45Drives
ENTERPRISE

ZFS Digital Guide

45Drives.com

Table Of Contents

Chapter 1: Understanding ZFS

What is ZFS?

ZFS (Zettabyte File System) is an advanced file system and volume manager designed to ensure data integrity, scalability, and ease of use. Originally developed by Sun Microsystems, ZFS combines volume management and a file system into a single solution, simplifying storage management. ZFS is widely used in enterprise environments and homelabs due to its powerful features:

Data Integrity: ZFS uses end-to-end checksums to detect and correct silent data corruption.

Copy-on-Write (COW): Rather than overwriting existing data, ZFS writes new data to a new location. This prevents data corruption during power loss or write failures.

Snapshots and Clones: Instant, space-efficient snapshots and writable clones enable data backup, versioning, and testing.

Built-in Compression and Deduplication: ZFS supports advanced storage-saving features, reducing disk usage without additional hardware overhead.

Resilvering: When a drive fails and is replaced, ZFS resilvers the data by reconstructing only the missing data from parity or mirrored devices. This ensures quick recovery and minimized downtime.

Self-Healing: ZFS automatically detects and corrects data corruption by comparing checksums. If corrupted data is detected, ZFS restores it from mirrored or parity data, ensuring data integrity.

Comparison to Traditional File Systems (EXT4/XFS):

EXT4/XFS: These are commonly used Linux file systems that provide strong performance but lack many advanced features, such as snapshots, integrated RAID management, and self-healing.

ZFS: Unlike EXT4 or XFS, ZFS combines file system and volume management, making it easier to handle large, complex storage systems. It also provides advanced features like resilvering, COW, and native compression/deduplication.

Introduction to RAID Types

ZFS includes built-in RAID-like functionalities, eliminating the need for separate RAID controllers or software. This integration simplifies storage management and offers superior flexibility. Here is a comparison of traditional RAID levels with ZFS equivalents:

Traditional RAID	ZFS Equivalent	Description	Performance	Storage Efficiency
Raid 0	**Striped zpool**	Stripes data across multiple disks for performance, no redundancy.	High read/write performance, no redundancy.	100% storage capacity used for data.
Raid 1	**Mirror**	Mirrors data for redundancy. Data is written identically to two drives.	Moderate performance (read from either disk).	50% storage efficiency (double storage used).
Raid 5	**RAIDZ1**	Data and single parity are distributed across at least 3 disks. Can tolerate 1 disk failure.	Good read performance, slower write due to parity calc.	Efficient with parity (loss of 1 drive’s capacity).
Raid 6	**RAIDZ2**	Data and dual parity are distributed across at least 4 disks. Can tolerate 2 disk failures.	Similar to RAID 5 but slightly slower writes.	Loss of 2 drives' capacity due to dual parity.
Raid 10 (1+0)	**Mirror Striped zpool**	Combines RAID 1 mirroring and RAID 0 striping for redundancy and performance.	High performance & redundancy, can tolerate multiple failures.	50% storage efficiency due to mirroring.

Condensed Explanation:

RAID 0 (ZFS Striped zpool): High performance, no redundancy. Total data loss if one drive fails.

RAID 1 (ZFS Mirror): Mirrors data for redundancy. Full capacity of one drive is used for backup.

RAID 5 (ZFS RAIDZ1): Data and parity distributed across drives for one drive failure tolerance. Balanced performance and capacity.

RAID 6 (ZFS RAIDZ2): Similar to RAID 5 but with dual parity for higher redundancy (2 drive failures).

RAID 10 (ZFS Mirrored Striped zpool): Combines mirroring and striping for maximum performance and redundancy. Ideal for high-performance, fault-tolerant setups.

Comparison to Traditional RAID Types (Hardware RAID and mdadm)

Windows Hardware RAID:

- **Advantages:** Hardware RAID provides good performance due to dedicated RAID controllers and offloads processing from the CPU.
- **Disadvantages:** Hardware RAID controllers can be expensive, and if the controller fails, rebuilding the RAID array on a new controller can be challenging. Limited flexibility compared to ZFS in terms of adding/removing drives.
- **ZFS Advantage:** ZFS is independent of hardware controllers, offering better portability and flexibility when moving pools between systems. It also provides advanced data integrity checks (self-healing, resilvering) that hardware RAID lacks.

mdadm (Linux Software RAID):

- **Advantages:** Free, flexible, and well-integrated with Linux. Supports many RAID levels (RAID 0, 1, 5, 6, etc.).
- **Disadvantages:** Lacks advanced features like checksumming, snapshots, or integrated file system management.
- **ZFS Advantage:** ZFS combines RAID management and file system features, reducing the need for separate RAID software (like mdadm) and offering advanced tools like snapshots, compression, and self-healing.

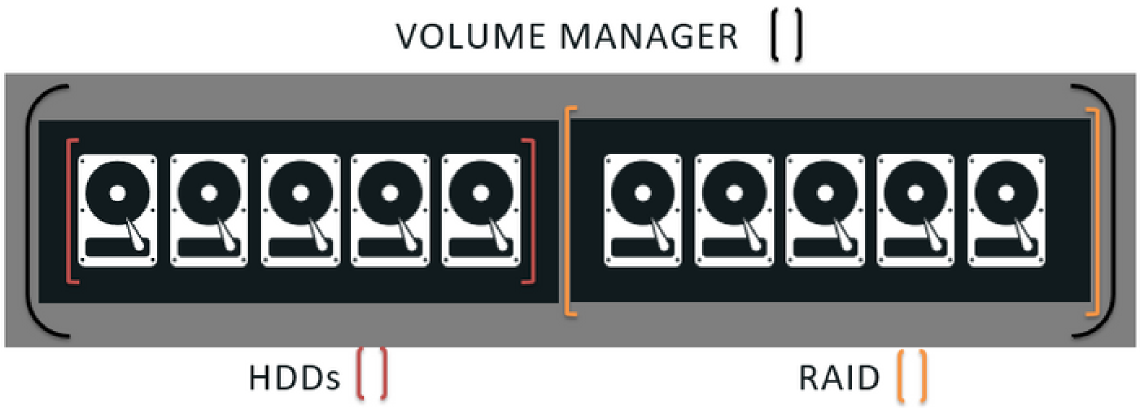
Volume Manager Overview

ZFS integrates volume management with its file system, eliminating the need for separate tools like Logical Volume Manager (LVM). This makes it simpler and more efficient to manage large-scale storage environments. Key features of ZFS volume management include:

- **Dynamic Resizing:** ZFS pools can be expanded instantly without the need for repartitioning or system restarts.
- **Storage Pools (Zpools):** A zpool is the fundamental unit of storage in ZFS. It aggregates one or more virtual devices (VDEVs), which themselves manage physical storage devices.
- **Device Independence:** ZFS pools can combine different storage device types (e.g., SSDs, HDDs, and NVMes) within the same pool, offering flexibility to scale both performance and capacity.

Comparison to Traditional Volume Managers

- **LVM (Logical Volume Manager):** In traditional systems, LVM is used to manage disk partitions separately from file systems like EXT4 or XFS. This approach requires more manual intervention when resizing or managing volumes.
- **ZFS:** ZFS simplifies storage by combining volume management and the file system into one. It dynamically adjusts storage pools without requiring separate tools or complex management.



Filesystem Overview

ZFS is both a volume manager and a file system, designed to handle large amounts of data with maximum efficiency and resilience. Key filesystem components of ZFS include:

- **Zpools (Storage Pools):** A zpool is an aggregation of one or more VDEVs (virtual devices) that manage physical storage devices. Zpools are the foundation for creating datasets, snapshots, and zvols.
- **Datasets:** Datasets are file systems within a zpool, allowing for flexible, individually configured sub-file systems. Datasets can have their own compression, deduplication, and quota settings.
- **Snapshots:** Snapshots are read-only, point-in-time versions of datasets. They are space-efficient and provide a quick recovery mechanism without consuming much additional storage.
- **Clones:** Writable copies of snapshots, allowing for instant testing or development environments without duplicating the entire dataset.
- **Zvols (ZFS Volumes):** Zvols are block devices created in ZFS pools. They are typically used for iSCSI or Fibre Channel targets, allowing applications to interact with ZFS as block storage rather than a file system.

***Current OS and Version of ZFS that 45Drives uses:**

- *Ubuntu 20.04: zfs-0.8.3-1*
- *Rocky 8: zfs-2.1.15-2*
- *Ubuntu 22.04: zfs-2.1.15-2*
- *Rocky 9: zfs-2.1.15-2*

Chapter 2: ZFS Storage Concepts and Architecture

Copy-on-Write (COW)

One of ZFS's fundamental features is Copy-on-Write (COW). In a traditional file system, overwriting data can lead to partial updates or corruption if there is a power failure or system crash during a write operation. With COW, instead of overwriting the original data, ZFS writes changes to a new block, and once the write is completed, it updates the metadata to point to the new block. This ensures that data is never lost or corrupted during writing operations.

How it Works:

When a file is modified, ZFS writes the changes to a new location, ensuring that if a failure occurs during the write, the old data remains intact. Once the write is successful, ZFS updates the pointers to the new data.

Advantages of COW:

- **Data Integrity:** Old data is preserved until new data is successfully written.
- **Atomic Writes:** Ensures that write operations are either fully completed or not at all, reducing the risk of data corruption.
- **Efficient Snapshots:** Because of COW, ZFS can take snapshots almost instantly by merely recording a point-in-time copy of the metadata without copying the actual data blocks.

The ZFS Storage Hierarchy

The ZFS storage hierarchy is structured to provide both flexibility and scalability. Below are the key components of the ZFS storage hierarchy and how they work together.

What is it?

- ZFS is built upon a flexible, hierarchical architecture designed to manage storage devices and data with ease. The structure allows administrators to configure and scale storage pools dynamically without complex setups. The ZFS hierarchy is organized into pools, virtual devices (VDEVs), datasets, and volumes.

What Makes a ZFS Storage Array?

ZFS storage arrays are built using storage pools, known as zpools, which aggregate multiple devices. ZFS pools are created from one or more VDEVs (virtual devices). The flexibility of VDEVs allows administrators to use different types of storage devices (e.g., HDD, SSD, NVMe) within the same pool to create highly optimized arrays.

Key concepts include:

- **VDEVs (Virtual Devices):** ZFS storage pools are made up of VDEVs, which can include mirrored devices, striped devices (for performance), and RAIDZ1/2/3 (for redundancy). Each pool must contain at least one VDEV, and VDEVs are themselves composed of physical storage devices (disks).
- **Storage Medium:** You can use various storage media (HDDs, SSDs, NVMes) to optimize pools based on performance, redundancy, or cost. ZFS allows combining different types of devices within a pool, enabling tiered storage setups.

