ORACLE

Oracle ZFS Storage Appliance as a Data Lake

May, 2024, Version [1.0]

Public

ORACLE

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

List of figures

List of tables

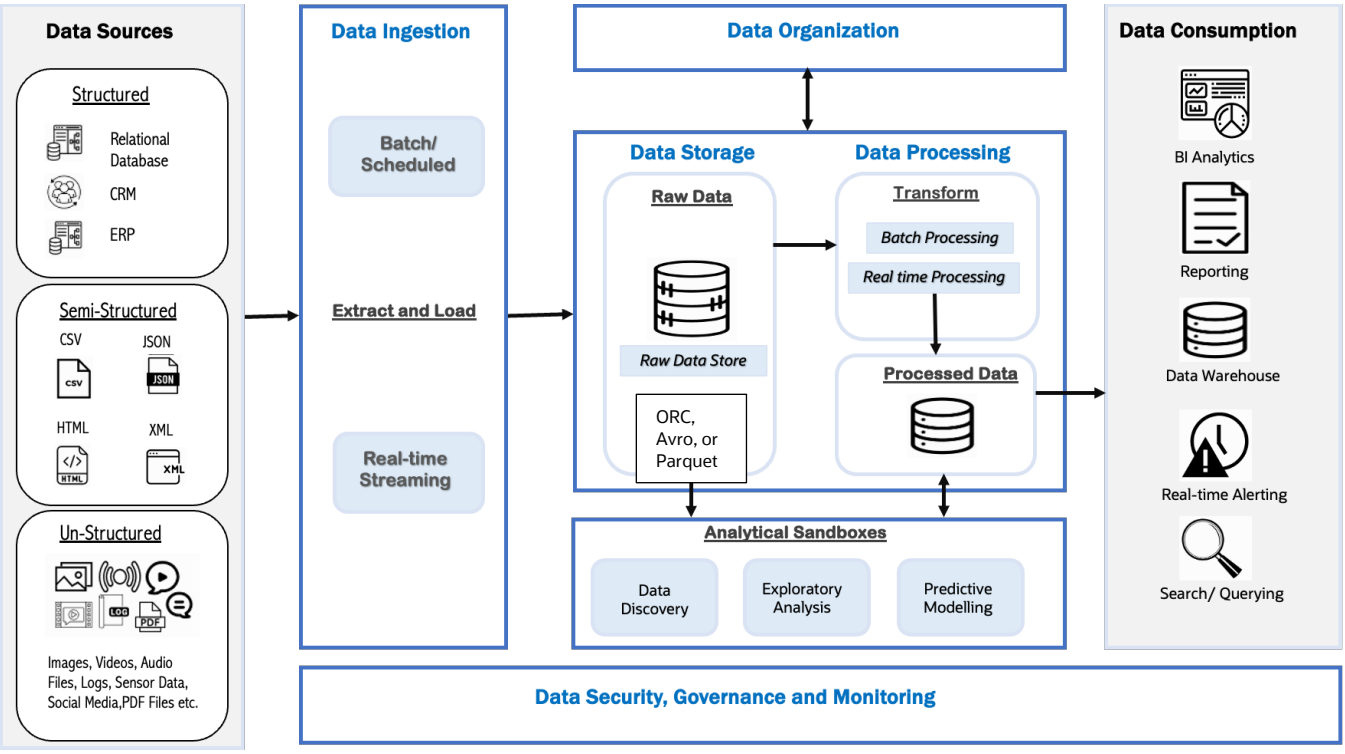
Introduction

For over a decade, leading organizations across diverse sectors like banking & finance and media & entertainment have relied on the cost-effective, petabyte-scale storage, and robust security and compliance features of on-premises Oracle ZFS Storage Appliances (ZFSSA). This solution brief is specifically intended for existing ZFSSA customers. It goes beyond the traditional role of storage, demonstrating how ZFSSA can serve as a foundation for building a data lake. By leveraging their existing investment in ZFSSA, customers can gain a secure, scalable, and cost-effective data lake solution. This approach unlocks additional value from their storage resources, transforming spare capacity into a powerful tool for data analytics.

Data Lake Architecture

Data lakes offer a central repository capable of storing and managing a vast and diverse range of data. This includes structured, semi-structured, and unstructured data, along with various file formats like JSON, ORC, AVRO, Iceberg, CSV, Parquet, and XML. Data lakes excel at ingesting and processing both batch and streaming data, making them powerful tools for fulfilling diverse data analysis goals.

Figure 1. Data Lake Architecture



The key components of a data lake architecture include:

- **Data Ingestion Layer:** This layer is responsible for collecting and ingesting data from various sources into the data lake. It includes tools and processes for data ingestion, validation, and transformation, ensuring that data is ingested efficiently and accurately.
- **Data Storage Layer:** The storage layer of a data lake provides scalable and cost-effective storage for storing raw data. It typically utilizes distributed file systems or object storage solutions to accommodate large volumes of data. Data is stored in big data file structures such as ORC, Avro or Parquet which provides both compression, and better access. See this article for insights.

