Platform
Architecture
Overview

ondat™

Contents

Introduction

Cloud native is fast becoming the de-facto standard in IT development as businesses seek to innovate quickly.
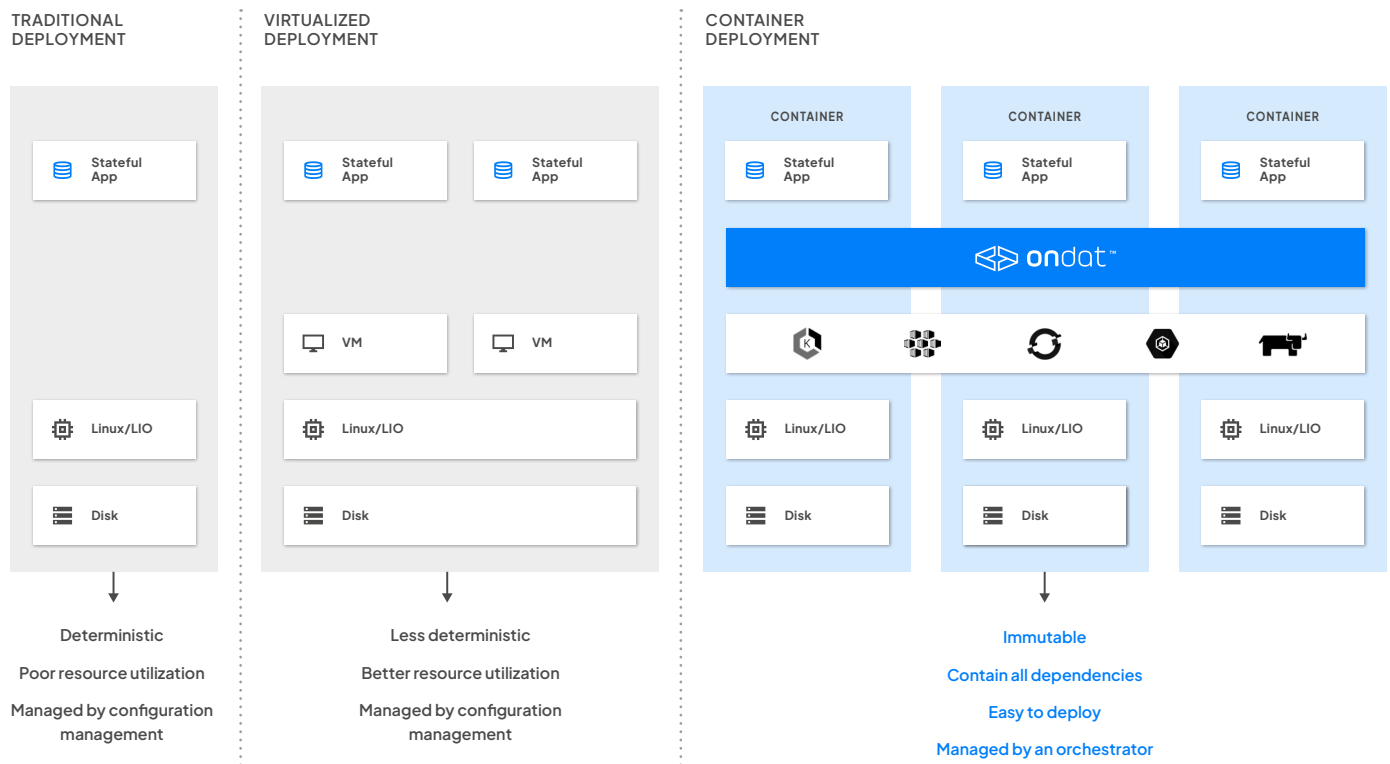
That means deploying apps anywhere in seconds, helping to improve time to market and ensuring a great end-user app experience.

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.

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.

TRADITIONAL DEPLOYMENT	VIRTUALIZED DEPLOYMENT	CONTAINER DEPLOYMENT		
		CONTAINER	CONTAINER	CONTAINER
Stateful App	Stateful App / Stateful App	Stateful App	Stateful App	Stateful App
		ondat™		
	VM / VM			
Linux/LIO	Linux/LIO	Linux/LIO	Linux/LIO	Linux/LIO
Disk	Disk	Disk	Disk	Disk
↓	↓	↓		
Deterministic	Less deterministic	Immutable		
Poor resource utilization	Better resource utilization	Contain all dependencies		
Managed by configuration management	Managed by configuration management	Easy to deploy		
		Managed by an orchestrator		

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 selfprovision storage without waiting months for other teams. This all allows you to respond to business change quickly.

