Protecting Kubernetes Apps Requires More than Data Backup

The importance of data protection for enterprises is greater than ever. Data protection includes concepts such as high availability, business continuity, backup and recovery, and disaster recovery. Whatever you call it, it's critical to implement, test, and maintain strategies to reduce risk to your reputation—and your revenue.

As data becomes more important in customer experience and business processes, there's also little tolerance for downtime that blocks access to critical information. According to a study by Uptime Institute, 41% of outages cost more than $1 million, indicating an increasing dependence on highly available (HA) data infrastructure.

However, just because data protection is critical doesn't mean enterprises are succeeding at it. In fact, most companies struggle to protect their data. This is especially true in Kubernetes environments where traditional data-protection methods that work well for applications running on a single host don't scale well in a distributed or multi–data center environment common in Kubernetes environments. Just backing up your data isn't enough.

"Without container-granular backup options for Kubernetes, enterprises are exposed to data loss,

downtime, and lost customer loyalty.”- Enrico Signoretti, GigaOm

This post will explain why traditional approaches to backup don’t work for Kubernetes and what to look for in a Kubernetes-native data-protection solution. Let’s dive in.

Data-protection Requirements for Kubernetes

Implementing data protection for Kubernetes with a solution designed for traditional applications is complex. The distributed nature of Kubernetes clusters and the need to protect data at multiple abstraction layers—such as underlying storage, storage resources (e.g., secrets, deployments, and PVCs), pods, nodes, namespaces, and entire clusters—makes all the difference.

A data-protection strategy designed for Kubernetes must be:

- Container-granular
- Application-consistent
- Kubernetes-aware
- Namespace-aware
- Multicloud

Container-granular Data Protection

Traditional data-protection methods are machine-focused, which means they protect the machine itself (e.g., the virtual machine). In doing so, they protect the app. This approach works well if the application runs on a single host. But it fails in Kubernetes where applications are normally deployed in multiple containers spanning multiple nodes in a cluster.

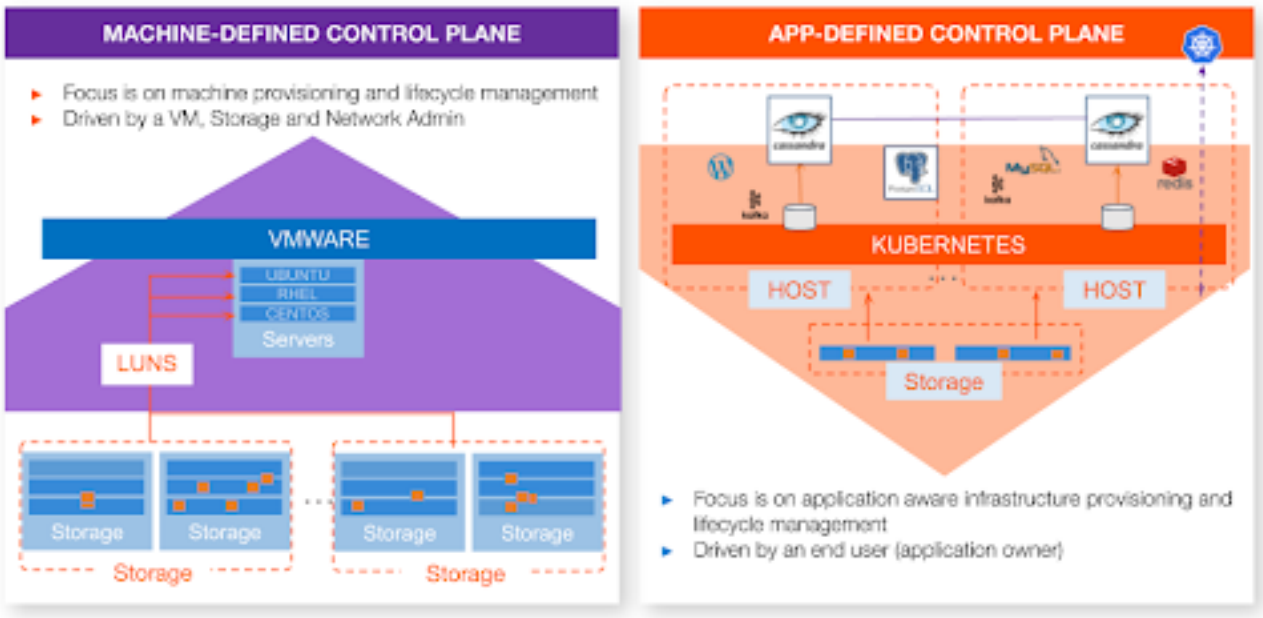


Figure 1: Kubernetes backups require an app-defined control plane.

