

TECHNICAL WHITE PAPER

Direct to Object with FlashBlade and Veeam Backup & Replication V12

Implementation and best practices for protecting
physical and virtual environments

Contents

Introduction 3

 Object Storage 3

FlashBlade 6

 Hardware Architecture 6

 Purity FlashBlade OS 8

 Object Store on FlashBlade 8

Implementation Guidelines 10

 FlashBlade Bucket Configuration 10

 Veeam Backup & Replication V12 13

 Object Repository Configuration 13

Conclusion 16

About the Author 17

Appendix A 17



Introduction

In today's data-driven world, IT managers and data protection practitioners face increasing challenges to ensure business continuity while protecting their organizations' critical data. As businesses adopt hybrid cloud strategies to manage ever-growing data volumes, data protection must evolve to meet these challenges. A new Pure Storage® data protection solution powered by Pure Storage® FlashBlade® and Veeam Backup & Replication V12 introduces new Direct-to-Object (D2O) enhancements, eliminating the need for additional storage gateways or intermediate backup repositories to streamline the backup process and provide a more cost-effective backup solution.

This white paper provides an in-depth analysis of object storage, Pure Storage FlashBlade Object capabilities, and Veeam Backup & Replication V12 Direct-to-Object repository feature. Targeted at backup administrators and IT managers, the objective of this paper is to educate readers about this new capability, its benefits, and how it can be utilized to optimize their data protection strategies. This document draws from official Pure Storage, Amazon S3, and Veeam documentation and best practices to give a comprehensive understanding of the solution enhancement feature and serve as an Implementation Guide.

Object Storage

Object storage is a modern and innovative data storage solution designed to address the challenges of handling massive amounts of unstructured data and scalability in the age of big data and cloud computing.

Traditional file and block storage systems were designed to handle structured data and hierarchical file systems. However, they have inherent limitations that object storage was created to address:

- **Scalability:** Object storage systems are inherently architected to scale horizontally and vertically to accommodate capacity and performance.
- **Performance:** Performance can degrade traditional storage systems as the number of files and directories increase, leading to longer access times and reduced throughput. Object storage systems, in contrast, maintain consistent performance levels even with a large number of objects due to their distributed architecture, flat address space, and GUID-based addressing.
- **Efficiency:** Traditional storage systems generally rely on RAID (redundant array of independent disks) for data protection. However, RAID can be less efficient and more prone to failure when dealing with large-scale data storage. Object storage systems use erasure coding that allows for more granular control of fault tolerance and storage efficiency than RAID. This provides higher storage efficiency, and higher levels of durability and data protection than some RAID levels.
- **Data management:** Traditional file systems rely on file names and directory structures, making it challenging to manage and search for data efficiently. Object storage uses rich metadata associated with each object, enabling advanced data management and search capabilities that streamline processes and improve accessibility.
- **Accessibility:** Traditional storage systems typically lack standardized APIs, making it more challenging for developers to integrate storage into applications and workflows. Object storage provides simple and standardized RESTful APIs that enable seamless integration with applications, improving accessibility and compatibility.
- **Cost-effectiveness:** Due to their inherent scalability, object storage systems typically reduce the overall cost of ownership. The pay-as-you-grow model¹ allows organizations the flexibility to only pay for what they need, reducing upfront costs.



Object Storage Architecture

The concept of "objects" lies at the core of object storage. An object is a discrete unit of data that consists of three main components:

1. **Data:** The actual content of the object, usually in the form of unstructured data
2. **Metadata:** Descriptive information about the object, such as creation date, object size, content type, and custom attributes defined by the user
3. **Unique identifier:** A globally unique identifier (GUID) which allows the system to locate and manage the object, regardless of its physical location

Objects are stored in a flat address space, meaning that there are no hierarchies of directories, which allows for easy retrieval.

Here is an overview on the process of how an object is stored:

1. **Object creation:** When an object is created, it is assigned a unique identifier or key. This key is generated either by the storage system or provided by the user.
2. **Metadata assignment:** Metadata is created for the object, which includes information such as object size, creation date, and content type. Users can also add custom metadata to describe the object's content or other relevant attributes.
3. **Data partitioning:** The object data is divided into chunks, which can be stored across multiple storage devices or nodes. This partitioning helps with load balancing, fault tolerance, and data redundancy.
4. **Data placement:** Using the unique identifier, the object storage system determines where to store the object chunks. This process usually involves algorithms like erasure coding to ensure data durability and redundancy.

When a user or application requires access to an object, they provide the unique identifier to the object storage system. The system then retrieves the object's metadata and data chunks, reassembles the object, and returns it to the user or application. This architecture enables virtually unlimited scalability and eliminates the need for complex directory structures.

Object Versioning

Versioning is a feature in object storage systems that allows you to preserve, retrieve, and restore every version of an object, providing a way to track changes, recover from accidental deletions or modifications, and maintain a history of the object's state over time.

Without versioning, an update overwrites the object, and the previous version of the object is lost. Similarly, when an object is deleted, it is permanently removed from the storage system.

With versioning enabled, changes are tracked with the version ID, which is separate from the object's GUID.

- When an object is updated or modified, instead of overwriting the existing object, the storage system creates a new version of the object, preserving the previous version(s).
- When an object is deleted, the storage system inserts a delete marker object, with the same object Key, but with a few key differences indicating that it is a placeholder for a deleted object. The object size is 0, indicating that there is no data associated with the object. The version ID is incremented, indicating that this is a new version of the object, even though



there is no data associated with it. Finally, there is no content type or other metadata associated with the object, since it is just a placeholder for a deleted object.

- The delete marker object becomes the new current version of the object. When listing objects in the storage system, the deleted object will not appear. However, users or applications can list and retrieve all the versions of an object using the object key and version identifier. This enables access to historical versions if needed.
- If versioning is disabled, it does not delete the existing versions. However, any new updates or deletions will overwrite the object as if versioning was not enabled.
- Version cleanup policies can be put in place by users or applications to reduce storage costs and clutter. They automatically delete older versions of objects after a specified period or transition them to different storage classes for long-term archiving.

Object Lock

Object Lock is a feature that allows users to create an immutable version of an object for a specific period of time. Once Object Lock is enabled, modifications to the locked object are controlled by the retention mode.