KEY BENEFITS

Lightweight

Ships as lightweight containers. Deploy as a DaemonSet across your Kubernetes nodes, orchestrated by our operator.

Simple

Designed to be simple to install – requiring only a few commands to achieve a working cluster.

Performant

Ondat container includes a purpose-built intelligent control plane and data plane.

Highly available

Includes replication for high availability to keep your data safe and encrypts in transit to keep it secure from prying eyes.
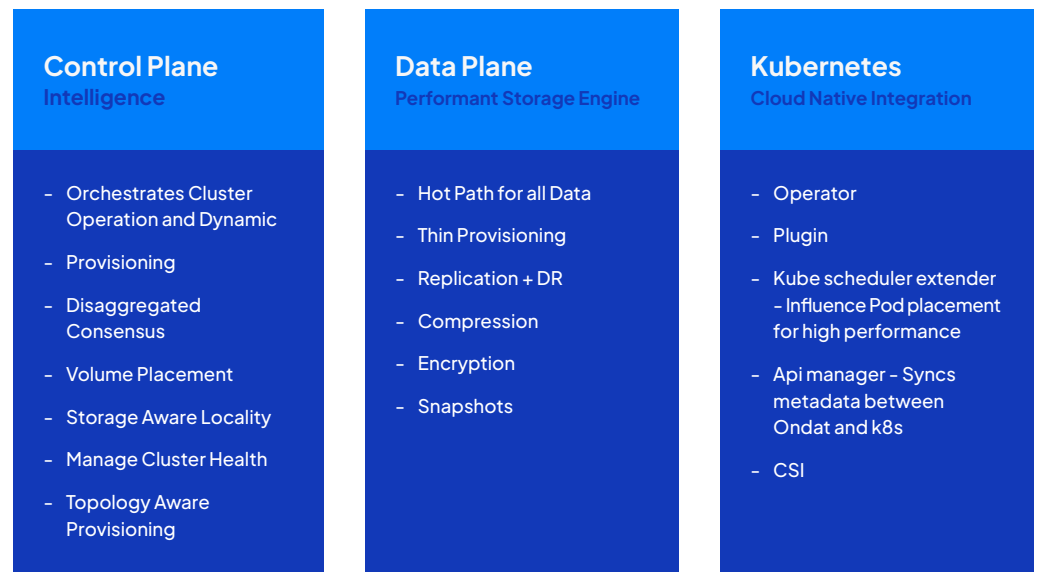
Deployment

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
Intelligence

- Orchestrates Cluster Operation and Dynamic
- Provisioning
- Disaggregated Consensus
- Volume Placement
- Storage Aware Locality
- Manage Cluster Health
- Topology Aware Provisioning

Data Plane
Performant Storage Engine

- Hot Path for all Data
- Thin Provisioning
- Replication + DR
- Compression
- Encryption
- Snapshots

Kubernetes
Cloud Native Integration

- Operator
- Plugin
- Kube scheduler extender – Influence Pod placement for high performance
- Api manager – Syncs metadata between Ondat and k8s
- CSI

