5 Traits of Effective Disaster Recovery

portworx

Michael Ferranti,

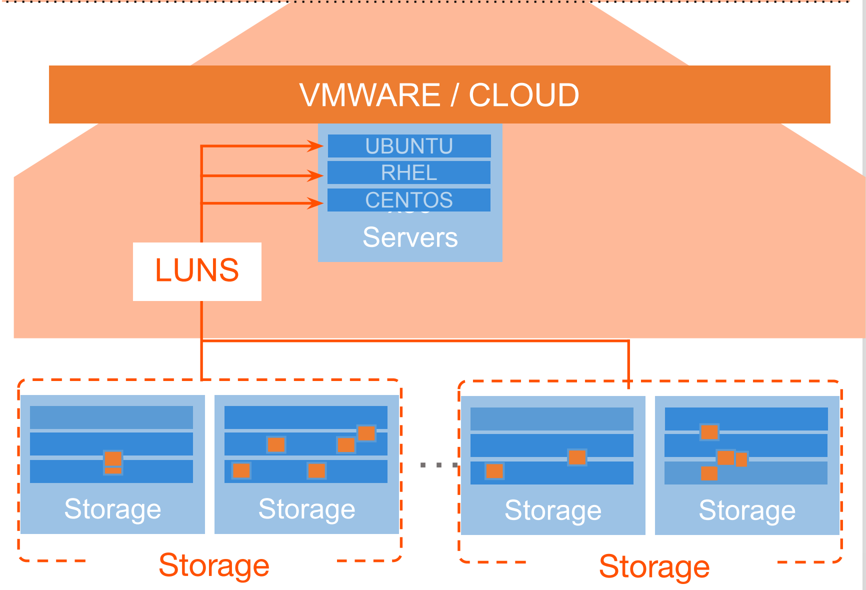
VP, Product & Market Strategy

- ✓ STATEFUL CONTAINERS SINCE BEFORE KUBERNETES
- ✓ CLOUD/SAAS PRODUCT & MARKETING BACKGROUND
- ✓ PASSIONATE ABOUT DISTRIBUTED SYSTEMS
- ✓ EX-CLUSTERHQ (FLOCKER), MAILGUN, RACKSPACE

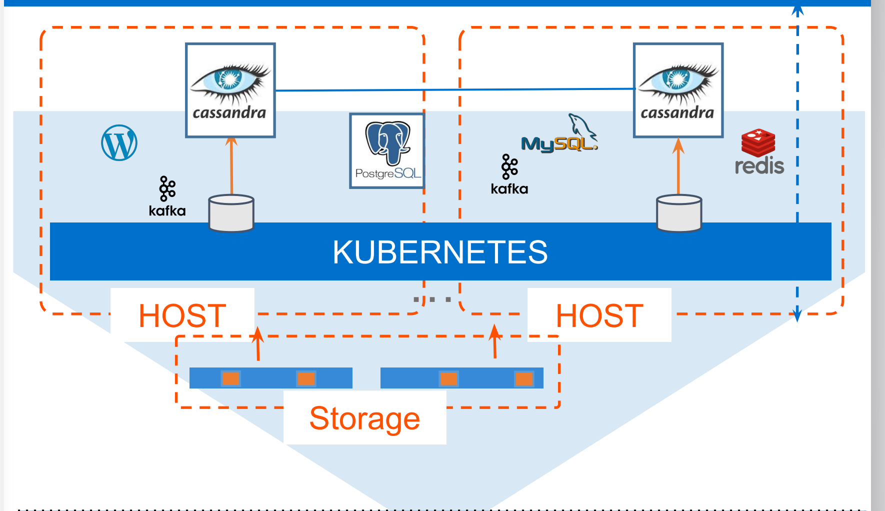
Machine Defined vs Application Defined

MACHINE DEFINED CONTROL PLANE

- ▶ Focus is on machine provisioning and lifecycle management
- ▶ Driven by a VM, Storage and Network Admin

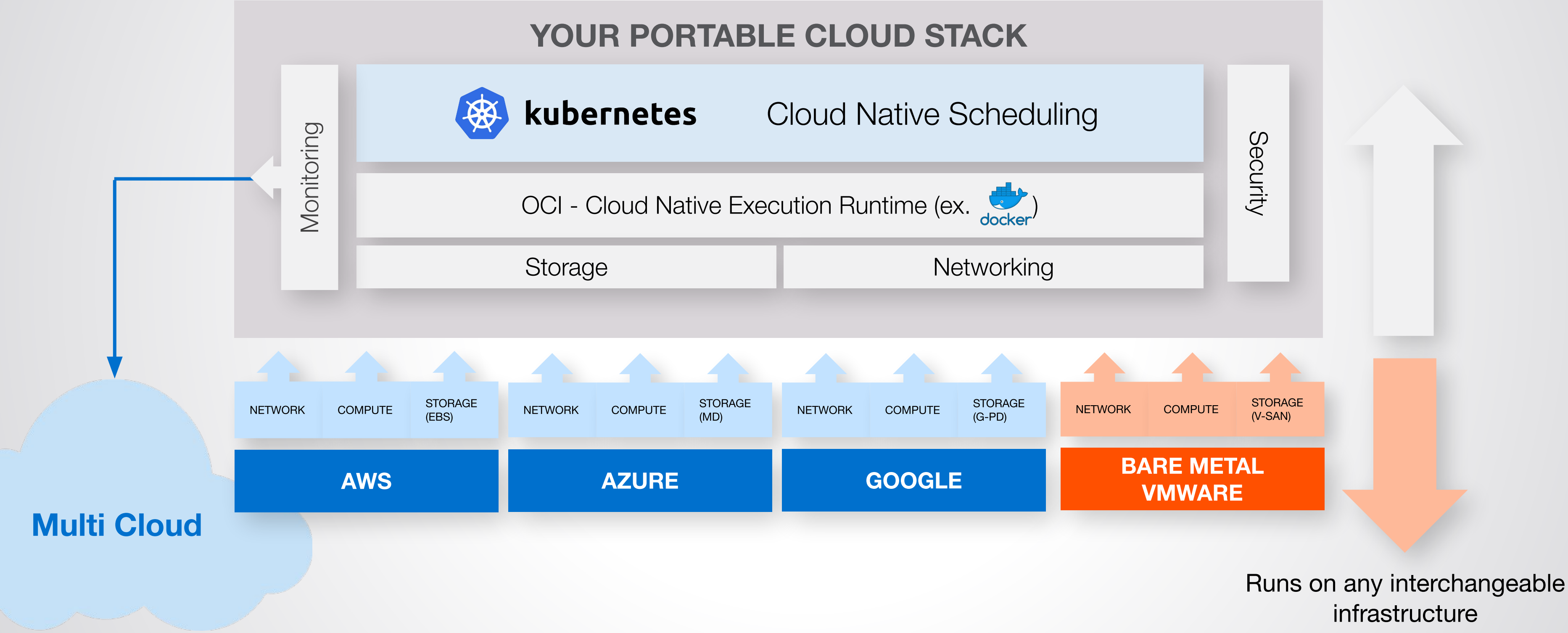


APP DEFINED CONTROL PLANE



- ▶ Focus is on application aware infrastructure provisioning and lifecycle management
- ▶ Driven by an end user (application owner)