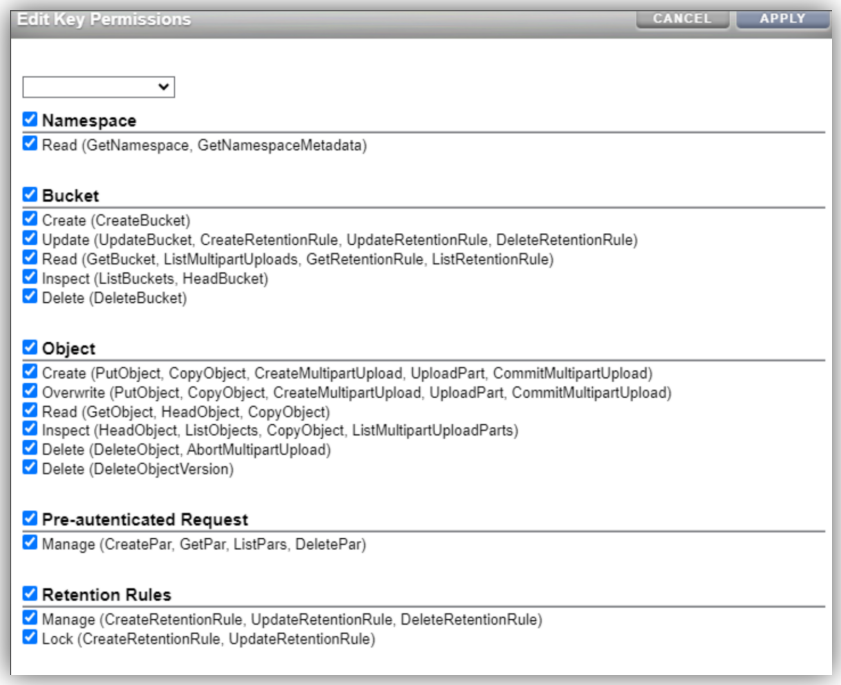
- **Data Processing Layer:** This layer enables data transformation, enrichment, and analysis within the data lake. It includes tools and frameworks for batch processing, stream processing, and interactive querying, allowing users to derive insights from the stored data.
- **Data Organization Layer:** The organization layer focuses on organizing and cataloging data within the data lake to make it discoverable, accessible, and understandable. It includes metadata management, data cataloging, and data governance capabilities to ensure data quality, lineage, and compliance.
- **Data Security and Governance Layer:** This layer ensures the security, privacy, and compliance of data stored and processed in the data lake. It includes access control mechanisms, encryption, data masking, auditing, and compliance monitoring tools to protect sensitive data and ensure regulatory compliance.

Leveraging Oracle ZFS Storage as a Data Lake

Oracle ZFS Storage offers flexible ingestion capabilities, allowing organizations to ingest structured, unstructured, and semi-structured data- from various sources, including databases, IoT devices, log files, and more. It offers protocol support for all the key industry standard storage protocols as well as Oracle's proprietary protocols that are optimized for Oracle-on-Oracle environments.

Table 1. ZFS Storage Capabilities as a Data Lake

Data Lake Features	ZFS Storage Capabilities as a Data Lake
Unified Storage	• Support for structured, semi-structured and un- structured data • Supports major industry standard protocols for data ingestion for e.g. HTTP/HTTPS, NFS, SMB, iSCSI, REST, FTP, JDBC etc.
Scalability	• Up to 25PB on-premises storage with support for block, file and object storage in the same platform.
Performance	• ZFSSA can be built with HDDs, SSDs, or both. With HDDs, its optional to incorporate read and/or write acceleration SSDs for improved read/writes • Offers multiple configurations to optimize various workload types for performance and/or throughput
Data Tiering	• Hybrid Storage Pool architecture offers data tiering for low latency I/O • ZFSSA combines different storage media - DRAM, flash (SSDs), and spinning disks - to deliver exceptional performance for both reading and writing data. ○ For reads, frequently accessed data is cached in high-speed DRAM for instant retrieval, with less-used data residing in a secondary flash cache (L2ARC). The remaining data sits on traditional spinning disks. This tiered structure ensures that critical data is always readily available for rapid reads. ○ Writes are handled intelligently as well. They are initially placed in DRAM for speed, then flushed to either flash or spinning disks based on the workload. Latency-sensitive applications benefit from writes being acknowledged after copying to a high-performance SSD, minimizing wait times. Conversely, throughput-intensive workloads can write data directly to spinning disks for faster transfers. This approach balances data integrity with optimal performance, catering to the specific needs of different applications.
Storage and Cost Efficiency	• Space efficient copy on write based snapshots, clones & replication • Storage efficiency with compression, deduplication further enables more storage for the same price points. • Cost effective data tiering with OCI object store enables low-cost, durable storage for archiving • ZFSSA offers two powerful controllers to manage 25PB storage which is unlike other solutions where compute must scale with storage, that increases the total cost of ownership (TCO).

Data Protection	• Block-level replication • Shadow migration • Supports tar & ZFS formats for backup/recovery
Security and Compliance	• Role-Based Access Control (RBAC) allows for granular control over user permissions within the storage system • Implements RAID protection, data encryption, immutable snapshots, retention locks, audit logs, ransomware recovery, and crypto replication for comprehensive data security and recoverability. • Compliance Ready: Cohasset's Compliance Assessment report on ZFS Storage • Oracle optimized object storage provides extensive permissions capabilities to separate bucket and object management, along reading and updating objects.  • Oracle optimized object storage also provides retention management by roles.
Integration	• Supports backups to OCI object store (standard and archival storage) • S3 API compatibility for seamless data transfer from Amazon S3
Investment Protection	• Existing customers who want to store and/or process data can leverage ZFSSA in the data lake capacity, thereby capitalizing on existing investment.

Oracle ZFSSA can store data in block, file and object formats and offers a cost-effective storage for long term retention or archival storage. The ZS9-2 ZFSSA appliance can scale up to 25PB which can accommodate a large number of starter data lakes. If organizations need to scale beyond that, they may use open-source tools like Ceph or GlusterFS to aggregate storage across multiple ZFS appliances, creating a distributed file system for their data lake. These open-source tools provide a unified namespace and management plane, allowing you to treat multiple ZFS appliances as a single entity for easier administration and scalability. This distributed architecture offers greater scalability, improved performance, and enhanced fault tolerance for handling massive datasets.