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 managedistributed 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 pervolume 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**
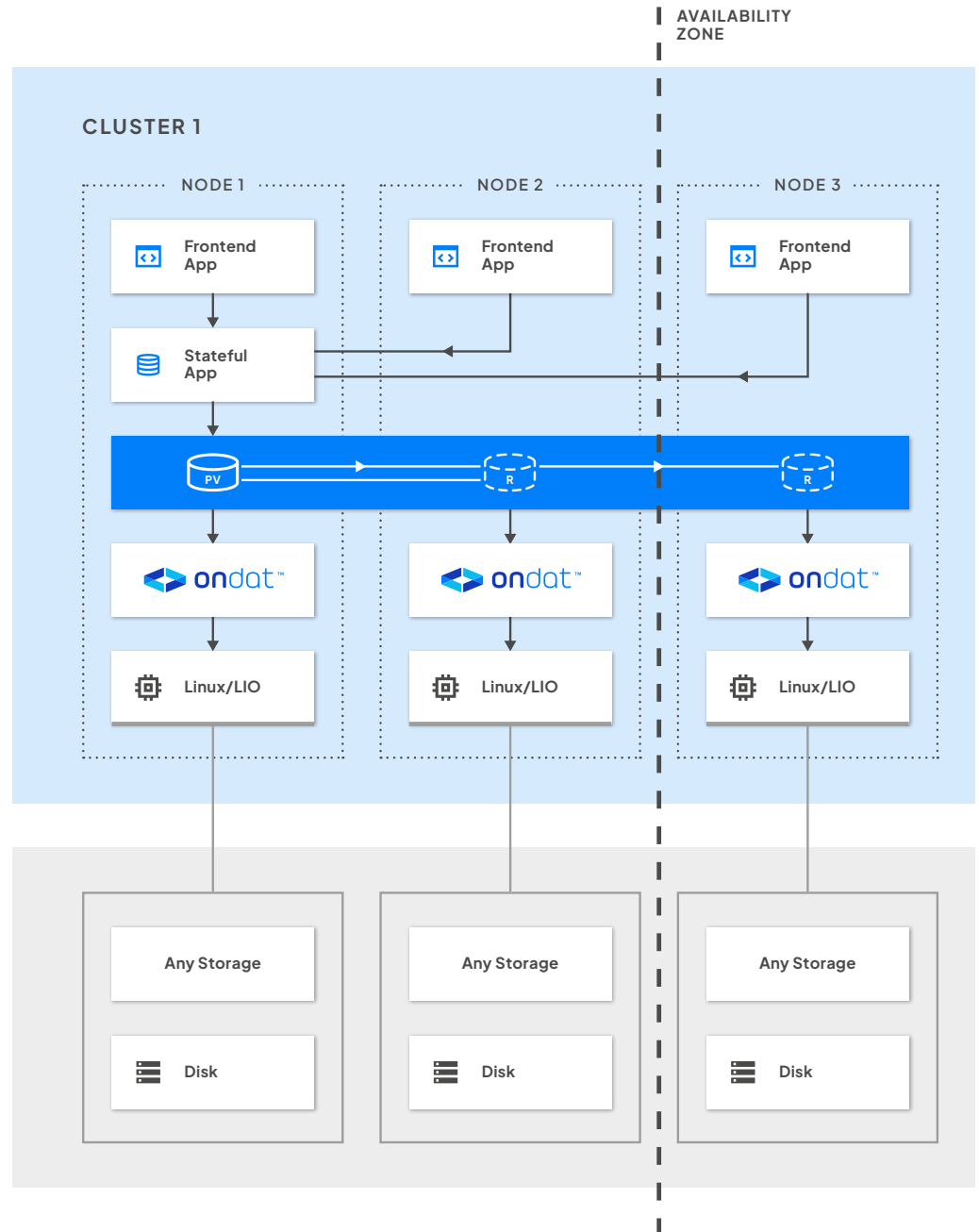
Ondat aggregates storage across all nodes in a cluster into a pool.

It allows volumes to be provisioned from the pool and for containers to mount those volumes from anywhere in the cluster.

Ondat 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.

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.

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.

```
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
```

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.

With 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.
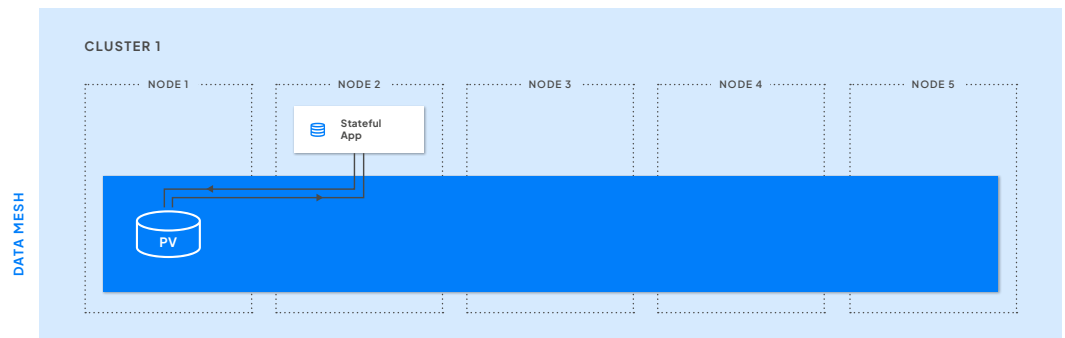
PV
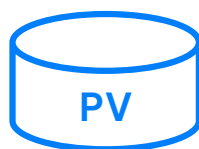
PERSISTENT
VOLUME

R
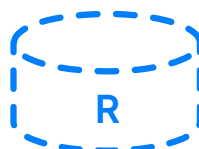
REPLICA

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).

