portworx

EXECUTIVE BRIEF

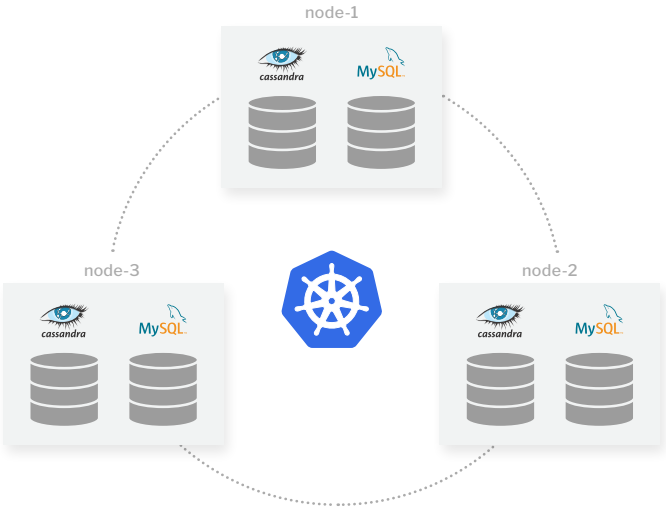
The Five Traits of Effective Disaster Recovery for Kubernetes

Disaster recovery is an essential capability for most enterprise applications.

In the pre-Kubernetes, pre-container world, backup and recovery solutions were generally implemented at the virtual machine (VM) level. This system works for traditional applications, when an application runs on a single VM. But when applications are containerized and managed with an orchestrator like Kubernetes, this system falls apart. Effective disaster recovery for Kubernetes must be designed for containerized architectures and natively understand the way Kubernetes functions.

Introduction

Traditional, VM-based backup and recovery solutions use snapshots that collect both too much and not enough information to be useful for a containerized application. This may seem paradoxical, but it is not. We say it's too much, because any particular VM will contain data from several applications. If you are trying to back up App 1 by taking a snapshot of a VM, you'll also get data from other applications. At the same time, it's not enough: App 1 will likely store data on other VMs as well, and that data would not be captured by a snapshot of a single VM.



Modern applications with distributed architectures need disaster recovery solutions that are able to locate all the relevant data and configuration information for a particular app and organize it in a way that allows the application to recover with zero recovery point objective (RPO) and near-zero recovery time objective (RTO).

An effective disaster recovery solution for Kubernetes needs to be:

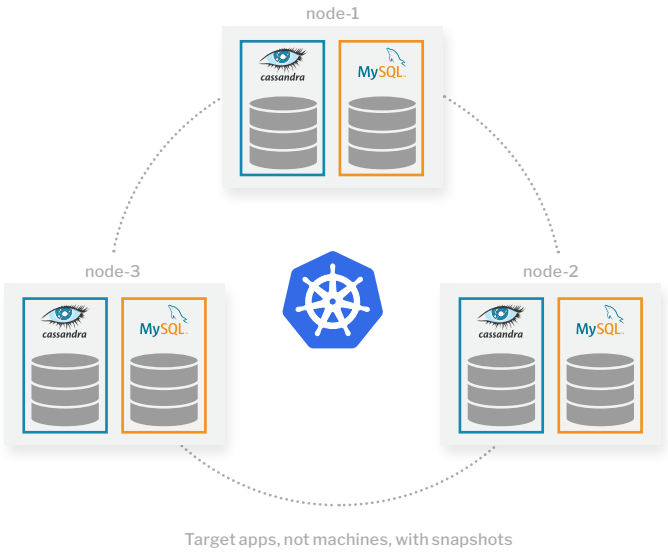
- Container Granular
- Kubernetes Namespace Aware
- Application Consistent
- Able to Backup Data and Configurations
- Multicloud and Hybrid Cloud Optimized

Unless a disaster recovery solution meets all five criteria, it won't be enough to ensure data-rich applications running on Kubernetes are able to meet service level agreements (SLAs) or legal requirements related to disaster recovery.

Let's discuss why each of these is important.

Container-Granular

A container-granular disaster recovery solution means that instead of backing up an entire VM or server, users can back up specific pods or groups of pods. This allows users to zero in on the application that needs to be backed up and snapshot only the containers that belong to that application.



