Essential Capabilities of Data Protection on Kubernetes

portworx
by Pure Storage

1. INTRODUCTION 3

2. DATA PROTECTION REQUIREMENTS FOR KUBERNETES 6

2.1 Container Granularity 6
2.2 Application-consistent data protection 7
2.3 Kubernetes-Aware data protection 8
2.4 Kubernetes Namespace-Aware Data Protection 9
2.5 Multi-Cloud 9

3. A FRAMEWORK FOR KUBERNETES DATA PROTECTION 10

3.1 Data Availability Within a Single Data Center 11
3.2 Backup and Recovery 12
3.3 Disaster Recovery Across Data Centers 12
3.4 Creating a Data Protection Policy for Individual Apps 12

4. DATA PROTECTION STRATEGY FOR KUBERNETES WITH PORTWORX 14

4.1 Local HA 14
4.2 Disaster Recovery with Portworx 15
4.2.1 Synchronous Cross-Datacenter DR 16
4.2.2 Asynchronous Cross-Datacenter Disaster Recovery with Portworx 17
4.2.3 PX-Backup: Backup and Recovery Tool for Kubernetes 18
Application-Aware Backups 19
Kubernetes-Aware Backups 19
Multi-Pod Backups 19
Kubernetes Namespace-Aware Backups 20
Multi-Cluster Support 20
Backup Audit 21
4.2.4 Other Backup and Snapshot Tools 21

5. CONCLUSION 22

1. INTRODUCTION

The importance of data protection in the enterprise is greater than ever. Data protection can refer broadly to concepts such as high availability, business continuity, backup and recovery, and disaster recovery. Whatever you call it, all enterprises must implement, test, and maintain strategies for data protection in order to avoid risking their reputation and revenue.

As data becomes more important in customer experience and business processes, there is also little tolerance for downtime that makes it impossible to access critical information. According to the Uptime Institute's study, 41% of outages cost over $1 million, which indicates the increasing dependence of enterprises on the Highly Available (HA) data infrastructure.

Consequently, a viable data protection strategy has to ensure that downtime is rare and that applications and their data can be restored quickly after downtime or data loss. Also, data protection strategies have to ensure data privacy and regulatory compliance (e.g., GDPR and CCPA), which become very important as more users expose their critical data online.

Notwithstanding pressing data protection demands, most companies struggle to protect their data. Traditional data protection methods that work well for applications running on a single host do not scale well in the distributed or multi-datacenter environment such as those common in Kubernetes environments. Traditional business continuity and disaster recovery (BCDR) approaches simply do not live up to the Recovery Point Objective (RPO) and Recovery Time Objective (RTO) requirements for most mission-critical applications running on Kubernetes.

According to 451 Research, most companies have high expectations regarding their RPO and RTO targets. Almost half (48%) of organizations have RTOs for mission-critical applications of less than an hour, and this number is 30% for business-critical applications. Similarly, 57% of organizations have RPO demands in the sub-hour timeframe for mission-critical apps and data. While mission-critical applications have predictably high recovery targets, many companies expect low RPO and RTO for non-critical applications, as well. For example, half of the customers interviewed by the 451 Research have RTO expectations of less than a day, even for less-critical apps and data.

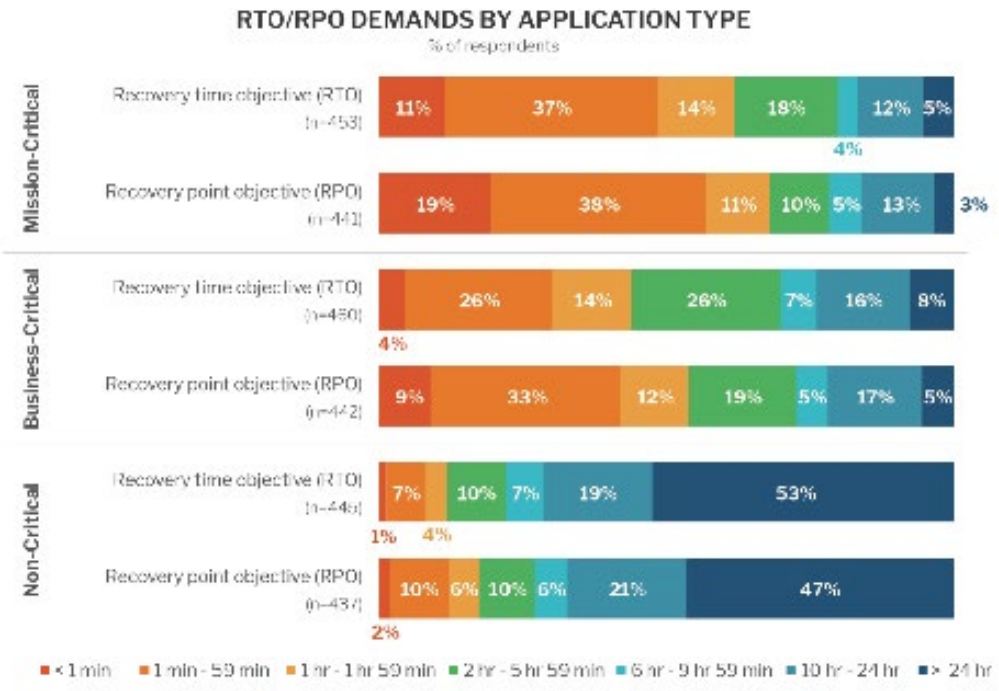


Table 1: RTO/RPO Demands by Application Type (Source: 451 Research: "Voice of Enterprise: Storage, Workloads, and Key Projects", 2019)

These expectations are in sharp contrast with reality, though. For example, according to the Uptime Institute, 50% of outages take 4 hours to recover, and 30% take over 5 hours to fix. Given the fact that most mission-critical applications require zero RPO and RTO of less than 15 minutes, this situation is not tolerable and requires an immediate response.

The data suggests that the actual state of data protection does not live up to IT professionals' expectations. As it turns out, daily backups and snapshots, the cornerstone of traditional BCDR approach, are not good enough, and organizations are pressed to implement new data protection strategies to meet their stricter data protection requirements.