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

All acks need to be received by the primary and the write needs to be acknowledged to the application.
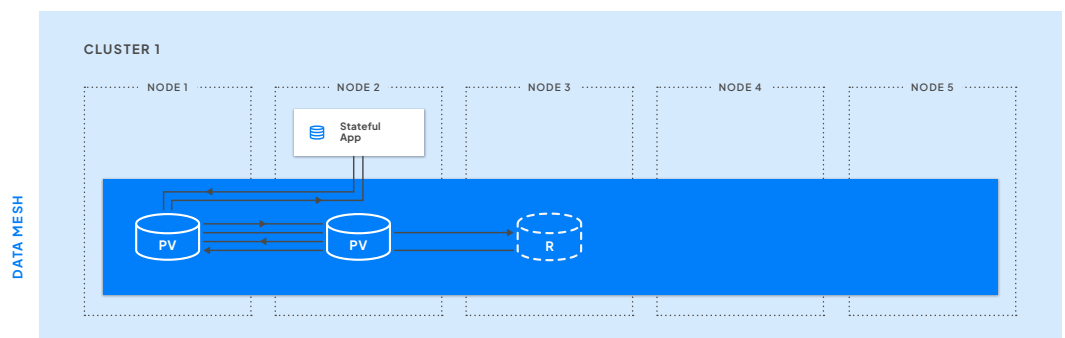
If a node fails, a replica is automatically promoted to become a new master and another replica is provisioned on an available node. Volume mount points move transparently to the application.
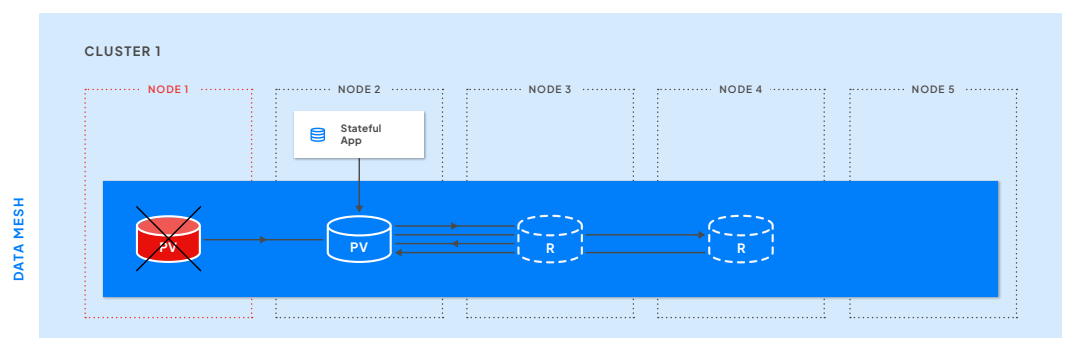
CLICK HERE TO LEARN MORE ON **REPLICATION FOR HA**

**Encryption
at rest and in
transit**

To protect your data, Ondat provides encryption of data-at-rest and data-in-transit at the PV level, 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 and in transit using AES-256 in XTS-AES mode with 512 bit keys as recommended by NIST. Usage of XTSAES 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 LEARN MORE ON **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.

Ondat SaaS Portal providing holistic view of the Kubernetes clusters and data resources
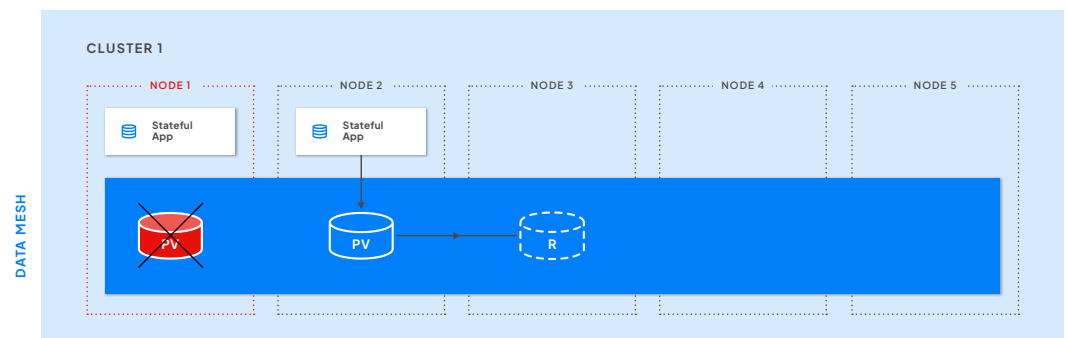