Comparison to Other Storage Arrays

Hardware RAID vs. ZFS: Traditional hardware RAID requires specialized controllers and lacks the data integrity, resilvering, and flexibility offered by ZFS.

mdadm (Linux Software RAID): While mdadm offers similar RAID configurations, it lacks the file system-level integration of ZFS, making it less efficient when dealing with data integrity and snapshots.

How Data is Organized in ZFS

Zpools: The foundational unit of storage in ZFS. A **zpool** is created from one or more VDEVs, which in turn manage the physical storage devices (disks). Zpools manage the underlying storage and handle tasks like redundancy, read/write performance, and scalability.

- **Dynamic Scalability:** Zpools can be expanded by adding more VDEVs, allowing storage to grow as needed.
- **Flexibility:** You can mix and match different storage devices (e.g., HDDs and SSDs) in a single zpool to optimize performance and cost.

VDEVs (Virtual Devices): Virtual devices (VDEVs) are the building blocks of a ZFS pool. Common VDEV types include:

- **Mirror VDEVs:** Provide redundancy by writing data to multiple devices.
- **RAIDZ1/2/3 VDEVs:** Similar to RAID 5/6, these provide redundancy and better storage efficiency by using parity across multiple disks.
- **Striped VDEVs:** Provide increased performance by spreading data across multiple devices but offer no redundancy.

Datasets: Datasets are logical divisions within a zpool. Each dataset can be configured with specific settings, such as **compression**, **deduplication**, and **quotas**. This flexibility allows you to fine-tune datasets to meet the requirements of different workloads.

How Data is Organized in ZFS

Snapshots and Clones:

- **Snapshots** are point-in-time versions of datasets or volumes that provide a read-only, spaceefficient copy of data. They are ideal for backups and versioning.
- **Clones** are writable copies of snapshots, making them perfect for testing, development, or recovery environments without requiring additional storage space for the unchanged data.
- **Zvols (ZFS Volumes):** Zvols are block devices in ZFS that act like physical hard drives. Zvols are commonly used with iSCSI targets or for applications that require raw block storage rather than filebased storage.

Chapter 3: Tying it all together

Now that we have our foundation laid out we can begin designing our zpool. While these configurations are by no means exhaustive, they will give a good breakdown on the considerations when mapping out the hardware and configuration of a zpool

For the purpose of this section we'll be using a Storinator F8X2 Hybrid server. This will gives use the flexibility for multiple workloads in a single system. This assumes there is adequate networking, and resources on the server

Storinator Hybrid - F8X2

Drive Bays: 24 HDD - 16 SSD bays

Dimensions: 30.5" x 17.12" x 6.86"

Maximum Capacity HDD*: 576TB

Maximum Capacity SSD*: 121.6TB

There are a host of different factors that go into creating a zpool but for the sake of simplicity we'll condense it to these 5 decisions

- **Device (SSD, HDD, NVMe, Size, Type)**
- **VDEV (Size, Amount of VDEVs, VDEV Type)**
- **Protocol (Block vs. File Storage: NFS, SMB, etc.)**
- **Tunings (Record Size, Compression)**
- **Support VDEVs (L2ARC, SLOG, Special VDEV)**

Device (SSD, HDD, NVMe, Size, Type)

SSD vs. HDD vs. NVMe:

- **SSD (Solid-State Drive):** SSDs provide high IOPS (Input/Output Operations Per Second) and low latency, making them ideal for workloads requiring fast random reads/writes. SSDs are suited for applications where performance is critical, but they come at a higher cost per GB than HDDs.
- **HDD (Hard Disk Drive):** HDDs offer larger capacities at a lower cost per GB, making them well-suited for workloads focused on sequential read/write operations or archival data where latency is less of a concern. Workloads involving large media files or backups can benefit from the high storage capacity of HDDs.
- **NVMe (Non-Volatile Memory Express):** NVMe drives provide even higher performance than SSDs, with lower latency and higher throughput due to their direct connection to the PCIe bus. These are excellent for workloads requiring the fastest possible data access, such as real-time analytics, large-scale virtualization, or high-frequency trading.
- **Size:** The size of the drives should be selected based on both current storage needs and anticipated future growth. Capacity requirements will vary depending on the workload (e.g., a few TBs for transactional data or many TBs for large file storage).
- **Enterprise Grade:** Always opt for enterprise-grade drives for critical workloads. These drives come with features like Power Loss Protection (PLP), higher endurance (e.g., TBW >2000) to withstand heavy write operations, and better warranties, ensuring stability and performance for 24/7 operations

VDEV (Size, Amount of VDEVs, VDEV Type)

VDEV Type:

- **Mirrored VDEVs:** Best for workloads that require high IOPS and redundancy. Mirroring writes data across multiple drives, providing fault tolerance and improved read performance. This setup is ideal when the workload demands fast data access and reliability.
- **RAIDZ2/RAIDZ3 VDEVs:** RAIDZ2 (double parity) and RAIDZ3 (triple parity) offer redundancy while maximizing capacity, making them suitable for workloads focused on throughput, especially when dealing with large sequential writes and reads. RAIDZ2 allows for two disk failures, and RAIDZ3 for three, ensuring higher fault tolerance.
- **VDEV Size and Number of VDEVs:** Increasing the number of VDEVs in the pool enhances parallelism in read/write operations. Workloads that benefit from high IOPS or require multiple users accessing data simultaneously (e.g., VMs, media editing) can see improved performance with more VDEVs.

Protocol (Block vs. File Storage: NFS, SMB, etc.)

- **Block Storage (iSCSI, Fibre Channel):** Block storage is well-suited for workloads where low-latency access is critical. It provides raw block-level access to storage, allowing applications (e.g., virtual machines or databases) to manage files directly, without the overhead of file-level management.
- **File Storage (NFS, SMB):** File storage protocols like NFS and SMB are better suited for shared environments where multiple users or systems need access to files. SMB is commonly used in cross-platform environments (e.g., Windows/macOS), while NFS is more common in Unix/Linux systems due to lower overhead and native integration.

Tunings (Record Size, Compression)

- **Record Size:** Match the record size to the type of workload
- For workloads with **random, small I/O operations,** such as transactional systems or file servers, smaller record sizes (e.g., 8KiB or 16KiB) are optimal. These settings allow for efficient use of storage and improve performance by minimizing fragmentation.
- For workloads that handle **large, sequential file operations,** such as media production or backup systems, larger record sizes (e.g., 1MiB) are more efficient, reducing the number of I/O operations and improving throughput.
- **Compression: LZ4 compression** is recommended for most workloads. It offers a good balance between performance and space savings, with minimal impact on CPU. For workloads where space optimization is crucial but CPU overhead is acceptable, stronger compression algorithms (e.g., GZIP) can be used.

Support VDEVs (L2ARC, SLOG, Special VDEV)

L2ARC (Read Cache):

For read-heavy workloads that frequently access the same data (e.g., virtual machine images, media libraries), adding an L2ARC device can improve performance by caching frequently accessed data on fast SSDs. This reduces the need to repeatedly access slower HDDs, improving overall read performance. Make sure you have enough ram or thast ram is maxed out the L2ARC will use ~ 3GiB of ARC per 1TB of L2ARC.

- Recommended to Mirror this support vdev.
- Can be removed from pool

SLOG (Separate ZFS Intent Log):

For workloads that rely on synchronous writes (e.g., databases or certain virtualized environments), anSLOG device can enhance performance. SLOG devices are typically fast, high-endurance SSDs (like NVMe) that quickly commit writes, ensuring faster acknowledgment and lower latency for critical transactions.

- Recommended to Mirror this support vdev.
- Can be removed from pool

Special VDEV:

A special VDEV is used to store metadata or small files separately from the main pool. This can significantly improve performance for metadata-heavy workloads (e.g., filesystems with many small files or databases with lots of indexing), as the metadata can be stored on faster devices like SSDs or NVMe. EXTREMELY RECOMMENDED to mirror, even going as far as a 3 device mirror. This is what houses the data about your data if this fails your pool fails

- Cannot be removed when implemented without destroying pool

Lets start with our workloads we have two:

- Database application
- Multiple users editing 4k video

Now that we've outlined our workloads, taking everything we discussed so far we know we need to figure out our 5 factors for the pool

Let's start with our database pool.

Key Characteristics:

- **Low Latency:** Databases need fast access times to ensure minimal delay in queries and transactions.
- **High IOPS:** High I/O operations, especially with many small, random reads and writes.
- **Data Integrity:** Ensuring that data is not corrupted and is always available for transactions.
- **Moderate to High Capacity:** Depending on the size of the database and potential growth, capacity should accommodate database expansion.
- **Sync Writes:** Databases often rely on synchronous writes, which requires fast acknowledgment of data commits.

When it comes to databases it's all about latency so the lower the latency the better, with that we know that SSDs outperform HDD's by a larger margin when it comes to latency and IOPS performance.

Device (SSD, HDD, NVMe, Size, Type)

- **Storage Medium:** Enterprise SSDs are ideal for databases due to their high IOPS and low latency, which are crucial for handling the random, small read/write operations typical in databases. SSDs excel in environments where responsiveness and speed are critical.
- **Drive Size:** Select the drive size based on both current and anticipated future needs. For example, if the database requires 2TB of storage now, it is recommended to plan for growth by selecting 4 x 1.92TB SSDs to account for overhead like snapshots, logs, and future expansion.
- **Enterprise Grade:** Enterprise SSDs or NVMe drives are essential for databases due to their durability, longer warranties, and critical features like Power Loss Protection (PLP), which safeguards against data corruption during power failures. High endurance (e.g., TBW >2000) is also critical for handling the continuous writes common in databases.

VDEV (Size, Amount of VDEVs, VDEV Type)

VDEV Type:

- **Mirrored VDEVs (RAID 10 equivalent):** Mirrored VDEVs offer the best balance between performance and redundancy for database workloads. Data is written to two or more drives, providing protection against drive failure and allowing parallel reads, which boosts performance and fault tolerance. Mirroring is ideal for workloads that demand low latency and high IOPS.
- **VDEV Size and Number of VDEVs:** Increasing the number of mirrored VDEVs can enhance performance. For example, using two mirrored VDEVs with 4 SSDs would provide both higher throughput and redundancy. This configuration can easily scale as the database grows

Protocol (Block vs. File Storage: NFS, SMB, etc.)

