White Paper

Advanced Data Management For Kubernetes

Powered by ROBIN Storage

Table of Contents

The Need for Data Management on Kubernetes

Kubernetes is gaining rapid adoption and enterprise customers are demanding the ability to run broader sets of workloads including stateful applications. Running stateful applications such as PostgreSQL, MySQL, MongoDB, Elastic Stack, Kafka, and MariaDB require advanced data management capabilities in order to:

- **Release new products and features faster:** Automated lifecycle management for app+data (not just the storage) is required to save valuable time at each stage of the lifecycle.
- **Collaborate quickly across teams:** Multiple teams (Dev/Test/Ops) need a mechanism to collaborate without procedural delays. CI/CD pipelines solve a part of the problem with automating the collaboration for code changes, but data is usually left out.
- **Recover from system failures and user errors:** App+data protection capabilities such as point-in-time snapshots, backup, and restore are required to recover from system failures and user errors.
- **Avoid infrastructure lock-in:** The ability to migrate from on-prem to cloud and vice versa, and among the public clouds is needed to avoid infrastructure lock-in.
- **Deliver predictable performance:** To guarantee QoS and to ensure high priority applications do not miss SLAs, you need the ability to set IOPS limits per app.
- **Eliminate security vulnerabilities:** Enterprise-grade security is required with authentication and encryption to ensure your data is safe.

Defining and Managing An Application

Kubernetes provides many useful constructs such as Pods, Controllers, PersistentVolumes etc. to help you manage your applications. However, there is no construct for an “Application”, i.e. a single entity that consists of all the resources that form an application. Users have to manually map the resources to an application and manage each resource individually for any lifecycle operation. The lack of a proper Application construct in Kubernetes poses a problem when it comes to performing operations that encompass a group of resources.

Frameworks such as Helm and Operators try to solve this problem by packaging resources together, but they do not solve it beyond the initial deployment. For example, how would one snapshot, clone or backup an entire helm release that spans PersistVolumeClaims, Secrets, ConfigMaps, StatefulSet, Pods, Services etc? Or how about snapshotting a web-tier, app-tier and database-tier each deployed separately using 3 different kubectl manifest files?

To facilitate this, ROBIN supercharges Kubernetes with the notion of an Application. An Application is a collection of Kubernetes resources that form a single unit on which a DevOps engineer can perform Data Management operations. ROBIN has the most extensible constructs to define an application. The following different types of applications are supported by ROBIN:

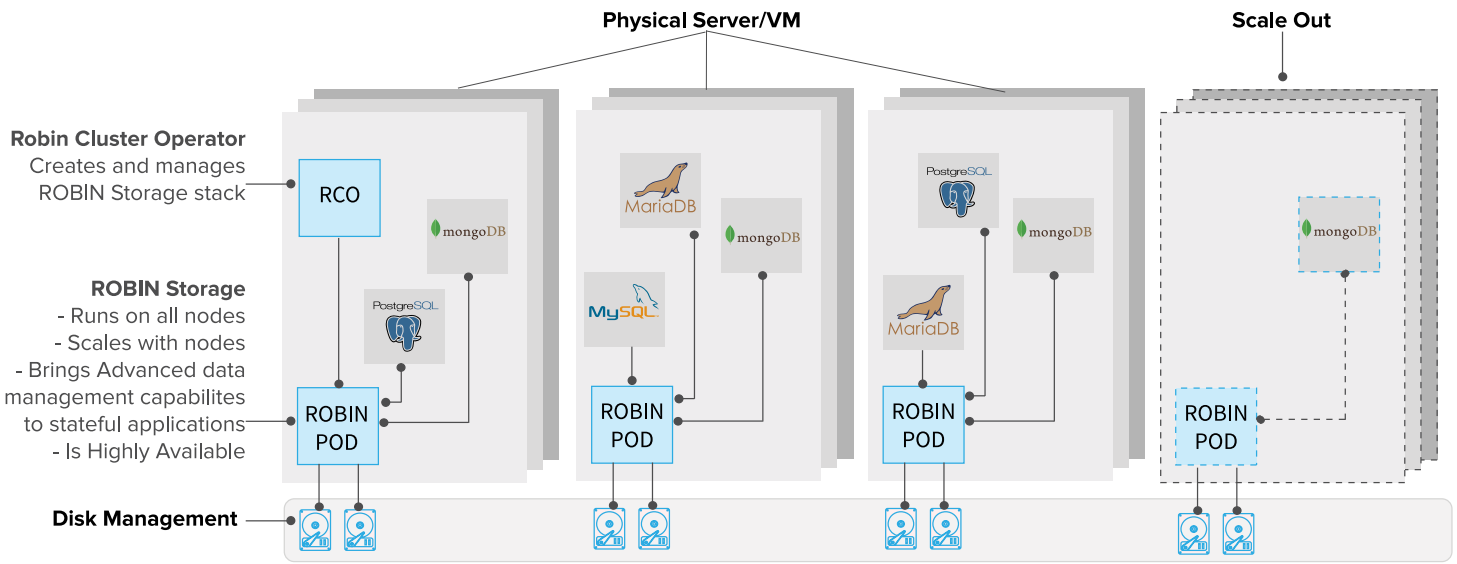
Application	Description
Helm Release	A Helm release that was deployed using helm install <chart-name>
Custom Operator	A custom resource that is spun up using a Custom Kubernetes Operator such as a couchbase operator
ROBIN FlexApp	A ROBIN defined construct which allows one to collect one or more Kubernetes resource into one unit using label-selector rules. For example, --select app=apache --select app=mysql would select Pods and PVCs for a web and database tier together into a single FlexApp object
ROBIN Application Bundle	This type of application is spun up using ROBIN's own Super Operator framework which allows one to deploy complex stateful workloads such as Cloudera, Oracle RAC, Splunk, SAP HANA, etc on Kubernetes.