> "Without container-granular backup options for Kubernetes, enterprises are exposed to **data loss, downtime and lost customer loyalty.** "
>
> - Enrico Signoretti, GigaOm

These problems are particularly acute for applications built to run on Kubernetes. The emergence of new deployment and production patterns for modern applications like those running on Kubernetes are incompatible with older data protection approaches. The reasons are multifold:

- **Distributed applications and distributed compute environments can't rely on machine-focused data protection.** Traditional data protection strategies focus on machines, but companies rely more and more on distributed applications that run across servers to serve their growing consumer base. These applications run in coordinated computer clusters, and a viable BCDR framework for these applications requires multi-node backup tools and a datacenter disaster recovery strategy.

- **Containers and Microservices create challenges for traditional data protection approaches.** More applications are developed and deployed as loosely coupled microservices and containers communicating across nodes. New backup and disaster recovery frameworks for these kinds of applications should be container-granular and be able to speak with containers and container orchestration abstractions.
- **Data expanse is immense.** Data is produced by more sources (e.g., IoT, cars, mobile devices) and transmitted across diverse environments, including multi-cloud, edge networks, and other environments. Backing up, aggregating, and managing data distributed across these environments efficiently is crucial for the effectiveness of modern analytics and monitoring pipelines that drive business intelligence.
- **Data protection needs to be flexible on a per application basis.** Different data protection policies need to be developed for different types of data and applications depending on their business impact and criticality.

These challenges cannot be addressed by the addition of new features to traditional backup platforms. What we need is a new Kubernetes *data management and protection system* that is compatible with new application deployment patterns, data sources, and data distribution patterns. These systems should be able to adapt the traditional data protection features to new realities, such as:

- Protection against data loss in multi-cluster and multi-datacenter contexts.
- Backup and recovery for multi-node and multi-container applications.
- Backup and recovery for entire applications, not just data.
- Data protection at the Kubernetes pod, tag, or namespace level.
- Enabling authentication and authorization for distributed data layers embedded into container orchestration systems.
- Encrypting distributed storage provided via a storage virtualization layer.
- Effective multi-datacenter backup and disaster recovery integrated into the container orchestration environment.
- Providing a backup/recovery model that works with GitOps/DevOps.

In this paper, we address these challenges focusing on the development of efficient data protection strategies for Kubernetes.

2. DATA PROTECTION REQUIREMENTS FOR KUBERNETES

Implementing data protection for Kubernetes is complex due to the distributed nature of Kubernetes clusters and the need to protect data at multiple abstraction layers, such as underlying storage, storage resources (e.g., PVC), pods, nodes, namespaces, and the entire clusters. In general, a data protection strategy for Kubernetes should satisfy the following requirements. It should be:

1. Container-granular
2. Application-consistent
3. Kubernetes-aware
4. Namespace-aware
5. Multi-cloud

Let's discuss these requirements in more detail.

2.1 Container Granularity

Traditional data protection methods are machine-focused, which means we protect the machine itself (e.g., virtual machine) and, in doing so, 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.

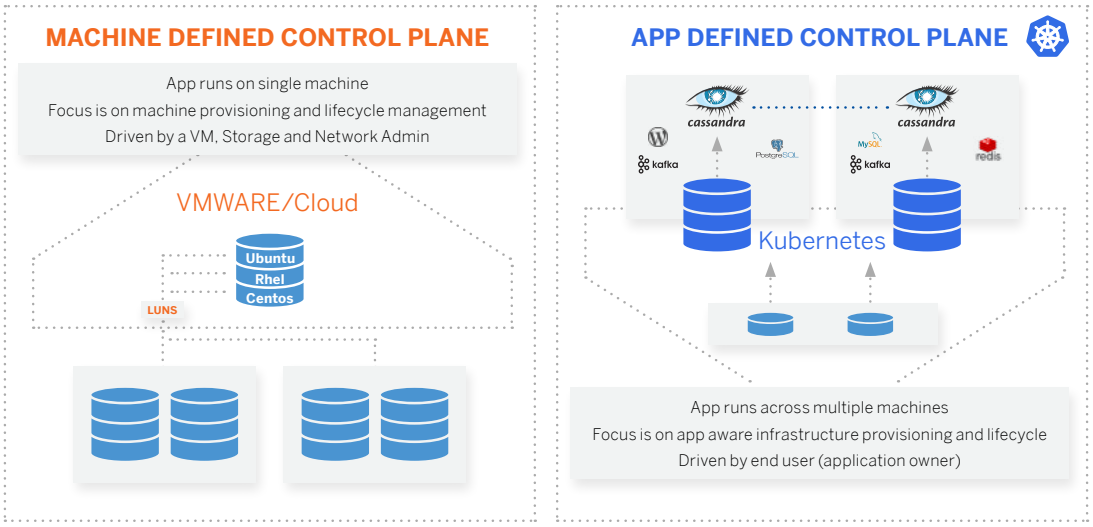


Image 1: Traditional backup assumed a machine-defined control plane, but Kubernetes uses an app-defined control plane

To protect applications on Kubernetes, you should be able to target specific containers. Take a look at the image below, for example. Here we see a Kubernetes cluster of three nodes, each running a separate MySQL instance and one Cassandra node. Note that Cassandra nodes participate in the same Cassandra ring. If we were to use the VM backup approach to back up Cassandra, we would end up with the snapshot that also contains the MySQL data. We don't achieve any granularity with this approach. Instead, if we had a backup solution tightly integrated with container runtime and Kubernetes abstractions like StatefulSets, we could create a snapshot specifically targeting the Cassandra StatefulSet. This backup solution could also leverage built-in replication of these abstractions as well as extra features for storage replication at the cluster level.

How do you back up 1 three-node Cassandra ring OR 3 one-node MySQL databases?

Other solutions use machine-based snapshots that fail in both instances.

Portworx backups are always container-granular.

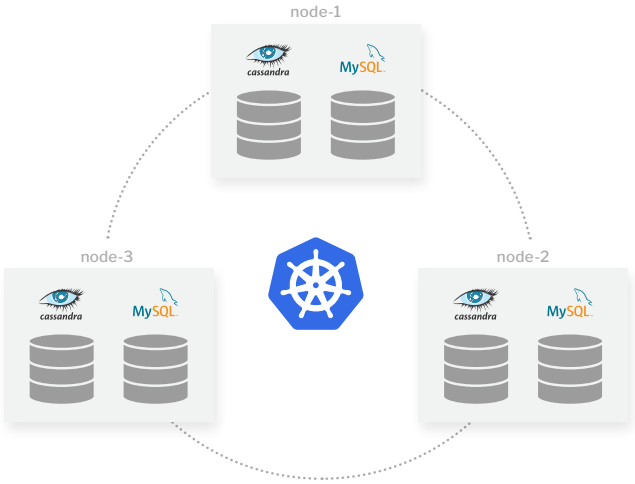


Image 2: Container Granularity with Portworx

2.2 Application-Consistent Data Protection

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 consistent backups.