Retention mode deals with changes to retention settings of an object. If retention mode is not set, an object's retention can be changed at any time irrespective of retention period. Object Lock supports two kinds of retention modes:

- **Governance mode:** In governance mode, the retention period for an object cannot be changed or deleted by any user, including the owner of the object. This mode is typically used in scenarios where regulatory requirements mandate that data be kept immutable for a specified period of time. Governance mode is often referred to as "write once, read many" (WORM) mode.
- **Compliance mode:** In compliance mode, the retention period for an object can be extended, but it cannot be shortened or deleted even by the owner of the object. This mode is typically used in legal scenarios where data may be subject to legal hold or other types of litigation.

Both governance and compliance modes enforce the immutability of data by preventing the object from being deleted or modified until the retention period expires or the legal hold is released. These modes provide a high degree of data protection and help ensure that data is stored securely and in compliance with applicable regulations and policies.

With respect to Object Lock, versioning provides a mechanism for the storage system to maintain multiple versions of an object, including the version of the object that is locked. Therefore, it is important to highlight that Object Lock does not prevent the creation of new versions of an object. If a new version of a locked object is created, the lock applies only to the original version, a new lock would be needed on the new version of the object.

Buckets

A bucket is a logical container, user-defined namespace within the object storage system. Versioning and Object Lock are features applied on the bucket, not the individual objects.



FlashBlade

FlashBlade is a unified fast file and object (UFFO) storage technology that can support thousands of clients while simultaneously hosting numerous file systems and multi-tenant object stores. FlashBlade is an all-flash, scale-out storage solution that is powered by a distributed metadata architecture, designed for enormous concurrency across all data types. FlashBlade can expand up to multi-petabyte capacity with linear-scale performance. It is regarded as a data hub because of its inherent scale-out design and ability to drive performance for any type of task. It allows companies to consolidate a variety of workloads on a single platform, from backups to analytics and AI.

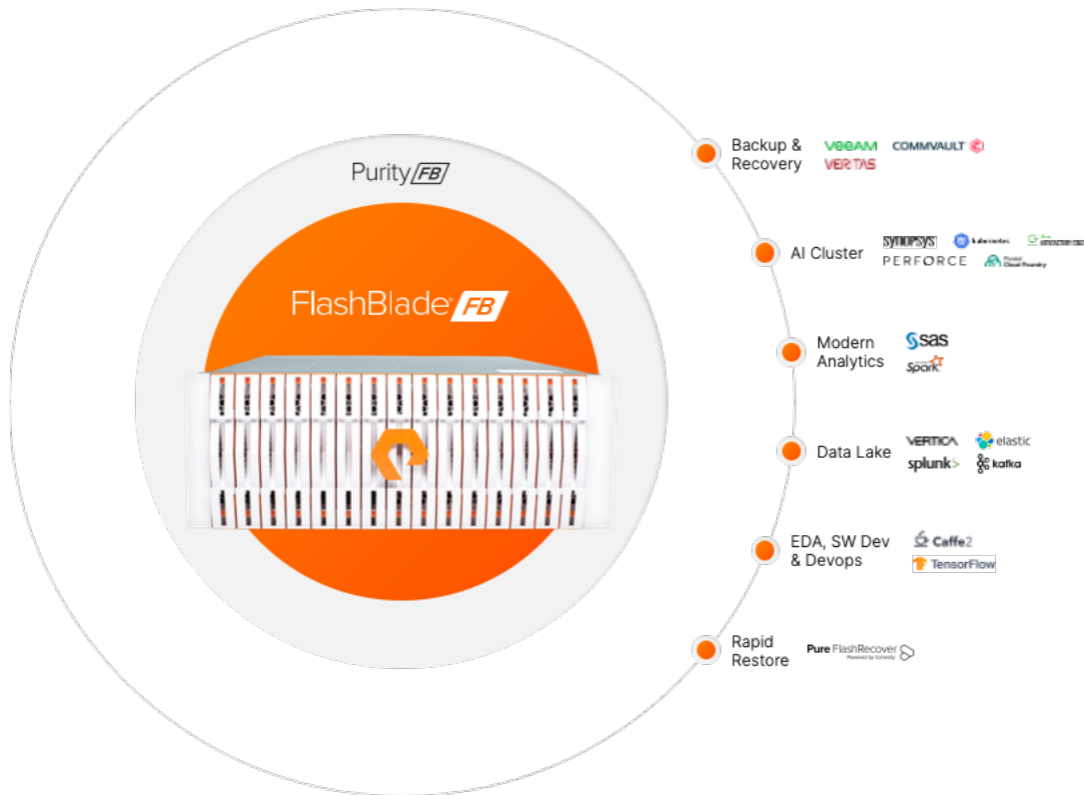


FIGURE 1 FlashBlade Unified Fast File and Object Storage Platform

Many organizations currently use FlashBlade to store their data protection backups, taking advantage of fast backup and restore performance while investing in a platform that also consolidates data lakes and other data silos.

Hardware Architecture

FlashBlade offers a unique, modular architecture that enables organizations to unlock new levels of power, space, and performance efficiency using an all-QLC design. The architecture disaggregates compute resources from storage, so that capacity and compute can scale independently for extreme flexibility in configuration. FlashBlade is a customizable platform that enables you to tailor your configuration for current workload requirements and non-disruptively upgrade to meet future needs. You can upgrade components on a schedule that is consistent with changing technologies to future-proof the system.



