IMPLEMENTATION GUIDE

Service Provider
Data Center

Implementation Guide for MinIO* Storage-as-a-Service

Learn how to deploy a storage-as-a-service (STaaS) solution based on MinIO* with Intel® technology to create a scalable, S3 object store that features high performance, strict consistency and enterprise security

This implementation guide provides key learnings and configuration insights to integrate technologies with optimal business value.

If you are responsible for...

- **Technology decisions:** You will learn how to implement a storage-as-a-service (STaaS) solution using MinIO*. You'll also find tips for optimizing performance with Intel® technologies and best practices for deploying MinIO.

MINIO

Introduction

MinIO* is a self-contained, distributed object storage server that is optimized for Intel® technology. MinIO provides a compelling storage-as-a-service (StaaS) object storage platform when combined with Intel's broad selection of products and technologies, such as Intel® Non-Volatile Memory express* (NVMe*)-based Solid State Drives (SSDs), Intel® Ethernet products and Intel® Xeon® Scalable processors, augmented by Intel® Advanced Vector Extensions 512 (Intel® AVX-512) single instruction multiple data (SIMD) instructions for x86 architecture. Collaborative open source developer communities are also available for MinIO.

An object storage solution should handle a broad spectrum of use cases including big data, artificial intelligence (AI), machine learning and application data. Unlike other object storage solutions that are built for archival use cases only, the MinIO platform is designed to deliver the high-performance object storage that is required by modern big data applications.

MinIO includes enterprise features such as:

- **Hyperscale** architecture to enable multi-data center expansion through federation
- **High performance** to serve large volumes of data needed by cloud-native applications
- **Ease of use** with non-disruptive upgrades, no tuning knobs and simple support
- **High availability** to serve data and survive multiple disk and node failures
- **Enhanced security** by encrypting each object with a unique key

Use the information in this implementation guide to deploy MinIO object storage and unleash the power of STaaS.

Solution Overview

MinIO consists of a server, optional client and optional software development kits (SDKs):

- **MinIO Server.** A 100 percent open source Amazon S3*-compatible object storage server that provides both high performance and strict consistency. Enterprise-grade encryption is used to help secure objects, and high-performance erasure

Table of Contents

code algorithms are used to provide data durability. With MinIO data protection, a cluster can lose up to half of its servers, and half of its drives, and continue to serve data. User and application authentication are provided via tight integration with industry-standard identity providers.

- **MinIO Client.** Called MC, the MinIO Client is a modern and cloud-native alternative to the familiar UNIX* commands like `ls`, `cat`, `cp`, `mirror`, `diff`, `find` and `mv`. The MinIO Client commands work with both object servers and file systems. Among the most powerful features of the MinIO Client is a tool for mirroring objects between S3-compatible object servers.
- **MinIO SDKs.** The MinIO Client SDKs provide simple APIs for accessing any Amazon S3-compatible object storage. MinIO repositories on GitHub* offer SDKs for popular development languages such as Golang*, JavaScript*, .Net*, Python* and Java*.

The use cases for MinIO span a wide variety of workloads and applications (see Figure 1).

