Whitepaper

Platform Architecture Overview

Build Better Faster

www.ondat.io

Contents

Introduction 3

About Ondat 3

Deployment 3

Inside the Ondat container

- The Control Plane 4
- The Data Plane 4

How Ondat works 4

Provisioning storage 5

Features

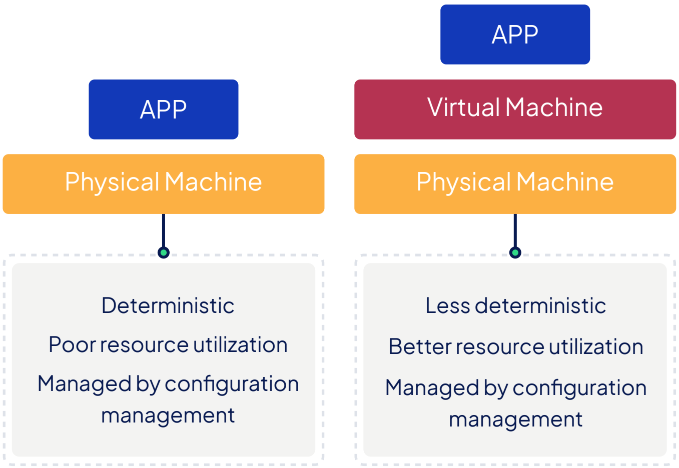
- Replication for high availability 5
- Encryption at rest 6
- Policy management 7
- Rapid failover (fencing) 7
- Management 8

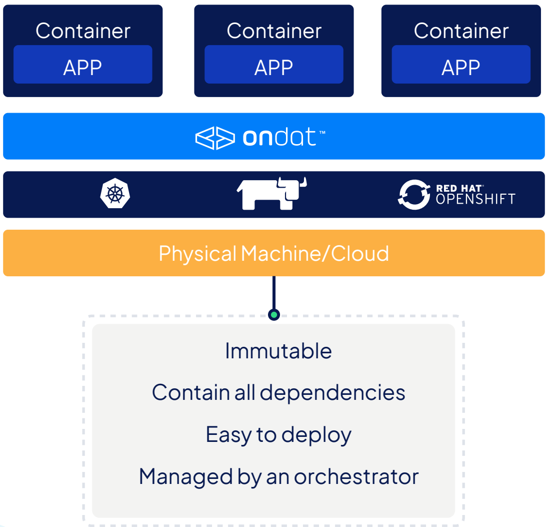
Conclusion: Ondat Self Evaluation Guide 8

Introduction

Cloud native is fast becoming the de-facto standard in IT development as businesses seek to innovate quickly. That means deploying apps anywhere in seconds, helping to improve time to market and ensuring a great end-user app experience. This new cloud native world sees adoption of containers and orchestrators to achieve many business and technology outcomes. During the cloud native journey, organizations will soon realize that they want to move stateful workloads to their new environments.

While containers provide well understood advantages over both physical and virtual machines, they are ephemeral filesystems that do not persist to disk.


APP
Physical Machine
Deterministic
Poor resource utilization
Managed by configuration management
APP
Virtual Machine
Physical Machine
Less deterministic
Better resource utilization
Managed by configuration management



To run applications which require persist storage within containers, we require a layer which can provide persistent disk storage to those containers, independent of the lifecycle of the containers themselves.

This paper is an architecture overview of how Ondat delivers a software-defined, cloud native storage solution.

About Ondat

Ondat is a software-defined, cloud native storage solution. We give you **total control** of your storage environment – whether on-premises or in the cloud. We deliver **persistent storage** to applications in containerized environments, helping you achieve all of the **business benefits** of this technology.

Our software is **built for developers and highly performant** allowing you to break lock-in, improve agility and respond to change quickly.