- **Block Storage (iSCSI, Fibre Channel):** Block storage is highly recommended for databases because it offers low-latency access to raw storage blocks. This allows the database management system (DBMS) to handle its own data structures directly, optimizing performance and reducing overhead. iSCSI is a commonly used block-level protocol that works well in database environments, providing flexibility and low-latency access.

Tunings (Record Size, Compression)

- **Record Size:** The ZFS record size should match the database block size to optimize performance:
- **PostgreSQL:** Typically uses 8KiB blocks, so the ZFS record size should be set to 8KiB for optimal performance.
- **MySQL/MariaDB:** Typically uses 16KiB blocks, so the ZFS record size should be set to 16KiB. By matching the record size to the database's internal block size, ZFS will handle I/O operations more efficiently, reducing overhead and improving both read and write performance.
- **Compression:** LZ4 compression should be enabled. LZ4 is a fast, low-overhead compression algorithm that can reduce the amount of data written to disk without significantly impacting performance. This is especially helpful in databases where the data can be compressed effectively, saving space and improving write performance

Support VDEVs (L2ARC, SLOG, Special VDEV)

- **SLOG (Separate ZFS Intent Log):** A SLOG device is essential for databases that rely on synchronous writes. SLOG stores the ZFS intent log on a fast, dedicated device (often NVMe or high-endurance SSD), improving the performance of synchronous writes by allowing the database to quickly commit transactions. This reduces write latency and ensures data integrity in the event of a power failure.

Example Database ZFS Pool Configuration:

Devices:

- **4 x 1.92TB Enterprise SSDs** or **NVMe drives** in 2 mirrored VDEVs for redundancy and high IOPS.
- **2x NVMe SSD** for SLOG to optimize synchronous write performance.

VDEVs:

- **mirrored VDEVs** (4 SSDs) to balance performance and redundancy, ensuring high IOPS and fault tolerance.

Protocol:

- **iSCSI (block storage)** for low-latency, direct access to storage blocks.

Tunings:

- **Record size:** 8KiB (PostgreSQL) or 16KiB (MySQL/MariaDB).
- **Compression:** LZ4 enabled for optimal space savings with minimal performance impact.

Support VDEVs:

- **SLOG:** Yes, to improve synchronous write performance and ensure data integrity.

Let's now cover our 4k editing workload.

Key Characteristics:

- **High Throughput:** Large 4K video files require fast sequential read and write speeds to handle editing and rendering efficiently.
- **Multiple User Access:** The pool must support multiple users editing and accessing large files simultaneously without causing performance bottlenecks.
- **Data Integrity:** Redundancy is essential to protect against data loss, as media assets are valuable.
- **High Capacity:** Large storage volumes are required to accommodate the substantial file sizes of 4K video projects.
- **Consistency:** The system must provide reliable, consistent performance over time, even when multiple users are working on the same project.

For this workload, HDDs may be more cost-effective for capacity, but SSDs or NVMe drives offer better performance, particularly in environments with multiple users or high read/write activity.

Device (SSD, HDD, NVMe, Size, Type)

Storage Medium: Enterprise HDDs (e.g., Seagate Exos) provide large capacity and reasonable performance for handling 4K media files, especially when cost is a factor. Enterprise SSDs or NVMe drives can be used in cache roles (L2ARC) or for higher performance editing workloads, especially in cases where fast sequential writes are crucial.

Drive Size: 4K video files are very large, so selecting large-capacity drives is essential. HDDs in the 12TB–18TB range are common for high-capacity workloads. If using SSDs or NVMe drives, capacities of 3.84TB or higher should be considered for primary storage or caching layers.

Enterprise Grade: Enterprise-grade drives are required for the 24/7 usage and heavy workloads involved in 4K video editing. These drives offer features such as Power Loss Protection (PLP) and higher endurance to prevent data loss and maintain performance over time, which is critical when dealing with high-value media assets.

VDEV (Size, Amount of VDEVs, VDEV Type)

- **VDEV Type: RAIDZ2 or RAIDZ3 VDEVs:** RAIDZ2 (double parity) provides a good balance between redundancy and capacity for 4K video editing. This configuration protects against two simultaneous drive failures per vdev while maintaining adequate performance for large sequential reads and writes. RAIDZ2 is generally preferred, but RAIDZ3 offers an extra layer of fault tolerance for critical projects and can be a good choice for critical archival data
- **VDEV Size and Number of VDEVs:** More VDEVs improve parallelism, which is important for environments with multiple users. For example, using **two RAIDZ2 VDEVs with 6 HDDs each** would provide high capacity and redundancy, ensuring that large files can be accessed or edited by several users simultaneously without bottlenecks.

Protocol (Block vs. File Storage: NFS, SMB, etc.)

- **File Storage (NFS, SMB):** For 4K video editing, file-level access is recommended. SMB is widely used in media production environments for cross-platform access (Windows/macOS), providing seamless integration with editing software. NFS may be better suited for Linux-based systems and has lower overhead, but may require more configuration to support mixed environments. File storage protocols allow for shared access, ensuring multiple users can work on the same project files without conflict.

Tunings (Record Size, Compression)

- **Record Size:** Large sequential file operations like video editing benefit from larger record sizes:
  - Set the **record size to between 512KiB or 1MiB** to handle large video files efficiently. This helps avoid excessive fragmentation and optimizes the pool for the large, sequential writes typical of video editing.
- **Compression:** LZ4 compression is recommended even in video editing pools. While video files are often already compressed, LZ4 can still reduce overhead for metadata and less compressible assets, improving space utilization without significant performance loss.

Example 4K Video Editing ZFS Pool Configuration:

Devices:

- **12 x 12TB Enterprise HDDs** in 2 RAIDZ2 VDEVs for large capacity and redundancy.
- **2x 1.92TB Enterprise SSD** or **NVMe drive** for L2ARC to improve read performance.
- **3x 480GB Enterprise SSD** or **NVMe drive** for Special vdev

VDEVs:

- **2 RAIDZ2 VDEVs** with 6 HDDs each to provide redundancy and high capacity for storing large video files.

Protocol:

- SMB for shared access across multiple users

Tunings:

- **Record size:** 1MiB to optimize for large, sequential file writes and reads.
- **Compression:** LZ4 enabled for metadata compression and minimal impact on performance.

Support VDEVs:

- **L2ARC:** Yes, to improve read performance by caching frequently accessed media.
- **Special VDEV:** Yes, to offload metadata and improve performance when handling large file libraries.

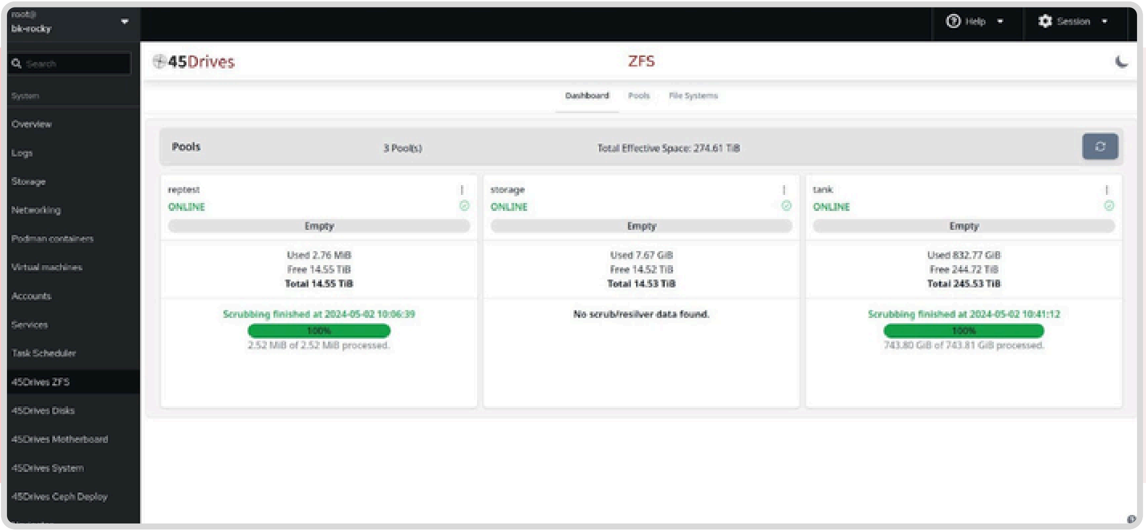
Chapter 4: Houston ZFS Administration

Now that we have the foundation for ZFS, let’s get to building a pool in the **Houston Command Center.**

Houston is a web-based management layer developed by 45Drives, built on the open-source Cockpit Project. Houston was developed to simplify server management, regardless of administrator experience level.

45Drives ZFS

- First, we'll go to the ZFS module. We can see we already have some pools, but we want to add another for an archive workload.

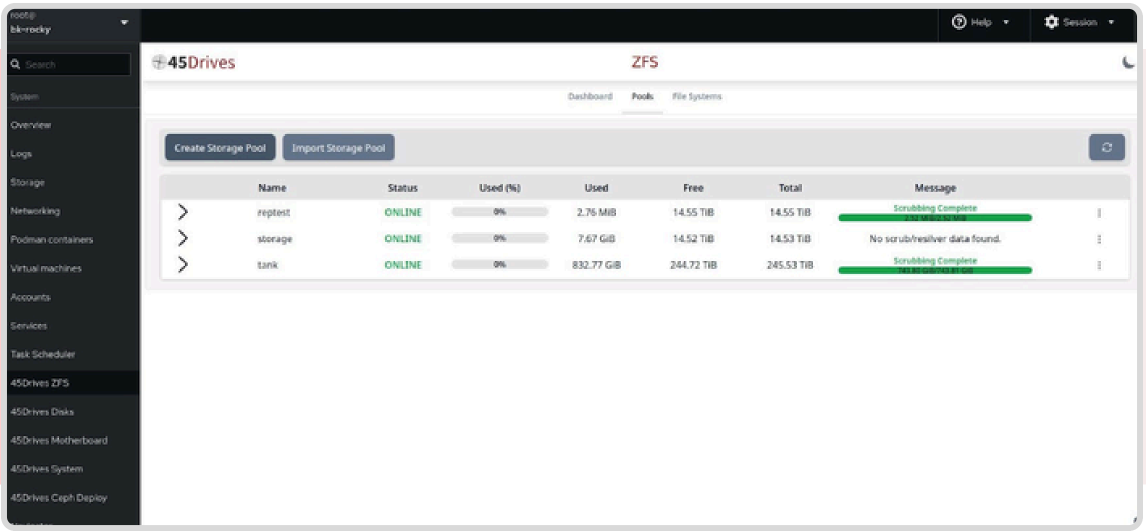


Pools 3 Pool(s) Total Effective Space: 274.61 TiB

reptest	storage	tank
ONLINE	ONLINE	ONLINE
Empty	Empty	Empty
Used 2.76 MiB Free 14.55 TiB **Total 14.55 TiB**	Used 7.67 GiB Free 14.52 TiB **Total 14.53 TiB**	Used 832.77 GiB Free 244.72 TiB **Total 245.53 TiB**
Scrubbing finished at 2024-05-02 10:06:39 100% 2.52 MiB of 2.52 MiB processed.	No scrub/resilver data found.	Scrubbing finished at 2024-05-02 10:41:12 100% 743.80 GiB of 743.81 GiB processed.