How do you take an application-consistent backup of a distribution application?

Other solutions don't differentiate between different data services.

Portworx backups understand the differences between apps to provide true app-consistency.

- Flush memory
- Status complete and return to CRD
- Freeze filesystems and snapshot
- Unfreeze filesystems

vs

- Flush & lock tables in background
- Status complete and return to CRD
- Freeze filesystems and snapshot
- Unfreeze filesystems
- Release table lock

Image 3: Application-Consistent Backups

2.3 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). They allow allocating a specific amount of storage to an app and configuring write/read access permissions, I/O limits, storage security, etc. Similarly, there are many other objects, such as Secrets, Service Accounts, and Jobs, that control how containers within a Pod communicate and how the data is accessed by various microservices.

Traditional data protection tools do not know how to interact with these abstractions to make true application backups which include Kubernetes objects, application configuration, and data. However, backing up an application without backing up these objects can increase the recovery time and lead to application crashes and hard-to-detect errors.

How do you backup entire apps to ensure minimal to no data loss and low RTO?

Other solutions only focus on data.

PX-DR captures data, application configuration, and Kubernetes objects, ensuring easy recovery.

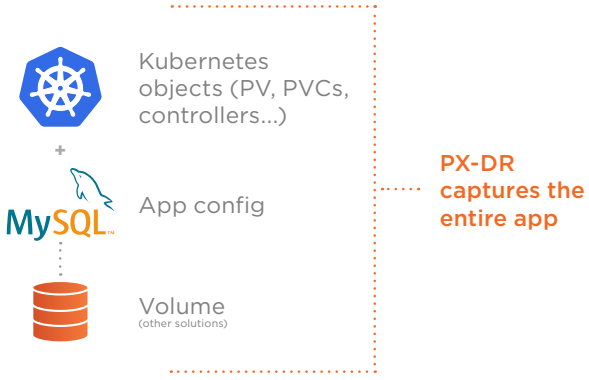


Image 4: Kubernetes-Aware Backups

For traditional applications, backing up configuration is equivalent to backing up the entire VM or configuration files within a host. The situation with Kubernetes is different. In Kubernetes, many application components may live within one or many Kuberentes nodes. All the components of the application—including Kubernetes resources, volumes, and configuration settings—are defined as YAML and tightly coupled. The configuration provided to Kubernetes via objects like ConfigMaps are interpreted by the containers at runtime and applications managed by this runtime. Managing these different layers of abstraction requires a Kubernetes-aware approach that allows data protection, backup and recovery and DR at the application granular level.

2.4 Kubernetes Namespace-Aware Data Protection

You may think of Kubernetes namespaces as logical partitions akin to logical disks that allow splitting 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., frontend apps vs. backend apps).

Consequently, 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 in order to perform a backup at the namespace level.

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

How do you backup and recover an entire Kubernetes namespace?

Other solutions require 8+ commands *per volume.*

Portworx backups can be applied to an entire namespace with a single command.

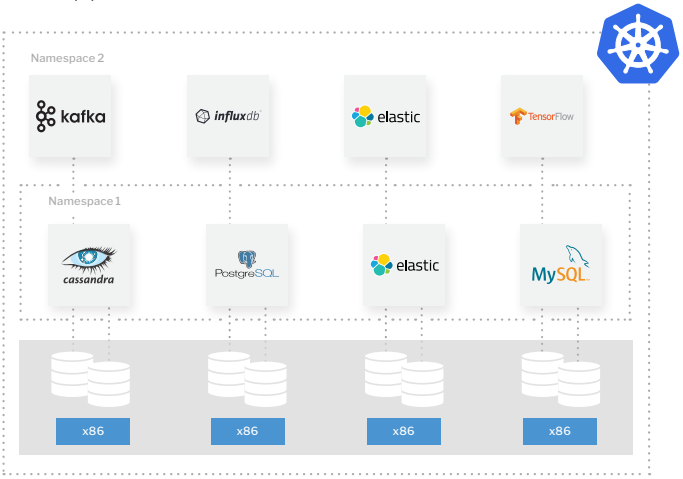


Image 5: Kubernetes Namespace-Aware Backups

2.5 Multi-Cloud

Moving data and replicating storage across clouds is hard. However, containerized applications and microservices are often deployed across clusters and datacenters located in public and private clouds. Thus, many companies require bidirectional data protection support to back up and restore apps and data across all of their clouds, both public and private.

To summarize, achieving near-instant recovery with no loss of data is hard for traditional applications and traditional backup solutions. It's even harder for multi-container applications running across different environments. The result of all this is that 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. This may lead to data loss, longer downtimes, and eventually a failure to meet your business mission. In what follows, we discuss how to implement a data protection strategy on Kubernetes that tackles the challenges we just discussed.

3. A FRAMEWORK FOR KUBERNETES DATA PROTECTION

A Kubernetes cluster is a complex entity that can span multiple nodes located in different cloud regions or even datacenters. This distributed architecture requires multi-layer data protection with built-in topology awareness, high availability requirements, and multi-cluster data management capabilities.

When developing a data protection strategy for Kubernetes, you should consider three layers of protection:

1. Data availability within a single data center
2. Backup and recovery within or across datacenters
3. Disaster recovery across data centers

You should also develop an individual approach for different types of applications depending on their business requirements and importance. In other words, each application running on your Kubernetes cluster should have an individualized data protection strategy. While this nuanced approach is impossible using traditional backup technologies that focus on machines, it is achievable with a Kubernetes-focused platform like Portworx.