That means a new stack has emerged



YOUR PORTABLE CLOUD STACK

kubernetes Cloud Native Scheduling

OCI - Cloud Native Execution Runtime (ex. docker)

Storage

Networking

Monitoring

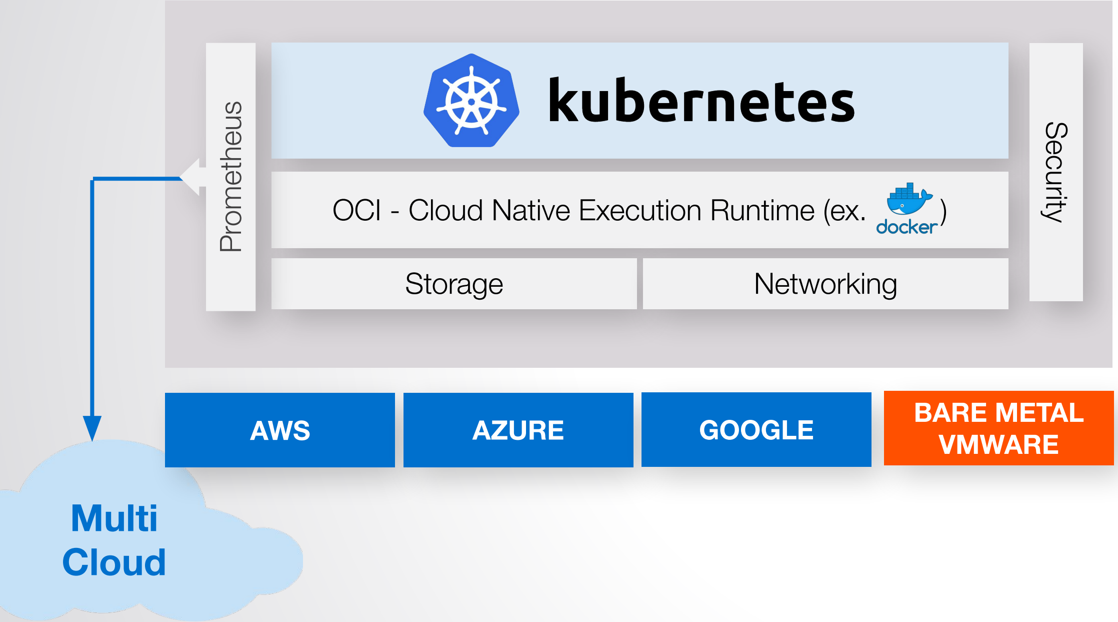
Security

AWS			AZURE			GOOGLE			BARE METAL VMWARE		
NETWORK	COMPUTE	STORAGE (EBS)	NETWORK	COMPUTE	STORAGE (MD)	NETWORK	COMPUTE	STORAGE (G-PD)	NETWORK	COMPUTE	STORAGE (V-SAN)

Multi Cloud

Runs on any interchangeable infrastructure

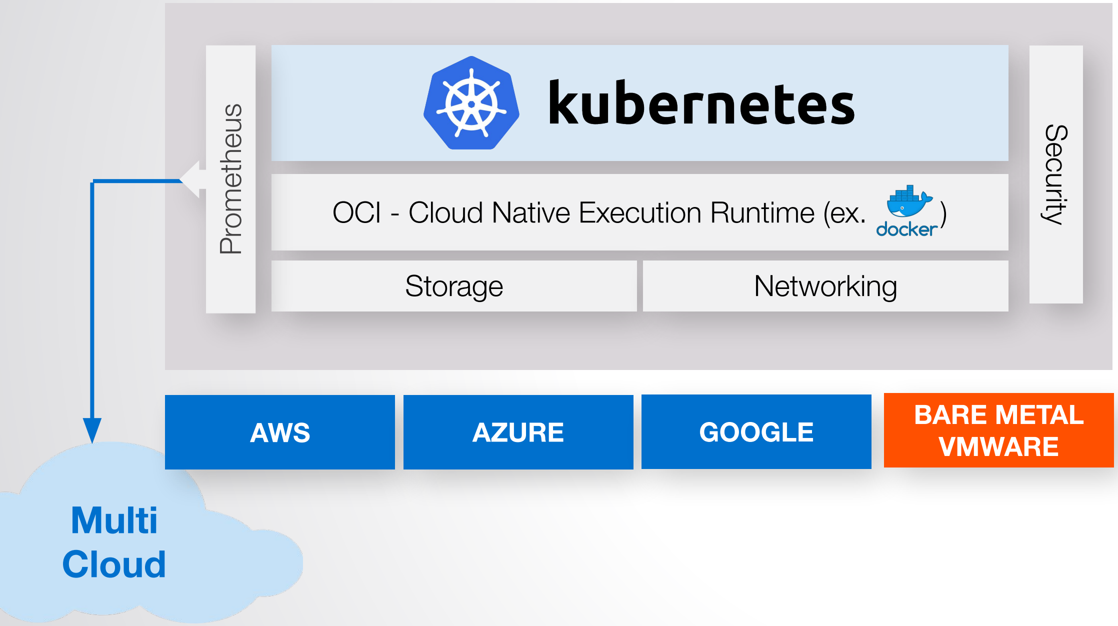
This new stack enables radical gains for your business



CLOUD NATIVE DELIVERS

- ▶ Self service for developers
- ▶ Fully automated
- ▶ Infrastructure agnostic SLAs
- ▶ Simple to adopt
- ▶ Low Touch Ops
- ▶ Optimization for cost

But...only if you can run most enterprise apps



kubernetes

OCI - Cloud Native Execution Runtime (ex. docker)

Storage

Networking

Prometheus

Security

AWS

AZURE

GOOGLE

BARE METAL VMWARE

Multi Cloud

BUSINESS REQUIREMENTS FOR ENTERPRISE APPS

- ⊗ Protect sensitive user data
- ⊗ Deliver performance within strict SLAs
- ⊗ Zero RPO DR for mission-critical apps
- ⊗ Backup and recovery for any app
- ⊗ Role-based access controls
- ⊗ Compliance and governance

Kubernetes Storage Platform: Essential to Cloud Native Success

1

ALL ENTERPRISE APPS HAVE DATA.

It needs to be available, performant and secure.