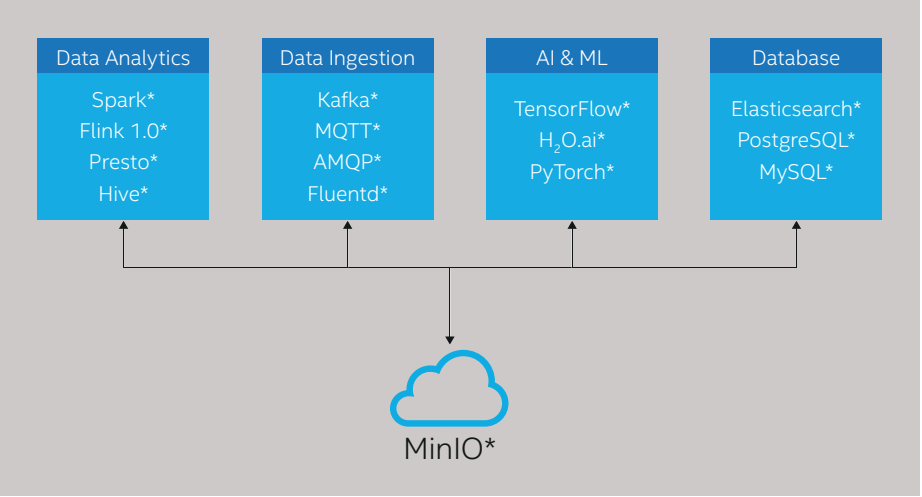


Figure 1. MinIO* object storage provides the high performance required by modern big data applications.

Intel® Technologies

Several Intel® technologies, both hardware and software, provide the performance and reliability foundation of a MinIO-based solution. See the "References" section for links.

- **Data durability and performance boost.** MinIO protects the integrity and durability of objects with erasure coding and uses hash checksums to protect against bitrot. These performance-critical algorithms have been accelerated using the SIMD instructions on Intel® architecture using Intel® Advanced Vector Extensions 2 (Intel® AVX2) and Intel AVX-512. Offloading these calculations has a positive impact on system performance.
- **Enhanced storage performance.** Intel® Serial AT Attachment (SATA)-based, Intel NVMe-based and Intel® Optane™ SSDs provide performance, stability, efficiency and low power consumption. Intel NVMe-based SSDs provide a fast storage layer which MinIO in turn translates into fast throughput for S3 object PUTs and GETs.
- **Linear scaling with Intel® processors.** The Intel Xeon processor Scalable family provides a wide range of performance options. High-end Intel® processors deliver energy efficiency and high performance for intensive STaaS workloads. An alternative is to choose a lower-performance option for less demanding STaaS workloads such as archive storage workloads.

- **Increased server bandwidth.** Intel® Ethernet Network Adapters provide flexible bandwidth options. They are available with 10, 25 and 40 Gigabit Ethernet (GbE) ports to support whatever network infrastructure may be deployed in your data center. The MinIO Server works fastest with high-speed, low-latency networking connections between servers in a cluster and between a MinIO cluster and its clients.

System Requirements

Software Requirements

- **MinIO software.** As of the publication date, the current version of MinIO is RELEASE.2019-04-04T18-31-46Z.
- **Compatible OS.** MinIO will work effectively with any Linux* distribution. Intel recommends that you deploy MinIO on one of the three major Linux distributions: CentOS*/Red Hat Enterprise Linux* (RHEL*), SUSE*, or Ubuntu*.

Hardware Requirements

MinIO was designed to run on industry-standard hardware. The notes in Table 1 provide guidance for selecting components.

Table 1. Server Configuration

Component	Range of Options	Notes
CPU	2x Intel® Xeon® Scalable processors	Erasure coding will take advantage of Intel® Advanced Vector Extensions 512 (Intel® AVX-512).
Memory	96 GB	MinIO does not require a large amount of server memory. 96 GB is a "balanced" memory configuration for both 1st and 2nd Generation Intel Xeon Scalable processors.
Data Storage	• SATA-based solid state drives (SSDs) • NVMe*-based SSDs • Intel® Optane™ Data Center SSDs	MinIO can use all the drives in a server. A common high-performance deployment choice is a 2U 24-drive chassis with SATA- or NVMe-based SSDs. For archive workloads, a 4U 45-drive chassis with high-density SATA-based SSDs works well. The use of faster drives allows MinIO to provide more throughput.
Network	10/25/40/50/100 GbE network interface cards (NICs)	The use of faster networks allows MinIO to provide higher levels of throughput. Bonding multiple networks together both creates a high-availability configuration and makes additional throughput possible.
Cluster Size	4 to 32 nodes	The minimum cluster size is 4 nodes. The maximum cluster size is 32 nodes. Multiple MinIO clusters can be federated to create larger clusters.