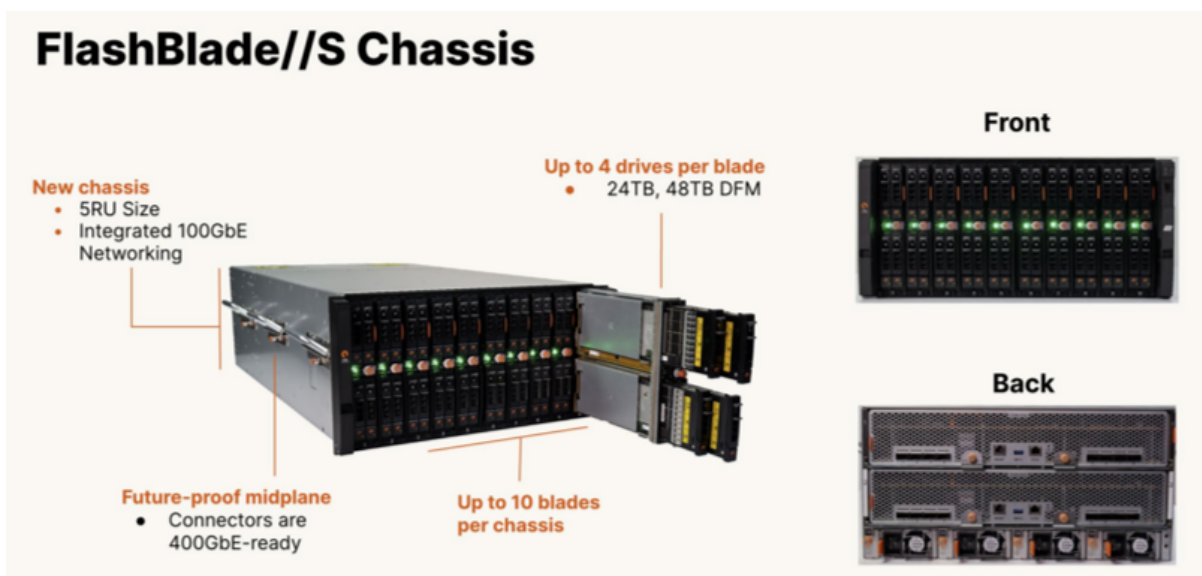


FIGURE 2 FlashBlade//S components

- **Chassis:** The FlashBlade chassis is 5RU high and has bays for mounting up to 10 blades. Fully populated with high-density blades, a chassis holds 1.92PB of physical flash with headroom for future density increases.

The chassis midplane distributes power and contains dual Ethernet links capable of operating at 100- Gbps to each blade. Blades connect to two Fabric I/O Modules (FIOMs) on the midplane. The FIOMs contain Ethernet switches that connect blades to the client network, or in multichassis systems, to eXternal Fabric Modules (XFM). In multi-chassis systems, the XFMs connect the chassis to each other and to clients.

- **Blades:** Each blade contains CPU, NICs, DRAM, and a minimum of one—and up to four—DirectFlash® Module (DFM) mounting slots. Blades use NVMe over on-board PCIe to communicate with their DFMs. A blade can be configured with either performance-optimized (24TB) or capacity-optimized (48TB) DFMs. Blades will operate with one, two, three, or four DFMs installed. All blades in the chassis must be configured with the same number of DFMs in each blade.
- **DFMs with QLC flash:** Architectures that use off-the-shelf, solid-state drives (SSDs) have an internal controller to manage the flash media on each specific drive. These systems do not have any visibility into what is happening at the system level. FlashBlade takes a different and innovative approach with proprietary DirectFlash modules (DFMs) that enable the storage operating system to manage the media on a global level. Global media management unlocks as much as 20% more capacity from NAND, compared to systems that use off-the-shelf SSDs, and delivers more consistent performance, better reliability, and higher media endurance without the need for a massive and expensive storage class memory (SCM) cache.
- **Networking:** The integrated networking in FlashBlade simplifies large-scale deployments by collapsing three networks (front-end, control, and back-end) into one high-performance, software-defined networking (SDN) fabric. This SDN is shared across the two fabric modules in the platform, and it hides the complexity of networking from the administrator.

FlashBlade virtualizes the network, so that no matter the size of the platform, it appears as one entity. This virtualization simplifies load balancing and cabling. Each blade can service and restart any client connection and run any protocol, and the platform is stateless because the logic can run anywhere.



Dual Fabric I/O Modules (FIOMs) interconnect blades, connect chassis (in multi-chassis systems), and connect blades to clients. The FIOMs have ethernet switches with eight (8) external ports each capable of 10, 25, 40, or 100 Gbps transmission rates. The switches have a total of 2TBps cross-sectional bandwidth. Each FIOM uses 50Gbps for interblade communication in the chassis. Both FIOM switches and blade NICs are capable of 100Gb/s for future expansion.

- **Expansion:** FlashBlade includes four base models that cover the space from ultra-dense to ultra-high performance. For efficient storage utilization, the minimum chassis configuration for all models include seven blades. As stated earlier, systems can be expanded by adding DFMs to blades, by adding blades to the chassis, and by adding additional chassis, either to initial configurations or as post-deployment upgrades.

FlashBlade supports multi-chassis configuration and expandable up to five chassis—that's 9.6PB capacity and 300GB/s throughput performance.

Purity FlashBlade OS

With the Purity//FB operating environment, FlashBlade hardware becomes a highly scalable UFFO storage system. It supports NFS, SMB, and HTTP protocols for file access, with access to objects via S3 APIs. Both file and object services are native, based on a common underlying key-value store; there is no protocol-on-protocol layering in Purity//FB.

Purity//FB's internal metadata structures are designed to accommodate tens of billions of files and/or objects, with a high degree of concurrency that exploits all system hardware resources regardless of client I/O load. Data layouts are designed to be resilient in the face of component failures, while at the same time automatically balancing utilization across all system resources.

Object Store on FlashBlade

Purity//FB supports an object store that provides client access to buckets of objects. It associates each bucket with an account that is also associated with users, each having access permissions to some or all of the account's object data.

- **Accounts and users:** Accounts define logical containers, or tenants, which are administrative entities that organize an object store's buckets and control client access to them. Administrators create users within accounts and assign policies that grant them permissions to perform actions on buckets and objects.

Purity//FB generates and uses access key pairs, each consisting of an access key ID and a secret access key, to authenticate users, or applications.

- **Access policies:** Administrators assign one or more access policies to each user to specify the permissible operations on buckets and objects. Policies grant rights to:
 - Display bucket information (e.g., list properties)
 - Control buckets (e.g., enable/suspend versioning)
 - Read, write, and delete individual objects
- **Buckets:** These are the upper level of the two-level S3 object hierarchy. Each object belongs to a single bucket, and each bucket is associated with a single account. Purity//FB tracks the number of objects in each bucket, the amount of physical flash they occupy, and each bucket's overall data compression ratio.

Administrators and users with the appropriate bucket control policy enabled may destroy buckets. Destroyed buckets are inaccessible by clients but are retained internally for possible recovery for 24 hours after which the software permanently eradicates them.



- **Versioning:** Purity//FB supports versioning of objects on a per-bucket basis. When versioning is enabled, overwriting an object creates a new current version, and retains previous versions per the bucket's lifecycle rules. Administrators or users with the appropriate bucket control policy enabled may enable versioning for a bucket. Once enabled, versioning can be suspended but not disabled.

Administrators can use lifecycle rules to limit retention of object versions. Rules specify either how long Purity//FB should retain objects or a date on which it should delete them. A rule can apply to an entire bucket, or its scope may be limited to objects whose IDs begin with a specified prefix.