2

YESTERDAY'S STORAGE CAN'T KEEP UP

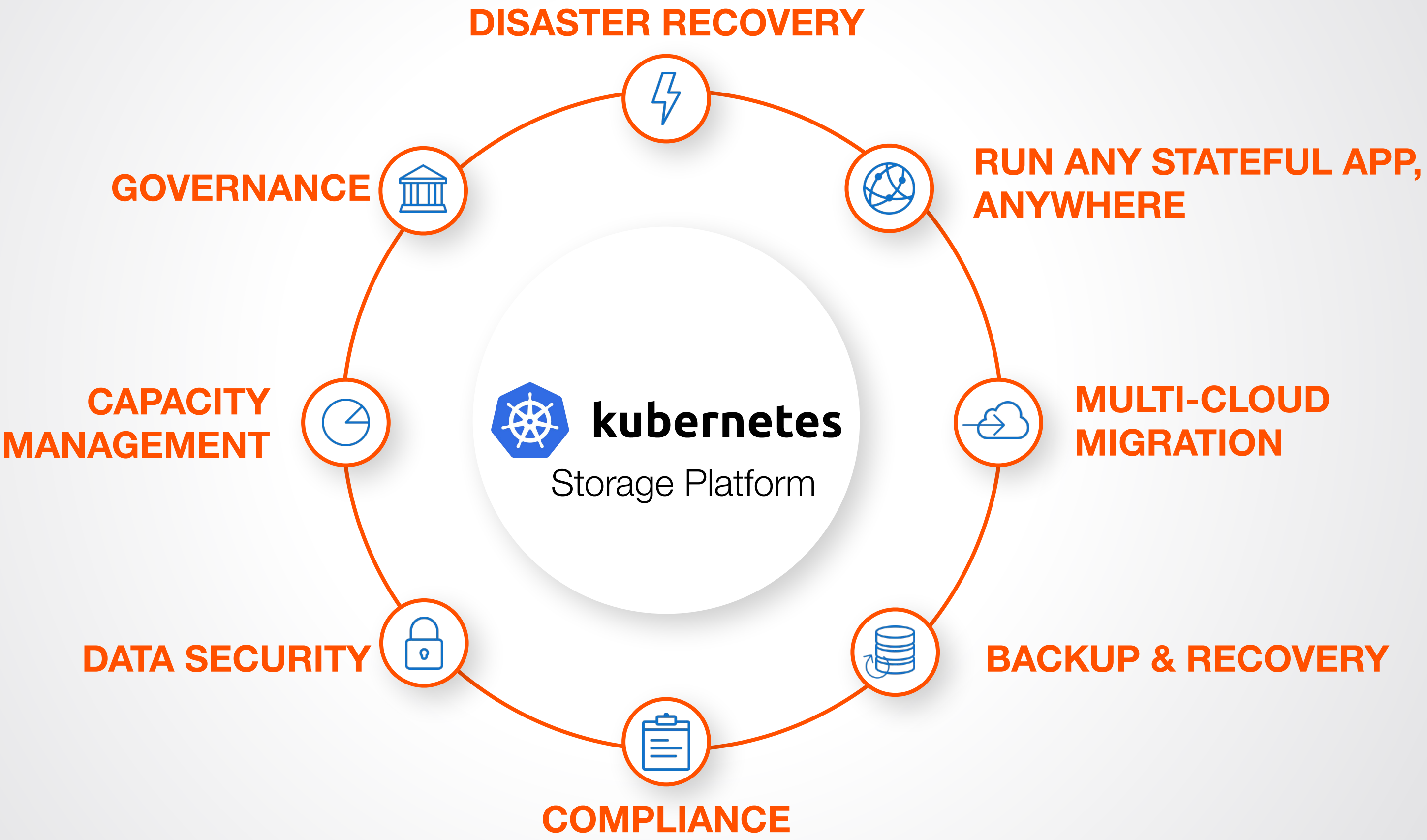
with the scale and dynamism of Kubernetes.

3

YOU CAN'T IGNORE THE PROBLEM.

Keeping data services outside of Kubernetes increases complexity, cost and lockin, while reducing agility.

A successful Kubernetes storage platforms gives you...

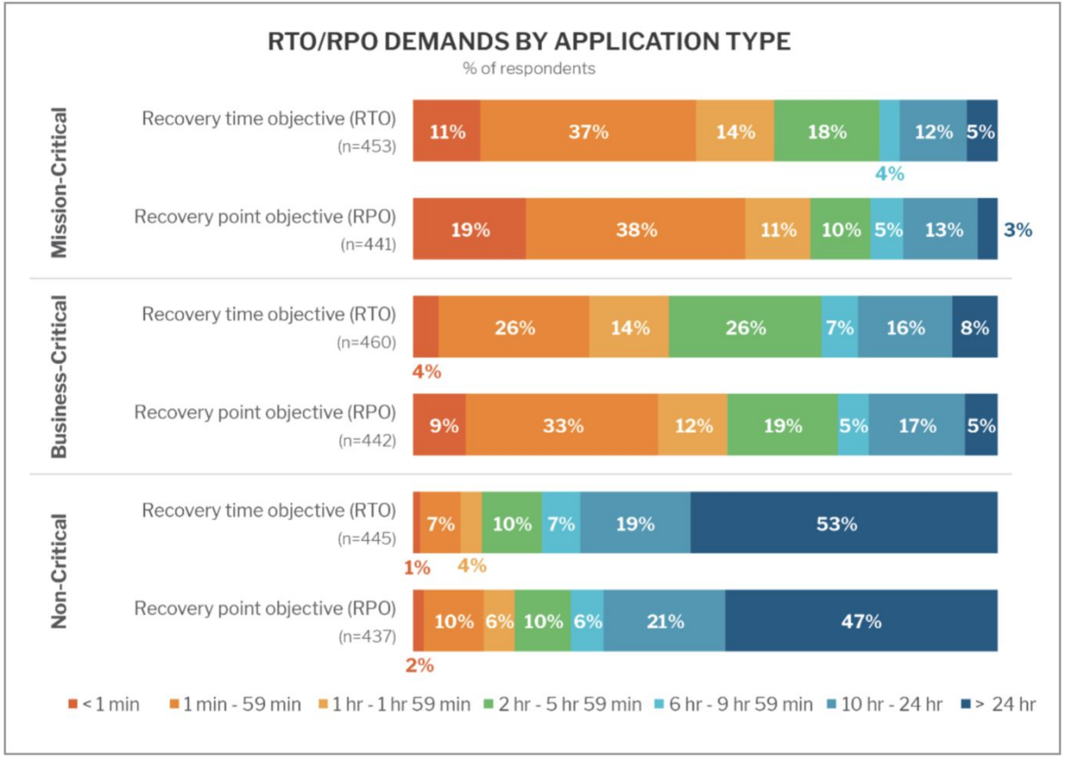


For mission-critical apps

48% demand RTO < 1 hr

57% demand RPO < 1 hr

Figure 1: RTO/RPO demands by application type



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

DR for containers is different than VMs

✓ **Container-granular**

Machine-based backups are no longer sufficient

✓ **Kubernetes namespace-aware**

Must speak the language of k8s

✓ **Application consistent**

Distributed systems require application consistency

✓ **Capable of backing up up data AND app config**

Just data is not enough. Neither is just app config.

✓ **Optimized for multi-cloud world**

Options for local and WAN-based DR

DR for containers is different than VMs

✓ Container-granular

Machine-based backups are no longer sufficient

✓ Kubernetes namespace-aware

Must speak the language of k8s

✓ Application consistent

Distributed systems require application consistency

✓ Capable of backing up up data AND app config

Just data is not enough. Neither is just app config.

