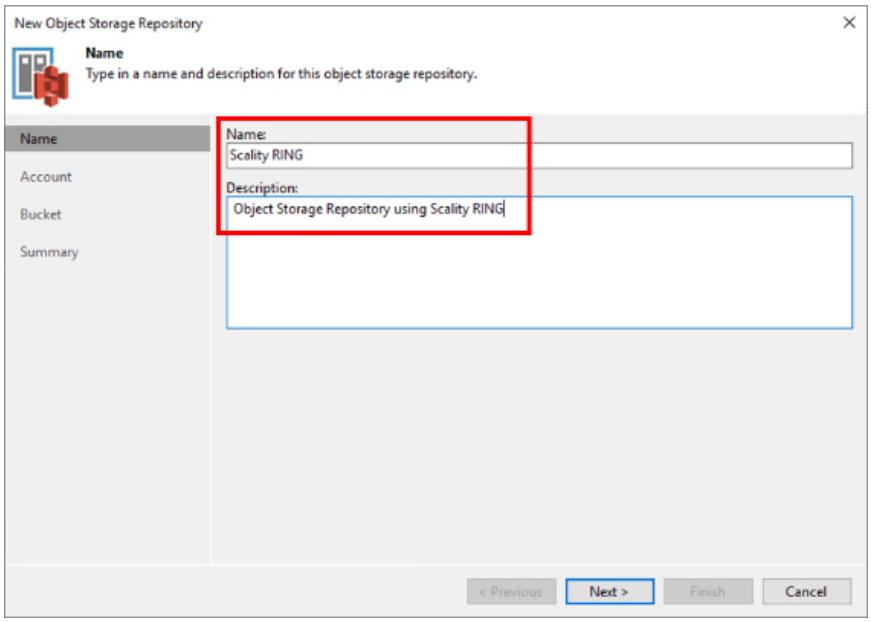
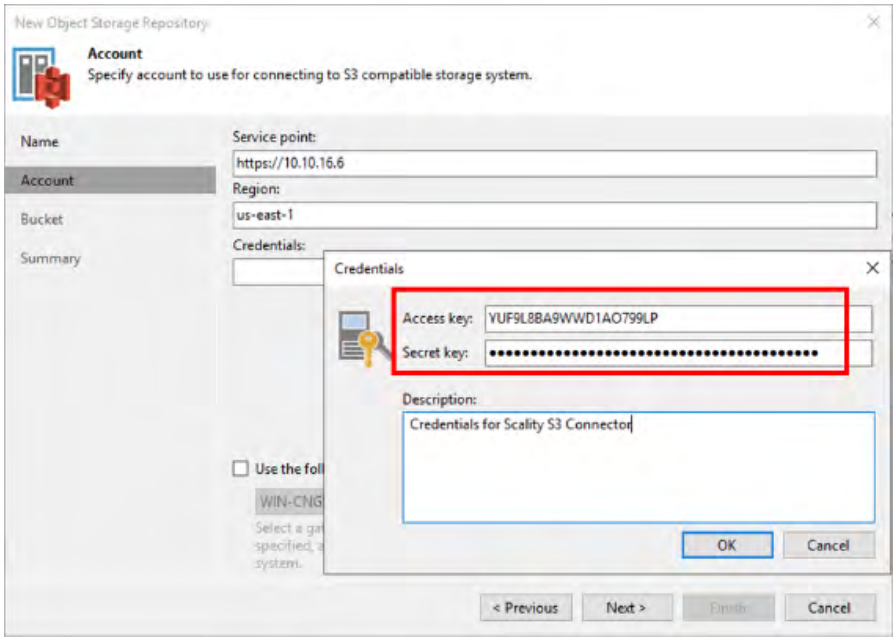


Figure 3 - Object Repository Name and Description.

4. Enter the HTTPS endpoint address and region for your Scality S3 connector (be aware that Veeam only supports TLS/HTTPS encrypted targets). To add credentials, click on the **Add** button and enter the **Access Key** and **Secret key** for your Scality S3 connector and click **OK.**



Leave the **Use Gateway Server** box unchecked and click **Next.**

5. On the **Bucket screen,** choose the Bucket and Folder where your backups will be stored. Check the box “Make recent immutable for xx days” to protect backups stored on Scality RING from malicious modification. Click the **Next** button to proceed. Review the Summary screen and click **Finish** to create the Object Storage Repository.