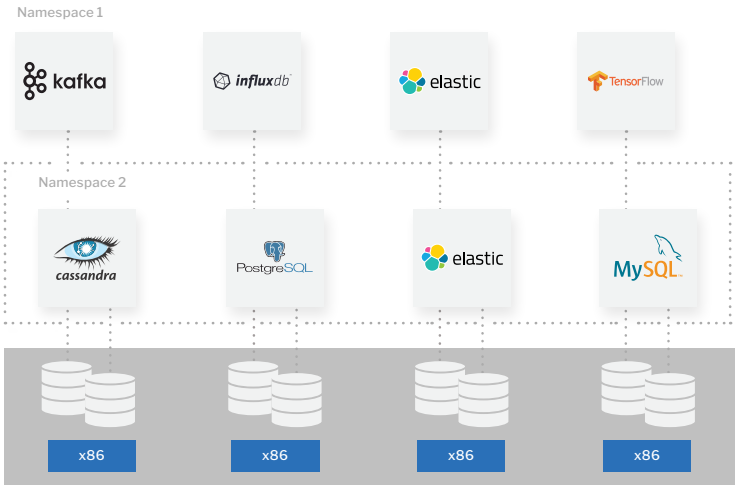
Let's say you have a three-node Kubernetes cluster with one three-node Cassandra ring and three one-node PostgreSQL databases, spread over three virtual machines. With a traditional disaster recovery solution, the only way to back up the cluster would be to back up the three virtual machines. This would lead to increased complications from extract, transform and load procedures, increased storage costs and increased RTO. The only way to back up enough data would be to back up way more than necessary.

With a container-granular approach, it's possible to back up just one single PostgreSQL database, or just the three-node Cassandra ring, on all three VMs, without picking up anything else.

Kubernetes Namespace Aware

Traditional backup and recovery solutions do not speak Kubernetes' language and are not designed to work in a way that makes sense in the Kubernetes ecosystem.

Namespaces in Kubernetes generally run several related applications. A frequent pattern in enterprise Kubernetes deployments is to have, for example, all of a company divisions' applications running on a single namespace. Given that use case, it's often necessary to back up all of the applications in a Kubernetes namespace together.

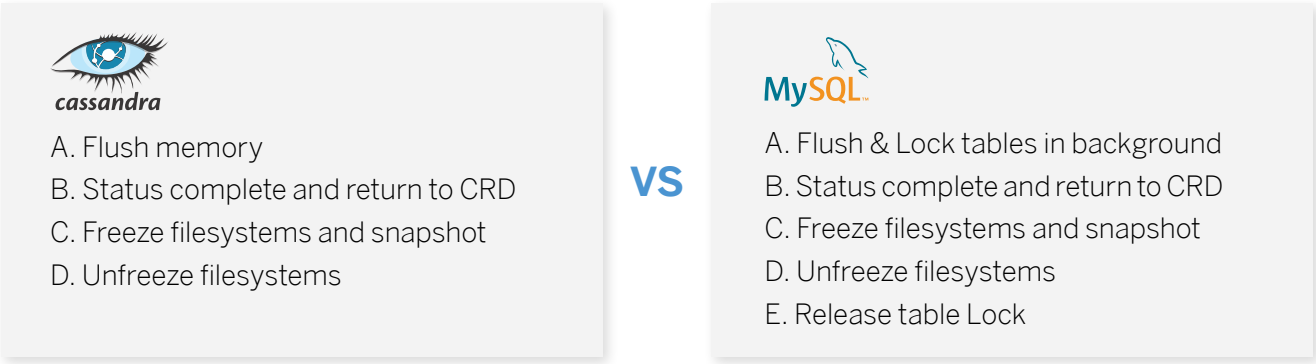


Like individual applications, however, namespaces are distributed over many virtual machines. Each virtual machine might also have pods from several different namespaces. Without a namespace-aware disaster recovery solution, full backups would require backing up and storing far more data than necessary. Even with this over-blown backup strategy, it would be difficult to restore the entire namespace in case of failure, leading to high RTO.

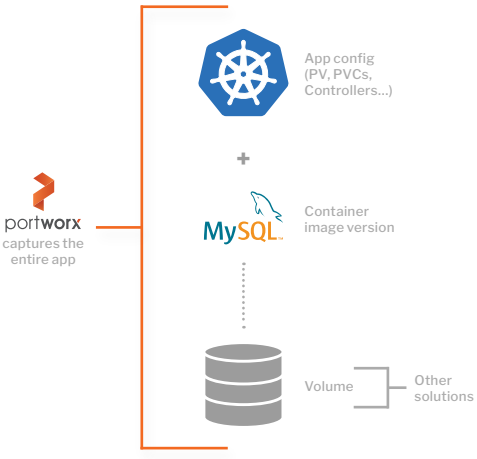
Application Consistent

Even if you wanted to solve the problems outlined above by simply backing up all the VMs in the system, it would be difficult to avoid data corruption with a traditional disaster recovery solution. To successfully back up a distributed application, without risk of data corruption, all pods in an application need to be locked while the snapshot is in progress. VM-based snapshots and serial snapshots can't achieve this, because they can't lock an entire application, spread over several VMs, and execute an application-consistent snapshot.