GbE = Gigabit Ethernet NVMe = Non-Volatile Memory express SATA = Serial AT Attachment

Installation and Configuration

There are six steps to deploying a MinIO cluster:

1. Download and install the Linux OS
2. Configure the network
3. Configure the hosts
4. Download the MinIO executables
5. Start the MinIO cluster
6. Test the MinIO cluster

Step 1: Download and Install the Linux OS of Choice

MinIO requires only the "basic server" option from the distribution. Often IT organizations have determined best practices for Linux OS provisioning and system hardening and these recommendations should be followed. Intel recommends that several utilities be added to the "basic server" bundle to manage Intel® devices. Additional services and libraries are also needed to support the health of the MinIO cluster. The following subsections provide information about the recommended additional services and libraries, utilities, Intel® tools and drivers and firmware.

Additional Services and Libraries

The following services and libraries should be present to support the MinIO cluster:

- Network time protocol (NTP) time server configured to synchronize time between all MinIO servers
- Domain Name Service (DNS)
- Secure Shell (SSH) libraries
- Secure Sockets Layer (SSL) libraries

Recommended Utilities

The following utilities are useful for managing hardware:

- numactl
- pci-utils (lspci)
- nvme-cli
- sdparm
- hdparm
- sysstat (perf, dstat)
- git
- python
- screen
- tree
- ipmitool
- wget
- curl
- vim-enhanced (or emacs)
- collectd
- parted

The Yellowdog Updater, Modified (YUM) tool can be used to install the above 16 utilities, as shown here:

```
yum install numactl pci-utils nvme-cli sdparm hdparm systat git python screen
tree ipmitool wget vim-enhanced curl collectd parted -y
```

Intel® Tools

The Intel® SSD Data Center Tool (Intel® SSD DCT) provides manageability and configuration functionality for Intel® SSDs with Peripheral Component Interconnect Express* (PCIe*) and SATA interfaces. This tool is used to upgrade firmware on SSD controllers and to apply advanced settings. This tool is separate from the nvme-cli utility, which is used only for managing NVMe-based devices and does not support updating drive firmware.

The Intel SSD DCT can be downloaded from the following link: https://downloadcenter.intel.com/download/28594/Intel-SSD-Data-Center-Tool-Intel-SSD-DCT-

Drivers and Firmware

Ensure that the following drivers and firmware are at their most current levels:

- **BIOS.** Follow the instructions provided by the server vendor.
- **Intel SSD.** Follow the instructions in the Intel SSD DCT Guide
- **Intel® NIC.** Download the latest driver and firmware packages. Note: 10/25/40 GbE NICs use the same i40e driver package.

Intel NIC drivers are available at the following link: https://downloadcenter.intel.com/download/24411/Intel-Network-Adapter-Driver-for-PCIe-40-Gigabit-Ethernet-Network-Connections-Under-Linux-?product=95260

Intel firmware is available at the following link: https://downloadcenter.intel.com/download/25791/Non-Volatile-Memory-NVM-Update-Utility-for-Intel-Ethernet-Adapters-710-Series-Linux-?product=95260

Step 2 - Configure the Network

Open Port 9000

By default, MinIO uses port 9000 to listen for incoming connections. If there is a need to employ a different port number, then this can be selected when the MinIO server is started. Ensure that the port is open on the firewall on each host.

Find the active zones:

```
firewall-cmd --zone=<my _ zone> --add-port=9000/tcp --permanent
```

Configure firewall:

```
firewall-cmd --zone=<my _ zone> --add-port=9000/tcp --permanent
```

Reload firewall:

```
firewall-cmd --reload
```

Choosing Hostnames

It is best to name the hosts with a logical sequence of hostnames. This is done to make starting and managing the MinIO cluster simple. For example, Table 2 shows a naming convention for N number of nodes.

Table 2. Hostname Examples

Node 1	minio1.example.com
Node 2	minio2.example.com
...	...
Node N	minioN.example.com

Enabling Jumbo Frames

The use of jumbo frames can improve network bandwidth. However, the maximum transmission unit (MTU) must be set globally for every host, client and switch. Intel recommends setting MTU = 9000 for client nodes, storage nodes and switches.