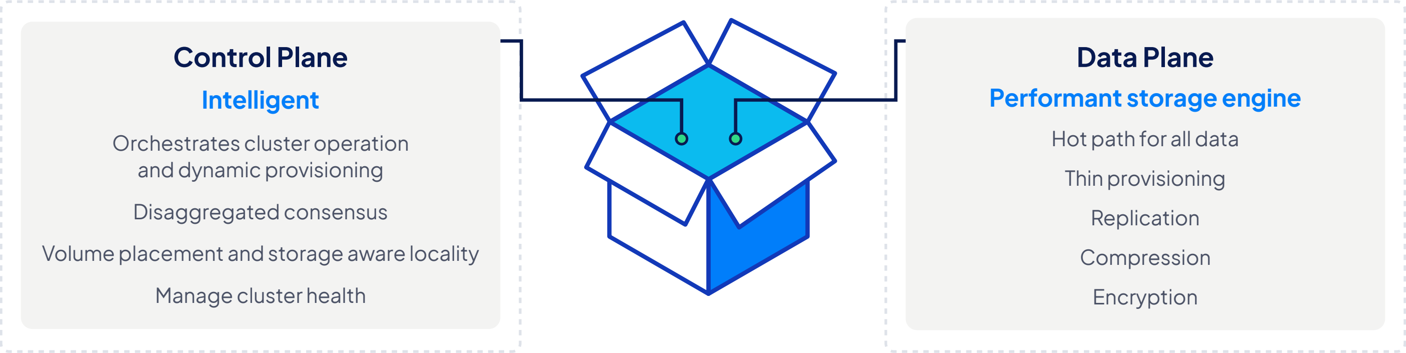
With Ondat, you can expect to save on infrastructure costs because you'll turn commodity hardware into enterprise grade storage. Your engineers will love that they can self-provision storage without waiting months for other teams. This all allows you to respond to business change quickly.

Deployment

Based on the principles of cloud native, Ondat ships as a container. Our software is deployed as a DaemonSet across your Kubernetes nodes, orchestrated by our operator. Ondat is designed to be simple to install – requiring only a few commands to achieve a working cluster.

Inside the Ondat container

Ondat consists of two fundamental components - an intelligent control plane and data plane.



Control Plane

Intelligent

Orchestrates cluster operation and dynamic provisioning

Disaggregated consensus

Volume placement and storage aware locality

Manage cluster health

Data Plane

Performant storage engine

Hot path for all data

Thin provisioning

Replication

Compression

Encryption

The Control Plane

The Ondat control plane orchestrates cluster operations such as volume placement, and reacts to node failure, dynamically promoting volume replicas and moving mountpoints as appropriate.

We use an external etcd cluster to store state and manage distributed consensus. To complement this, a gossip protocol is established between all the nodes to monitor cluster health.

The Data Plane

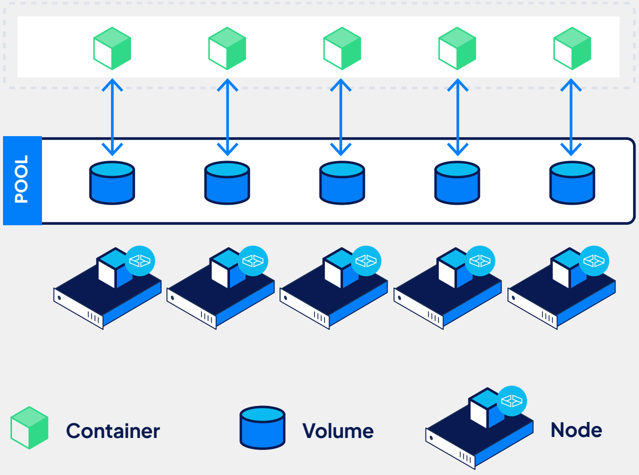
The data plane is our end-to-end block storage implementation. Utilizing a patented on-disk format, the data plane stores user volume data in BLOB files on hosting nodes. The data plane is written in fast C++ and is the only layer through which user volume data travels. We apply various transforms to data before committing to disk, including encryption and compression, using the LZ4 algorithm. These transforms are controllable on a per-volume basis.

Ondat compression is enabled by default. Performance is generally increased when compression is enabled due to fewer read/write operations taking place on the host disks.

How Ondat works

Ondat aggregates storage across all nodes in a cluster into a pool. It allows volumes to be provisioned from the pool and for containers to mount those volumes from anywhere in the cluster. Ondat transparently redirects reads and writes to the appropriate volume, so the container is unaware of whether it is accessing local storage or remote storage. Volumes are thin provisioned to avoid consuming disk space unnecessarily.

Ondat features are all enabled/disabled by applying labels to volumes. Labels can be passed to Ondat via PersistentVolumeClaims (PVCs) or can be applied to volumes using the Ondat CLI or GUI.