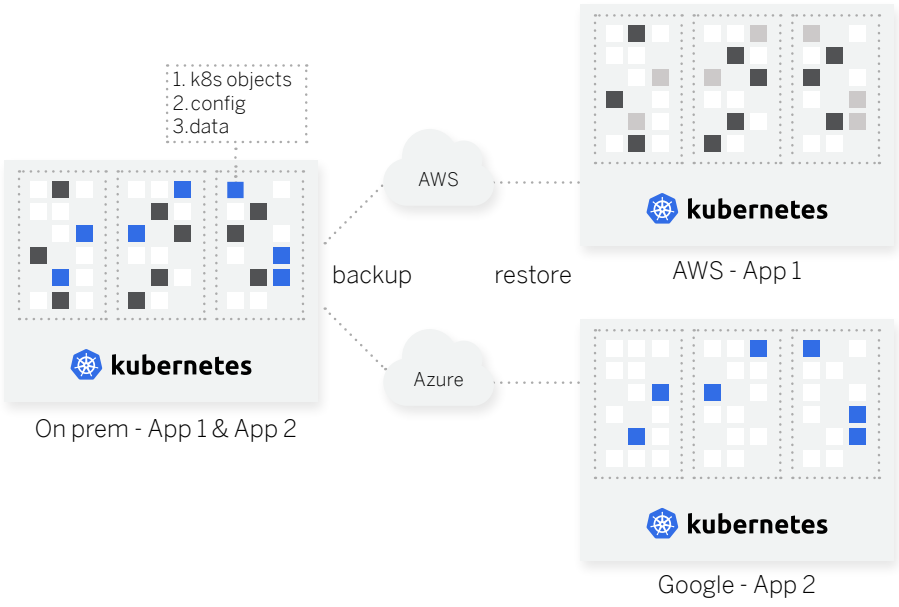


Image 6: App-Granular Backup and Restore in Kubernetes

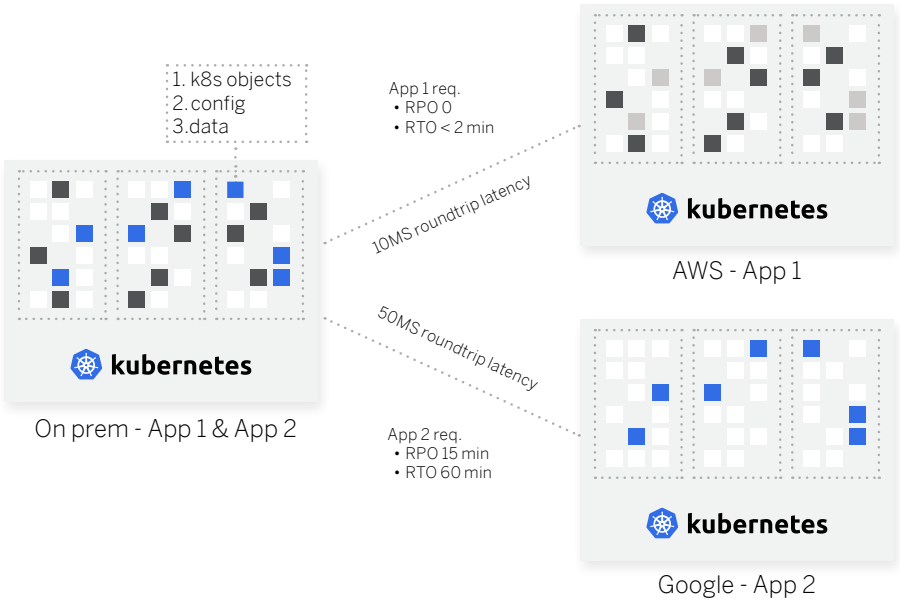


Image 7: App-Granular disaster recovery across environments

3.1 Data Availability Within a Single Data Center

Data protection of a stand-alone Kubernetes cluster boils down to ensuring the high availability of Kubernetes components and applications and cross-node replication of data.

The bare minimum requirement for the HA in Kubernetes is having at least 3 nodes for etcd quorum. However, for your Kubernetes cluster to be truly highly available, the HA should be ensured at other levels, as well. In particular, you should implement HA for:

- **Kubernetes components.** Kubernetes consists of several master and node components, including `kube-apiserver`, `etcd`, `kube-proxy`, `kubelet`, etc. You need to ensure that these components are highly available, too. Even if your cluster has two healthy nodes but `kube-apiserver` is down on all of them, the cluster won't be able to serve client requests and will run in a read-only mode.
- **Kubernetes applications.** HA for Kubernetes apps can be ensured with a proper replication strategy for StatefulSets and Deployments. You can also prepare for traffic spikes using Horizontal Pod Autoscaler (HPA) or a cluster-wide autoscaling tool.
- **Persistent Volume (PV) data.** HA for Persistent Volumes should be also enabled using cross-node replication. Persistent volumes should be evenly distributed across availability zones or data centers to ensure better fault tolerance and data redundancy.
- **Cloud components.** If you provide external access to your Kubernetes clusters via cloud load balancers, you should ensure their HA, as well. If you have one load balancer and it is down, you won't be able to provide external access to the cluster.

Also, it's important to ensure HA for data in your Kubernetes cluster. This can be achieved with storage replication, regular volume backups, storage-aware orchestration, and hyperconvergence of apps and data.

We will see how to achieve local HA for Kubernetes applications using the Portworx Storage Platform for Kubernetes in section 4.

3.2 Backup and Recovery

A viable data protection plan should include a backup strategy as a major component. Traditional backup tools primarily focus on backups of servers whereas for Kubernetes we need a multi-functional solution capable of backing up applications. For Kubernetes this means not just data but also Kubernetes resources (Persistent Volumes, Secrets) and configuration.

To achieve this, a backup tool for Kubernetes should have a built-in understanding of Kubernetes API and resources as well as domain-specific knowledge needed for application-consistent backups. Also, this tool should have a Kubernetes topology awareness to be able to back up not just individual pods but groups of pods, or namespaces and manage these backups across clusters and data centers. Building such a tool requires a deep rethinking of the backup strategy and aligning it with principles embedded in containerization, microservices, and Kubernetes.

We will see how to backup and recovery for Kubernetes applications using the Portworx Storage Platform for Kubernetes in section 4.

3.3 Disaster Recovery Across Data Centers

Local cluster High Availability can't protect against the downtime of the entire datacenter. And often even an application-centric backup and recovery program can't provide the RPO and RTO levels required for business continuity planning. However, a comprehensive data protection strategy should account for this scenario. To prepare for datacenter downtime, you should have an efficient cross-cluster application migration and disaster recovery program in place.

DR for Kubernetes involves having 'active' and 'standby' Kubernetes clusters located in different datacenters. An active cluster replicates its data and configuration to the standby cluster on a set timeline based on RPO and RTO requirements so it's up-to-date in case of a disaster. In the case of the Active cluster failure, the role of the active cluster is automatically overtaken by the standby cluster. The frequency of data and application configuration transfer between data centers could be every hour, every 15 minutes, or even synchronously, depending on the desired RPO and RTO. Since not every application in a cluster has the same business requirements, each app needs to have its own DR policy.