You need the ability to target specific containers to protect applications on Kubernetes. Figure 3 shows a Kubernetes cluster of three nodes, each running a separate MySQL instance and one shard of a three-node Cassandra database. If you used the VM backup approach to back up Cassandra, the snapshot would also contain the MySQL data because Cassandra and MySQL run on the same host. Likewise, if you wanted to back up MySQL, the backup would include a third of the Cassandra data. The reason: machine-based backup approach lacks container-granularity.

Figure 2: Kubernetes backup solutions require container-granularity.

Application-consistent Data Protection

Because many NoSQL databases and other modern data-analytics services are made up of multiple shards, taking a backup requires making copies of multiple volumes at the same time. This is known as an application-consistent backup. Application-consistent backups use domain-specific knowledge of the application to capture its state in a consistent way. For example, an application-aware backup tool for Cassandra can flush all pending write operations to disk and take a backup of all nodes in a Cassandra ring simultaneously. Snapshot procedures often differ across databases (e.g., Cassandra is different from Kafka, which is different from Elasticsearch), and your backup tool should be aware of these differences; otherwise, you risk data corruption. Since these distributed applications make up the bulk of apps running on Kubernetes, you need a data-protection solution capable of taking both application-aware and application-consistent backups.

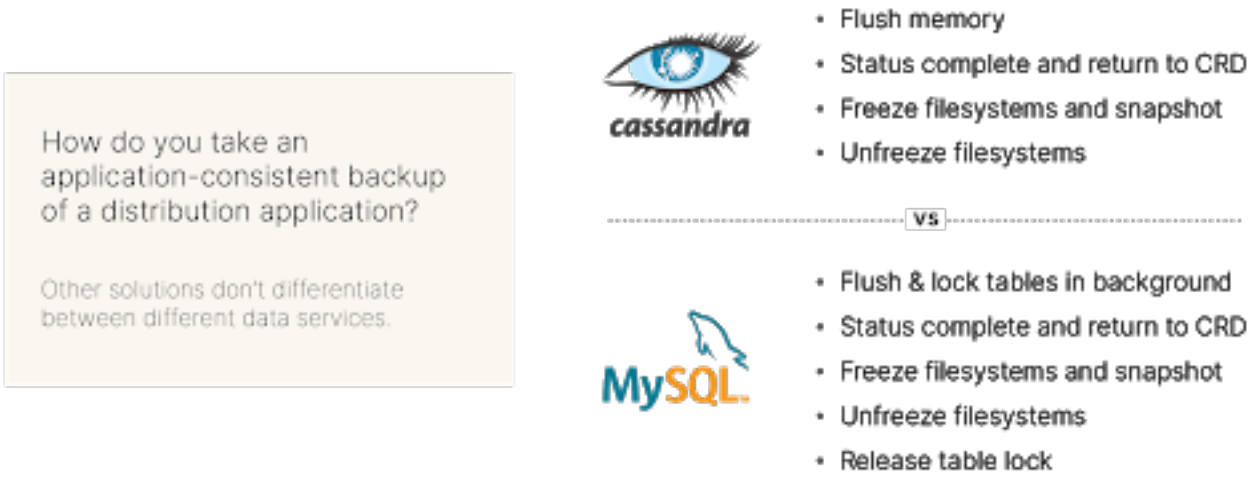


Figure 3: Each Kubernetes application must be backed up with application-specific rules.

Kubernetes-aware Data Protection

Kubernetes consists of many abstractions that wrap applications and their data and provide interfaces for container-orchestration services. For example, the underlying storage is provided to pods via Kubernetes persistent volumes (PVs), which enable you to allocate a specific amount of storage to an app and configure write/read access permissions, I/O limits, storage security, and more. Similarly, many other objects—such as secrets, service accounts, and deployments—control how containers within a pod communicate and how various microservices access the data.

Traditional data-protection tools don’t know how to interact with these abstractions to make true application backups that include Kubernetes objects, application configuration, and data. However, if you don’t back up these objects, it can increase the recovery time and lead to application crashes and hard-to-detect errors.