Provisioning storage

Users can provision and manage a volume with standard Kubernetes semantics via a Kubernetes PVC. It is a secure native Kubernetes integration where:

- Namespaces segregate the scope of control aligned to K8S namespaces.
- Kubernetes RBAC manages permissions to namespaces and volumes.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-vol-1
  annotations:
 volume.beta.kubernetes.io/storage-class: fast
spec:
  accessModes:
 - ReadWriteOnce
  resources:
 requests:
   storage: 5Gi
---
apiVersion: v1
 kind: Pod
 metadata:
   name: d1
 spec:
   containers:
     - name: debian
       image: debian:9-slim
       command: [“/bin/sleep”]
       args: [ “3600” ]
       volumeMounts:
         - mountPath: /mnt
           name: v1
   volumes:
     - name: v1
       persistentVolumeClaim:
         claimName: my-vol-1
```

Applications create a Ondat volume through the PVC specifying size. Volumes are dynamically provisioned instantly, and mountable on any node immediately, without having to detach and reattach physical volumes.

Provisioning storage directly to the application (rather than the operating system) allows storage to be declared and composed as part of application instantiation through Kubernetes. This enables developers to deploy and provision storage resources and services alongside CPU, networking and other application resource.

Features

Replication for high availability

Replication is the process by which one or more replica volumes can be kept in sync with a single master volume. High availability refers to the ability to switch between the master and replicas at will, so if the master is suddenly unavailable (for whatever reason), a replica can be promoted to master. This is essential for any organization wanting to run stateful applications in containers. Without it, the business risks data loss or downtime.

Wih replication disabled, a Ondat volume saves data to a single node in a cluster. When a node fails, access to the Ondat volume is suspended for the duration of the node failure, thus causing outage for the application using the volume.

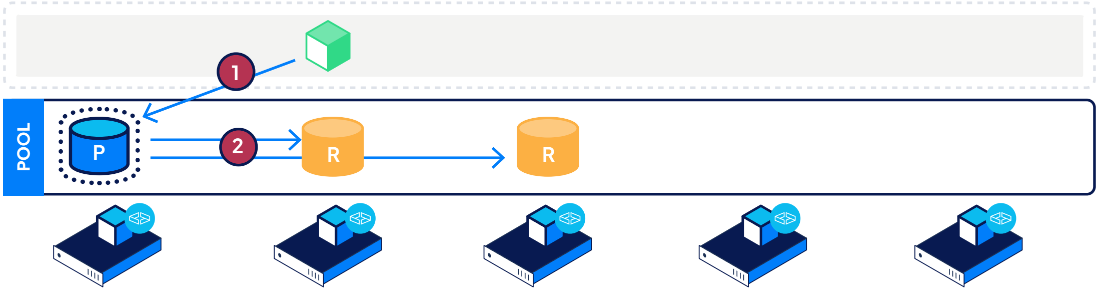
When enabled, under node failure condition, Ondat volume replication will transparently promote a replica node to master. Mount endpoints migrate to the new master, and applications continue without requiring maintenance or downtime. From the perspective of the application, the only visible effect is a small pause in IO while the failover takes place.

This allows applications backed by Ondat volumes to be turned into HA applications without extra development work or application refactoring.

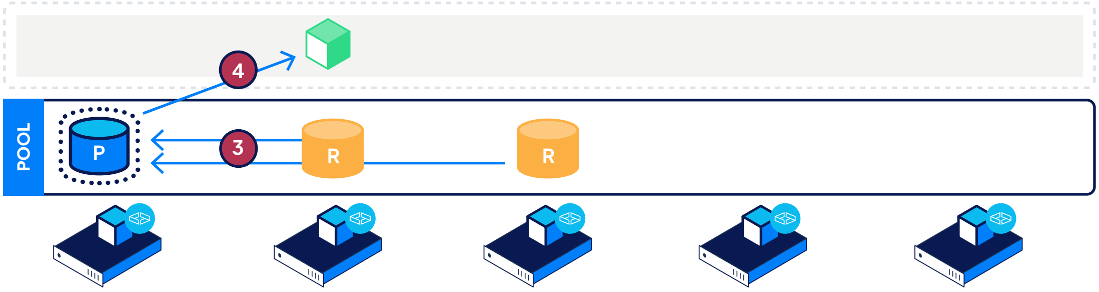
How replication for HA works

Ondat protects data from a disk or node failure and ensures a strong consistency model. Replication is synchronous between a primary volume and user defined number of replicas (up to 5).

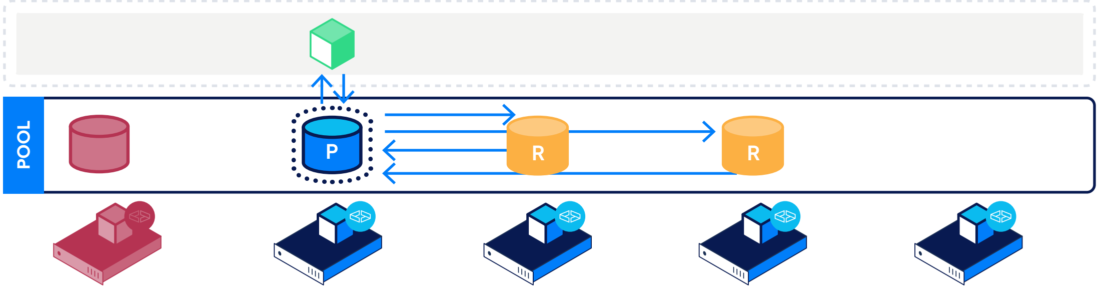
1. Data is sent to the primary first and then sent in parallel to all the replicas.
2. Then sent in parallel to all the replicas.



3/4. All acks need to be received by the primary and the write needs to be acknowledged to the application.