Enabling Bonding

Network bonding offers redundancy for the networks that connect MinIO hosts and MinIO hosts and clients. Bonding can also increase the network bandwidth between clients and hosts. When using a bonded interface, the transmit hash policy should use upper-layer protocol information, which allows the traffic to span multiple slaves.

The transmit hash policy is set with the following command.

```
$ echo "layer3+4" > /sys/class/net/bond0/bonding/xmit _ hash _ policy
```

Adding MinIO Host and Cluster Names to the DNS

Add all of the MinIO hosts, hostnames and IP addresses to the DNS.

Create a DNS entry with a name and IP address for the cluster. Use a round-robin algorithm to point incoming requests to individual hosts in the MinIO cluster.

Advanced Network Tuning Parameters

Configuring interrupt request (IRQ) affinity to assign interrupts and applications to the same core can have a positive impact on network performance. To configure IRQ affinity, stop irqbalance and then either use the set_irq_affinity script from the i40e source package (recommended) or pin queues manually. With each interrupt mapped to its own CPU, performance can increase when the interrupt handling is done on the cores closest to the device.

The following example sets the IRQ affinity to all cores for the Ethernet adapters.

First, disable user-space IRQ balancer to enable queue pinning:

```
systemctl disable irqbalance
systemctl stop irqbalance
```

Then set IRQ affinity to all cores for ethX devices:

```
[path-to-i40epackage]/scripts/set _ irq _ affinity -x all ethX
```

Detailed instructions for setting IRQ affinity are provided in the *Intel® X710/XL710 Linux Performance Tuning Guide*, available at the following link: https://www.intel.com/content/dam/www/public/us/en/documents/reference-guides/xl710-x710-performance-tuning-linux-guide.pdf

Testing the Network

The last step to configuring the network is to verify connectivity and name resolution between MinIO nodes. Each node should be able to connect with all of the other nodes. Run a command such as the following on all MinIO hosts:

```
for i in $(seq 1 16); do ping -t2 host$(i); done
```

Step 3 - Configure the Hosts

Kernel Tuning Parameters

Most Linux systems should provide adequate performance out of the box. If you want to tune the OS for maximum performance, then Intel recommends applying specific kernel settings to the sysctl configuration file. The settings in the /etc/sysctl.conf file will be used to override the default kernel parameter values and survive system reboots.

"Appendix A: Linux Kernel Tuning Parameters" contains a list of recommended tuning parameters and values. Upload the text in Appendix A to each host, append the text to the sysctl.conf file, and then refresh the new configuration.

```
cat tuning.txt >> /etc/sysctl.conf
sysctl -p
```

Preparing the SSD Media

If the drives already contain data, or their provenance is unknown, Intel recommends conditioning the drives by running the following command twice (or more) on each drive:

```
nohup dd if=/dev/zero of=/dev/<drive> oflag=direct bs=2M &
```

This can also be scripted:

```
for i in {0..24}; do  nohup dd if=/dev/zero of=/dev/nvme$i{i}
oflag=direct bs=2M &; done
```

Partitioning and Formatting Devices

All of the devices that will be used with MinIO must be partitioned and formatted with a file system. The XFS file system is recommended, but the ext4 file system is also valid. To make starting and managing the MinIO cluster simple, name the drive partitions and mount points logically, as shown in the following example. Repeat this task on each host.

Partition:

```
parted /dev/nvme0n1 name 1 ”minio-data1”
parted /dev/nvme1n1 name 1 ”minio-data2”
parted /dev/nvme2n1 name 1 ”minio-data3”
```

Create a file system:

```
mkfs -t xfs /dev/nvme0n1
mkfs -t xfs /dev/nvme1n1
mkfs -t xfs /dev/nvme2n1
```

Mount the devices:

```
mount /dev/nvme0n1 /mnt/minio-data1
mount /dev/nvme1n1 /mnt/minio-data2
mount /dev/nvme2n1 /mnt/minio-data3
```

Step 4 - Download the MinIO Executables

Download the MinIO server binary to each node in the cluster and add executable permissions with the following commands. Note: This command will retrieve the latest stable build.

```
wget https://dl.minio.io/server/miniorelease/linux-amd64/minio
chmod +x minio
```

Download the MinIO Client (MC) to a client or laptop with the following command. The MinIO Client allows you to manage and test the cluster. Note: This command will retrieve the latest stable build.

```
wget https://dl.minio.io/server/minio/release/linux-amd64/minio
chmod +x minio
```

Step 5 - Start the MinIO Cluster

Starting a MinIO cluster is done by simply starting the MinIO executable on each node in the cluster. If you have followed the advice on suggested naming conventions for host names and mount points, then the command line that is used to start the cluster will be quite elegant.