As a unified storage platform, ZFSSA can manage its own internal metadata, including file attributes, block size information, and object size data. ZFS ensures reliable access to data in data lakes through a layered metadata management system. Embedded metadata on each data block allows for quick retrieval without relying on a central

source, while the stash layer synchronizes comprehensive metadata copies across nodes for data consistency – critical aspects for efficient data discovery, maintaining data integrity, and enabling data governance within large and ever-growing data repositories.

This allows for basic data access and retrieval. However, ZFSSA benefits significantly from leveraging external meta store solutions. These external meta stores, like Apache Hive Meta store or Apache Atlas, provide a central repository for comprehensive meta data details including schema information, lineage tracking, and access control. This enables efficient data discovery, enforces data governance, streamlines data processing, and fosters collaboration – all essential for maximizing the value of your data lake.

Use Case: Batch Ingestion for Financial Data to Oracle ZFS Storage Appliance

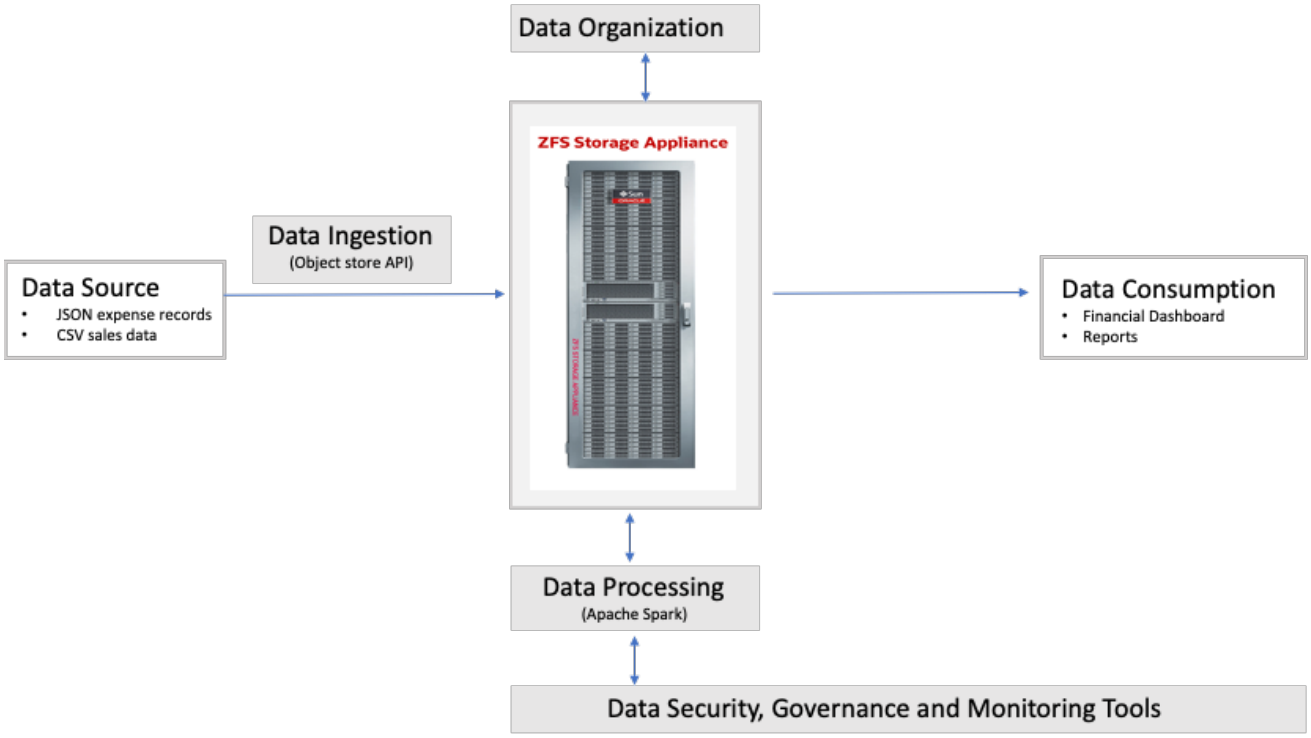
This section outlines the batch Ingestion use case with Oracle ZFSSA. Batch ingestion is used for loading large datasets at scheduled intervals. Batch processing then analyzes these datasets at once.

In financial institutions, data integrity, security, and regulatory compliance are critical. Archiving financial data is essential for regulatory compliance, historical analysis, and disaster recovery purposes. In this use case, we'll demonstrate how to set up a batch ingestion pipeline to archive financial data in CSV format to an Oracle ZFS Storage Appliance (ZFSSA) object store using OCI Object Store API. The financial data includes daily transaction records, account balances, market data, portfolio holdings, and trade history. By archiving this data to the ZFSSA, organizations can ensure data integrity, compliance with regulatory requirements, and have access to historical financial data for analysis and reporting purposes. The setup provides a cost-effective and scalable solution for data archival, leveraging the robust storage capabilities of the ZFSSA.

Batch processing data pipeline automates and optimizes the process of archiving data for compliance and cost-effective storage for later use. It involves:

1. **Data Ingestion:** Financial data from various sources (transactions, ERP, CRM) is gathered using ELT/ETL tools and stored in a central data lake.
2. **Data Archiving:** This data can be archived for the duration set by the administrator and used later for processing if needed. * Archival data can also be protected with a retention lock.