45Drives ZFS ➜Pools

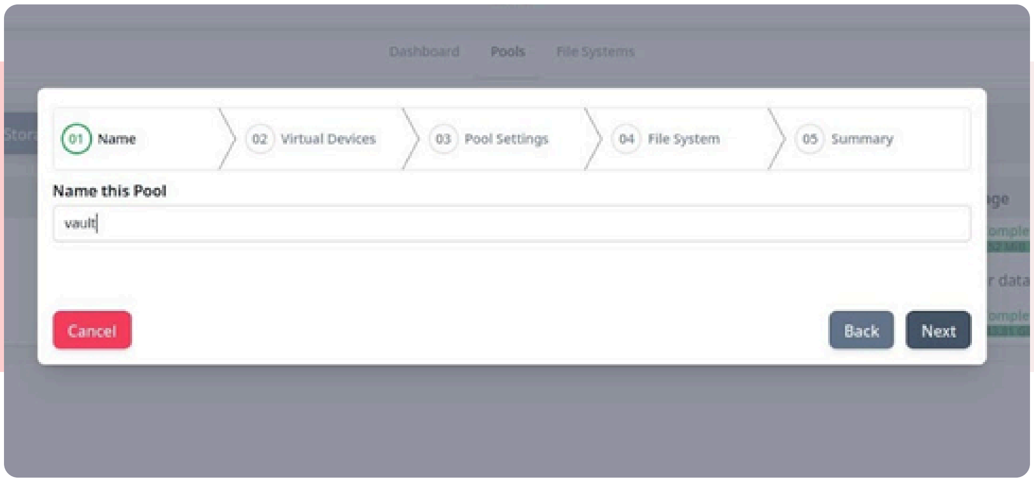
- Once we click on pools, we can click "Create Storage Pool".



Name	Status	Used (%)	Used	Free	Total	Message
reptest	ONLINE	0%	2.76 MiB	14.55 TiB	14.55 TiB	Scrubbing Complete
storage	ONLINE	0%	7.67 GiB	14.52 TiB	14.53 TiB	No scrub/resilver data found.
tank	ONLINE	0%	832.77 GiB	244.72 TiB	245.53 TiB	Scrubbing Complete

45Drives ZFS ➜ Pools ➜ Create Storage Pool

- Give it a name.

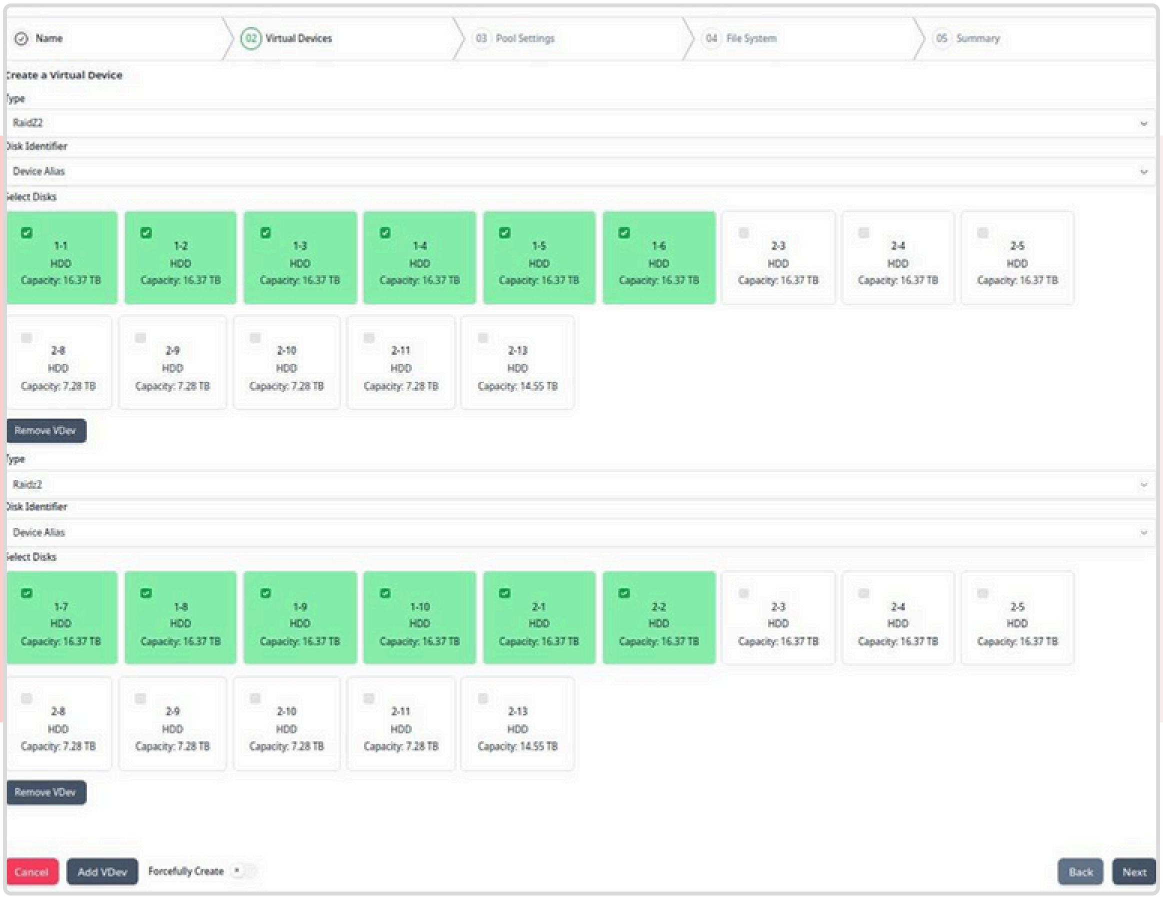


Name this Pool

vault

45Drives ZFS ➜ Pools ➜ Create Storage Pool ➜ Virtual Devices

- We'll create 2 raidz2 vdevs, consisting of 6 drives each.



Create a Virtual Device

Type: RaidZ2

Disk Identifier: Device Alias

Select Disks:

✓ 1-1 HDD Capacity: 16.37 TB
✓ 1-2 HDD Capacity: 16.37 TB
✓ 1-3 HDD Capacity: 16.37 TB
✓ 1-4 HDD Capacity: 16.37 TB
✓ 1-5 HDD Capacity: 16.37 TB
✓ 1-6 HDD Capacity: 16.37 TB
2-3 HDD Capacity: 16.37 TB
2-4 HDD Capacity: 16.37 TB
2-5 HDD Capacity: 16.37 TB
2-8 HDD Capacity: 7.28 TB
2-9 HDD Capacity: 7.28 TB
2-10 HDD Capacity: 7.28 TB
2-11 HDD Capacity: 7.28 TB
2-13 HDD Capacity: 14.55 TB

Type: Raidz2

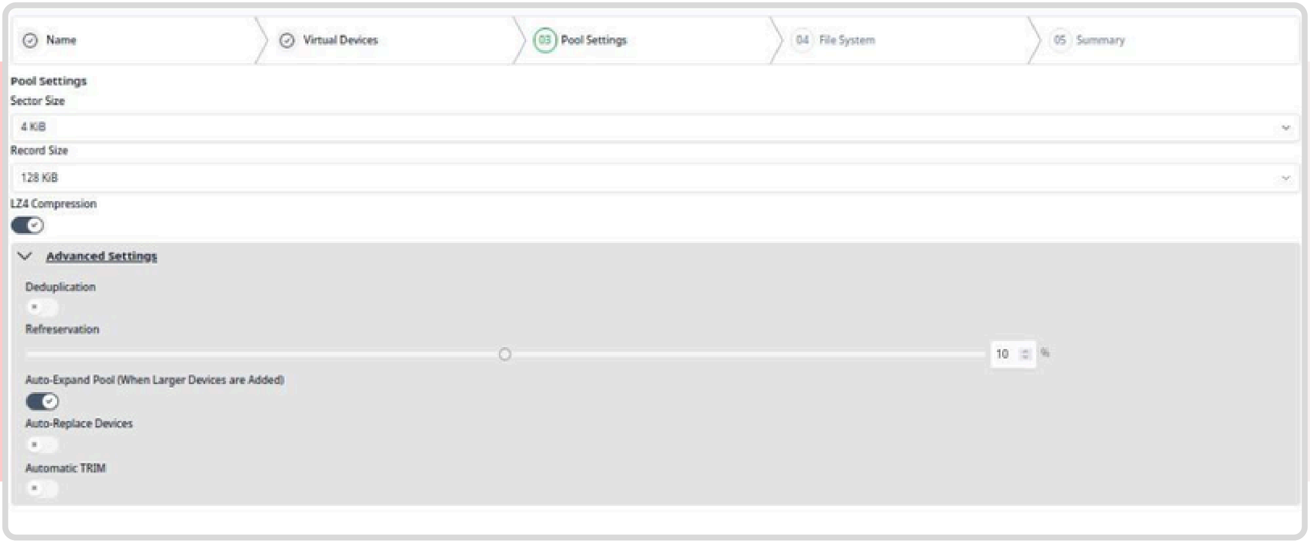
Disk Identifier: Device Alias

Select Disks:

✓ 1-7 HDD Capacity: 16.37 TB
✓ 1-8 HDD Capacity: 16.37 TB
✓ 1-9 HDD Capacity: 16.37 TB
✓ 1-10 HDD Capacity: 16.37 TB
✓ 2-1 HDD Capacity: 16.37 TB
✓ 2-2 HDD Capacity: 16.37 TB
2-3 HDD Capacity: 16.37 TB
2-4 HDD Capacity: 16.37 TB
2-5 HDD Capacity: 16.37 TB
2-8 HDD Capacity: 7.28 TB
2-9 HDD Capacity: 7.28 TB
2-10 HDD Capacity: 7.28 TB
2-11 HDD Capacity: 7.28 TB
2-13 HDD Capacity: 14.55 TB

45Drives ZFS ➔ Pools ➔ Create Storage Pool ➔ Virtual Devices ➔ Pool Settings

- For our pool settings, we'll leave as the defaults.