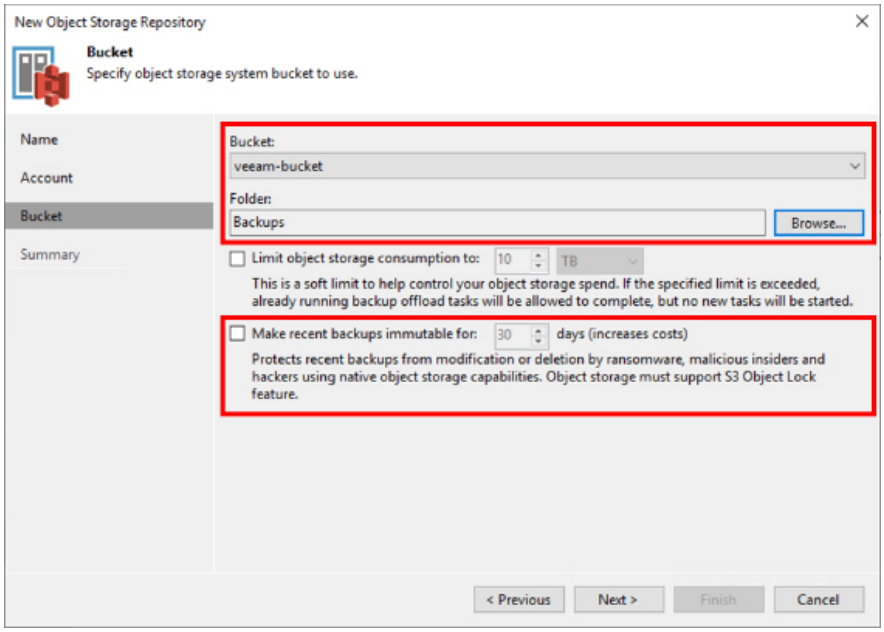


Figure 5 - Object Repository Bucket

6. Under **Backup Infrastructure** pane, click on **Scale-out Repositories** and click the **Add Scale-out Repository** button on the top menu bar. Type in a name and description for this new scale-out repository, which will use the Scality S3 connector for the Cloud Tier target. Click **Next** to continue.

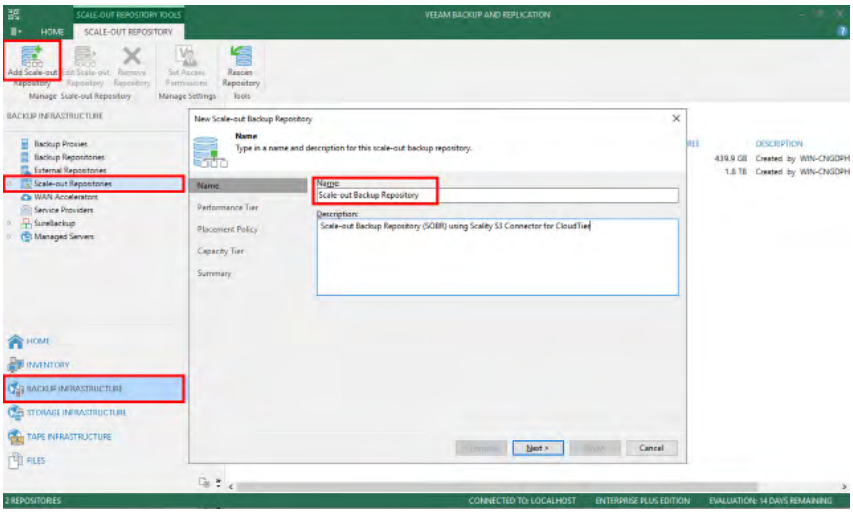


Figure 6 - Scale Out Repository

7. On the **Extents** screen, click on the **Add** button to add an extent to the scale-out repository. Add the primary repository (Performance Tier) that was created for your backups. Click **OK** to add the extent, and click **Next** to continue.