Figure 2. Batch Ingestion



Implementation Steps for Financial Reporting with Batch Ingestion and Processing

The following experiment lays out the steps used to set up data pipeline for batch ingestion and processing:

Steps:

1. Create a data source
2. Set Up ZFS Storage Configuration
3. Ingest Data via OCI Object Store API into ZFSSA:
4. Archiving – setting up retention period

1. Create a data source

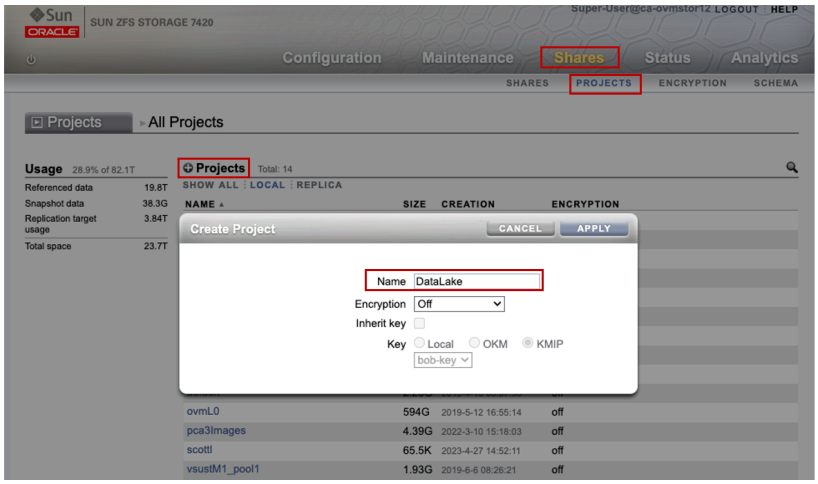
a. Choose a virtual machine (VM) as the environment for hosting your data source

b. Prepare your financial data in either CSV or JSON format and store it within the VM.

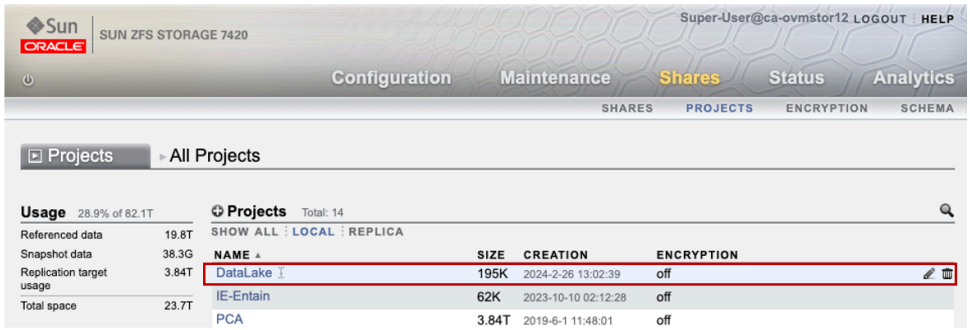
2. Set Up ZFS Storage Configuration

a. Create and configure Share

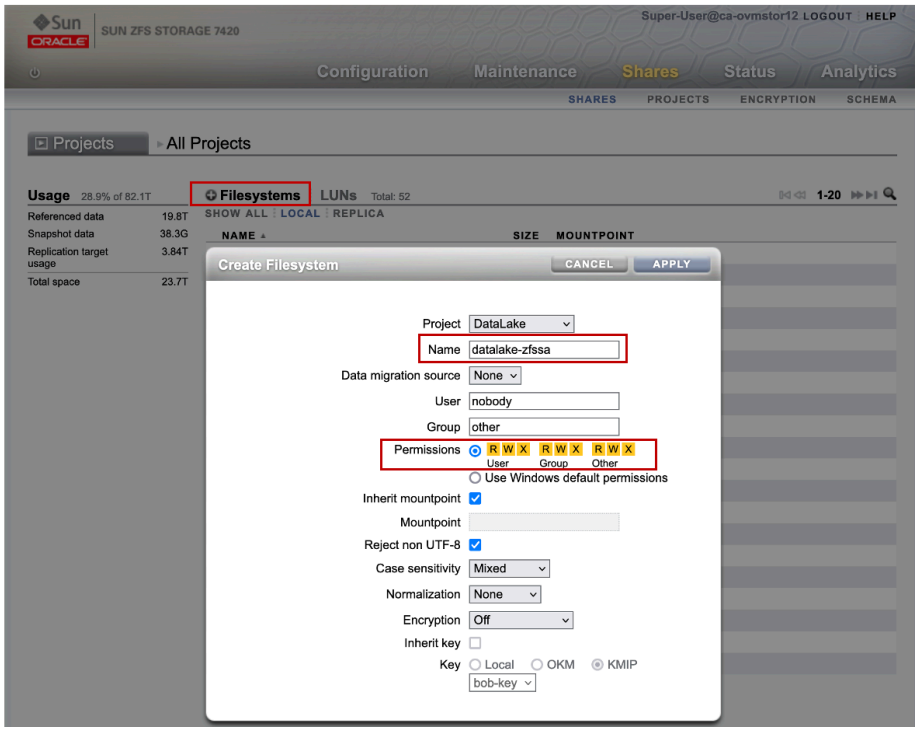
i. Log into Oracle ZFS Storage Appliance Browser User Interface (BUI) and navigate to Shares->PROJECTS-> Click on +icon next to Projects and a dialog box opens->Enter the project Name->click Apply