Pool Settings

Sector Size: 4 KiB

Record Size: 128 KiB

LZ4 Compression: ✓

Advanced Settings

Deduplication: ✗

Refreservation: 10 %

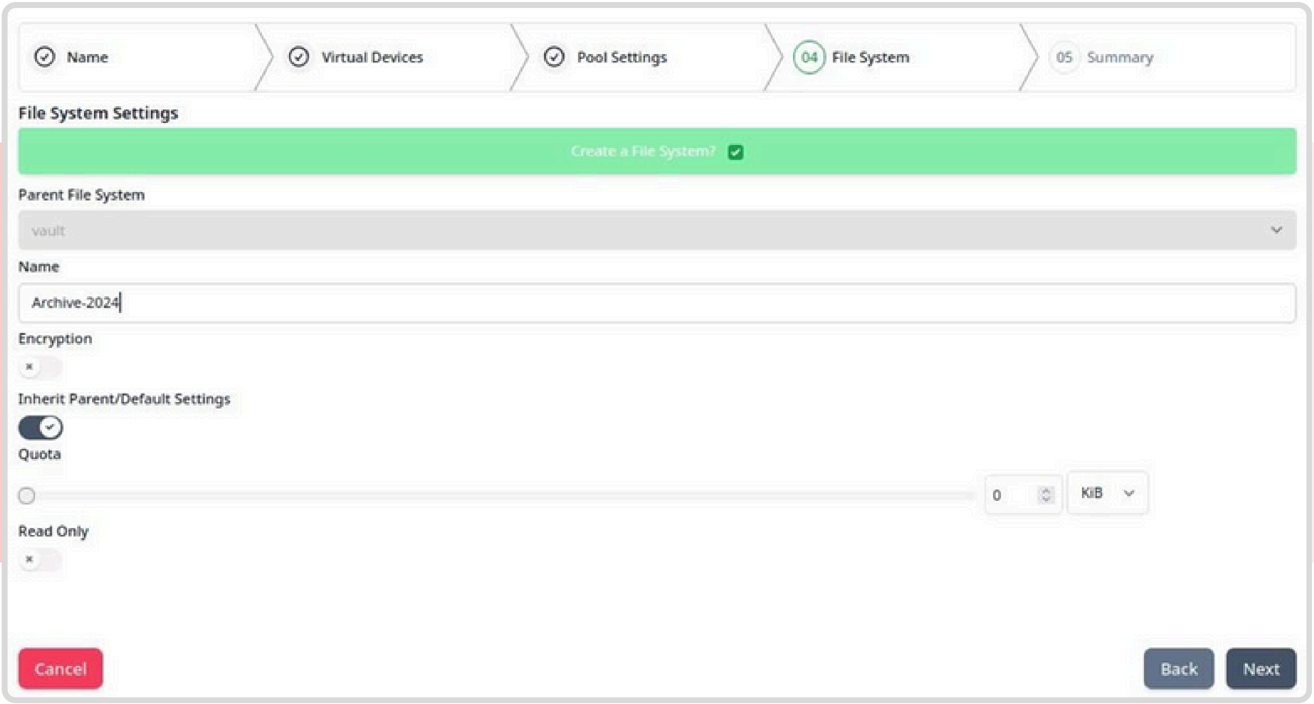
Auto-Expand Pool (When Larger Devices are Added): ✓

Auto-Replace Devices: ✗

Automatic TRIM: ✗

45Drives ZFS ➔ Pools ➔ Create Storage Pool ➔ Virtual Devices ➔ Pool Settings ➔ File System

- Next we'll want a filesystem. We'll give it a name, and leave these settings as the default



File System Settings

Create a File System? ✓

Parent File System: vault

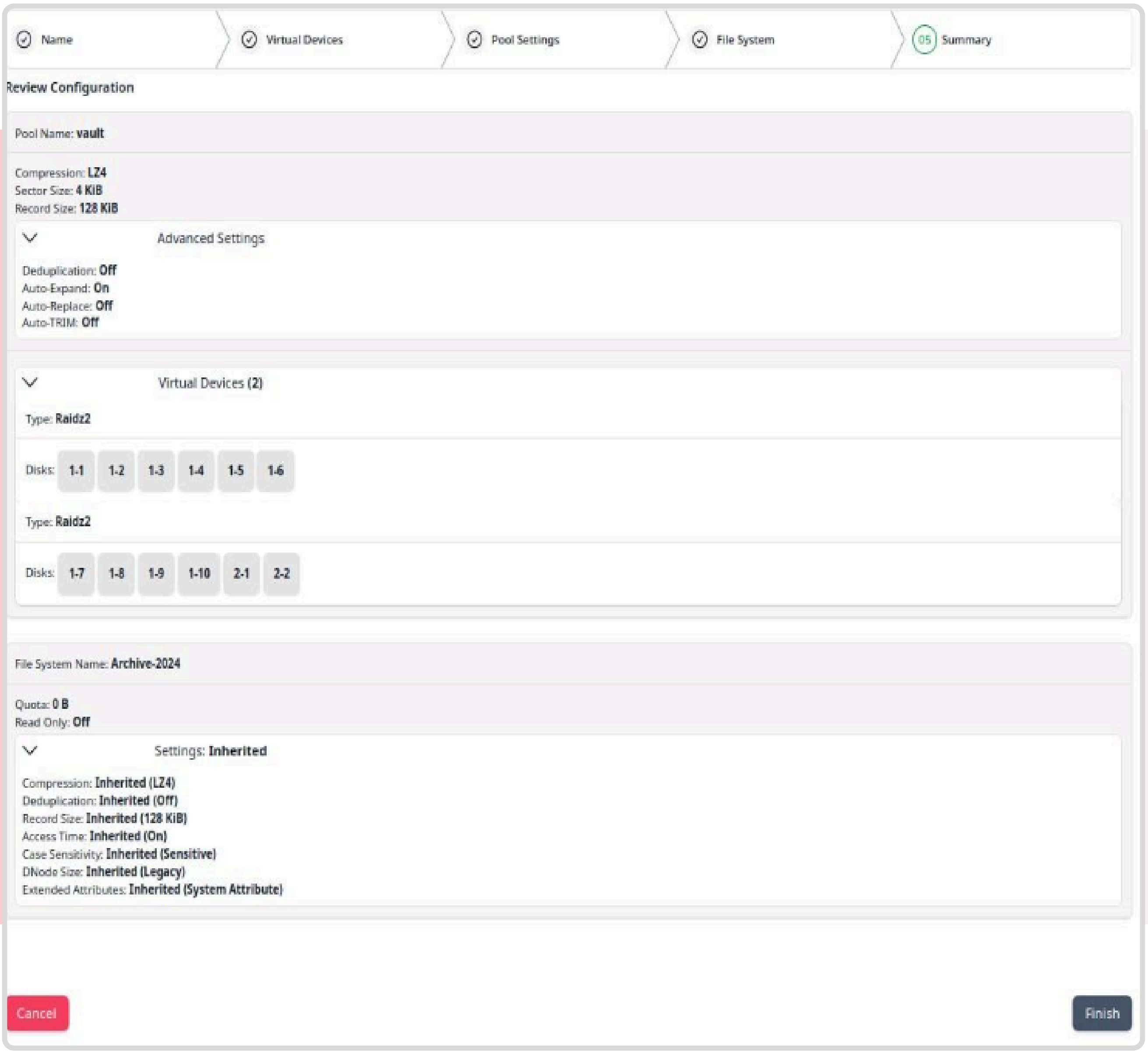
Name: Archive-2024

Encryption: ✗

Inherit Parent/Default Settings: ✓

Quota: 0 KiB

Read Only: ✗



Review Configuration

Pool Name: **vault**

Compression: **LZ4**
Sector Size: **4 KiB**
Record Size: **128 KiB**

Advanced Settings

Deduplication: **Off**
Auto-Expand: **On**
Auto-Replace: **Off**
Auto-TRIM: **Off**

Virtual Devices (2)

Type: **Raidz2**

Disks: 1-1 1-2 1-3 1-4 1-5 1-6

Type: **Raidz2**

Disks: 1-7 1-8 1-9 1-10 2-1 2-2

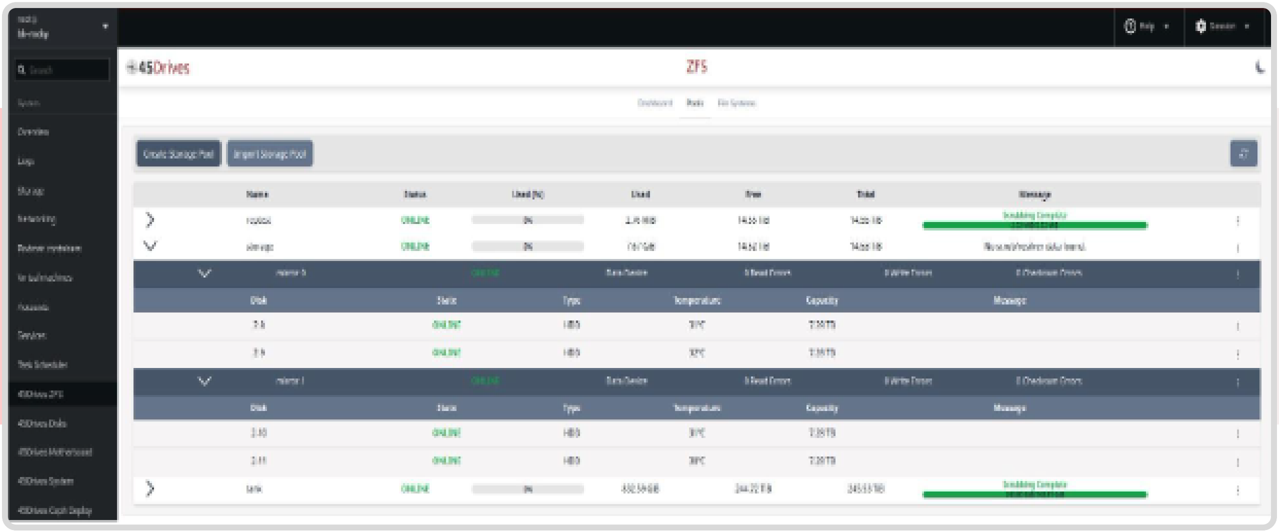
File System Name: **Archive-2024**

Quota: **0 B**
Read Only: **Off**

Settings: **Inherited**

Compression: **Inherited (LZ4)**
Deduplication: **Inherited (Off)**
Record Size: **Inherited (128 KiB)**
Access Time: **Inherited (On)**
Case Sensitivity: **Inherited (Sensitive)**
DNode Size: **Inherited (Legacy)**
Extended Attributes: **Inherited (System Attribute)**

- Now, we'll take a look at one of our other pools. When we click the drop down, we can see the status of the pool and its drives.