- **Replication:** Purity//FB can optionally replicate bucket contents to corresponding buckets on one or two target FlashBlade systems, or to the Amazon Web Services public cloud. The software replicates individual objects asynchronously as updates occur. The CLI and GUI display the latest completed object replication and the replication backlog for each bucket.
- **Object lock:** Purity//FB supports object lock data protection and management framework, providing features like retention mode, and retention period that help protect the locked versions of objects in accordance with regulatory and compliance requirements. Purity//FB, goes one step further with Freeze Locked Objects capability. While object lock protects versioned objects from immediate and permanent data loss, it is unable to protect objects created in a bucket with versioning disabled. Freeze locked objects, will extend retention mode, and retention period capability on those unversioned objects, providing the same immutability against immediate deletion, or alteration. Once enabled, Freeze Locked Objects can only be revoked by contacting Pure Storage Technical Support.
- **Retention lock:** An even further layer of protection is provided by Purity//FB. Retention lock protects the bucket from permanent deletion by privileged users. Retention lock is not the same as Object Lock. While Object Lock protects a specific version of the object, based on life cycles rules managed by an external application like Veeam Backup & Replication V12, or internally managed by the FlashBlade Administrator. Retention lock is a bucket level protection:
 - Prevents an administrator or privileged user from destroying the bucket if the bucket has any objects, even if those objects are marked for deletion.
 - If bucket level retention is used (default is 0), it prevents an administrator or privileged user from reducing bucket level retention.
 - Prevents an administrator or privileged user from disabling object versioning.
 - Prevents deletion of buckets via S3 scripts. Buckets can only be deleted via Purity//FB administrative interfaces and are not eradicated until their respective eradication pending periods have elapsed.
 - Once enabled, retention lock can only be altered, or revoked by contacting Pure Storage Technical Support.

Retention Lock provides extra security for the on-premise Object storage from inadvertently, or maliciously wiping the company's most valuable asset, their data. Therefore it's recommended to enable Retention Lock.

FlashBlade hardware and Purity//FB software are indeed "future proof." The hardware design anticipates future component developments, but more importantly for users, FlashBlade shows that the Purity//FB software architecture delivers scalability that realizes the potential of whatever technologies comprise the hardware on which it runs.



Implementation Guidelines

Coupled with Veeam Backup & Replication V12 (VBR), Pure Storage FlashBlade delivers on the promise of cloud-like simplicity and agility with consistent high performance, predictable Recovery Point Objectives (RPOs), and low Recovery Time Objectives (RTOs) for the most critical workloads.

(Figure 3), illustrates the FlashBlade, powered by Veeam Backup & Replication V12, high level solution for VMware infrastructure, but equally applies to Microsoft Hyper-V infrastructure as well.

While installing the Pure Universal API (USAPI) plug-in, for VMware workloads, is optional, it enables VBR to orchestrate hardware based snapshots of the source Data Stores if they reside on Pure Storage FlashArray™, and If VBR proxies are Fibre channel or iSCSI capable. This enables VBR to backup directly from the Pure Storage FlashArray storage, alleviating backup traffic off the production VM Network.

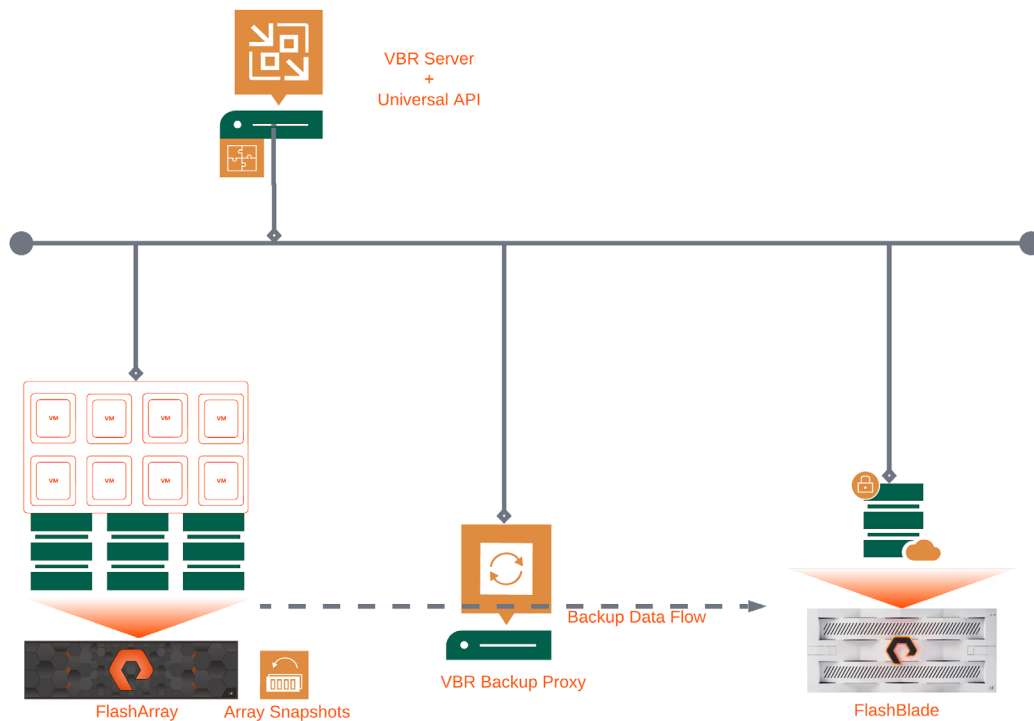


FIGURE 3 FlashBlade powered by Veeam Backup and Replication V12 Solution

FlashBlade Bucket Configuration

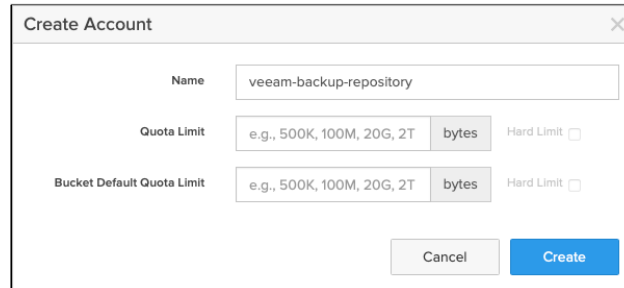
Pure Storage FlashBlade can be managed with a web-based graphical user interface (GUI), remote Secure Shell command-line interface (CLI), or programmatically using a rich library of application programmable interfaces (APIs). We will be using the GUI for demonstrations.

In FlashBlade GUI, click on the **Storage** menu item. Click on the **Object Store** tab, this is where you can manage all object storage related tasks.