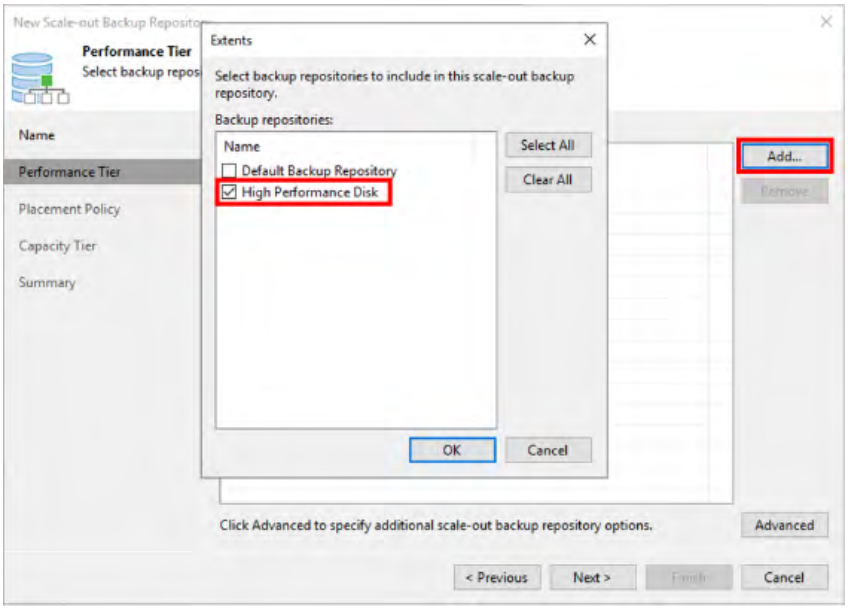


Figure 7 - SOBR Extents

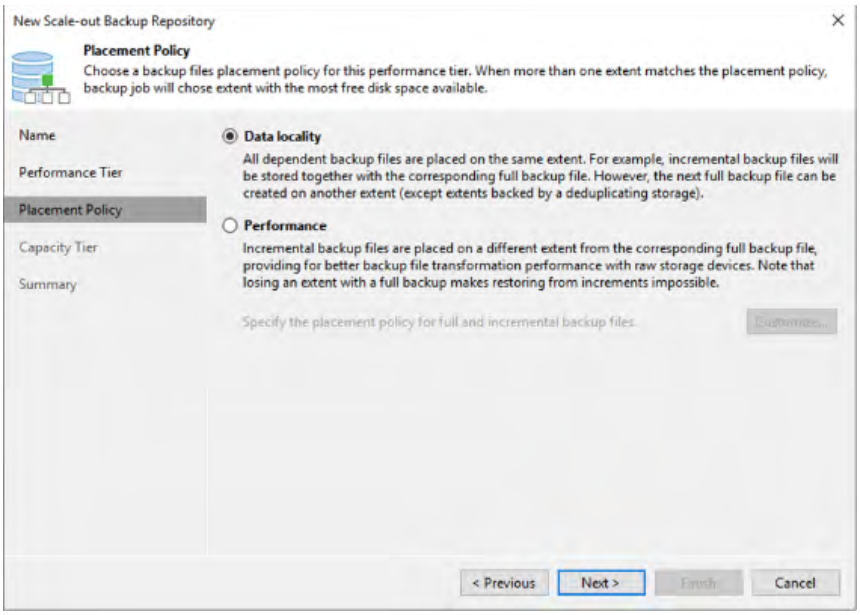


Figure 8 - SOBR Placement Policy.

8. On the **Placement Policy** screen, keep the default **Data Locality** policy. Click **Next** to proceed.

9. On the **Capacity Tier** screen, check the box to extend the scale-out repository with object storage. Then select the Scality repository that will be used.

   Check the box to **Move Backup Files to Object Storage** and set the number of days for old backups to start trimming. Also check the box "Copy backups to object storage as soon as they are created"to create a copy of the backups stored on the performance tier of Scality RING, to improve backup durability and reduce the RPO in case the performance tier goes offline because of an outage (such as a ransomware attack for example). Click **Apply** to apply these changes.

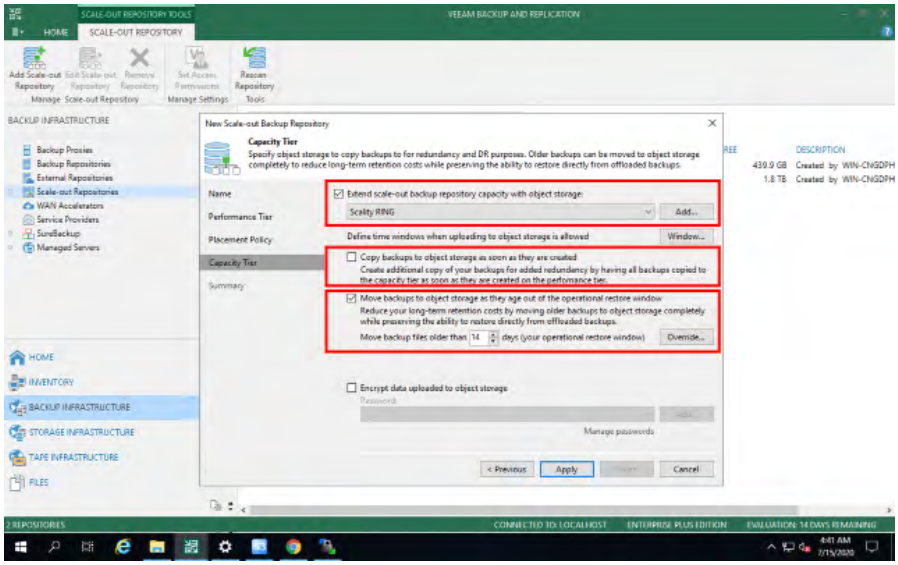


Figure 9 - SOBR Placement Policy.

10. On the **Summary** screen, click **Finish** to create the scale-out repository.