Creating a Script to Start the MinIO Cluster

When starting a MinIO cluster, it is recommended to create a shell script containing the needed commands. After creating this script, copy it to all nodes in the cluster. The following example script starts a distributed MinIO server on a 32-node cluster where each node has 24 drives, the cluster uses an access key of “minio” and a secret key of “minio123”.

```
export MINIO_ACCESS_KEY=minio
export MINIO_SECRET_KEY=minio123
./minio server http://node{1...32}.example.com/mnt/export{1...24}
```

Starting the MinIO Cluster

Run the MinIO cluster shell script on each host. The MinIO executable will seek to connect with the MinIO servers running the other nodes specified in the command line. Once all of the nodes connect with each other, the cluster will report that it has started.

Run the shell script as a background task. As an alternative, add the MinIO cluster startup commands to the init/service scripts on each host as described next.

Using Init or Service Scripts to Start MinIO

MinIO cluster commands can be added to the init/service scripts of each host. A collection of example scripts for systemd, sysvinit and upstart is located at the following GitHub download page: https://github.com/minio/minio-service

Step 6 - Test the MinIO Cluster

MinIO Server comes with an embedded web-based object browser. To access the MinIO object server, point a web browser to the DNS name of the cluster, or to an individual node in the cluster. From the web browser you will be able to view buckets and objects stored on the server.

```
http://minioclustername.example.com:9000
```

Accessing and Managing the MinIO Cluster

There are multiple methods to access the MinIO cluster.

MinIO Client (MC)

The MC provides a modern alternative to UNIX commands like `ls`, `cat`, `cp`, `mirror`, `diff` and `find`. It supports both filesystems and Amazon S3-compatible cloud storage service (Amazon Web Services Signature* v2 and v4).

The MC is also used to manage the MinIO Server. The range of commands that it supports includes starting and stopping the server, managing users and monitoring CPU and memory statistics.

For instructions on using the MC to manage the MinIO Server, refer to this documentation: https://docs.minio.io/docs/minio-admin-complete-guide.html

AWS Command-Line Interface (CLI)

The AWS CLI is a unified tool to manage AWS services. It is frequently the tool used to transfer data in and out of AWS S3. It works with any S3-compatible cloud storage service.

For instructions on using the AWS CLI with the MinIO Server, refer to this documentation: https://docs.minio.io/docs/aws-cli-with-minio

S3cmd CLI

S3cmd is a CLI client for managing data in AWS S3, Google Cloud Storage or any cloud storage service provider that uses the S3 protocol. S3cmd is open source and is distributed under the GPLv2 license.

For instructions on using S3cmd with the MinIO Server, refer to this documentation: https://docs.minio.io/docs/s3cmd-with-minio

MinIO Go Client SDK for Amazon S3-Compatible Cloud Storage

The MinIO Go Client SDK provides simple APIs to access any Amazon S3-compatible object storage. A quick-start guide shows you how to install the MinIO Go Client SDK, connect to MinIO and provide a walkthrough for a simple file uploader.

The MinIO Go Client SDK quick-start guide is available at the following link: https://docs.minio.io/docs/golang-client-quickstart-guide

Operating MinIO

Parity and Erasure Coding

MinIO helps protect data against hardware failures and silent data corruption using erasure code and bitrot checksums. Erasure code is a mathematical algorithm used to reconstruct missing or corrupted data. MinIO uses Reed-Solomon code to shard objects into data and parity blocks. MinIO's erasure code algorithms are performed inline and are accelerated by using the Intel AVX-512 instruction set. This increases the performance of the MinIO object storage solution. Erasure code is a powerful technique of providing high durability with low storage overhead.