Object Store Accounts

Accounts tile, is where to create an object store account. Creating a bucket on FlashBlade starts by creating an object store account. This account is an internal container used by FlashBlade to manage multi-tenancy. You can have up to 2000 Object store accounts. Object store accounts are not visible or used by applications consuming the bucket(s).



The 'Create Account' dialog box contains the following fields and controls:

- Name:** A text input field containing 'veeam-backup-repository'.
- Quota Limit:** A text input field with a dropdown menu showing 'e.g., 500K, 100M, 20G, 2T' and a unit selector set to 'bytes'. A 'Hard Limit' checkbox is to the right.
- Bucket Default Quota Limit:** A text input field with a dropdown menu showing 'e.g., 500K, 100M, 20G, 2T' and a unit selector set to 'bytes'. A 'Hard Limit' checkbox is to the right.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

FIGURE 4 Create Object Store Account Dialog box

Click on (+), **Create an Account** dialog box (Figure 4), allows you to choose a **Name** for the object store account, the name must be between three and 63 characters of only lower case letters, numbers and '-'. It should begin and end with a lowercase letter or number. The name cannot contain two consecutive '-' and cannot be a 12 digit number.

You can also set soft, and hard quota limits. Soft quota limits set the usage threshold at which alerts are raised, and hard limits set the usage limit at which all writes are halted until usage decreases below the set limit.

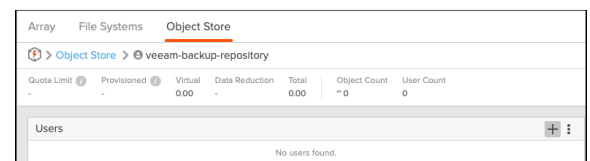
Quota Limit, sets limits at the object store account level and supersedes **Bucket Default Quota Limit** which is the per bucket quota limit. If all buckets under an object store account have collectively exceeded the account's hard limit, writes to any bucket in that account will be halted regardless of their set hard limits. Click the **Create** button to save changes.

Users, Permissions, and Access Keys

Click the object account just created to drill into the tenants properties. Here is where you can create users, access keys, and buckets. You can add up to 100 users per object store bucket.

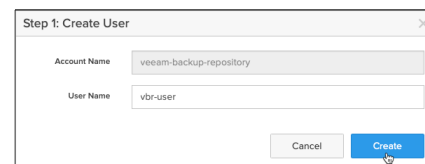
Click (+) in the Users tile (Figure 5). A wizard for creating a user, access keys, and permissions will start.

1. **Create User** dialog box (Figure 6). Add the appropriate name in the **User Name** field, and click the **Create** button.
2. The **Add Access Policies to user** dialog box (Figure 7) will open. You can set the permissions the user is allowed to perform against the bucket and its objects in this window. While it's possible to use the already predefined access policies, it's recommended to create an access policy with all required permissions².
3. Click the Add button to save selections.
4. Add Access Key to user dialog box (Figure 8).



The 'Tenant screen' shows the 'Object Store' tab for 'veeam-backup-repository'. It includes a table with columns: Quota Limit, Provisioned, Virtual, Data Reduction, Total, Object Count, and User Count. Below the table is a 'Users' section with a '+ +' button and the text 'No users found.'.

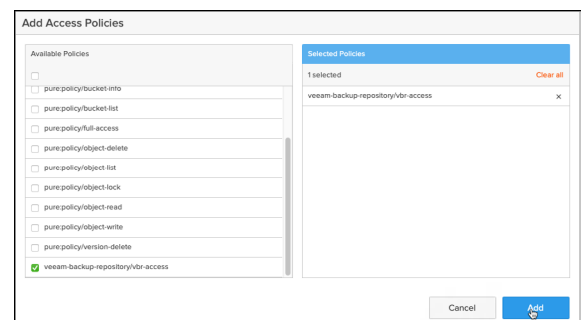
FIGURE 5 Tenant screen Dialog box



The 'Step 1: Create User' dialog box contains the following fields and controls:

- Account Name:** A text input field containing 'veeam-backup-repository'.
- User Name:** A text input field containing 'vbr-user'.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

FIGURE 6 Create User Dialog Box



The 'Add Access Policies' dialog box shows a list of 'Available Policies' on the left and a 'Selected Policies' list on the right. The 'Available Policies' list includes:

- ☐ pure:policy/bucket-info
- ☐ pure:policy/bucket-list
- ☐ pure:policy/full-access
- ☐ pure:policy/object-delete
- ☐ pure:policy/object-list
- ☐ pure:policy/object-lock
- ☐ pure:policy/object-read
- ☐ pure:policy/object-write
- ☐ pure:policy/version-delete
- ☒ veeam-backup-repository/vbr-access

The 'Selected Policies' list shows '1 selected' and 'veeam-backup-repository/vbr-access'. At the bottom are 'Cancel' and 'Add' buttons.

FIGURE 7 Access permissions for user dialog box



In this window, you have the option to **Create** or **Import** an Access Key. If you select **Create a new key**, you will be prompted to copy the key/secret to clipboard, or download a csv or json copy³.

Creating a Bucket

Open the FlashBlade GUI, Click the Storage → Object Store.

In Accounts Tile, select the appropriate tenant.

Navigate to the Buckets tile (Figure 9).

Click (+) to open the Create Bucket window (Figure 10). Bucket names should not contain uppercase letters; only “-”, and numbers are allowed.

You can set a soft, or hard quota on the created bucket. If Quota Limit is left empty, the bucket will inherit the object store account (Tenant) Bucket default quota limit setting, if its enabled.

Enable Object Lock

Object Lock, and consequently VBR, requires versioning to be enabled on the object bucket.

On the FlashBlade GUI, Click Storage → Object Store tab, select the appropriate Object Store Account. Identify the appropriate bucket name, and select () on the right hand side of the bucket row, within Buckets tile (Figure 11).

In the Edit Bucket dialog window (Figure 12) set the following:

- Versioning: enabled (required by VBR)
- Expand Retention Settings
 - Object Lock: checked
 - Freeze Locked Objects: default unchecked.
 - Default Retention: none.
 - Default Retention Mode: none.
 - Retention Lock: ratcheted (locked)

VBR will manage default retention and the default retention mode.