	Name	Status	Used (%)	Used	Free	Total	Message
>	[illegible]	ONLINE	0%	[illegible]	14.53 TiB	14.53 TiB	Scrubbing Complete
v	storage	ONLINE	0%	7.67 GiB	14.52 TiB	14.53 TiB	[illegible]
>	tank	ONLINE	0%	[illegible]	[illegible]	[illegible]	Scrubbing Complete

	mirror-0	ONLINE	Data Device	0 Read Errors	0 Write Errors	0 Checksum Errors

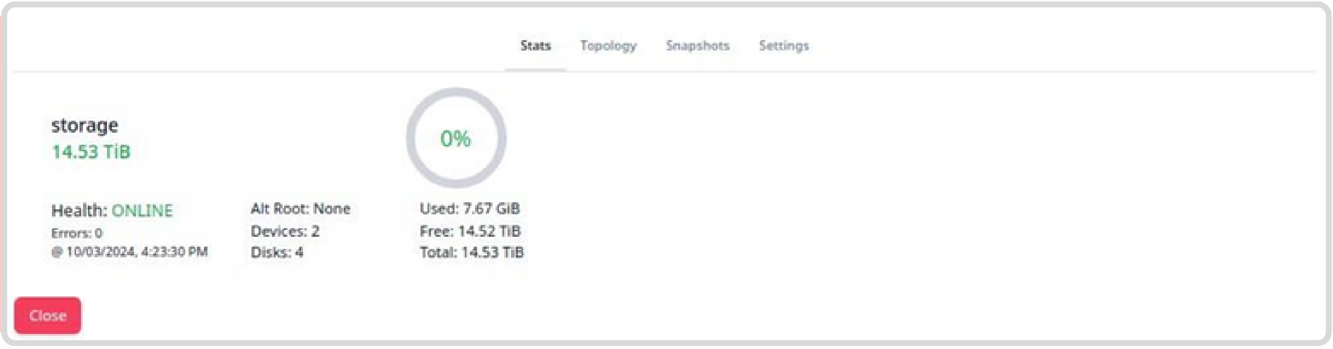
Disk	Status	Type	Temperature	Capacity	Message
[illegible]	ONLINE	HDD	[illegible]	7.28 TB	
[illegible]	ONLINE	HDD	[illegible]	7.28 TB	

	mirror-1	ONLINE	Data Device	0 Read Errors	0 Write Errors	0 Checksum Errors

Disk	Status	Type	Temperature	Capacity	Message
[illegible]	ONLINE	HDD	[illegible]	7.28 TB	
[illegible]	ONLINE	HDD	[illegible]	7.28 TB	

**If we want to get more information, we can click the hamburger button (the three vertical dots) on the right of the pool and click "Pool Details".*

- Starting with stats, we can get a quick glance:



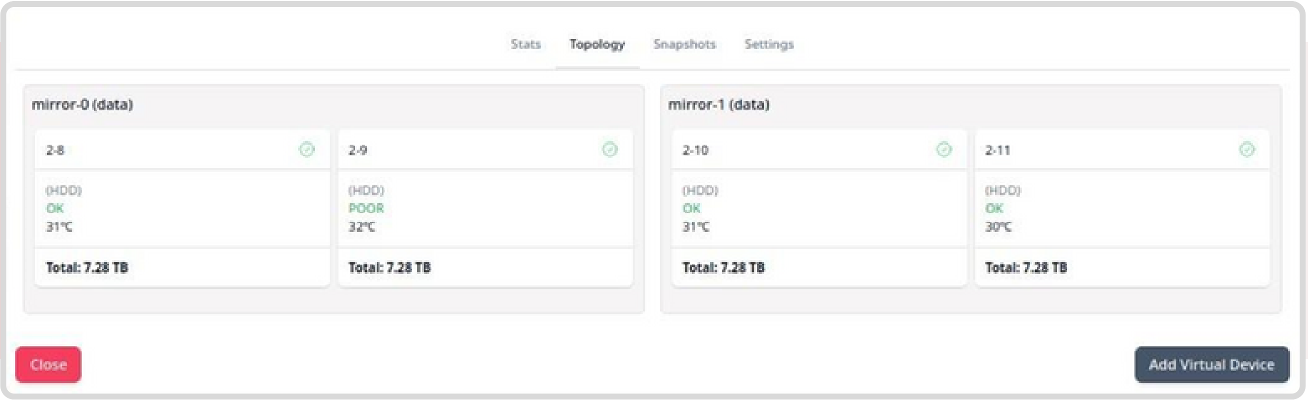
storage
14.53 TiB

Health: ONLINE
Errors: 0
@ 10/03/2024, 4:23:30 PM

Alt Root: None
Devices: 2
Disks: 4

Used: 7.67 GiB
Free: 14.52 TiB
Total: 14.53 TiB

- Next, Topology where we can see out vdev layout



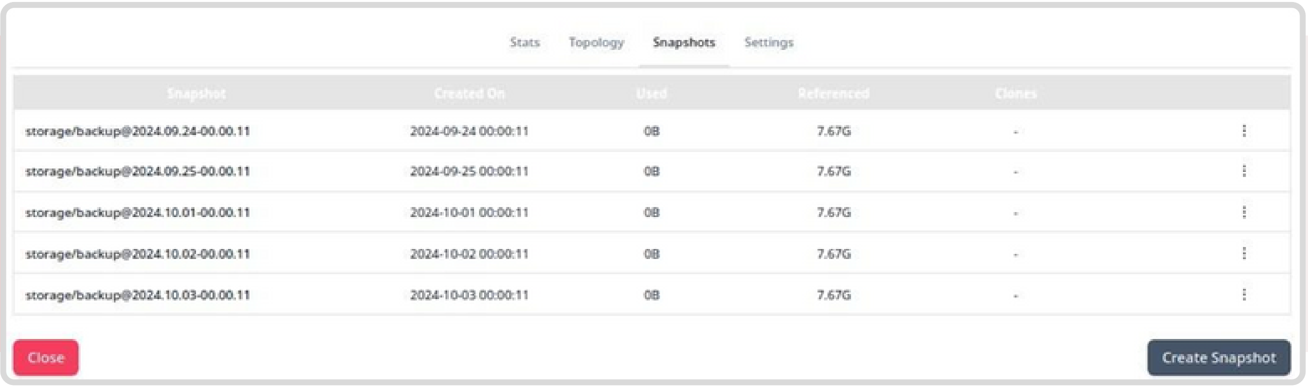
mirror-0 (data)

2-8	2-9
(HDD) OK 31°C	(HDD) POOR 32°C
Total: 7.28 TB	Total: 7.28 TB

mirror-1 (data)

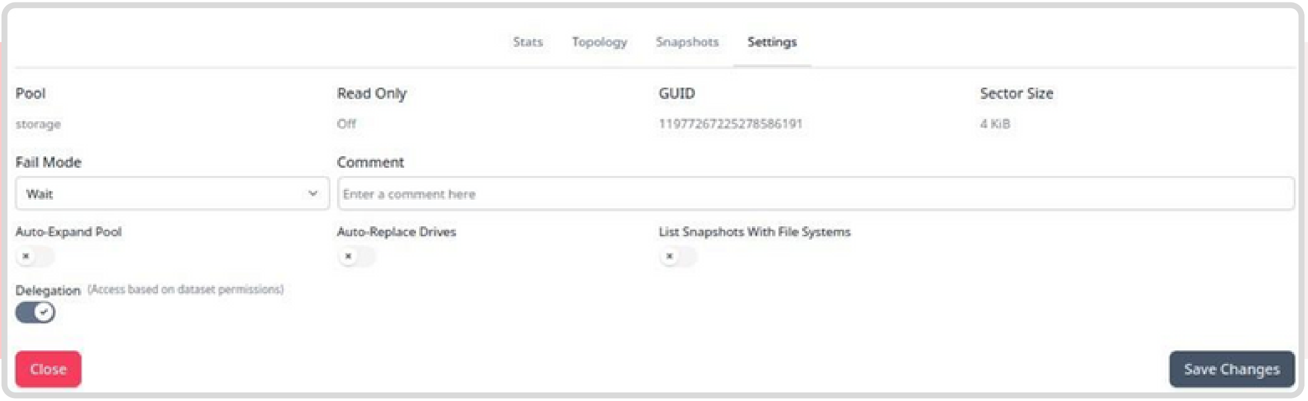
2-10	2-11
(HDD) OK 31°C	(HDD) OK 30°C
Total: 7.28 TB	Total: 7.28 TB

- Then, our snapshots on the pool.



Snapshot	Created On	Used	Referenced	Clones	
storage/backup@2024.09.24-00.00.11	2024-09-24 00:00:11	0B	7.67G	-	⋮
storage/backup@2024.09.25-00.00.11	2024-09-25 00:00:11	0B	7.67G	-	⋮
storage/backup@2024.10.01-00.00.11	2024-10-01 00:00:11	0B	7.67G	-	⋮
storage/backup@2024.10.02-00.00.11	2024-10-02 00:00:11	0B	7.67G	-	⋮
storage/backup@2024.10.03-00.00.11	2024-10-03 00:00:11	0B	7.67G	-	⋮

- Finally, some additional settings for our pool.