Dealing with Hardware Failures

MinIO is purposely designed to survive multiple hardware failures. During disk and server failures, the MinIO cluster continues to serve objects normally. Unlike other solutions, there is no urgency to replace failed drives, even next business day is considered overkill for maintaining a MinIO cluster. This relaxed approach to system maintenance means you can reduce the human labor, and potential for human errors, needed to operate a MinIO cluster.

Healing Objects

The heal command can be used to check the health of individual objects and the health of all objects on the system. When a new disk is used to replace a failed disk, the heal command can be used to recreate any erasure coded object data that was previously stored on the failed disk.

The following MC command is used after replacing a disk. It recursively heals all buckets and objects on the MinIO cluster 'myminio'.

```
mc admin heal --recursive myminio
```

Updating MinIO

Upgrading a MinIO cluster is a two-step process. First, run the `minio update` command on each node in the cluster to download a new MinIO server executable. Then restart the MinIO server cluster using the `mc admin service restart` command. This will restart the entire cluster, an action which is non-disruptive to the clients connected to the cluster.

```
$ minio update

Update to RELEASE.2019-04-04T18-31-46Z [yes]: yes

Minio updated to version RELEASE.2019-04-04T18-31-46Z
successfully.

$ mc admin service restart myminio

Restart command successfully sent to `myminio`.

Restarted `myminio` successfully.
```

Checking MinIO Cluster Status

To check the status of the MinIO cluster use the `mc admin info` command:

```
$ mc admin info play
   127.0.0.1:9000
   Status : online
   Uptime : 20 hours
  Version : 2019-04-04T20:48:34Z
   Region : us-east-1
  Storage : Used 16 GiB
```

Monitoring MinIO Using Prometheus*

MinIO includes built-in support for exporting Prometheus*-compatible data on an unauthenticated endpoint. This enables Prometheus monitoring for MinIO server deployments without sharing server credentials and eliminates the need to run an external Prometheus exporter.

Details on monitoring MinIO are provided here: https://docs.minio.io/docs/minio-monitoring-guide.html

Support for MinIO

MinIO Slack* Channel

There is a large community of people devoted to supporting MinIO and answering questions about MinIO on a public Slack* channel.

Sign up for the Slack channel here: https://slack.min.io

MinIO SUBNET

MinIO SUBNET is an annual subscription that combines software and services to facilitate the production success of MinIO customers. While MinIO is available under the open source Apache V2 license, customers may choose to purchase the software on a subscription basis. Their reasons for doing so differ, but they are unified in the value they see in the software coupled with a desire to have a deeper relationship with the team behind MinIO.

Sign up for MinIO SUBNET here: https://min.io/subscription

Scaling MinIO Clusters with Federation

The designers of MinIO believe that building durable large-scale object stores is best done by creating individual clusters which are then connected together to create a federation of clusters. This approach was learned after studying the scaling methods used by the hyperscalers that employ a method of scaling that joins simple building blocks together. A large MinIO cluster is created first by deploying a single MinIO cluster and then joining additional MinIO clusters together to create a federation of clusters and a single global namespace (see Figure 2).

There are multiple benefits to MinIO's cluster and federation architecture:

- Each node is an equal member of a MinIO cluster. There is no master node.
- Each node can serve requests for any object in the cluster, even concurrently.
- Each cluster uses a Distributed Locking Manager (DLM) to control updates and deletes to individual objects.
- The performance of individual clusters remains constant as more clusters are joined to the federation.
- Failure domains are kept within a single cluster. An issue with one cluster does not affect the entire federation.
- Individual clusters can range in size from 4 to 32 nodes and they can employ different server hardware, drive types and sizes.