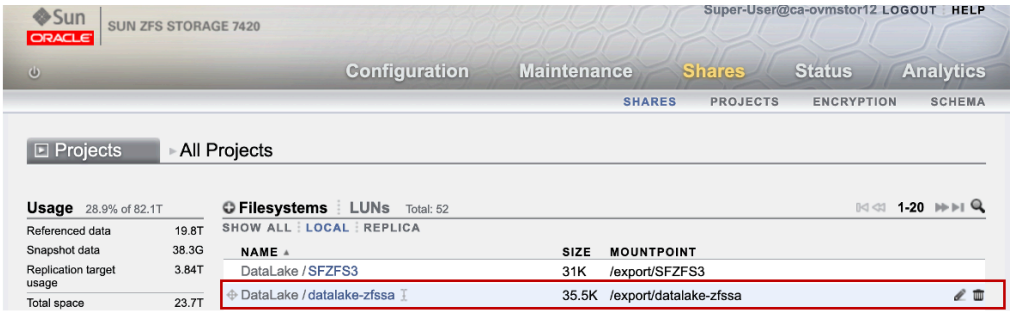
ii. Click the pencil icon for the DataLake project just created to create the filesystem share.



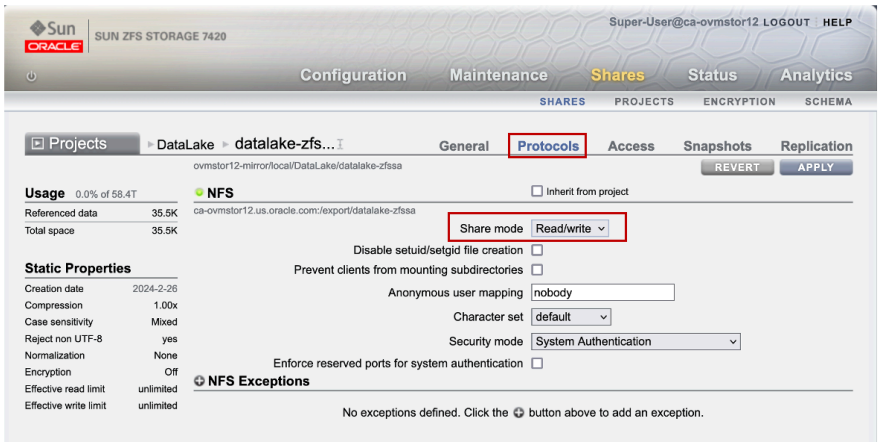
iii. Click the +icon **-> + icon** next to Filesystem and a dialog opens->Enter a filesystem Name and enable permissions for User, Group and Other-> Click Apply.



iv. This will create the file share and its mount point can be seen: /export/datalake-zfssa. Click on the pencil icon for the file system you just created to configure the filesystem.

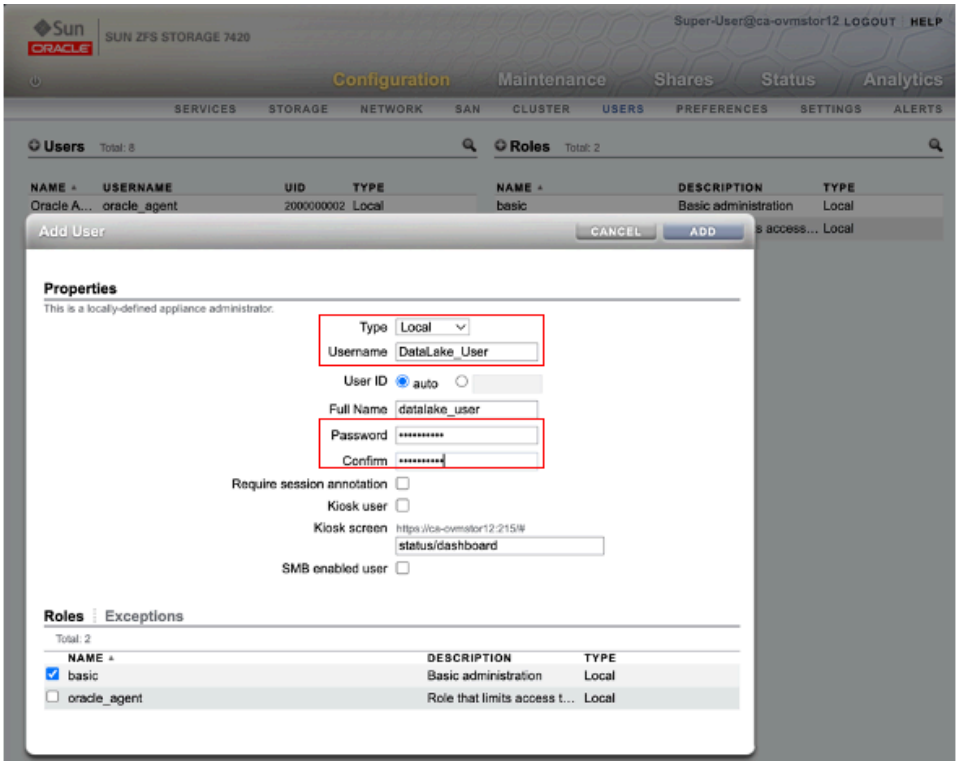


v. Navigate to Protocols and scroll down to NFS-> select the Share mode from the drop-down as 'Read/write-> click Apply.