Pool	Read Only	GUID	Sector Size
storage	Off	11977267225278586191	4 KiB

Fail Mode: Wait

Comment: Enter a comment here

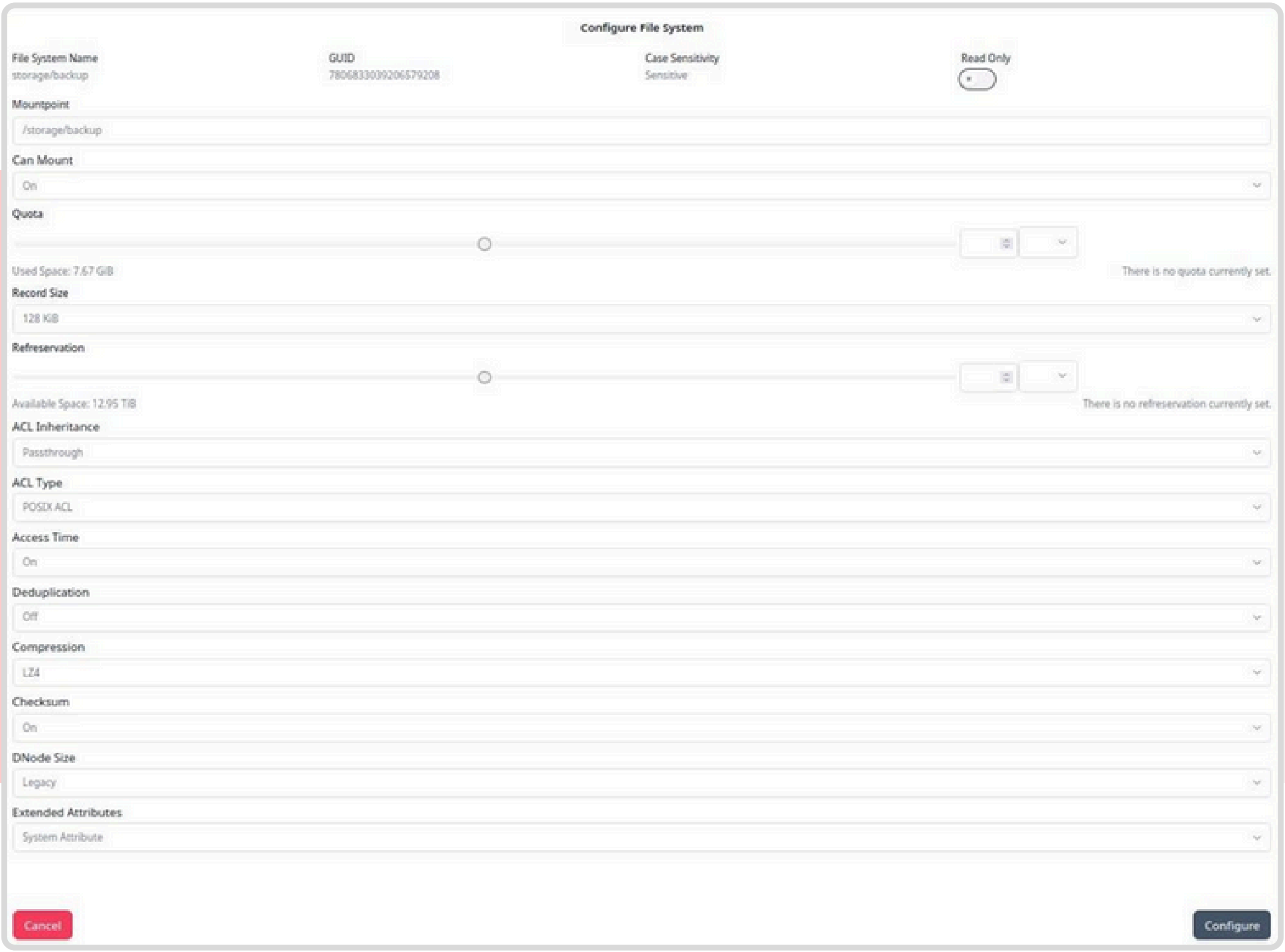
Auto-Expand Pool: ✗

Auto-Replace Drives: ✗

List Snapshots With File Systems: ✗

Delegation (Access based on dataset permissions): ✓

- Last up for the ZFS module is our file system tab, where we can create additional filesystems, or make changes to existing file systems.


Configure File System

File System Name	GUID	Case Sensitivity	Read Only
storage/backup	7806833039206579208	Sensitive	

Mountpoint: /storage/backup

Can Mount: On

Quota

Used Space: 7.67 GiB — There is no quota currently set.

Record Size: 128 KiB

Refreservation

Available Space: 12.95 TiB — There is no refreservation currently set.

ACL Inheritance: Passthrough

ACL Type: POSIX ACL

Access Time: On

Deduplication: Off

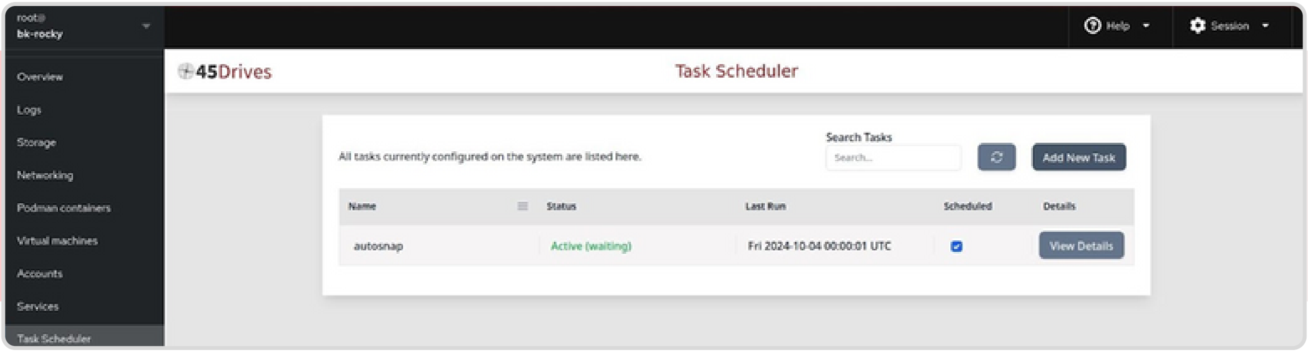
Compression: LZ4

Checksum: On

DNode Size: Legacy

Extended Attributes: System Attribute

- Let’s say we want to create a local snapshot every hour and send it to a remote system every hour as well.



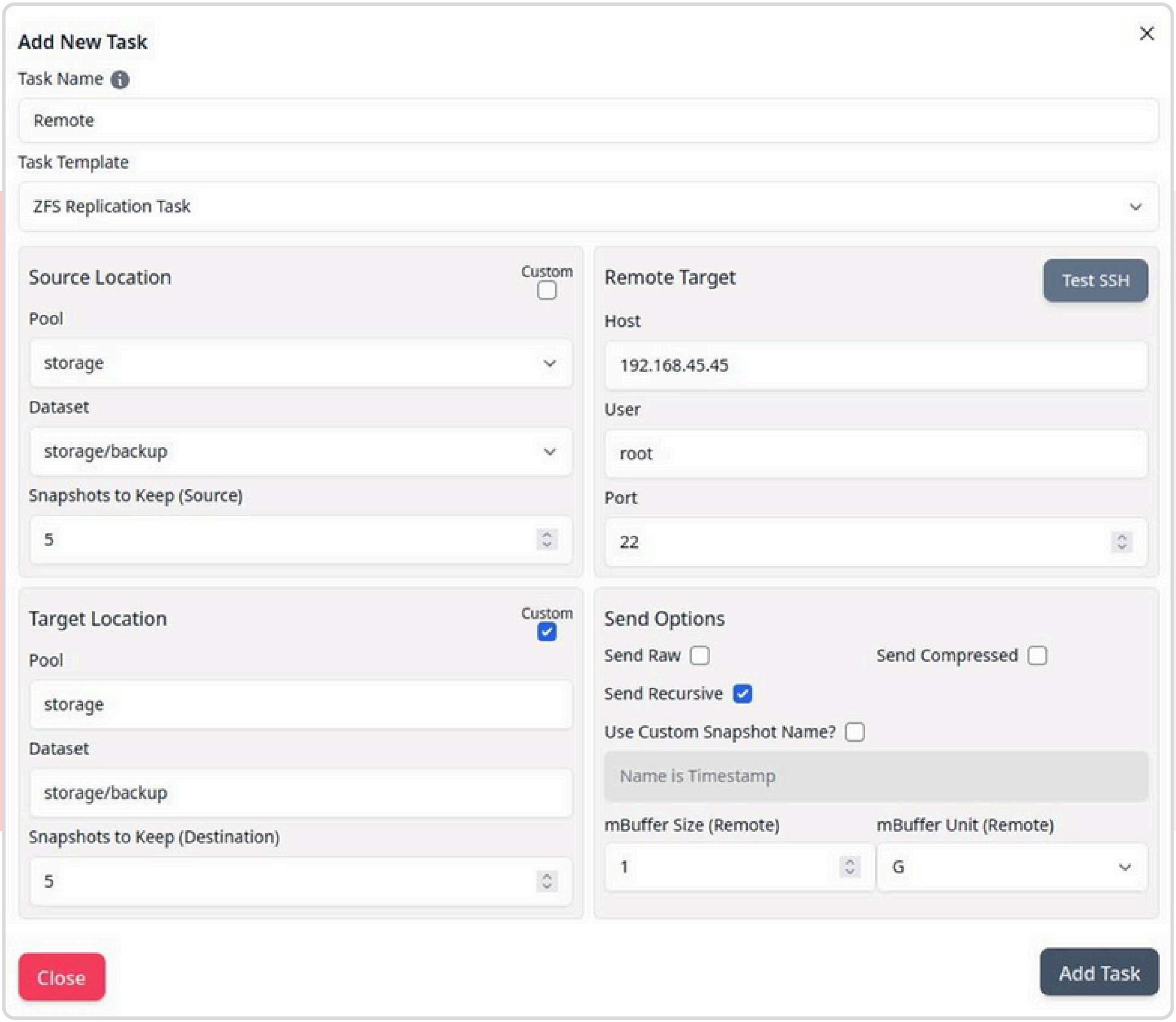
Task Scheduler

All tasks currently configured on the system are listed here.