We will see how to achieve a flexible DR program for Kubernetes applications using the Portworx Storage Platform for Kubernetes in section 4.

3.4 Creating a Data Protection Policy for Individual Apps

Kubernetes clusters may consist of applications with different degrees of importance for your enterprise mission. We can divide these apps into three basic categories: mission-critical, business-critical, and non-critical applications. Having a data protection policy customized for different categories of apps is the way to spend your BCDR budget efficiently.

There is a model you can use to determine what level of data protection you need for each app. In this model, the level of data protection depends on the application's criticality, compliance impact, and revenue impact.

Apps with low direct revenue impact include Continuous Integration and Continuous Development (CICD) Pipelines, test environments, sandboxes, etc. For example, the CICD pipeline is important for the effective testing, development, and deployment of applications following the agile method; however, it does not immediately affect the customer experience and revenue flows. Therefore, its criticality level may be assessed as medium-low. While each app is different, apps with a similar level of criticality may be able to tolerate the RPO of 4 hours and RTO of 1 hour, as an example. As far as the data protection level is concerned, these applications can most likely do well with local HA as their data protection strategy. Additionally, some of them may require periodic backups 'just in case.'

Let's look at an app with a higher level of criticality.

Customer engagement applications belong to this second group. The compliance impact of these apps is high because they contain customer data, and revenue impact is medium. Often, these types of apps require more stringent levels of data protection with lower RPO and sometimes RTOs—for instance, 1 hour or less. As a result, customer engagement applications may require a local data protection strategy combined with cross-cluster migration and backup.

Apps with high criticality and business impact levels include transaction processing apps and other mission- and business-critical applications. Such applications can't tolerate any data loss (require zero RPO) and often have an RTO of max 5 minutes because of their high compliance and revenue impact. Mission-critical apps require a multi-datacenter protection strategy with strong HA for individual clusters in addition to backup and recovery focused on managing compliance impact.

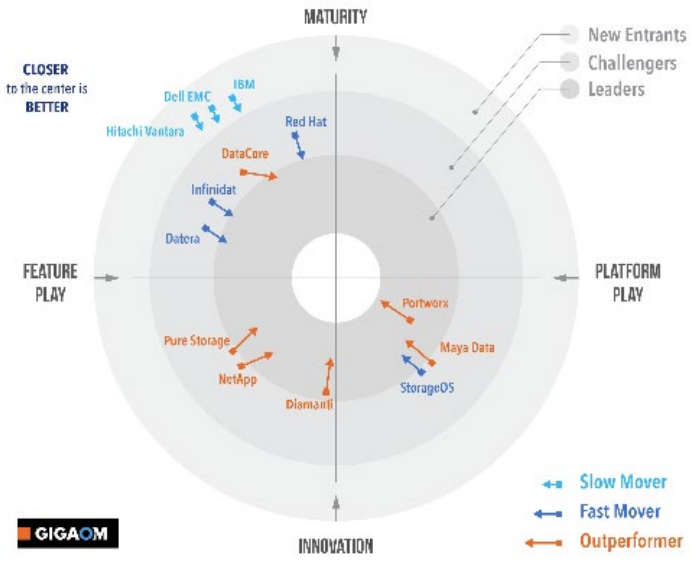
Sample App Type	Criticality	RPO	RTO	Data Protection Strategy
CICD Pipelines	Medium	4 hours	1 hour	Local HA and periodic backups "just in case"
Custom Engagement Apps	Medium-High	1 hour	1 hour	Local HA with regular backups to meet compliance needs
Transaction Processing Apps	High	0	5 minutes	Full backup and disaster recovery for multi-datacenter

Table 2: Data Protection Requirements for Different Categories of Apps

The above are simply examples of typical enterprise applications and the types of data protection that they often require. Your requirements for individual apps will likely differ. However, what will remain is that different apps require different data protection strategies, and you need a platform that allows you to apply data protection at an application-level of granularity. In what follows, we'll discuss how to implement this model for applications using built-in Kubernetes features and the Portworx Enterprise storage platform.

4. DATA PROTECTION STRATEGY FOR KUBERNETES WITH PORTWORX

FIGURE 1. GIGAOM RADAR DATA STORAGE FOR KUBERNETES



The Portworx Enterprise Storage platform is an end-to-end storage and data management solution for Kubernetes. Portworx provides such features as storage aggregation and on-demand provisioning, backup and recovery, storage security, monitoring, and more in distributed container environments. Portworx was recognized by GigaOm Research as the #1 Kubernetes data storage platform in its Radar report.

At the most basic level, Portworx works as a storage cluster that aggregates underlying storage capacity and pools it into a unified storage layer. Portworx automatically partitions the layer into different storage classes that can be provided to containers using Kubernetes persistent volumes. The Portworx storage layer is elastic and easily scalable. Instead of managing individual storage devices for each node in your cluster, you can leverage Portworx storage virtualization to create a unified storage pool.

On top of data aggregation, Portworx offers many data management services, including local HA, backup, and disaster recovery features. These features are implemented in several Portworx tools, including PX-DR, PX-Backup, 3DSnaps, PX-Migrate, PX-Secure, and volume snapshots, among others. Using the Portworx platform, you can develop different strategies for your apps and data depending on their RTO and RPO requirements, criticality, and impact.

With Portworx, you can ensure local HA, cross-cluster, and multi-datacenter disaster recovery as well as effective container-granular and app-consistent backups. Let's see how.

4.1 Local HA

Local High Availability ensures that data is always available to customers and applications even if the individual node running the app becomes inactive, unavailable, or fails. Local HA can be achieved with Portworx volume replication.

Portworx volumes can be created with a replication factor that specifies the number of data replicas to keep in the cluster. The Portworx engine evenly distributes these replicas across the fault domains (e.g., server racks on premises or Availability Zones in the cloud) to ensure that the data is present on a configurable number of nodes in the cluster.

All of the replicas are kept up-to-date using synchronous replication. Portworx replicates each write to any of the volumes across all replicas in the cluster. Consequently, if the node or application fails, the new application instance spun up on a new node can have instant access to a copy of its data.