When deploying a cluster, it is recommended that you use a programmable DNS, such as CoreDNS*, to route HTTP(S) requests to the appropriate cluster. Global configuration parameters can be stored and managed in etcd (an open-source distributed key-value store). Documentation of MinIO federation feature is located here: https://docs.min.io/docs/minio-federation-quickstart-guide.html

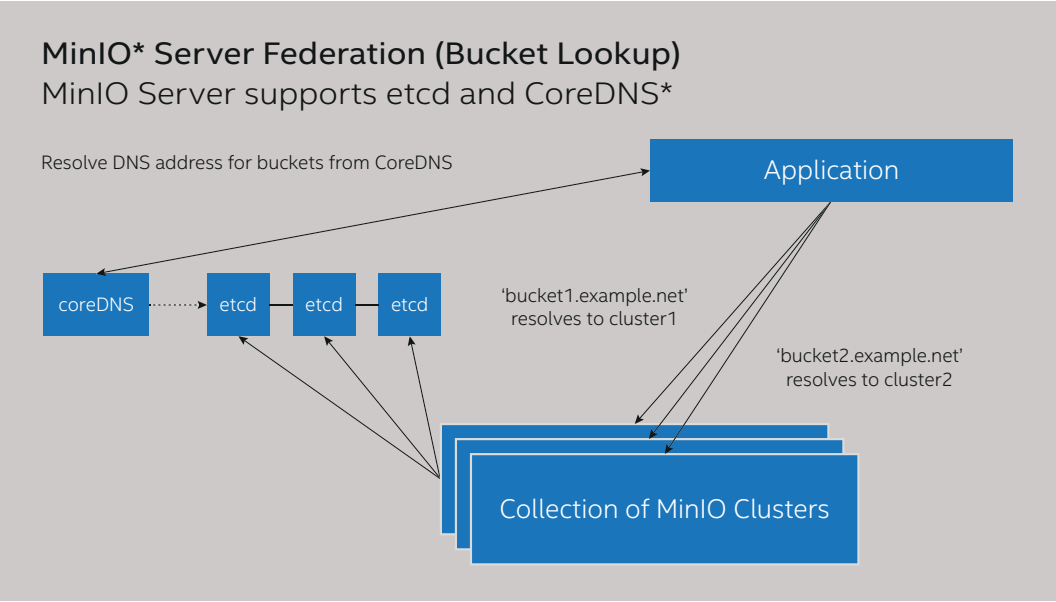


Figure 2. Federation is used to build very large MinIO* clusters with a single global namespace.

Automating MinIO Deployments Using Kubernetes*

MinIO is lightweight and container-friendly. When combined with Kuberbetes*-like orchestration platforms, MinIO object storage resources can be managed just like compute resources. With Kubernetes, launching a new MinIO instance is as easy as `helm install stable/minio`.

By using persistent volumes (PVs) and persistent volume claims (PVCs), Kubernetes makes it easy to abstract physical storage details from your application. MinIO Servers can be used to aggregate persistent volumes (PVs) into a scalable distributed object storage server that can be provisioned as needed.

At the time of deployment, the MinIO Server makes a request for a PVC, which will be used for disk space and memory. Kubernetes fulfills the claim by mapping it to a matching PV. If a MinIO container goes down, Kubernetes automatically deploys a replacement MinIO container and ensures that the PV is attached to the new replacement container.

MinIO Best Practices

Cluster Sizing

Cluster sizing with MinIO is as simple as calculating the number of nodes and the number of drives in a cluster to determine the total capacity of the cluster. A single MinIO cluster can be created with anywhere from 4 to 32 nodes. Individual servers from Dell, Supermicro and others feature 24, 32, 45 or even 90 drives. Table 3 shows the total capacity of several example clusters.

When choosing the size of the nodes and the size of the cluster, pay attention to the failure domain. Generally, Intel recommends keeping the failure domain as small as possible. This keeps the blast radius of operational problems as small as possible.

Table 3. Cluster Storage Capacities

Cluster Size	4 Nodes	12 Nodes	24 Nodes	32 Nodes
Drives per Node	12 x 8 TB	24 x 8 TB	24 x 8 TB	32 x 8 TB
Total Capacity	384 TB	2,304 TB	4,608 TB	8,192 TB

Performance and Networking

MinIO is able to consume fast networking technologies, even 100 GbE, in each server. Your cluster will benefit from using fast networking in each MinIO node. The choice of networking speed is largely determined by the types of applications running on your clients. For example, if you have graphics processing unit (GPU)-accelerated AI workloads, then you will most likely benefit from creating a high-performance MinIO object storage cluster.