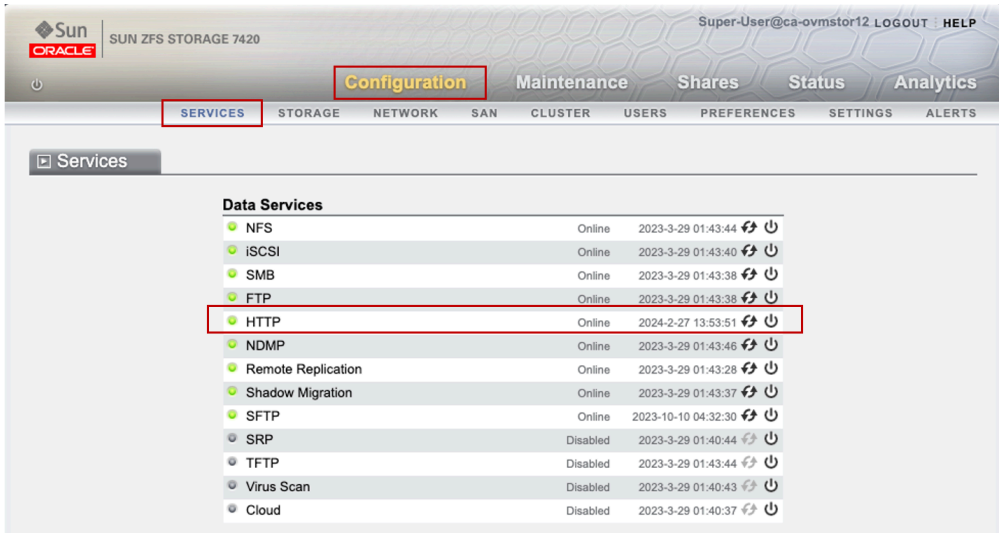
b. Create a user

i. Within the BUI, navigate to Configuration ->USERS-> click the +icon **-> + icon** next to Users and a dialog opens-> Select the Type as 'Local' from the drop-down menu ->Enter a username and provide a password-> Click Add.

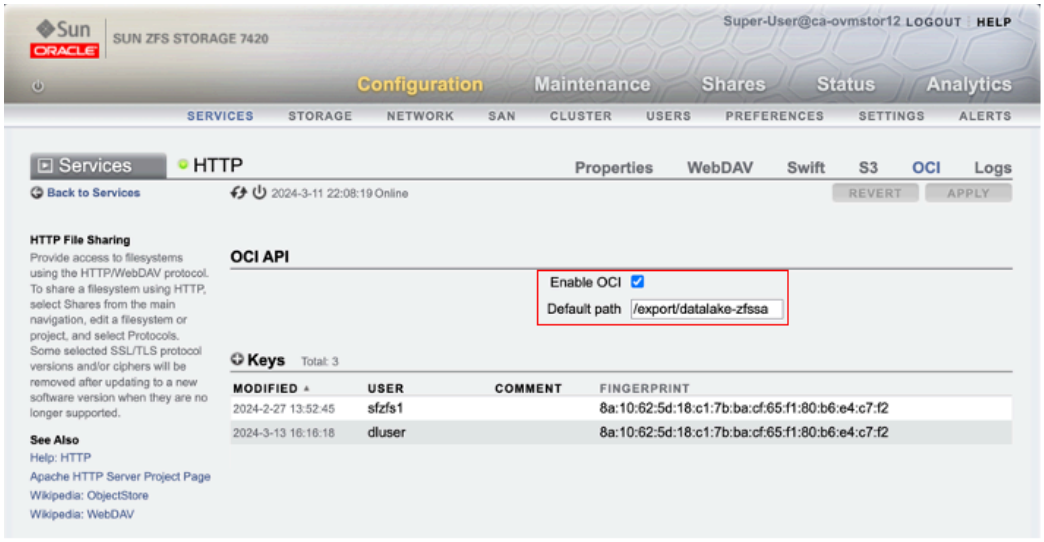


c. Configure protocol service

i. Navigate to Configuration -> SERVICES -> HTTP->Click the power-on icon to turn it on & the circle to the left of HTTP will turn green.

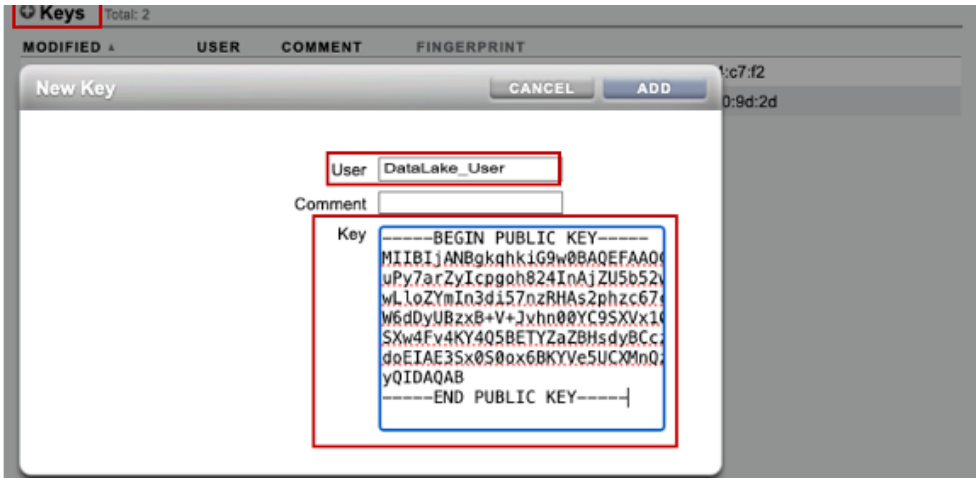


ii. Double click on HTTP to enter the set-up screen -> Click OCI->under OCI API, check 'Enable OCI' checkbox and in Default path enter the mountpoint (/export/datalake-zfssa) for the share which you will create for data lake.



iii. Click on the + icon**-> + icon** next to Keys, to create the user key. Enter a name for User -> then enter public key credentials generated as part of the private/public key pair -> click Apply.