Portworx is so effective in managing data replicas across the Kubernetes fault domains (racks, nodes) thanks to its storage-aware scheduling functionality implemented in STORK (Storage Orchestration Runtime for Kubernetes). STORK is tightly integrated with kube-scheduler and other Kubernetes controllers. If a Kubernetes scheduler is configured to use an extender, it makes two API calls before making a pod placement decision: Filter and Prioritize. The 'filter' request allows STORK to filter out nodes without the storage driver on them. This reduces the number of failed pod scheduling attempts and results in faster scheduling of applications on the node with their data.

Also, STORK uses a 'prioritize' request to single out nodes that host a replica of the app data. Portworx checks which persistent volume claims (PVCs) are used by the pod and then queries the storage driver for the location of the pod's data. STORK uses this information to rank various nodes as to which node would provide the best performance when accessing the persistent storage from that node.

As a result, STORK makes sure that the app has fast access to its data whenever the critical downtime or failure is detected. In this way, Portworx ensures high availability and fast failover for affected applications.

In addition to data and app hyperconvergence, STORK provides such features as failure-domain awareness, storage health monitoring, and snapshot-lifecycle features for stateful apps on Kubernetes.

In summary, these features enable local HA for your Kubernetes clusters. This is a good starting point for any workload and perhaps the only data protection capability you need to implement for low-priority workloads, such as development and testing pipelines.

4.2 Disaster Recovery with Portworx

Portworx Enterprise includes PX-DR, a package that implements cross-datacenter disaster recovery for containerized applications running in Kubernetes as well as other orchestration systems.

Cross datacenter Disaster Recovery (DR) is a data protection strategy that includes one or more datacenters that host a backup (standby) cluster for the active Kubernetes cluster. Having an up-to-date standby cluster effectively protects against the downtime scenarios where the entire Kubernetes cluster becomes inactive due to the datacenter outage. Cross-datacenter DR is the optimal choice for business-critical and mission-critical applications that have low or zero RPO and low RTO requirements.

The PX-DR is based on the container-granular approach to disaster recovery where containers and Kubernetes pods are treated as atomic units of backup. Instead of backing up everything that runs on a

virtual machine or bare metal server, Portworx gives users the ability to back up specific pods and groups of pods running on particular nodes. This means that one application can use PX-DR synchronous replication of data and configuration to achieve Zero RPO DR, while another can use asynchronous DR to achieve 15 minute or 1 hour RPO. At the same time, PX-DR users have an option to back up entire Kubernetes namespaces and clusters when needed.

PX-DR can operate in two modes: 1) synchronous cross-datacenter DR and 2) asynchronous cross-datacenter DR. The choice of a mode depends on your Kubernetes cluster deployment architecture.

In both modes, the cross-cluster synchronization process works as follows. First, PX-DR pairs two clusters using the generated pair key. These two clusters are active clusters (a currently operating cluster) and the standby cluster serving as a backup destination.

Once the clusters have been paired, PX-DR periodically backups application configuration, data, namespaces, and other objects indicated by the user to the standby cluster and manages the standby cluster's lifecycle.

Image 8: PX Disaster Recovery with Active and Standby Clusters

In both modes, the standby cluster does not only have all backups but all Kubernetes controllers and processes needed to supersede the active cluster if it becomes unavailable. All PVCs on the standby cluster also map to the same volumes and replicas in the active cluster.

4.2.1 Synchronous Cross-Datacenter DR

Synchronous cross-datacenter DR works when Portworx is installed as a single stretch cluster across multiple Kubernetes clusters spanning across a metropolitan area network (MAN) (for example, a campus network in which two data centers are located in close proximity). The requirement of this mode is for the Kubernetes nodes to be in the same cloud region (possibly different zones) and for both active and standby clusters to be in datacenters located within a 50-mile distance. Also, Portworx synchronous disaster recovery recommends roundtrip network latency between nodes to be lower than 10 milliseconds.

Low network latency allows an active cluster to continuously replicate data to the standby cluster and enables the fast failover of the entire Kubernetes cluster if the active cluster goes down. In particular, the synchronous DR mode has zero RPO and RTO of less than a minute. Also, the mode has a built-in fault-domain detection, so the backup replicas can be evenly distributed across the nodes of a standby cluster.

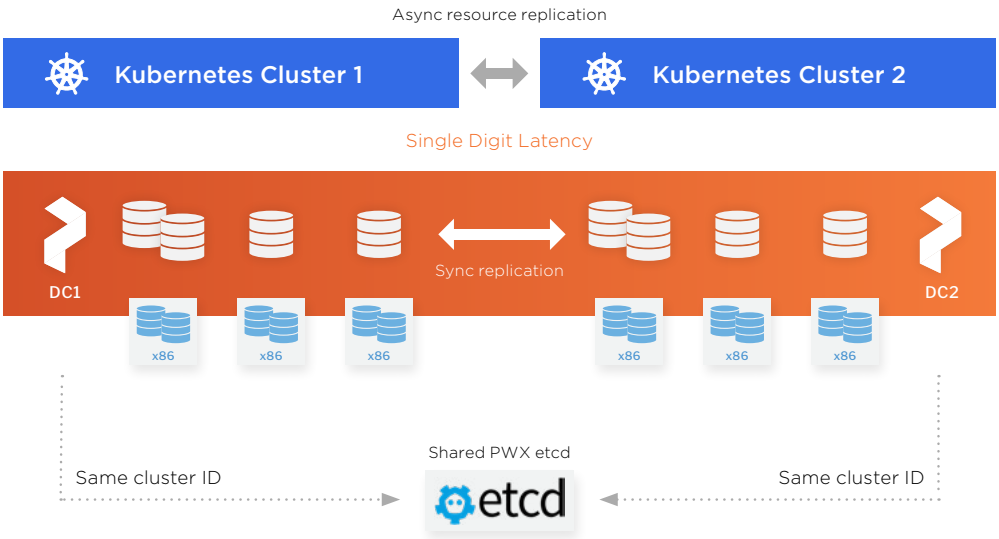


Image 9: Synchronous DR with Portworx

4.2.2 Asynchronous Cross-Datacenter Disaster Recovery with Portworx

In the asynchronous mode, Kubernetes clusters are located in different regions or datacenters and have high latency between them. To connect these datacenters, a separate Portworx cluster is installed in each Kubernetes cluster deployed in these datacenters. Incremental changes in Kubernetes applications and Portworx data are continuously sent to the standby cluster according to the migration schedule specified by the user. Even though active and standby clusters are eventually synchronized according to the schedule, due to the higher network latency, this mode will have an RPO of 15 minutes. The RTO of less than 60 seconds is ensured for this mode, as well.