ROBIN Storage: Manage App+Data as a Single Entity

ROBIN Storage is a purpose-built container-native storage solution that brings advanced data management capabilities to Kubernetes. It is a CSI-compliant block storage solution with bare-metal performance that seamlessly integrates with Kubernetes-native administrative tooling such as Kubectl, Helm Charts, and Operators through standard APIs.

ROBIN Storage is application-aware. The "Application" construct, as defined above, provides the context for all ROBIN Storage operations. All lifecycle operations are performed by treating app+data as a single entity. For example, when you snapshot a MongoDB application, ROBIN Storage captures the entire application topology and its configuration (i.e., specs of Pod, Service, StatefulSet, Secrets, ConfigMaps, etc), and all data volumes (PersistentVolumeClaims) to create a point-in-time application checkpoint.

ROBIN Storage for Kubernetes - GKE, OpenShift, CSP



Key Features

Data Protection and Security

- Protect app+data with replication, snapshots, backup & recovery to run always-on applications
- Secure data with encryption at rest and in motion
- Safeguard against data corruption with checksum error-detection

Automated Application Management

- Bring automated management of app+data (not just storage) to kubectl, Helm, and Operators
- Enable Quick and easy deployment of enterprise workloads on any Kubernetes distribution

High Performance at Scale and QoS Guarantee

- Get high performance enterprise-grade storage trusted and validated by Google
- Experience bare-metal performance with the flexibility and scale of software defined storage
- Guarantee QoS for high priority applications by setting IOPS limits per application

DevOps Collaboration for Stateful Applications

- Enable collaboration across geos and teams by cloning app+data in minutes
- Quickly share app+data among Dev, QA, and Production teams to shorten release cycles

Hybrid and Multi-Cloud Flexibility

- Enable easy movement of app+data, between on-prem and cloud(s)
- Avoid infrastructure lock-in, run your applications on most cost-effective infrastructure

Registering Helm Releases as Applications

Before data management can be performed on a collection of PersistentVolumeClaims, Pods, etc all the relevant Kubernetes resources that together deliver a service to the end user must be collected together into one single unit, which ROBIN tracks as an Application. Application resources created by frameworks such as Helm or Operators can be registered as an "App" with ROBIN.

The comman `kubectl robin app register` is used to register an existing Helm Release, Custom Operator etc with ROBI

Command to register a Helm release:

```
$ kubectl robin app register <appname> --app helm/<helm-release-name>
```

Name	Description
<a name>	The name to give to this app in ROBIN. To make it easy to associate with a specfic Helm release that this app refers to it is best to use a name that matches the helm release name
helm/<helm-release-name>	Name of the Helm release that is obtained by running helm list

Example: Register Postgres Helm Release as an App in ROBIN:

```
$ helm install --name pgdb stable/postgresql
```

```
$ kubectl robin app register mydb --app helm/pgdb
```

App+Data Time-Travel with Snapshots

Snapshots allow you to restore your application's state to a point-in-time. If you make a mistake, such as unintentionally deleting important data, you can simply undo it by restoring a snapshot. With ROBIN Storage you can seamlessly travel between different application states (backward and forward), even if the application's topology and configuration has changed over time. With ROBIN's app+data time-travel your developers are free to run what-if analysis quickly and restore a safe app+data checkpoint after experimentation.