1. Use the OCI documentation to create public/private keys.



3. Ingest Data via OCI Object Store API into ZFSSA

a. Create a config file for creating a bucket in Oracle ZFSSA

i. Note: To access ZFSSA via OCI CLI **-> CLI**, download the OCI cli tool from here.

1. [Default]
2. user=ocid1.user.oc1..dluser
3. fingerprint=7e:b4:31:e6:e3:e4:9d:b3:88:a3:74:d7:13:f0:9d:2d
4. key_file=~/.oci/oci_api_key.pem
5. tenancy=ocid1.tenancy.oc1..nobody
6. region=us-ashburn-1

1. Default OCI Location: This entry is identified as the default location to use for OCI operations if you don't specify one, as multiple locations can be configured in the file.
2. ZFS User ID: This identifies the user for ZFS access, formatted as "ocid1.user.oc1..{zfs user}".
3. API Public Key Fingerprint: Remember from the last blog post, this fingerprint is crucial. It identifies the specific API public key entry on ZFS that matches the private key you'll be sending.
4. Private Key File: This file holds the private API key corresponding to the public key added to ZFS.
5. OCI Client Configuration (Not Used by ZFS): Though not used by ZFS, this setting is mandatory for the OCI client. Use the previously identified default entry.
6. OCI Client Configuration (Not Used by ZFS): Similar to the previous point, this setting is required for the OCI client but not used by ZFS.
7. Optional Passphrase (if applicable): If the private key was created with a passphrase, enter it here. This passphrase unlocks the private key.

b. Creating a bucket in ZFSSA

i. See the OCI documentation on bucket creation. I created the following bucket.

```
oci os bucket create -c datalake-zfssa -ns datalake-zfssa --endpoint
http://ca-ovmstor12.us.oracle.com/oci --profile default --name
datalake_files
```

where:

--endpoint: it is the <url>/oci. For my ZFSSA it is

```
--endpoint http://ca-ovmstor12.us.oracle.com/oci
```

-ns: this is the namespace and is the share on the ZFSSA. In my config, it is:

```
-ns datalake-zfssa
```

-c: this is the compartment-id and is also the share on the ZFSSA. In my config it is:

```
-c datalake-zfssa
```

--name: the name of the bucket I want to create, in my case it is:

```
--name datalake_files
```

```
{
  "data": {
    "approximate-count": null,
    "approximate-size": null,
    "auto-tiering": null,
    "compartment-id": "datalake-zfssa",
    "created-by": "DataLake_User",
    "defined-tags": null,
    "etag": "bd49bfa296eb505db25bbcbb2bab23af",
    "freeform-tags": null,
    "id": "bd49bfa296eb505db25bbcbb2bab23af",
    "is-read-only": null,
    "kms-key-id": null,
    "metadata": null,
    "name": "datalake_files",
    "namespace": "datalake-zfssa",
    "object-events-enabled": null,
    "object-lifecycle-policy-etag": null,
    "public-access-type": "NoPublicAccess",
    "replication-enabled": null,
    "storage-tier": "Standard",
    "time-created": "2024-03-14T00:18:51+00:00",
    "versioning": "Disabled"
  },
  "etag": "bd49bfa296eb505db25bbcbb2bab23af"
}
```


c. Ingesting data into the bucket

i. See the OCI documentation on uploading an object storage object into a bucket. I uploaded the *sample.csv* file I had created for this experiment into *datalake_files* bucket in ZFSSA.

```
oci os object put -ns datalake-zfssa -bn datalake_files --file
/home/opc/sample.csv --name sample.csv --metadata '{"description":"Sample
CSV data"}' --profile default --endpoint http://ca-
ovmstor12.us.oracle.com/oci
```

4. Archiving – Setting up retention period

The following document provides information to configure and use the file retention feature within ZFSSA OS version: 8.8.45 and above:
https://support.oracle.com/knowledge/Sun%20Microsystems/2867335_1.html#aref_section25

Important Requirements and Settings:

The following are requirements to use the file retention feature:

- The ZFS Appliance MUST be on OS version 8.8.45 (i.e. 2013.06.05.8.45) or higher.
- Deferred updates MUST first be applied on the storage pool.
- Filesystems that are intended to contain file retention data can ONLY be created in a storage pool that has redundancy (i.e. Mirror or RAID-Z).
- If using OS version(s) 8.8.45 - 8.8.50 and log devices are present they MUST be mirrored, otherwise file retention policy cannot be set.

Non-mirrored log devices are permitted in OS versions 8.8.51 and higher.

- If using OS version(s) 8.8.45 - 8.8.56 and metadevices are present they MUST be mirrored, otherwise file retention policy cannot be set.

Non-mirrored metadevices are permitted in OS versions 8.8.57 and higher.

- The file retention feature is enabled at the initial point of creation on a new filesystem. File retention CANNOT be enabled on an existing filesystem.
- OS 8.8.45 requires the NTP service to be functional. The ZFS Appliance MUST be able to reach an NTP server, and the sync_always setting MUST be enabled.
- Root login MUST be disabled for the BUI (https) and CLI (ssh).
- Each administrator should have their own account for auditing purposes.

There could be another use case for batch ingestion and processing where the batch data after ingestion needs to be processed for generating reports. In that case at regular intervals, a batch processing framework (e.g., Apache Spark) cleanses, transforms, and calculates metrics on the data in batches. Financial reports (income statements, balance sheets, etc.) are then generated using reporting or data visualization tools, providing valuable insights for decision-making.