Figure 4: Kubernetes backups must include configuration, container image version and volumes.

Kubernetes Namespace-aware Data Protection

Backing up a single Kubernetes application requires container-granularity, application-awareness, and Kubernetes-awareness. But what if you want to back up 100 individual applications? You'll also need Kubernetes namespace-awareness. Think of Kubernetes namespaces as logical partitions akin to logical disks that allow you to split a single Kubernetes cluster into homogenous regions with shared resources, app groupings, and access permissions. Two important use cases for Kubernetes namespaces are distributing resources among IT units and grouping applications that constitute some part of the stack (e.g., front-end apps vs. back-end apps).

Consequently, while a DevOps team might be concerned with backing up only a single app, Kubernetes namespace admins may want to back up *all* applications running in a particular namespace at a single time. Doing this manually is hard because there are simply too many pods to back up. The result is a large number of manual operations and time-consuming ETL operations.

Unfortunately, traditional backup tools don't understand Kubernetes API and can't back up at the namespace level, leaving the only options as manual or machine-granular backups. Creating a namespace-aware backup script to fill this gap requires deep knowledge of Kubernetes, which most companies lack. Thus, you need a backup solution built with Kubernetes namespace-awareness in mind as well as the ability to back up individual Kubernetes applications.

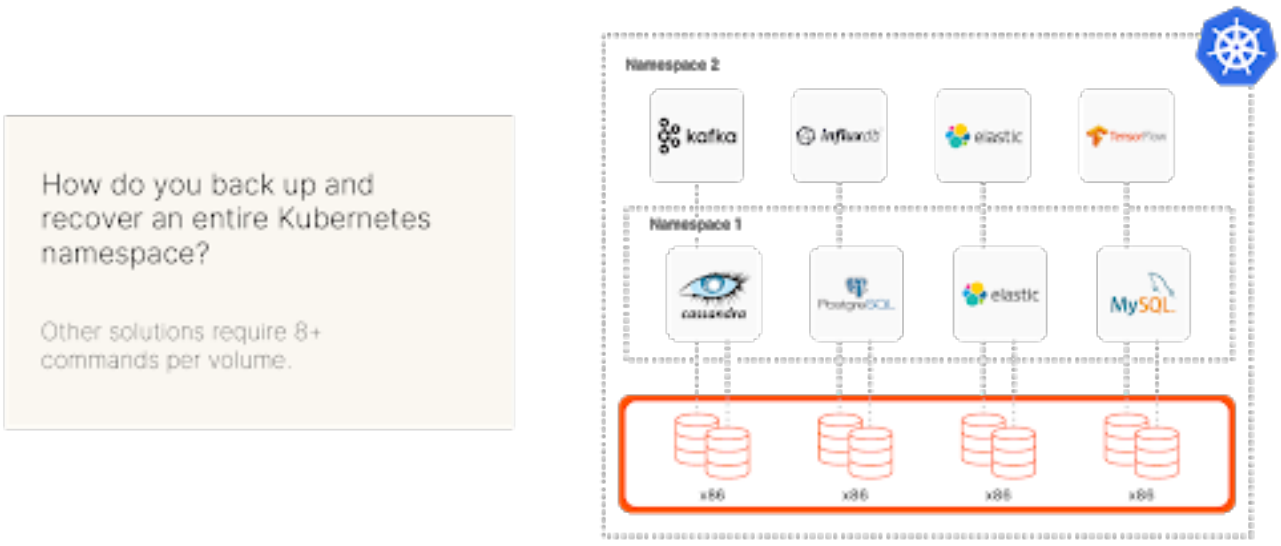


Figure 5: Kubernetes backup solutions must be able to take advantage of namespaces.