Summary

Cloud service providers are facing spiraling data volumes, inefficient storage appliances and increasing demand from customers for cost-effective storage. Software-defined STaaS solutions can solve these challenges by abstracting the storage software from the storage hardware. A STaaS solution based on MinIO supports a wide variety of use cases, including AI and machine learning, advanced analytics, archival and application data. MinIO is optimized for Intel architecture and provides extreme scalability, high performance and availability, excellent security and ease of use. This implementation guide shows how to combine MinIO with Intel technologies such as NVMe-based Intel SSDs, Intel Ethernet products, and Intel Xeon Scalable processors. Using these guidelines, you can build a MinIO-based STaaS solution that provides the performance you need to successfully compete in the fast-growing STaaS market.

References

Here's a summary of the primary external references relevant to this document:

MinIO:

- MinIO home page
- MinIO federation documentation
- MinIO Documentation
- MinIO Subnet

Intel:

- Intel® Cloud Insider Program
- Intel® Storage Builders Program
- Intel® Xeon® Scalable processors
- Intel® Solid State Drives
- Intel® Ethernet products
- Intel® Cache Acceleration Software (Intel® CAS)
- Intel® Intelligent Storage Acceleration Library (Intel® ISA-L)
- Tuning Throughput Performance for Intel® Ethernet Adapters

Appendix A: Linux* Kernel Tuning Parameters

Most Linux* systems should provide adequate performance out of the box. If you want to tune the OS for maximum performance, then Intel recommends the following kernel settings for software-defined storage (SDS) solutions.

```
kernel.pid _ max=4194303
fs.file-max=4194303
vm.swappiness = 1
vm.vfs _ cache _ pressure = 10
vm.min _ free _ kbytes=1000000
net.core.rmem _ max=268435456
net.core.wmem _ max=268435456
net.core.rmem _ default=67108864
net.core.wmem _ default=67108864
net.core.netdev _ budget=1200
net.core.optmem _ max=134217728
net.core.somaxconn=65535
net.core.netdev _ max _ backlog=250000
net.ipv4.tcp _ rmem=67108864 134217728
268435456
net.ipv4.tcp _ wmem=67108864 134217728
268435456
net.ipv4.tcp _ low _ latency=1
net.ipv4.tcp _ adv _ win _ scale=1
net.ipv4.tcp _ max _ syn _ backlog=30000
net.ipv4.tcp _ max _ tw _ buckets=2000000
net.ipv4.tcp _ tw _ reuse=1
net.ipv4.tcp _ tw _ recycle=1
net.ipv4.tcp _ fin _ timeout=5
net.ipv4.udp _ rmem _ min=8192
net.ipv4.udp _ wmem _ min=8192
net.ipv4.conf.all.send _ redirects=0
net.ipv4.conf.all.accept _ redirects=0
net.ipv4.conf.all.accept _ source _ route=0
net.ipv4.tcp _ mtu _ probing=1
```

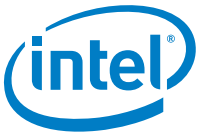
Solutions Proven by Your Peers

A STaaS platform, powered by Intel technology, provides high performance and easy manageability. This and other solutions are based on real-world experience gathered from customers who have successfully tested, piloted and/or deployed the solutions in specific use cases. Intel Solutions Architects are technology experts who work with the world's largest and most successful companies to design business solutions that solve pressing business challenges. The solutions architects and technology experts for this solution reference architecture include:

- **Daniel Ferber**, Solutions Architect, Intel Sales & Marketing Group
- **Karl Vietmeier**, Solutions Architect, Intel Sales & Marketing Group

Find the solution that is right for your organization. Contact your Intel representative or visit **intel.com/CSP**.

Solution Provided By:



0519/JS/CAT/PDF 338993-001EN