5/6. If a node fails, a replica is automatically promoted to become a new master and another replica is provisioned on an available node. Volume mount points move transparently to the application.



Click here to read the documentation on replication for HA

Encryption at rest

To prevent bad actors from viewing data offline e.g. by stealing disks, etc., Ondat includes encryption of data at rest, using keys that only you hold access to. This is an important distinction between encryption of Ondat volumes and encryption of devices offered by cloud providers.

Ondat encrypts data at rest using AES-256 in XTS-AES mode with 512 bit keys as recommended by NIST. Usage of XTS-AES encryption enables the use of the AES-NI instruction set when available to minimize the CPU overhead and latency of encrypting volumes. The keys and initialization vectors used to encrypt volumes are generated using the **crypto/rand** package. Each volume is encrypted with a unique 512bit key.

Ondat has no access to, nor means to recover these keys allowing you to make iron clad guarantees about who has access to your encryption keys. This also means that data can effectively be destroyed by deleting the volumes' encryption keys.

Encryption keys are stored as Kubernetes secrets. For increased security, we recommend the usage of a **Kubernetes KMS plugin** to protect the secrets using the envelope encryption scheme of a KMS provider.

Click here to read more about how Ondat encryption works.

Policy management

Ondat policy management enables compliance with corporate policy (e.g. replica count, encryption, compression) while retaining developer agility. Ondat rules control features based on volume labels/ namespaces. To grant a user or group access to a namespace, a policy needs to be created mapping the user or group to the namespace. Policies control access to Ondat namespaces. Policies can be configured at the group or user level so access can be controlled granularly.

Users can belong to one or more groups to control their namespace permissions. Additionally, user specific policies can be created to grant a user access to a namespace. Users can belong to any number of groups and have any number of user level policies configured.

Rapid failover (fencing)

The Kubernetes [StatefulSet] controller is the standard controller for running stateful workloads on Kubernetes. It provides volume templating, strong guarantees about pod creation order, and enforces serialization of mounts and unmounts such that a given volume can never be mounted twice.

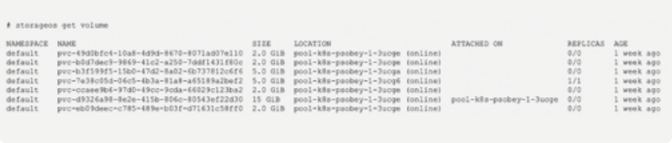
To provide these guarantees, the StatefulSet controller is highly conservative with respect to restarting pods – specifically it tries hard to ensure that a given pod is completely dead with its volume unmounted before scheduling a replacement. Manual intervention is normally required before a StatefulSet will failover to another node.

When enabled for a volume, Rapid Failover will use Ondat awareness of node health to influence StatefulSet pod failover.

Click here to read the documentation on Rapid failover (fencing)

GUI - Visualize the storage environment for ease of use

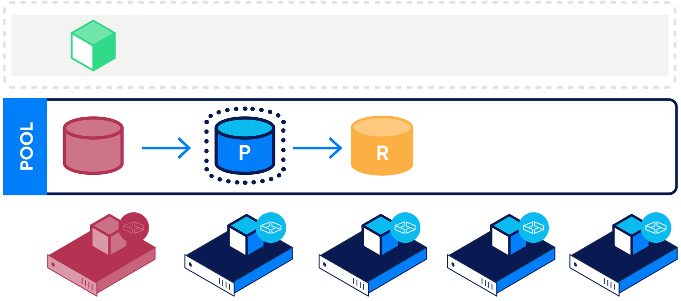
CLI - Open source to manage cluster-wide configuration



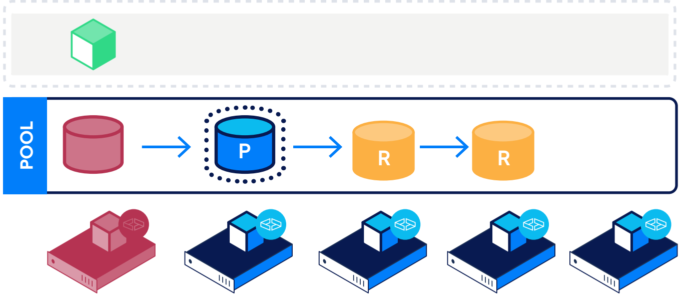
```
# storageos get volume