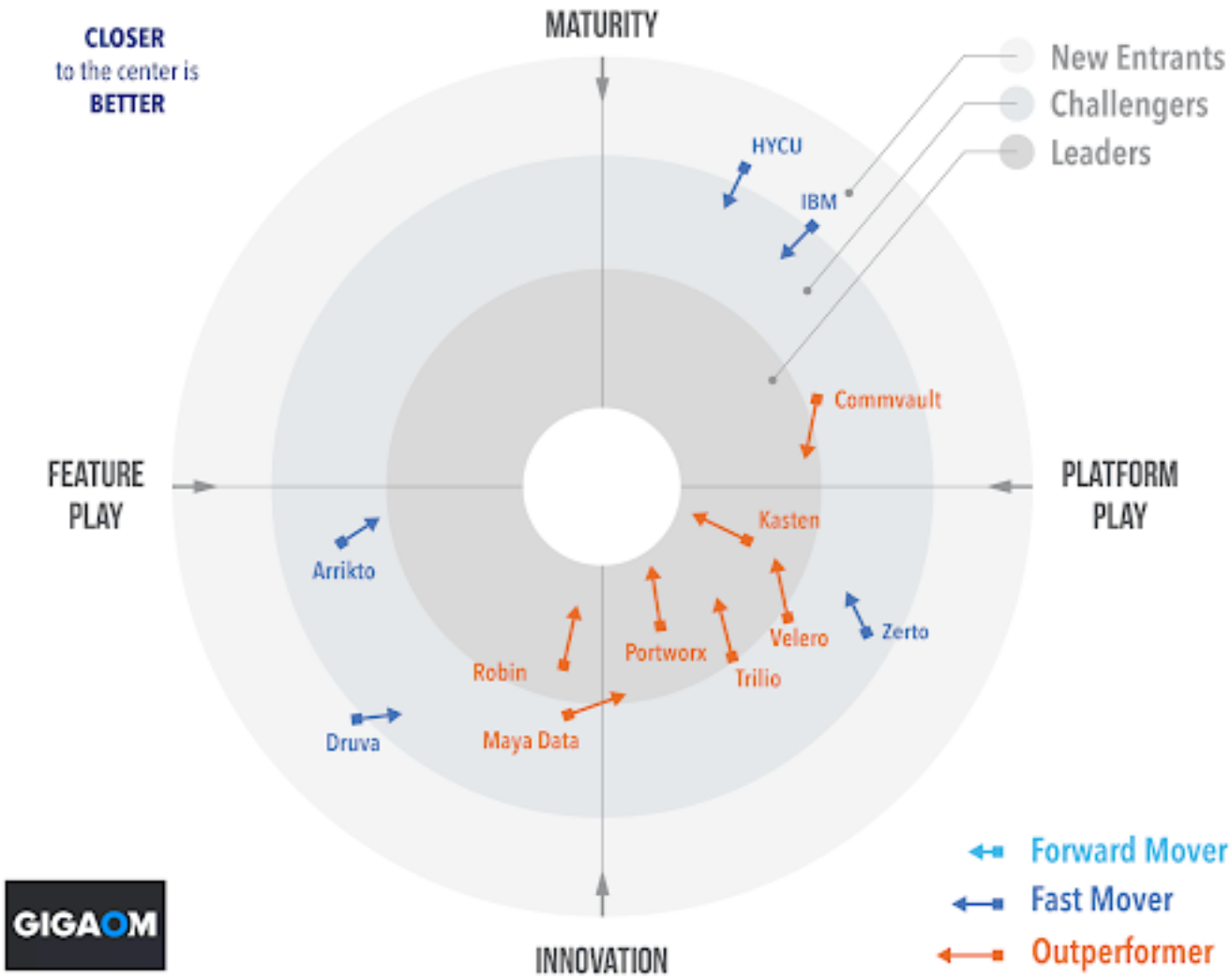
Multi-cloud Data Protection

Moving data and replicating storage across clouds is hard. However, containerized applications and microservices are often deployed across clusters and data centers located in public and private clouds. As a result, many companies require bidirectional data-protection support to backup and restore apps and data across all their clouds, both public and private.

To summarize, achieving near-instant recovery with no loss of data is hard for traditional applications and backup solutions. It's even harder for multi-container applications running across different environments. As a result, when you rely on traditional backups for your Kubernetes applications, your apps—and your business—are left exposed. Backing up entire machines leads to slow and expensive ETL procedures when you restore, causing data loss, longer downtimes, and eventually a failure to meet your business mission.

How to Protect Your Kubernetes Applications

Luckily, just because you can't protect your Kubernetes applications with a traditional solution, doesn't mean the task is impossible. Portworx by Pure Storage® was recently named a leader in Kubernetes data protection by GigaOm.



GIGAOM RADAR FOR KUBERNETES DATA PROTECTION