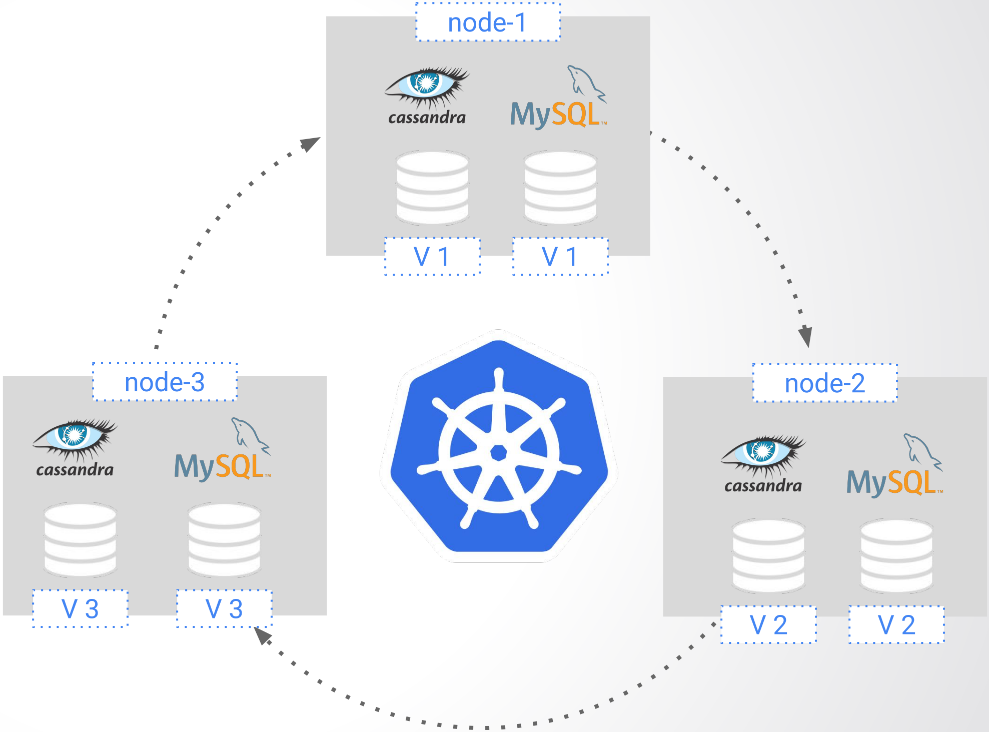
✓ Optimized for multi-cloud world

Options for local and WAN-based DR

Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

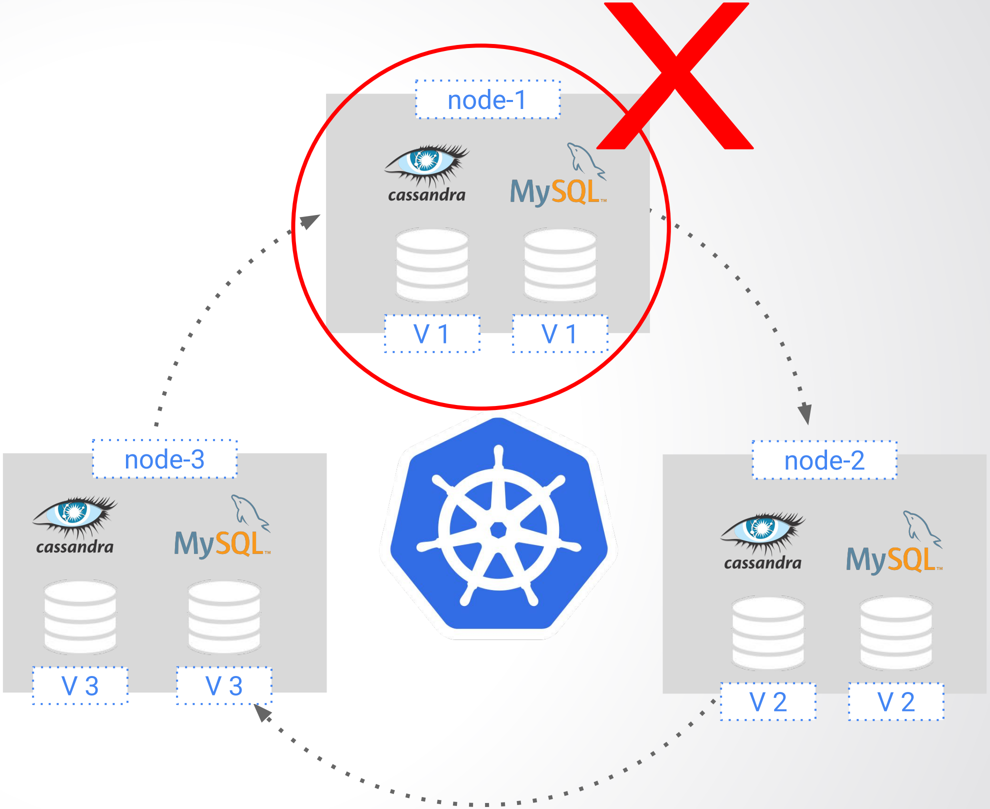
How do you back up only the application in question?



Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

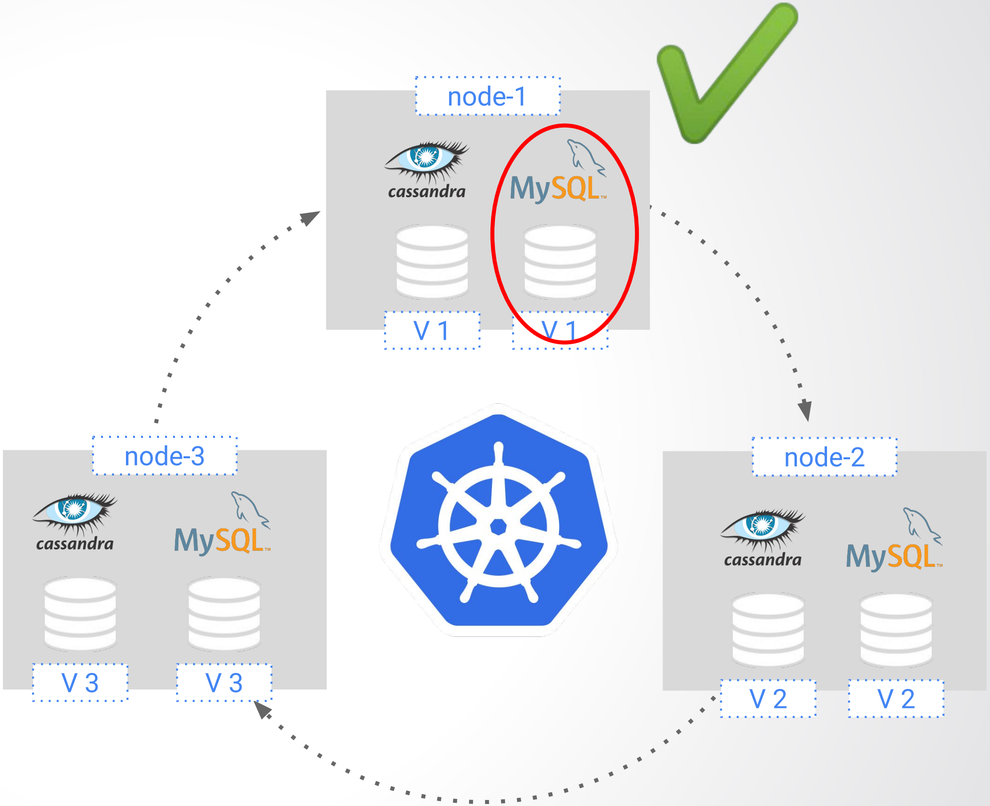
How do you back up only the application in question?



Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

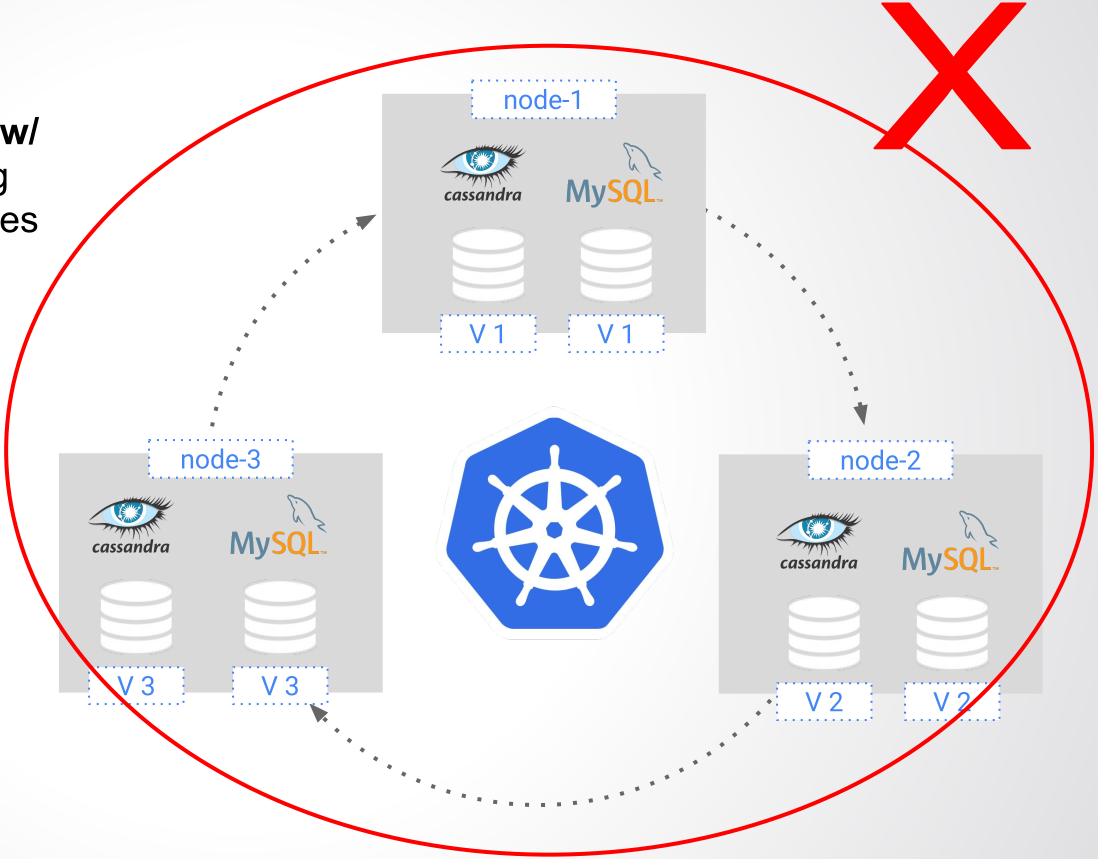
How do you back up only the application in question?



Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

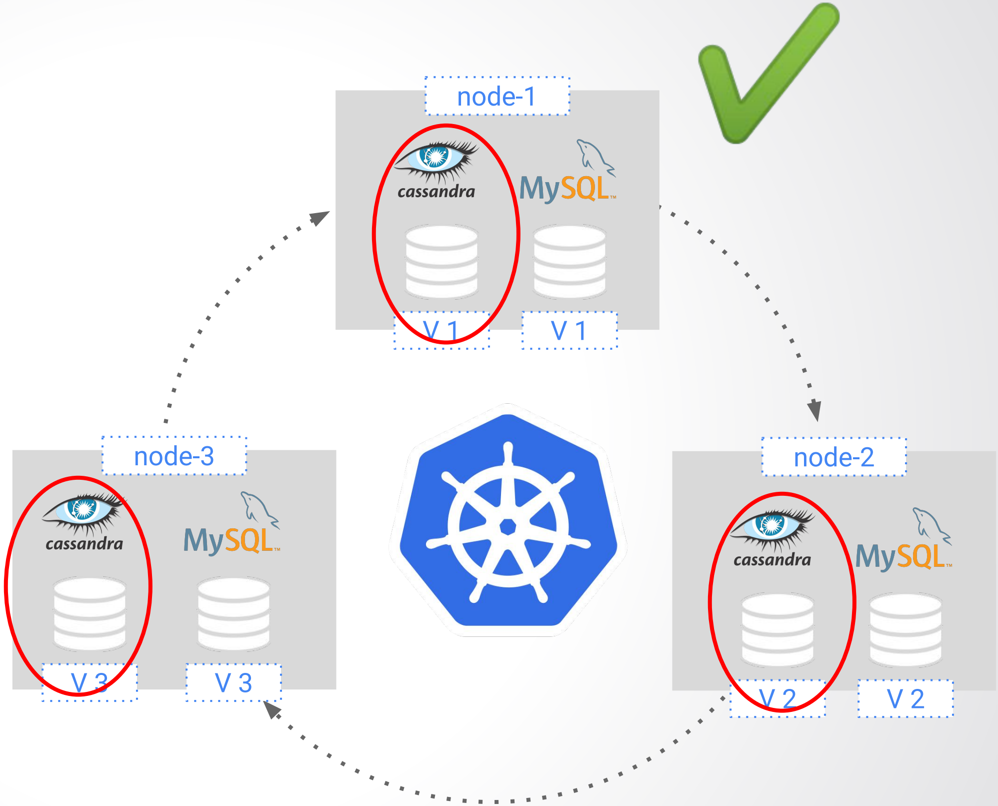
How do you back up only the application in question?



Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

How do you back up only the application in question?