Similarly to the synchronous mode, in the asynchronous mode the standby Kubernetes cluster has running controllers, configuration, and PVCs that map to local volumes, so it's ready to become active any time.

Also, users can schedule synchronization between the active and the standby clusters using migration schedule policies. They allow setting an interval for migration and specify objects and volumes covered by the schedule. You can also decide whether an application should start once it's migrated using the `startApplications` flag.

As with snapshots, users can specify `preexec` and `postexec` rules for migrated volumes. This may be useful when you migrate databases that require all pending write operations to be flushed to disk before making a migration. As part of the Asynchronous DR feature, Portworx can migrate PV, PVCs, Deployments, StatefulSets, ConfigMaps, Services, Secrets, and other important Kubernetes resources.

Image 10: Asynchronous DR with Portworx

4.2.3 PX-Backup: Backup and Recovery Tool for Kubernetes

Traditional backup and restore solutions are usually implemented at the virtual machine (VM) level. This approach does not scale well into the containerized environment where multiple pods related to different applications can run on a single node. Distribution of multiple containers/microservices over the cluster is the key architectural pattern that ensures cost savings and container orchestration, but, in effect, it makes traditional backup methods obsolete.

Another limitation of the traditional backup tools is that a Kubernetes pod encompasses not just data but many resources (Kubernetes manifests), configuration objects, container settings, persistent volumes, metadata, and more that are needed for the application to run efficiently in the Kubernetes environment. We need a way to perform a full, Kubernetes-consistent backup of applications on Kubernetes to secure fast and efficient failover of applications in the Kubernetes cluster.

So, to back up applications on Kubernetes efficiently we need a different, container-granular approach to backup and restore. Such an approach is offered by the PX-Backup, a component of Portworx Enterprise. Instead of backing up everything that runs on a single host, PX-Backup gives users the ability to back up specific pods and groups of pods in a fashion compatible with how containers and Kubernetes work. By offering container-granularity, we avoid costly and error-prone ETL procedures that would be required if we backed up all VMs in their entirety. By only backing up the specific applications desired, we minimize storage costs and keep recovery time objectives (RTO) low.

PX-Backup ships with the following features:

- Full application-consistent backups for Kubernetes resources
- Container data lifecycle management from the centralized and user-friendly interface
- Cataloging relevant backup metadata
- Auditing backup history
- PX-Backup supports backups for applications storing their data on both Portworx Enterprise as well as directly on cloud block storage from Azure, AWS, and Google Cloud managed via the Kubernetes CSI plugin.

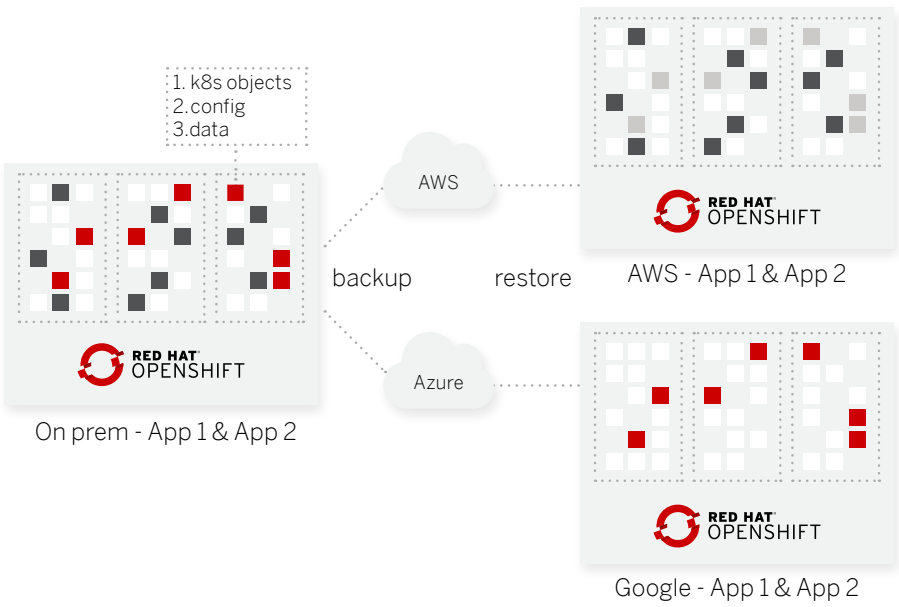


Image 11: App-Granular Backups with Portworx

Application-Aware Backups

PX-Backup is a toolkit that satisfies all major requirements for the efficient application backup policy on Kubernetes. With PX-Backup, you can ensure that your application state is consistent after the restore. In particular, you can specify pre- and post-backup rules to be executed before and after the backup is made. This is especially relevant in the case of databases where certain operations might be pending when taking the snapshot. For example, with PX-Backup, you can specify a pre-snapshot rule telling the database to flush all write operations to disk before making a snapshot. Correspondingly, you can specify database-specific actions such as compaction or data repair to ensure that the backup image is restored consistently.

Kubernetes-Aware Backups

Knowledge of Kubernetes resources and objects, such as Secrets, ConfigMaps, and Persistent Volumes, is baked into the PX-Backup process. Whenever you take a snapshot of your Kubernetes application with PX-Backup, it copies all configuration and resources that back up the app in a consistent way in a single command. This backup can be easily restored on Kubernetes without the need to re-create multiple Kubernetes objects and configurations. This leads to a much faster recovery of your apps on Kubernetes unattainable to most backup solutions. Your RTO objectives can be met with this approach with no pain.

Multi-Pod Backups

Kubernetes administrators may need to back up multiple applications at the same time. PX-Backup allows creating snapshots of multiple groups of Pods based on labels and other user-specified constraints. PX-Backup maintains the metadata about the backed-up pods so you can easily audit the process and restore the same pods to the same place.

Kubernetes Namespace-Aware Backups

This ability to backup multiple pods can be extended to entire namespaces. As we've already mentioned, namespaces are logical partitions of the Kubernetes clusters, which allow sharing cluster resources between applications and teams. For example, a developer team can use a 'Dev' namespace that has a resource quota of .4 CPU, 10 RAM, and 500GB of storage. This namespace can be used to run multiple applications related in some way. PX-Backup provides an option to back up the entire namespace, not just a single application. The tool will maintain the metadata of location where the backup was performed so you can easily restore applications to the same namespace.