FlashBlade can provide even more protection on buckets, and objects with Retention Lock. Retention Lock has two modes:

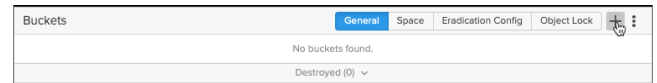


FIGURE 8 Access Key dialog box



FIGURE 9 Buckets Tile

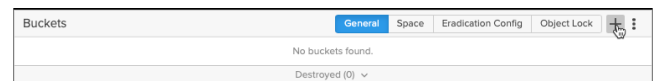


FIGURE 10 Create Bucket dialog box

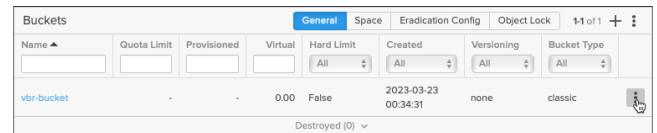


FIGURE 11 Editing Bucket dialog box

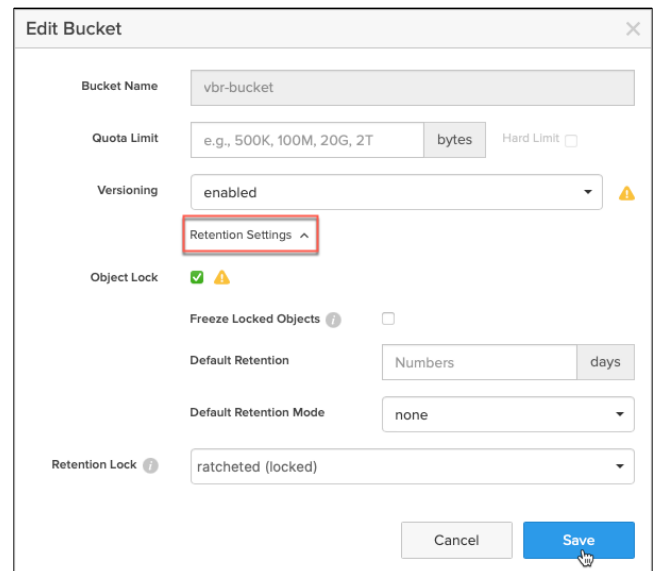


FIGURE 12 Editing Bucket dialog box



- Un-locked mode (default mode).
- Ratcheted (locked) mode.
 - Buckets can only be destroyed when empty.
 - Automatic bucket eradication is disabled.
 - Changes to retention mode, and retention periods are disabled.

Bear in mind, any changes to Retention Lock requires contacting Pure Storage Technical support.

Click the Save button to apply the changes. Now, the S3 bucket is ready to be consumed by VBR.

Data Network

Accessing a bucket requires at least one valid data virtual IP address (data vip). Data-VIPs are configured through the FlashBlade GUI, **Settings** → **Network** → **Subnets & Interfaces**⁴. Data-VIPs can also be configured through CLI⁶.

Veeam Backup & Replication V12

Organizations generate large amounts of new data every day, and its backups must be stored somewhere. Veeam Backup & Replication V12 brings an object-first approach, simplifying the backup infrastructure. The large amounts of backup data can be stored efficiently with FlashBlade's highly scalable hardware and software architecture. VBR can establish thousands of connections to FlashBlade to handle massive amounts of backup and recovery jobs efficiently.

Object Repository Configuration

Object Repository setup on VBR follows the same approach as all other VBR repositories.

1. First, make sure that you have the necessary credentials and permissions to access the FlashBlade bucket created in the [previous section](#).
2. In the VBR console on the navigation panel, select **Backup Infrastructure** down in the views section. Hover the mouse over the working area, right-click, select **Add Object Storage Repository**. Adding Storage repository wizard will start.
3. Choose **S3 Compatible**, next, enter a name, and description for your repository. Leave **limit concurrent tasks** unchecked. Click **Next**.
4. Under **Service Point**, provide the FlashBlade data virtual IP address (data vip), or FQDN if the data-vip is configured in DNS.
 - a. You don't need to provide "https:// ..." as VBR will handle that automatically. VBR will **only** connect to the S3 bucket over SSL.
5. **Region** is a required free-text field. It is typically used for on-premise object storage to denote data center location.
6. **Credentials**: Using the **Add** button, provide the **Access Key** and **Secret Key** generated in the previous section by FlashBlade.



7. **Connection Mode:** (Figure 13) Defines how VBR will connect to the FlashBlade object bucket backup repository:

- a. **Direct:** Any backup job controlled by VBR server⁵ will be able to utilize the proxy server(s) to transfer VM Backups or file share backups to FlashBlade object storage repository.

Proxy server(s), must have network access to the FlashBlade data-VIP (Service Point), otherwise the backup job will fail. Object bucket credentials will be shared with the proxy server(s) at the start of the backup job.

- b. **Through a gateway server:** VM Backups or file share backup data will be sent to the FlashBlade Object Repository through a Veeam Gateway Server. VBR will determine which Gateway Server(s) to use from the list of configured VBR infrastructure Gateway Servers⁶.

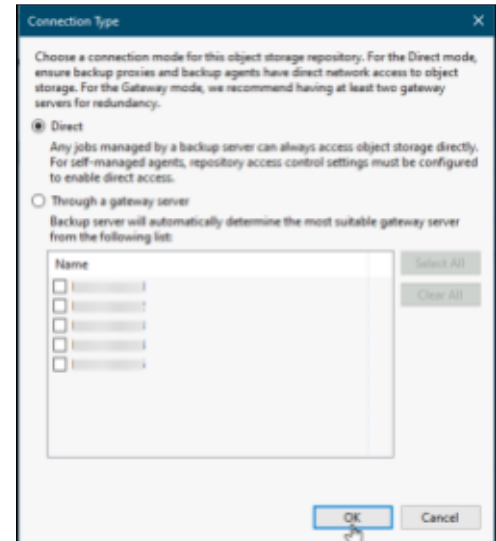


FIGURE 13 Connection Mode

8. Click the **Next** button. VBR will may prompt you to accept the untrusted certificate if the FlashBlade doesn't have a properly signed certificate^{7,8}.
9. In the **bucket configuration** dialog window, you will need to configure:
- a. **Bucket:** Click on the **Browse** button to list the bucket created [previously](#).
 - b. **Folder:** Specify the folder where backups will be stored. Click **Browse** and select a folder if you already created one. You can also create a new folder by clicking the **New Folder** button. Click OK to save the selection.
 - c. **Limit Object Storage Consumption to (# TB):** Provide a soft quota on the bucket to help with storage spending, particularly in the public cloud. This is a user preference. It's helpful to set an upper value on the bucket from within VBR since the AWS S3 API does not provide the mechanism to report on object bucket consumption.
 - d. **Make recent backups immutable for (# days):** VBR will check whether versioning and Object Lock are enabled. If Object Lock and versioning are not detected, VBR will not allow you to define the immutability period.