DR for containers is different than VMs

✓ Container-granular

Machine-based backups are no longer sufficient

✓ Kubernetes namespace-aware

Must speak the language of k8s

✓ Application consistent

Distributed systems require application consistency

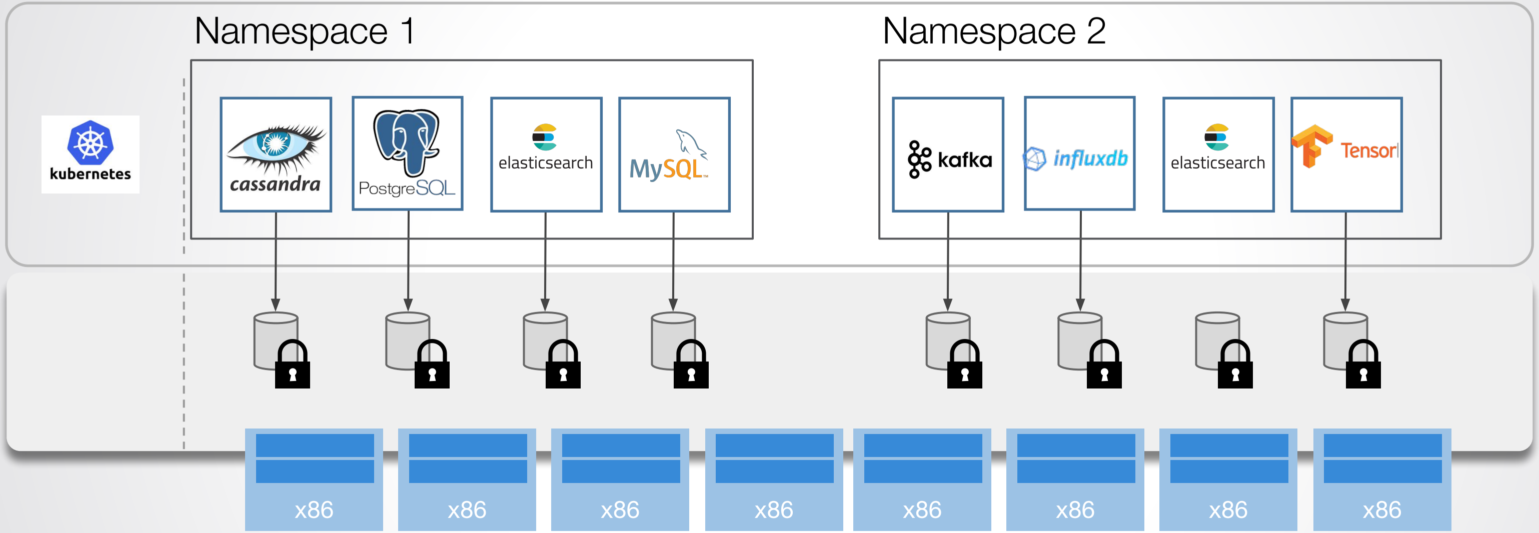
✓ Capable of backing up up data AND app config

Just data is not enough. Neither is just app config.

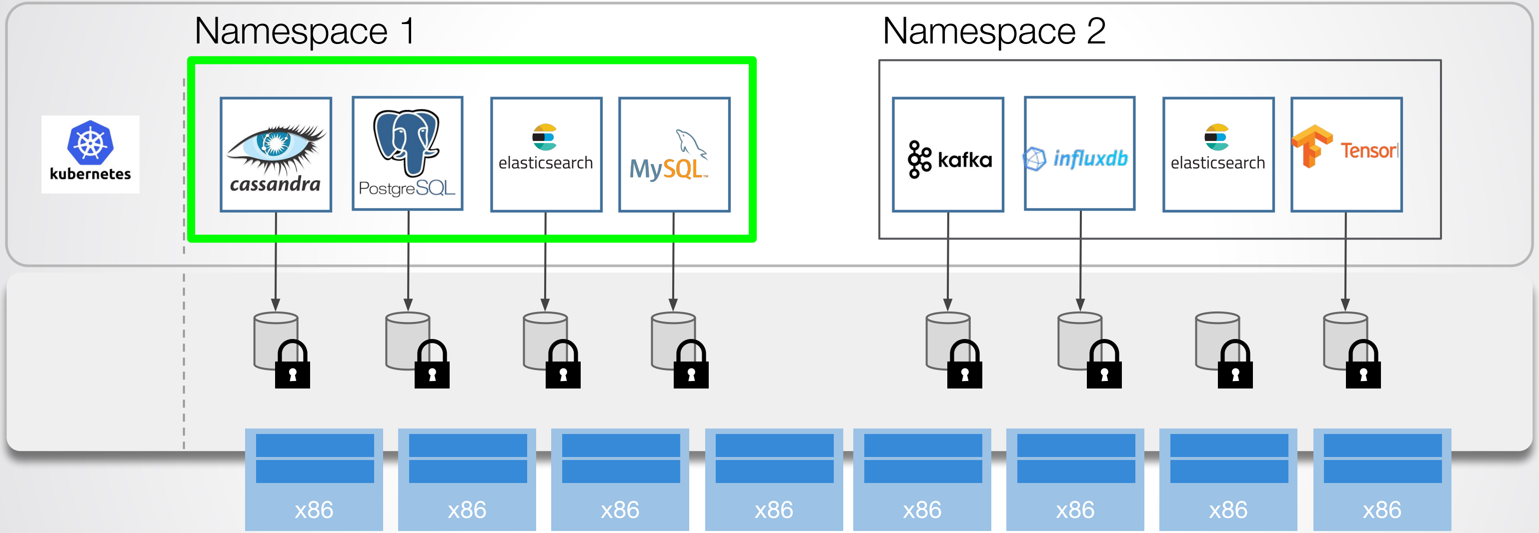
✓ Optimized for multi-cloud world

Options for local and WAN-based DR

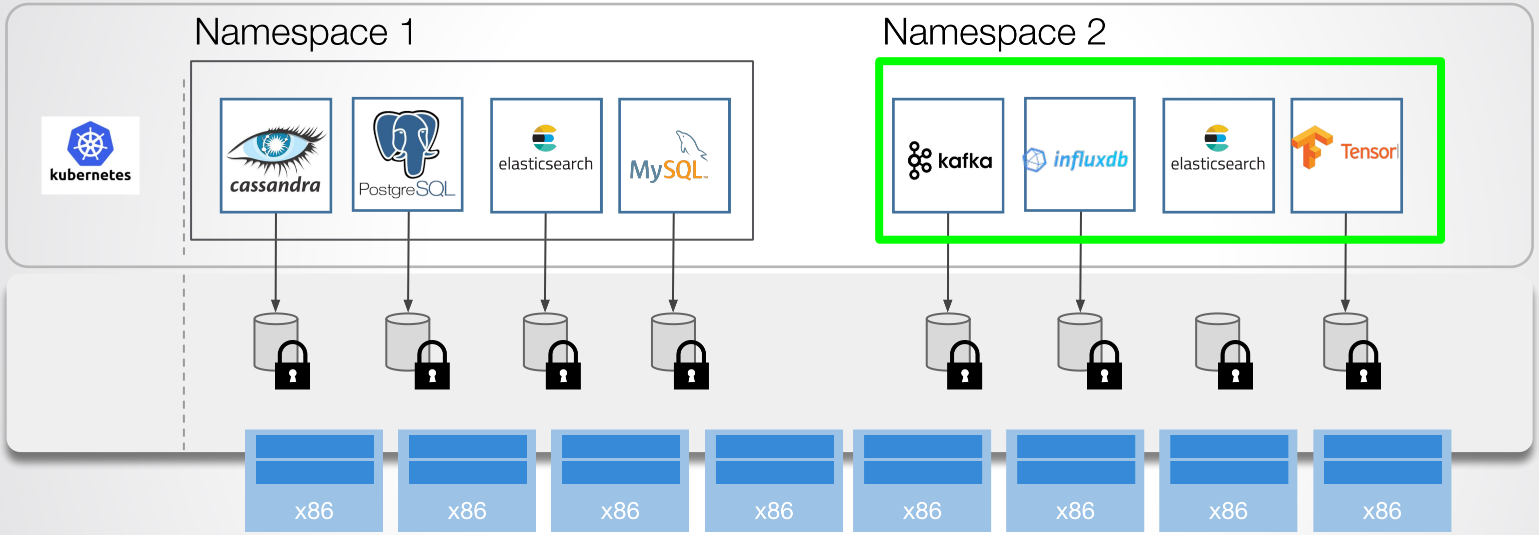
We must extend that container-granularity to namespaces