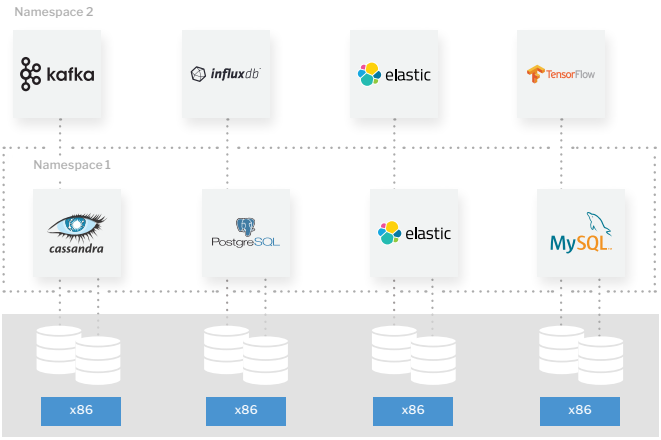


Image 12: Namespace-Aware Backups

Multi-Cluster Support

PX-Backup allows managing application backups for multiple clusters from one central location. For example, you might have application backups from the AWS cluster stored in S3 and application backups from Azure or Google Cloud. Whenever you want to restore the app, you can select a cluster it was backed up from, and PX-Backup uses the backup metadata to perform a restore. It's very convenient when you run several Kubernetes clusters on different clouds or on-premises. Having a centralized interface for managing backups for these clusters provides visibility into the source environments and allows managing the full lifecycle of your backups.

Also, PX-Backup is seamlessly integrated with cloud drives from GCP, AWS, and Azure, which allows making backups with cloud storage provided by these platforms and import backups from them.

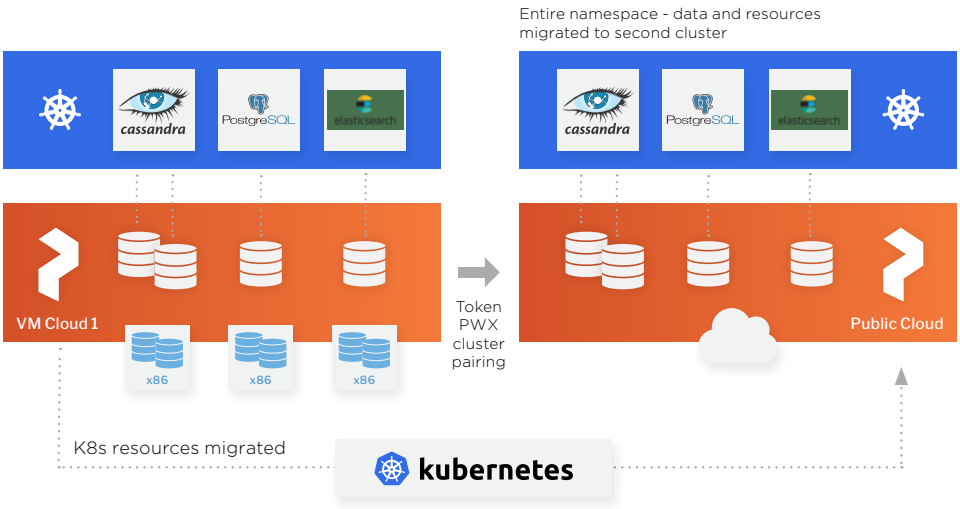


Image 13: Cross-Cluster Namespace Migration

PX-Backup also provides a number of utilities for managing backups, including point-in-time restore of data and filtering backups on cluster(s), namespace(s), and labels. All these features allow automating backup management for your applications on Kubernetes.

Backup Audit

PX-Backup tracks information about the application backup's source cluster. This allows managing the backup lifecycle long after the original source cluster has been deleted. The ability to use backup audit logs can help with data life cycle management and auditing data backup assets.

4.2.4 Other Backup and Snapshot Tools

Portworx provides several additional tools for manual backups and snapshots of applications in Kubernetes. These are volume snapshots, 3DSnaps, application backups, and application cloning.

With volume snapshots, you can create copies of persistent volumes attached to your Kubernetes pods. You can create one-time and scheduled volume snapshots which then can be referenced in new persistent volume claims whenever you need to restore.

One can store snapshots of volumes locally or in the cloud using STORK (Storage Orchestration Runtime for Kubernetes). Cloud snapshots can be automatically uploaded to the configured S3-compliant endpoint (e.g., AWS S3).

Also, you can use 3DSnaps to create manual application-consistent backups. Similarly to PX-Backup, 3DSnaps allow users to specify pre-snapshot and post-snapshot rules to be executed before and after the snapshot. For example, this feature allows users to pause the database writes before the snapshot is taken and resume I/O after the process is over.

With Portworx, you can also perform full application backup manually using `applicationBackup` and `backupLocation` custom resources. You can specify backup targets at the namespace level and create backup schedules for each namespace or an individual application. Additionally, Portworx provides a way to clone an application between namespaces using an `ApplicationClone` custom resource.

5. CONCLUSION

The goal of data protection strategy is to safeguard data against theft and accidental loss and meet low RPO and RTO targets. Enabling an efficient data protection strategy becomes harder as your production systems become more complex and distributed. When running your applications in containers on Kubernetes, you should use container-granular, application-consistent, namespace-aware, and multi-datacenter BCDR strategy for your workloads.

In this paper, we discussed major data protection requirements for different types of applications depending on their significance, criticality, and revenue impact. Portworx local HA tools, PX-Backup, and PX-DR tools allow meeting these requirements for Kubernetes applications. With Portworx replication and storage-aware scheduling, you can ensure local storage HA. Combined with the regular application-consistent backups done with PX-Backup, this strategy can guarantee effective data protection for the applications and data with the medium impact profile.

You can also enable efficient data protection for critical workloads using synchronous and asynchronous disaster recovery tools. Synchronous PX-DR will be the best choice if your cluster spans the MAN and has low latency whereas an asynchronous DR is a good choice for multiple clusters distributed across wide area networks (WANs). In both cases, Portworx provides tools needed to ensure automatic detection of cluster failures, cluster synchronization, and automatic promotion of standby clusters to active clusters.

Implementing data protection at local and multi-cluster levels will protect your clusters against many potential hazards while providing visibility into the lifecycle of your applications, data, and backups via Portworx BCDR monitoring and auditing features available for multiple clusters from a single user interface.

portworx
by Pure Storage

Portworx, Inc.
4940 El Camino Real, Suite 200
Los Altos, CA 94022
Tel: 650-241-3222 | info@portworx.com | www.portworx.com