Search Tasks

Name	Status	Last Run	Scheduled	Details
autosnap	Active (waiting)	Fri 2024-10-04 00:00:01 UTC	✓	View Details

- We'll want to keep the 5 most recent snapshots on the source and destination:



Add New Task

Task Name: Remote

Task Template: ZFS Replication Task

Source Location (Custom ☐)

Pool: storage

Dataset: storage/backup

Snapshots to Keep (Source): 5

Remote Target

Host: 192.168.45.45

User: root

Port: 22

Target Location (Custom ✓)

Pool: storage

Dataset: storage/backup

Snapshots to Keep (Destination): 5

Send Options

Send Raw ☐ Send Compressed ☐

Send Recursive ✓

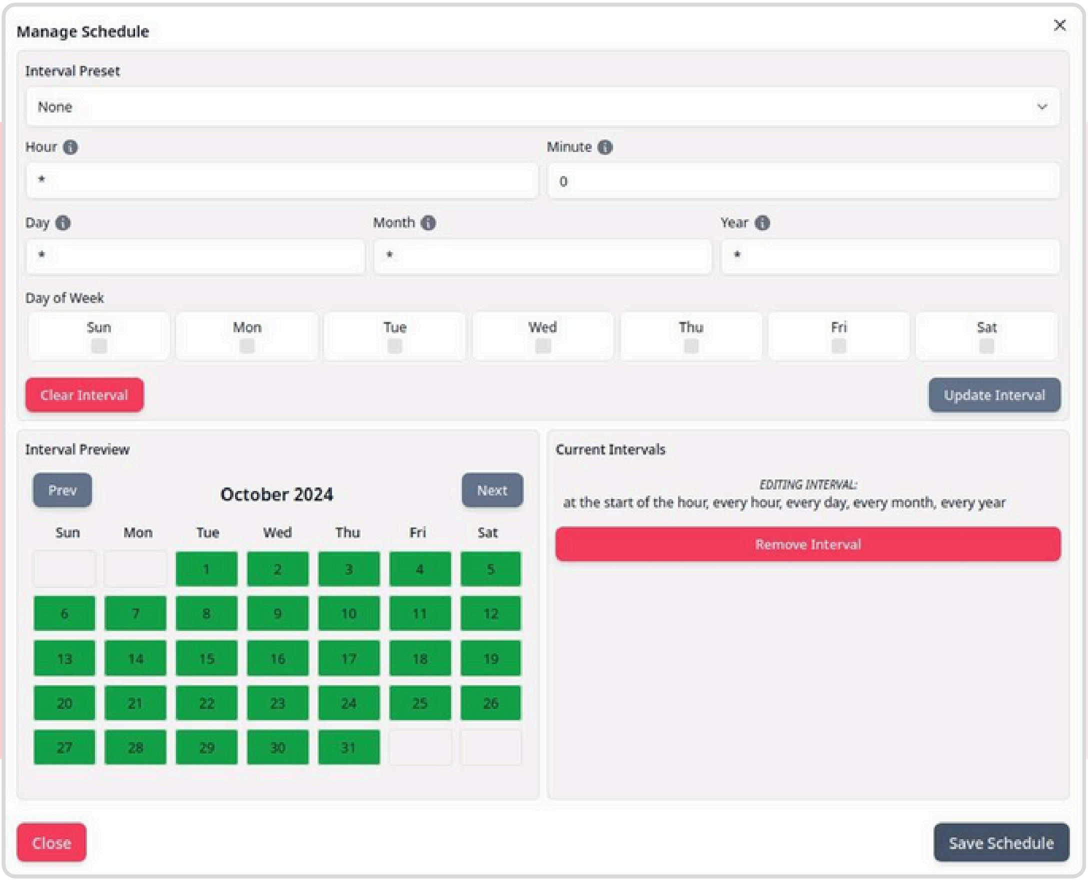
Use Custom Snapshot Name? ☐

Name is Timestamp

mBuffer Size (Remote): 1

mBuffer Unit (Remote): G

- Now, we can set up the intervals to take on the local, and send to the remote server hourly.



Manage Schedule

Interval Preset: None

Hour	Minute
*	0

Day	Month	Year
*	*	*

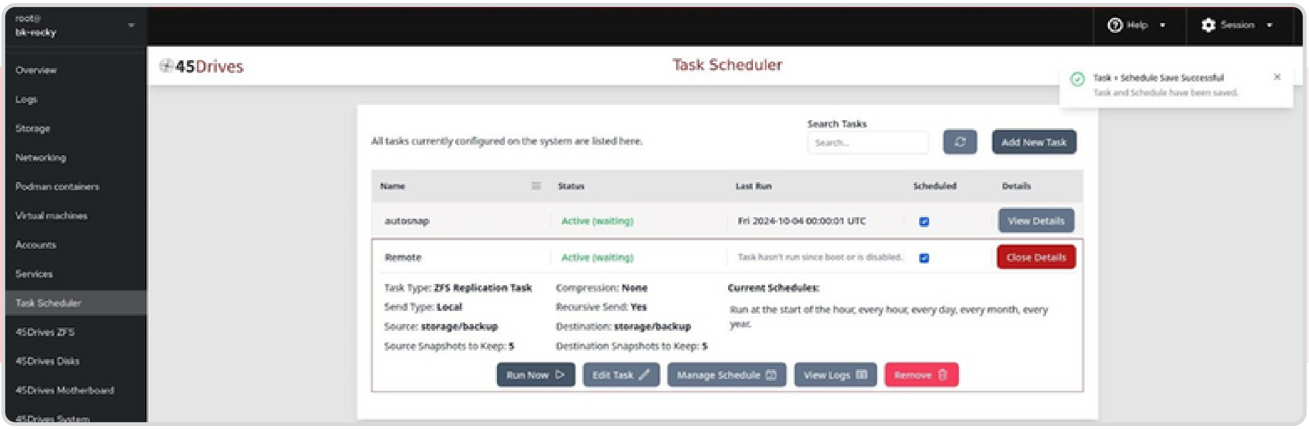
Day of Week: Sun Mon Tue Wed Thu Fri Sat

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Current Intervals

EDITING INTERVAL:
at the start of the hour, every hour, every day, every month, every year

- Once we click Save Schedule, we can see it's been created. We can get more info by clicking *View Details*.



Task + Schedule Save Successful
Task and Schedule have been saved.

All tasks currently configured on the system are listed here.

Name	Status	Last Run	Scheduled	Details
autosnap	Active (waiting)	Fri 2024-10-04 00:00:01 UTC	✓	View Details
Remote	Active (waiting)	Task hasn't run since boot or is disabled.	✓	Close Details

Task Type: **ZFS Replication Task**
Send Type: **Local**
Source: **storage/backup**
Source Snapshots to Keep: **5**
Compression: **None**
Recursive Send: **Yes**
Destination: **storage/backup**
Destination Snapshots to Keep: **5**
Current Schedules:
Run at the start of the hour, every hour, every day, every month, every year.

Chapter 5: About 45Drives

About 45Drives

At 45Drives, we serve as the single point of accountability for open-source virtualization and storage solutions — including hardware, software, and service. Our initial architecture design, configuration, ongoing support, and end-to-end services guarantee seamless management and optimization at every stage. We do it all without any lock-in through our ‘New Enterprise’ Business model.

Whether you are new to ZFS, or you’re an experienced system administrator simply looking for more information about this file system, our superior line of storage solutions can provide you with the necessary infrastructure to implement a highly customized architecture for your exact needs.

Learn more about our product solutions at: www.45drives.com/products

We also encourage you to visit our YouTube channel and explore our dedicated ZFS Playlist for your learning enjoyment. Watch at: https://bit.ly/45Drives-ZFS

ZFS Webinar

45Drives offers a variety of free public and private webinars, including **ZFS 101: Everything You Need to Know About ZFS File Storage**. In this webinar, and as a complement to the knowledge gained from this e-book, we discuss the fundamentals of the ZFS file system.

Through the personal touch of learning from our storage experts, we cover topics like RAID types, volume manager, filesystem, the ZFS storage hierarchy, and how ZFS can easily be managed with the 45Drives Houston UI interface. By using this book and attending our ZFS webinar, you will gain the necessary information to begin your journey with the ZFS File System.

Visit https://www.45drives.com/contact/webinar/ today!

Chapter 6: Glossary of Terms

Glossary of Terms in ZFS

- **ARC (Adaptive Replacement Cache)**: A key caching component in ZFS that dynamically uses available RAM to store frequently accessed data, significantly speeding up data retrieval and improving overall system performance.

- **Atomic Writes**: Atomic writes ensure complete and consistent data operations such that a write operation to the filesystem is either fully completed or not performed at all. This feature prevents data corruption by maintaining data integrity during system crashes or power failures, ensuring that partial writes do not remain, thereby safeguarding the file system's consistency.

- **Checksum**: A method used by ZFS to verify data integrity by generating a unique value based on the contents of a data block, allowing for error detection and correction.

- **Compression**: A feature in ZFS that reduces the amount of storage space required by encoding data more efficiently, leading to cost savings and improved performance in most scenarios—laz4 is the default choice.

- **Deduplication**: A data reduction feature in ZFS that eliminates redundant copies of data to save space within a storage pool.

- **Hardware RAID**: Utilizes a dedicated controller to manage disk arrays for enhanced redundancy and performance. By using its own processor, it offloads data operations from the host CPU, offering configurations like RAID 0, 1, 5, and 6 for improved data management and storage efficiency in enterprise environments.

- **Hot Spares**: Additional disk drives in a Zpool that are kept on standby to automatically replace a failed disk, minimizing downtime and maintaining data availability.

- **L2ARC (Level 2 Adaptive Replacement Cache)**: An extension of ZFS's caching mechanism that allows frequently accessed data to be stored in faster storage media, such as SSDs, to improve read performance.

- **Mirror:** A type of virtual device setup in ZFS that duplicates data onto two or more disks simultaneously, ensuring immediate data redundancy and enhanced reliability.

- **Quota**: A limit set on the amount of storage space a dataset can use, preventing any single dataset from consuming all available storage in a Zpool and ensuring fair resource distribution among multiple datasets.

Glossary of Terms in ZFS

- **RAIDZ**: RAID-like configurations used in ZFS to ensure redundancy and protect against disk failure, promoting continuous data availability.

- **Replication**: Creating copies of ZFS datasets across different systems or locations to ensure data redundancy and improve disaster recovery capabilities.

- **Resilvering**: Resilvering in ZFS is the process of repairing and rebuilding data on a disk after a failure or when a new disk is added to a pool. It ensures that all data is correctly mirrored or distributed across the disks in accordance with the chosen redundancy scheme, thereby maintaining data integrity and preventing potential data loss.

- **Snapshot**: A snapshot is a read-only copy of a filesystem or volume at a specific point in time, thus allowing users to preserve the state of the data without interrupting operations, enabling data recovery, rollback, and historical analysis.

- **Software RAID**: Software RAID the system's CPU to manage data redundancy and distribution across multiple disks without the need for dedicated RAID hardware. It integrates directly with the filesystem, providing flexibility and cost-effectiveness while maintaining data integrity and fault tolerance.

- **Write Cache (ZIL - ZFS Intent Log)**: A special area in ZFS used to store synchronous writes temporarily before they are committed to the main storage pool, promoting improved performance while ensuring data integrity.

- **Zvols**: Zvols, or ZFS volumes, are block devices created within the ZFS filesystem, allowing them to emulate traditional block storage devices. They provide flexible storage solutions by supporting features like snapshots and replication. Zvols are ideal for use cases requiring block-level access, such as databases or virtualization, and integrate seamlessly with ZFS's data compression and deduplication features.

45Drives
ENTERPRISE

Contact Us Today

www.45Drives.com/contact

info@45Drives.com

1-866-594-7199

Written by:
Zachary Perry, 45Drives Senior Data Storage Specialist
Ashley MacDonald, 45Drives Technology Education Manager