We must extend that container-granularity to namespaces



We must extend that container-granularity to namespaces



DR for containers is different than VMs

✓ **Container-granular**

Machine-based backups are no longer sufficient

✓ **Kubernetes namespace-aware**

Must speak the language of k8s

✓ **Application consistent**

Distributed systems require application consistency

✓ **Capable of backing up up data AND app config**

Just data is not enough. Neither is just app config.

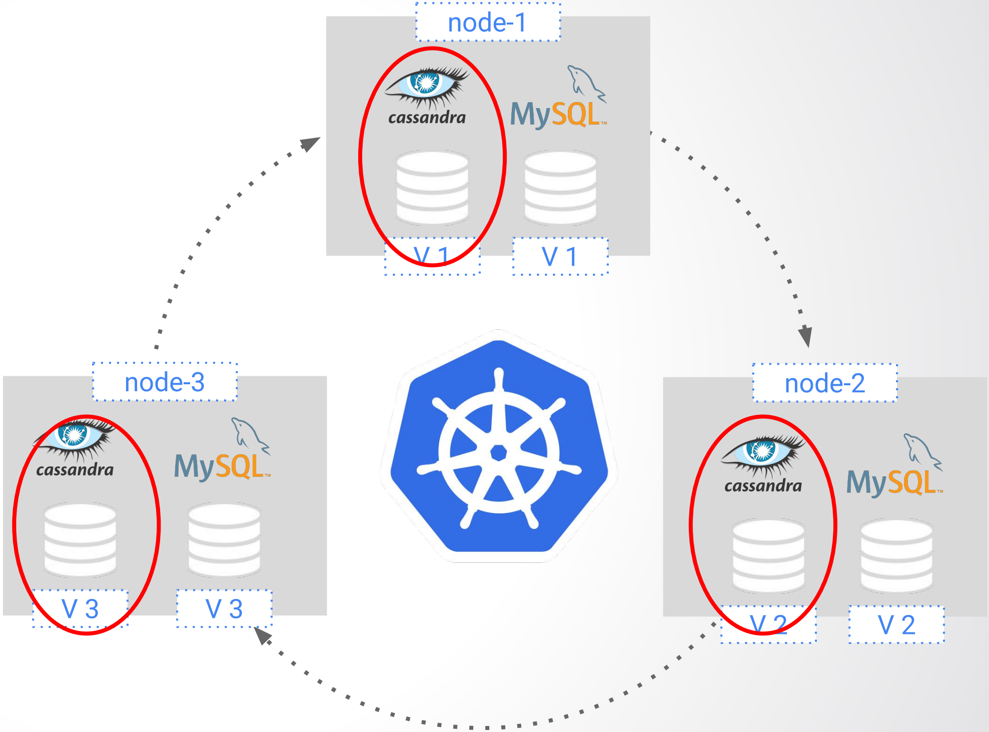
✓ **Optimized for multi-cloud world**

Options for local and WAN-based DR

Three-node Kubernetes cluster w/

- 1 three-node Cassandra ring
- 3 one-node MySQL databases

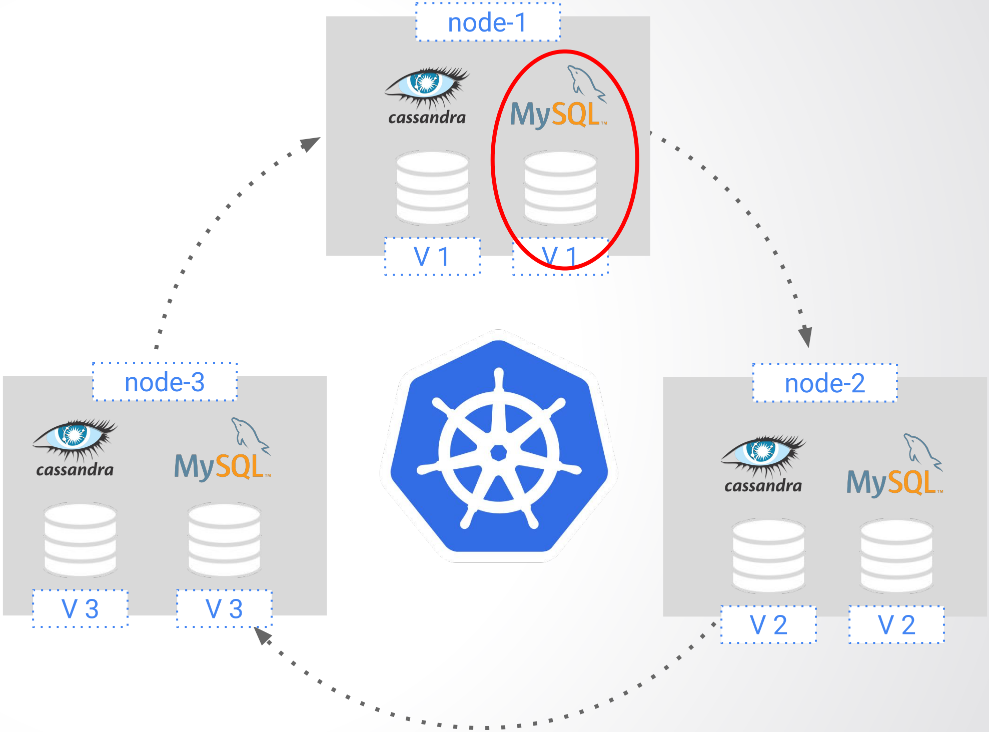
How do you back up only the application in question?