CLICK HERE TO READ THE **DOCUMENTATION ON POLICY MANAGEMENT**
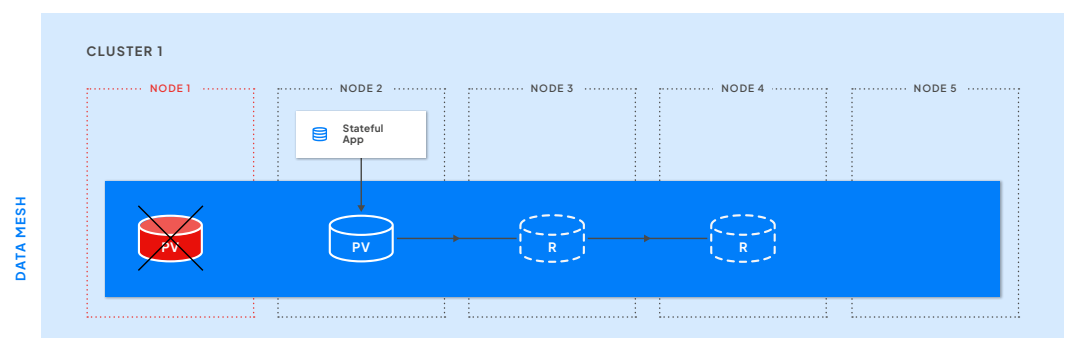
Rapid failover (fencing)

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.

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

CLICK HERE TO READ THE **DOCUMENTATION ON RAPID FAILOVER (FENCING)**

Management

Ondat managment features include:

> Ondat can be managed through a Command Line Interface (CLI) to manage cluster-wide configuration. **Docs: Installation and Usage**

> Ondat provides a **SaaS Platform** to view and manage your clusters. **Ondat SaaS platform installation guide**

> **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{namespace="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{namespace="mysql",pool="default",type="presenta
tion",volume_id="a1ddf1bb-dda3-5ab5-bda7-cfea0e9c7ccb",volume_
name="dev-mysql-0"}
1.077248e+06
```

PersistentVolumes (12) **StorageClasses (11)** Nodes (3)

StorageClass Name	Provisioner	FS Type
storageos-rep9	csi.storageos.com	ext4
storageos-rep5	csi.storageos.com	ext4
storageos-rep7	csi.storageos.com	ext4
storageos-rep10	csi.storageos.com	ext4
storageos-rep6	csi.storageos.com	ext4
storageos-rep8	csi.storageos.com	ext4

StorageClass details

Compression	Encryption	Replicas
False	False	2

Failure Mode	Allow Volume Expansion	Volume Binding Mode
-	True	Immediate

Reclaim Policy	Is Default Class	Mount Options
Delete	False	None

Manifest

```
1  allowVolumeExpansion: true
2  apiVersion: storage.k8s.io/v1
3  kind: StorageClass
4  metadata:
5    annotations:
6      kubectl.kubernetes.io/last-applied-configuration: >
7        {"allowVolumeExpansion":true,"apiVersion":"storage.k8s.io/v
```

Cluster summary   ✎ Edit Cluster

Cluster Name	Cluster ID	Connection Status	Licence	Cluster Location
test-bench	f92b48cd-7884-468a-9c22-2bc4b29c4a56	● Good	Free Trial	☁ Google Cloud

PersistentVolumes (12) StorageClasses (11) **Nodes (3)**

Node Name	Health	Age
gke-test-bench-default-pool-0f6533b1-bk62	● ONLINE	21 days 11 hours 29 minutes
gke-test-bench-default-pool-0f6533b1-1cvc	● ONLINE	21 days 11 hours 39 minutes
gke-test-bench-default-pool-0f6533b1-nax7	● ONLINE	21 days 11 hours 35 minutes

Register for our SaaS Platform

Learn how Ondat can help you scale persistent workloads on Kubernetes.

Register

For information, please write to us at

info@ondat.io