NAMESPACE  NAME                                        SIZE     LOCATION                          ATTACHED ON               REPLICAS  AGE
default    pvc-49d0bfc4-10a8-4d9d-8670-8071ad07e110  2.0 GiB  pool-k8s-psobey-1-3ucge (online)                            0/0       1 week ago
default    pvc-b0d7dec9-9869-41c2-a250-7ddf1431f80c  2.0 GiB  pool-k8s-psobey-1-3ucge (online)                            0/0       1 week ago
default    pvc-b3f599f5-15b0-47d2-8a02-6b737812c6f6  5.0 GiB  pool-k8s-psobey-1-3ucga (online)                            0/0       1 week ago
default    pvc-7e38c05d-06c5-4b3a-81a8-a65189a2bef2  5.0 GiB  pool-k8s-psobey-1-3ucg6 (online)                            1/1       1 week ago
default    pvc-ccaee9b6-97d0-49cc-9cda-66029c123ba2  2.0 GiB  pool-k8s-psobey-1-3ucge (online)                            0/0       1 week ago
default    pvc-d9326a98-8e2e-415b-806c-80543ef22d30  15 GiB   pool-k8s-psobey-1-3ucge (online)  pool-k8s-psobey-1-3ucge  0/0       1 week ago
default    pvc-eb09deec-c785-489e-b03f-d71631c58ff0  2.0 GiB  pool-k8s-psobey-1-3ucge (online)                            0/0       1 week ago
```

Click here to read the documentation on Policy management



When enabled for a volume, Rapid Failover will use Ondat awareness of node health to influence StatefulSet pod failover.



For certain workloads this provides faster failover behaviour than the StatefulSet controller alone.

Management

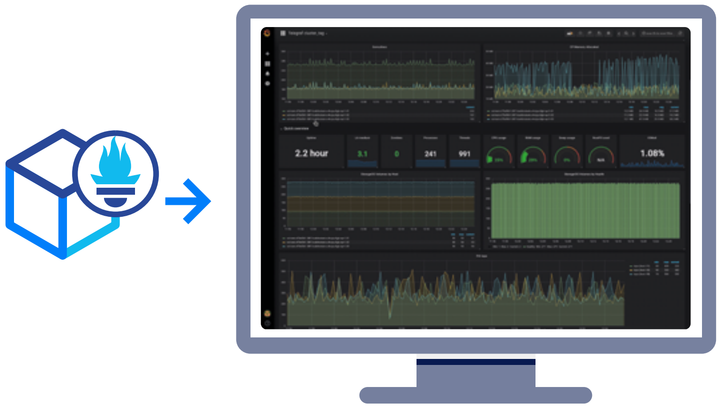
Ondat management features include:

- Ondat can be managed through a Command Line Interface (CLI) to manage cluster-wide configuration. **Docs: Installation and Usage**
- Ondat provides a GUI for cluster and volume management. The GUI is available at port 5705 on any of the nodes in the cluster. **Docs: Login and managing cluster nodes and pods**
- **Prometheus** endpoints expose metrics about Ondat artefacts (such as volumes), as well as internal Ondat components. Customers may scrape these metrics using Prometheus itself, or any compatible client, such as the popular **Telegraf** agent shipped with InfluxDB. **Docs: Metrics**

```
# TYPE storageos_volume_frontend_read_bytes_
total counter

storageos_volume_frontend_read_bytes_total{na
mespace=”mysql”,pool=”default”,type=”presenta
tion”,volume_id=”48459472-80a5-96ee-9a5d-
486319ccc5bd”,volume_name=”prod-mysql-0”}
1.077248e+06

storageos_volume_frontend_read_bytes_total{na
mespace=”mysql”,pool=”default”,type=”presenta
tion”,volume_id=”a1ddf1bb-dda3-5ab5-bda7-
cfea0e9c7ccb”,volume_name=”dev-mysql-0”}
1.077248e+06
```



Get started today with Ondat

Visit us at www.ondat.io or email us at info@ondat.io