Successful snapshots that minimize the risk for data corruption have to be application consistent and designed



for distributed architectures. This means the ability to lock all pods that belong to an application and to execute the snapshot simultaneously.



Data and Configuration Backups

The goal of a disaster recovery system is not just preventing data loss—it is also to keep keep RTO low. You need the application up and running again as soon as possible.

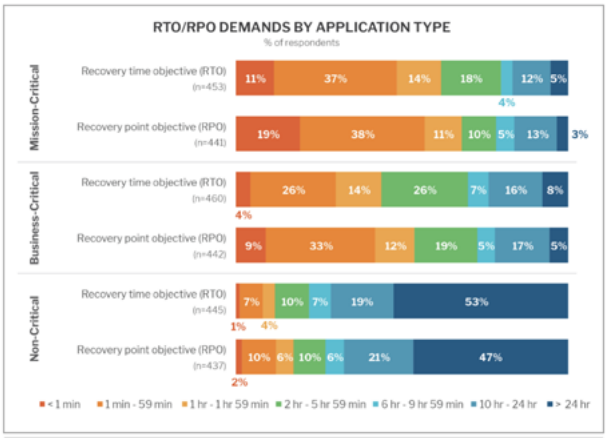
This requires backing up both application data and configuration information. If configuration information isn't included with the backup, the application will have to be rebuilt in place—a slow, manual and potentially error-prone process. If only configurations are saved, however, you could lose all your data.

A true enterprise-class disaster recovery system for Kubernetes will include both data and configuration backups, allowing you to redeploy the application after a failure in one or two commands.

Optimized for Multicloud and Hybrid Cloud

The vast majority of enterprises have applications running in at least two environments. This could mean multiple on-premises data centers or multiple Amazon Web Services (AWS) regions. In the context of disaster recovery, it's common to have one data center be the primary site and to have a second backup site. There are also, however, many companies that use a combination of the public clouds and on-premises data centers to run applications and to meet their business requirements. In most cases, businesses choose the best architecture based on their RPO and RTO requirements.

Figure 1: RTO/RPO demands by application type



Source: 451 Research, Voice of the Enterprise: Storage, Workloads and Key Projects 2019

It's crucial for disaster recovery solutions to work with these architectures to support varying levels of RPO and RTO. An effective disaster recovery solution should be able to provide both synchronous and asynchronous data replication, depending on the latency between the active and the backup cluster.

For mission-critical apps

48%

demand RTO < 1 hr

57%

demand RPO < 1 hr

Synchronous replication, which allows an RTO and RPO of zero, is possible when the round-trip latency between active and backup sites is generally under 10 milliseconds—often, when the active and backup clusters are running in data centers in the same metropolitan area.

In some cases, businesses need to maintain geographic distance between the active and backup sites. In that case, RTO can still be at or near zero, but the increased latency means that data can't be synchronously replicated without massive performance problems. If an RPO of fifteen minutes or an hour is acceptable for the particular application, however, this should be one of the disaster recovery options.

An enterprise-class disaster recovery solution for Kubernetes should give users the option of either synchronous or asynchronous data replication that works with a multi-cloud or hybrid cloud architecture. This gives users the ability to select the disaster recover option that works for their data center architecture as well as their business requirements.

Conclusion

As enterprises move mission-critical applications to Kubernetes, it's essential that they rethink disaster recovery. It's entirely possible to meet SLAs related to disaster recovery while running an application on Kubernetes, but it requires taking an approach that is designed for Kubernetes and natively understands how Kubernetes works. Traditional, VM-based disaster recovery solutions don't translate to a cloud-first, containerized application and can't provide the disaster recovery protection that enterprises need for their mission-critical, data-rich applications.

The Portworx Enterprise Storage Platform was purpose-built for containers and Kubernetes. It makes zero-RPO and near-zero RTO disaster recovery possible for applications running on Kubernetes, with container-granular, namespace-aware, application consistent disaster recovery. Failovers can be completely automated, keeping RTO as low as possible.

Curious how the Portworx Enterprise Storage Platform can provide enterprise-class disaster recovery that understands how Kubernetes works? Schedule a demo today.