The general recommendation is to have an immutability period the same or lower than the retention period defined in the backup job. VBR will automatically add 1 to 10 days on top of what the user defined for immutability to objects still needed within the defined retention period, this setting is called **block generation**⁹.

10. Click **Next** to configure the **Mount Server**.
11. **Mount Server** dialog window requires you to specify a server you will be using to mount VM disks directly from the object storage repositories to the VMware environment. Mount Server will present itself as an NFS Server serving a Datastore named "VeeamBackup_{MountServer}" to the VMware environment, with VM disks objects streamed from the object storage repository.
12. The **Instant recovery write cache folder** field is necessary to track changes made to VMs recovered with VBR Instant Recovery feature⁷.
13. VBR will automatically configure a helper appliance service to perform health checks and validity of the objects for every Object bucket. The **Configure** button will allow you to change the default server to another managed server in the VBR infrastructure.
14. Click the **Next** button, then **Apply** button to proceed with creating the object repository.



Repository Permissions for Veeam Agents

You must set up access permissions to the Object Repository if you intend to backup virtual and physical machines with VeeamAgents in direct connection mode.

Veeam Agents communicate with the object storage using one of the following modes:

- **Gateway server:** With this connection mode, Veeam Agents access object storage through a Gateway server. Access to object storage is managed by a Gateway Server assigned in the VBR console. Backup data is sent from the Veeam Agent computer to the Gateway Server, then from the Gateway Server to the object storage.
- **Direct:** Veeam Agents access object storage directly. Backup data is sent from the Veeam Agent computer to the object storage. Permissions to the Veeam Agent's access to object storage is managed by the VBR Server.

To grant access permissions to Veeam Agents:

1. Navigate to **Backup Infrastructure View**.
2. On the inventory pane, click on **Backup Repositories**.
3. Right-click on the appropriate object repository and select **Access Permissions** (Figure14).

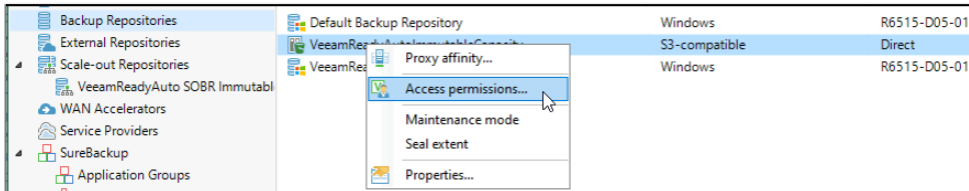


FIGURE 14 Access Permissions

4. Under the **Security** Tab, choose **Agents share credentials to direct storage repositories [direct to object]** (Figure 15)].

Note: Veeam Backup & Replication V12, can also integrate with IAM/STS¹⁰ to access the object storage repository in direct mode. As of the writing of the document, this feature is not yet supported by Pure Storage FlashBlade.

5. Under the **Standalone Applications** Tab (Figure 16), choose **Allow to everyone**. If you want only specific users to be able to run backups, click the **Add** button to add the necessary users and groups to the list.

Note: If you're using a SOBR, then only the Standalone **Applications** option will be available in Access Permissions. The **Security** Tab will be under the specific Object repository configured within that SOBR.

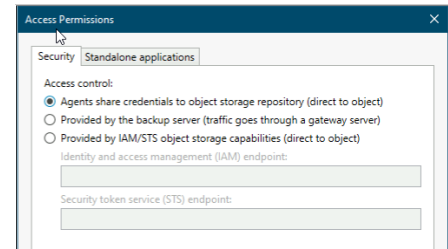


FIGURE 15 Security

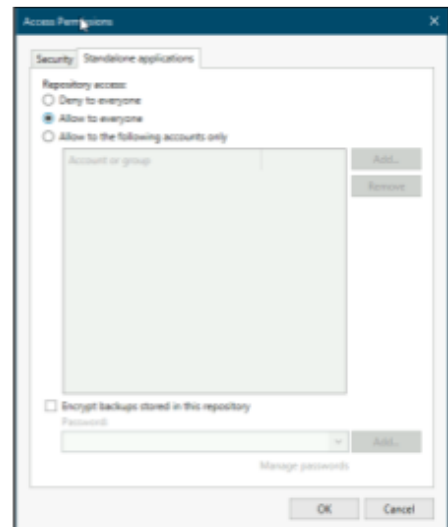


FIGURE 16 Security



Scale-out vs. Traditional Object Repositories

With a FlashBlade//S200 (4X24×10), and VBR n12, we were able to observe how the FlashBlade balanced the workload across all its components eliminating the need to use multiple data-VIPs. There were no observed performance differences between Scale-Out or Simple repository architectures.

- **Throughput:** The scale-out backup repository (SOBR) tested consists of four object buckets, with four data-VIPs, and performed almost identical to using a single object bucket. VBR Proxy Servers established 100s of concurrent connections (5000+ in total), balanced across all FlashBlade nodes.
- **Backup:** A capacity impact test pushed FlashBlade to near 50% full with VM backup data (100+TB after VBR compression) to measure how VBR balancing the capacity across multiple object buckets with a SOBR could possibly yield better performance rather than just Single Object Bucket. Again, there was no difference in performance between the two configurations.
- **Restore:** Using SOBR Object Repositories could be advantageous with restores. Since each object bucket requires a **VBR** Mount Server, it helps to have multiple Mount Servers involved, especially when restoring a large number of VMs.

Capacity Consumption with Object vs. File Repositories

It is important to note that the actual difference in capacity consumption between traditional and object repositories will depend on various factors, including backup policies, data types, and the efficiency of deduplication and compression.

With a traditional backup repository, VBR stores the backup files in a deduplicated and compressed proprietary format with the extension .vbk (for full backups) and .vib (for incremental backups). These files contain the backed-up data, along with metadata about the backup job. On restore, VBR will recall the needed files, unwrap compression and deduplication, and present the VM data back to VMWare.

With an object backup repository, there is no concept of files, VBR stores the backup data as deduplicated and compressed objects. On restore, VBR will stream the needed objects based on an optimized index, and present the VM data back to VMWare.