ROBIN allows creating point-in-time snapshots of the app+data, which captures:

- The entire application topology and its configuration (i.e., specs of Pod, Service, StatefulSet, Secrets, ConfigMaps, etc), and
- Snapshot of all data volumes (PersistentVolumeClaims)

ROBIN snapshots at the storage layer are based on Redirect-on-Write based technology, which means that snapshots can be taken in sub-seconds even for terabyte sized data volumes. Because only the changed blocks are tracked between snapshots, ROBIN snapshots are very space efficient.

An application snapshot is created using the command below:

```
$ kubectl robin snapshot create <snapname> <appname>
```

Command	Description
snapname	A name that would be assigned to the snapshot being taken
appname	Name of the application being snapshotted. appname is obtained by running robin app list command

DevOps Collaboration using App+Data Clones

With ROBIN Storage, you can clone entire application environments, and share them across teams. While developing new features, your Dev team can use a clone of the production environment as the starting point to minimize errors. Upon finishing the implementation, the Dev team can create a clone of their environment and handover to QA for testing. Clones are independent of the parent application. Changes made to the clone are not visible in the parent application from which the clone was created. Similarly, any changes made to the parent application are not visible to the clone.

Typically cloning an application requires significant manual or scripting work and coordination across the storage layer and the application metadata layer (Pod, StatefulSet, PVC, Service etc). With ROBIN one can clone app+data from a previously taken snapshot with one single command. Once created, the clone is ready for use right away.

At the storage layer, ROBIN Storage's thin cloning technology is leveraged. Which means that a cloned PersistentVolume is not physical copy of the source volume, instead it is a virtual copy, where only metadata is physically copied with the actual data being shared between the clone and the source. Changes made to either the clone or the source are localized into their own "writable" data streams, whereas any common unmodified data is shared between them.

An application can be cloned using the following command:

```
$ kubectl robin clone create <clone-app-name> <snapshotid>
```

Command	Description
`<clone-app-name>`	A name to give to the cloned application
`<source-app-snapshotid>`	Snapshot ID of the source application. This is obtained by running `robin snapshot list` command

Backup & Restore App+Data to Recover from System Failures

Running any stateful application, such as MySQL, Postgres, Oracle, MongoDB, Cassandra, etc. in production would require that these databases be backed periodically up to ensure safety from data loss or corruption. A "Backup" is a full copy of the application snapshot that resides on a completely different storage media than the application's data. Therefore, backups are independent of the application and hence are useful to restore an entire application in the event of catastrophic failures, such as disks errors, server failures, data centers going offline etc.

ROBIN Storage allows one to:

- Backup App+Data to an external storage repository such as S3 Object Store, Google Cloud Storage etc
- Make App+Data portable across Kubernetes clusters spanning on-prem, public and hybrid cloud environments

To accomplish this an external storage repository such as Amazon S3 or Google GCS bucket it first registered with the ROBIN cluster and then attached to an application. After this, either manually through `kubectl robin backup` (or `kubectl robin s-`) command or automatically through schedules (`kubectl robin app config --backup-schedule`) snapshots of the application are backed up (i.e., physically copied) from the primary storage to the secondary storage repository.

An application can be backed-up using the following command:

```
$ kubectl robin backup create <backup-name> <snapshotid>
```

App+Data Portability across Clouds

Kubernetes-based architecture gives you complete control of your infrastructure. With the freedom to move your workloads across private and public clouds, you can avoid vendor lock-in. Using ROBIN Storage, you can easily move apps+data, across Kubernetes clusters spanning on-prem, hybrid, and public clouds.

An application can be moved using the following command:

```
$ kubectl robin app move <app-name> -from <cluster1> to <cluster2>
```

Conclusion

ROBIN Storage introduces the concept of "Application" to Kubernetes. With the "Application" construct, ROBIN Storage encapsulates all Kubernetes resources related to an application, including the entire application topology and its configuration (i.e., specs of Pod, Service, StatefulSet, Secrets, ConfigMaps, etc), and all data volumes (PersistentVolumeClaims) into a single entity. All lifecycle operations are performed on "App+Data" as a single entity.

By creating the App+Data single entity, ROBIN Storage brings advanced data management capabilities to Kubernetes. It seamlessly integrates with Kubernetes-native administrative tooling such as Kubectl, Helm Charts, and Operators through standard APIs. ROBIN Storage provides automated provisioning, point-in-time snapshots, backup and recovery, application cloning, QoS guarantee, and multi-cloud migration for stateful applications on Kubernetes.

ROBIN.IO

Suite 600, 224 Airport Pkwy
San Jose CA 95123
info@robin.io

Website: www.robin.io
Twitter: twitter.com/RobinSystems
LinkedIn: www.linkedin.com/company/robin-systems
Facebook: www.facebook.com/RobinSystems