Other Use Cases

This table provides examples of how ZFS can be used as a data lake foundation, tailored to specific industries and requirements. ZFS offers flexibility to support both stream and batch processing pipelines and can integrate with additional open-source and proprietary tools needed to build pipelines for various analytical objectives. (Note: These tools are not part of the core Oracle solution but offer potential integration options for specific use cases.)

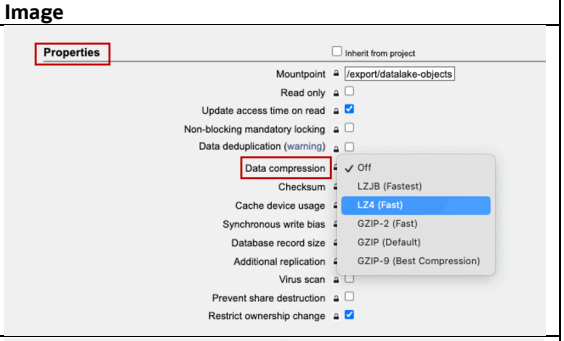
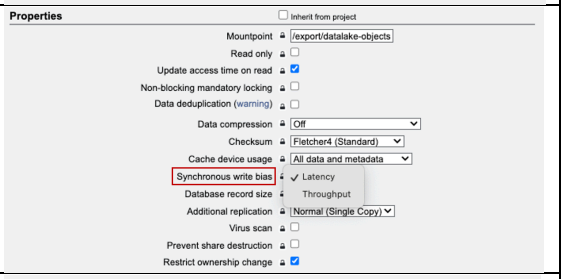
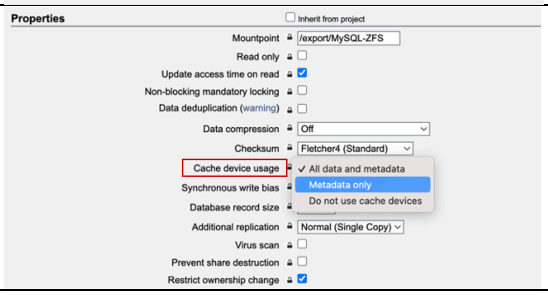
Table 2. Data Lake Pipeline Use Cases

Industry	Use Case Name	Data Source	Data Type	Processing Frequency	Additional Tools	Considerations
Banking & Finance	**Fraud Detection**	Transaction logs, customer data, network activity	Structured, semi-structured	Real-time (streaming)	Apache Spark, Kafka	Low latency, real-time alerts, high data volume
Banking & Finance	**Risk Analysis**	Market data, economic indicators, social media sentiment	Structured, unstructured	Batch	Spark, Hadoop, Hive	Historical analysis, complex calculations, regulatory compliance
Media & Entertainment	**Personalized Content Recommendations**	User behavior, content metadata, streaming data	Unstructured, semi-structured	Real-time (streaming) and batch	Spark, Kafka, Elasticsearch	Real-time recommendations, user segmentation, offline analysis
Media & Entertainment	**Content Analytics**	Viewing statistics, social media engagement, content creation tools	Unstructured, semi-structured	Batch	Spark, Hadoop, Hive	Identify trends, predict audience preferences, optimize content strategy
Healthcare	**Real-time Patient Monitoring**	Sensor data, medical records, vital signs	Structured, semi-structured	Real-time (streaming)	Apache Flink, Apache Pulsar	Low latency, anomaly detection, critical care monitoring
Healthcare	**Medical Research**	Clinical trials data, genomic data, medical images	Structured, unstructured	Batch	Spark, Hadoop, R	Statistical analysis, disease modeling, drug discovery
Retail	**Product Recommendations**	Sales data, customer behavior, product information	Structured, semi-structured	Real-time (streaming) and batch	Spark, Kafka, Neo4j	Personalized recommendations, inventory optimization, campaign targeting
Retail	**Demand Forecasting**	Sales history, weather data, social media trends	Structured, unstructured	Batch	Spark, Hadoop, Prophet	Predict future demand, optimize inventory levels, improve pricing strategies
IoT	**Industrial Asset Monitoring**	Sensor data, machine logs, performance metrics	Structured, semi-structured	Real-time (streaming) and batch	Apache Kafka, Prometheus, Grafana	Anomaly detection, predictive maintenance, operational efficiency
IoT	**Smart City Analytics**	Traffic data, energy consumption, public safety data	Structured, semi-structured	Batch	Spark, Hadoop, Tableau	Analyze city trends, optimize resource allocation, improve citizen services

Best Practices for Optimizing ZFSSA according to Data Lake Use Case Requirements

The following steps describe best practices for tuning ZFSSA to obtain optimal results for specific use cases.

Table 3. Best practices for tuning ZFSSA

BUI Label	CLI property name	Value	Image
Data Compression	**Compression** *Controls how data stored in filesystem is compressed*	• Set to **LZ4**	
Synchronous Write bias	**logbias** *controls the behavior of ZFS when synchronizing data writes to the underlying storage pool*	• Set to **Latency** for prioritizing low latency • Set to **Throughput** for prioritizing high throughput	
Cache device usage	**secondarycache** *defines how dedicated cache devices are used for storing data and metadata within a ZFS storage pool*	• Set to **All data and metadata** for smaller files accessed frequently and write heavy workloads • Set to **do not use cache devices** for most data lake workloads with large files, read heavy workloads & cost efficiency • Not recommended setting **Metadata only**	

Conclusion

Turning your existing Oracle ZFS Storage Appliances into data lake makes smart business sense. It's secure, scales easily, and maximizes your current storage investment. This lets you use spare capacity for valuable data analysis, giving you the insights, you need to make data-driven decisions.

ORACLE

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

May 2024
Authors: Sheetal Sabharwal