The **Storage Optimization** settings inside of a backup job are directly related to the space consumed by the backup object in the case of an object repository, or space consumed by the backup file in the case of file repository. Stored file, or object, is deduplicated and compressed (default settings). For example, if the storage optimization setting is 1MB (the recommendation), it means the backup object or file will be constructed with up to 1MB in size, before data reduction. Stored object or file, will be smaller than 1MB, depending on the amount of data reduction observed.

With forever forward incrementals, both File and Object repositories have shown to consume comparable storage capacity.

Conclusion

Veeam Backup & Replication V12's object first approach unlocks the full capabilities of Pure Storage FlashBlade. The solution provides robust and efficient modern data protection. Veeam's ease-of-use, mobility with self-describing backup objects, managed immutability, and intelligent tiering, can ensure the security and availability of organizations critical data.

Meanwhile, the high-performance and scalability of FlashBlade combined with the Pure Storage commitment to maintain ease of manageability enhances this partnership by accelerating the backup and recovery process. Altogether, this powerful partnership offers organizations a comprehensive and reliable data protection solution, ensuring business continuity and minimizing the risks associated with data loss and downtime.





About the Author

Tamer Swidan is a senior solution architect with Pure Storage. He is responsible for defining Pure Storage solutions and reference architectures for protecting and recovering primary workloads such as Oracle, SQL, and VMware. Tamer has 21 years' experience working in and with data protection hardware and software solutions, serving as an end user, a subject matter expert consultant, and data protection solution architect.

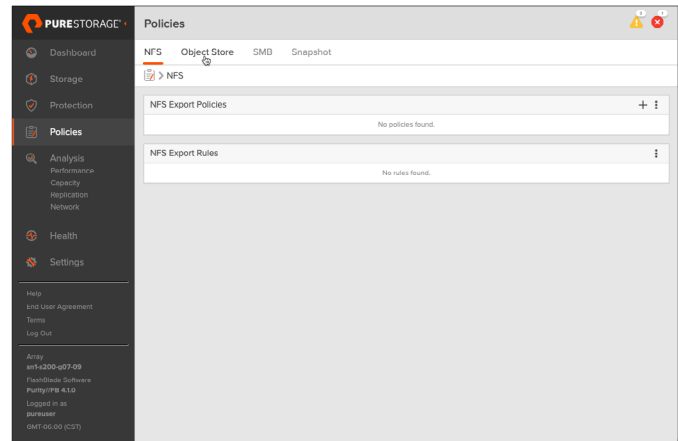
Appendix A

Configure Object Access Policy for Veeam Backup & Replication V12

FlashBlade Policies for Object storage, provides granular permissions on objects, and buckets, to further restrict users to what they need to do, rather than providing full access.

The following steps, provide an example on how to configure granular permissions to the Veeam Backup & Replication user, in order to properly manage the Object Repository.

- From the FlashBlade GUI, click on **Policies**, then click on the **Object Store** tab.
- On **Object Store Access Policies** tile, Click (+)
- Dialog Box: **Step 1: Create Object Store Access Policy**, opens
 - Account:** From the pull-down menu, select the Object Store Account.
 - Name:** Provide a name for the access policy, (recommend *VBR-Repository*)
- Click **Create**.



Step 1: Create Object Store Access Policy

Account

veeam-ready

Name

vbr-repository

Description (Optional)

Add a description for this policy. The description cannot be supplied or changed later.

Cancel

Create



- Dialog Box: **Step 2: Add Rule to Policy ...**, opens:
 - **Name:** Provide a rule name (Recommend *VBRAccess*).
 - **Effect:** Defines whether the permissions are to **deny**, or, **allow** access.
 - **Actions:** will contain a list of all S3 object permissions currently available. Click on (+).
 - List of S3 permissions required:
 - s3:AbortMultipartUpload
 - s3>CreateBucket
 - s3>DeleteBucket
 - s3>DeleteObject
 - s3>DeleteObjectVersion
 - s3:GetBucketLocation
 - s3:GetBucketVersioning
 - s3:GetObject
 - s3:GetObjectLegalHold
 - s3:GetObjectLockConfiguration
 - s3:GetObjectRetention
 - s3:GetObjectVersion
 - s3:ListAllMyBuckets
 - s3:ListBucket
 - s3:ListBucketMultipartUploads
 - s3:ListBucketVersions
 - s3:ListMultipartUploadParts
 - s3:PutLifecycleConfiguration
 - s3:PutObject
 - s3:PutObjectLegalHold
 - s3:PutObjectRetention
 - Click the **Add** button to commit changes

Step 2: Add Rule to Policy 'veeam-ready/vbr-repository'

Name: vbrAccess
Rule name consisting of letters and numbers, with no hyphens or underscores.

Effect: Allow
Allow S3 requests that match all of the criteria below. 'Allow' rules are additive.

Actions: s3:ListBucketMultipartUploads, s3:ListBucketVersions, s3:ListMultipartUploadParts, s3:PutLifecycleConfiguration, s3:PutObject, s3:PutObjectLegalHold, s3:PutObjectRetention
List of permissions to grant, in addition to any from other rules and policies.

☐ Ignore Action Restriction Enforcement

Resources: *
List of bucket names and object paths, with a wildcard (*) to specify objects in a bucket; e.g., bucket1, bucket1/*, bucket2, bucket2/*.

Optional Conditions ▾

Skip **Add**

Optional Conditions include additional granular controls, like only allow certain source IPs to access the bucket, or allow or deny access to only certain objects in the bucket.

- 1 Learn more about the Pure Storage Evergreen® storage program.
- 2 Please refer to Appendix-A for the specific permissions needed and steps on how to configure Access Policies.
- 3 Once you close this window, you will no longer be able to see the secret key ID, you will have to create a new Access Key/Secret.
- 4 Refer to FlashBlade user's Guide for Steps on how to setup a Data-VIP.
- 5 As compared to backup jobs controlled by Veeam Agents running in managed mode.
- 6 For more details, and limitations, please refer to VBR User Guide for VMware vSphere.
- 7 Please refer to the "Array Certificates" section on the FlashBlade user's Guide for instructions on how to import properly signed certificates.
- 8 If you click "Continue", VBR will disable self-signed certificates after some time, therefore, if you don't intend on installing a properly signed certificate, it would be advantageous to click on "View", and install the self-signed certificate.
- 9 For more details on Block Generation please refer to Block Generation - User Guide for VMware vSphere (veeam.com).
- 10 For details on IAM/STS please refer to Access Permissions - User Guide for VMware vSphere (veeam.com)