Three-node Kubernetes cluster w/

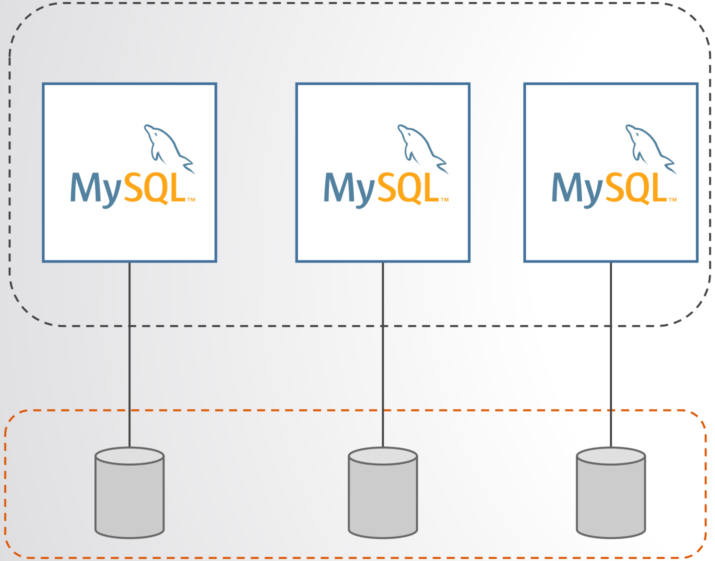
- 1 three-node Cassandra ring
- 3 one-node MySQL databases

How do you back up only the application in question?



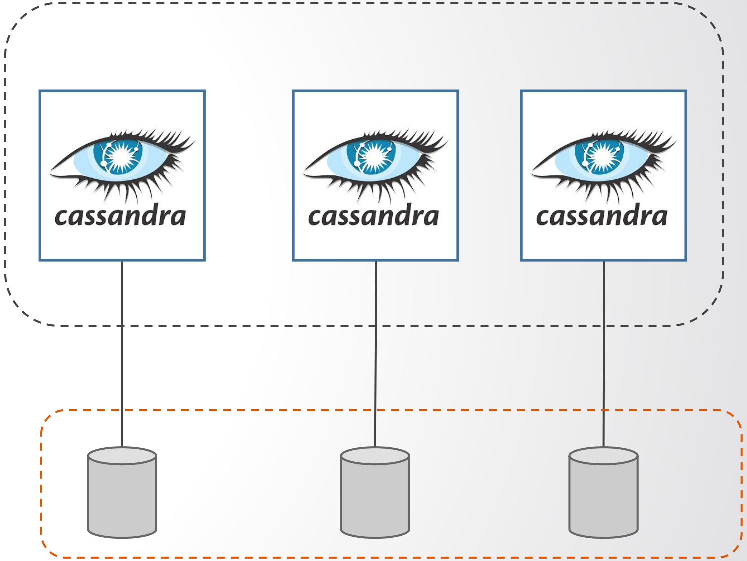
App-consistent backups means understanding apps

1. Flush & Lock tables in background
2. Status complete and return to CRD



3. Freeze filesystems and snapshot
4. Unfreeze filesystems
5. Release table Lock

A. Flush memory
B. Status complete and return to CRD



C. Freeze filesystems and snapshot
D. Unfreeze filesystems

DR for containers is different than VMs

✓ **Container-granular**

Machine-based backups are no longer sufficient

✓ **Kubernetes namespace-aware**

Must speak the language of k8s

✓ **Application consistent**

Distributed systems require application consistency

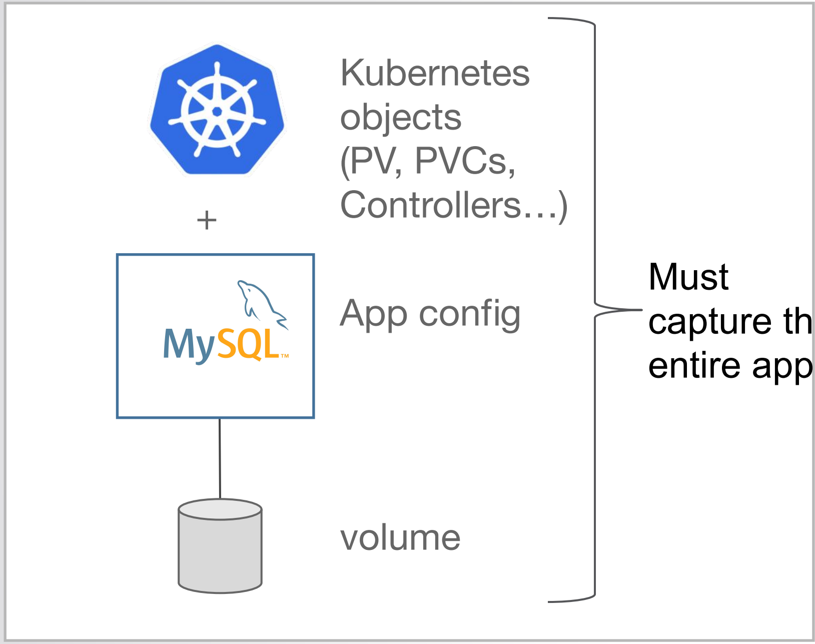
✓ **Capable of backing up up data AND app config**

Just data is not enough. Neither is just app config.

✓ **Optimized for multi-cloud world**

Options for local and WAN-based DR

Multi-cloud migration for Kubernetes requires App + Data



Kubernetes objects (PV, PVCs, Controllers…)

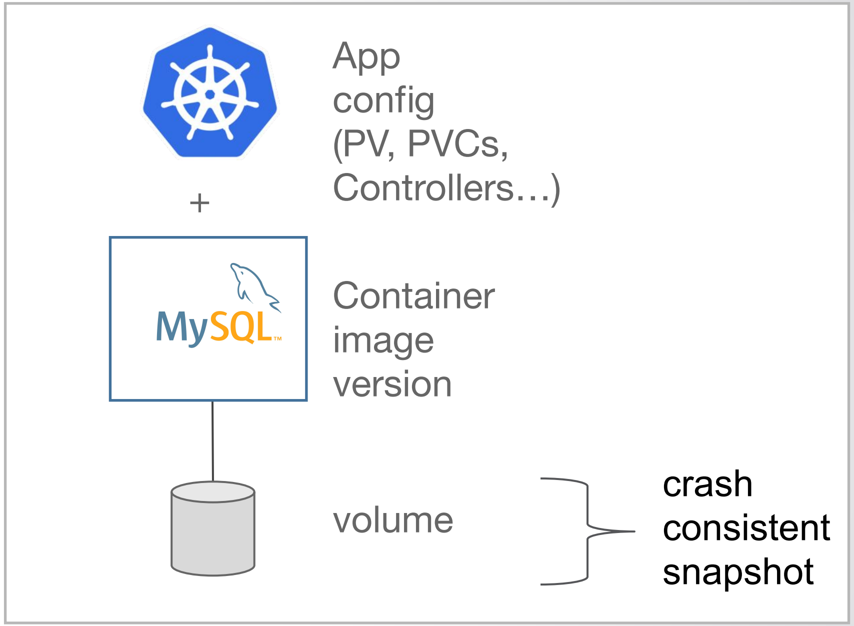
+

App config

volume

Must capture the entire app

Seamless DR, faster recovery



App config (PV, PVCs, Controllers…)

+

Container image version

volume

crash consistent snapshot

Other DR solutions

DR for containers is different than VMs

✓ **Container-granular**

Machine-based backups are no longer sufficient

✓ **Kubernetes namespace-aware**

Must speak the language of k8s

✓ **Application consistent**

Distributed systems require application consistency

✓ **Capable of backing up up data AND app config**

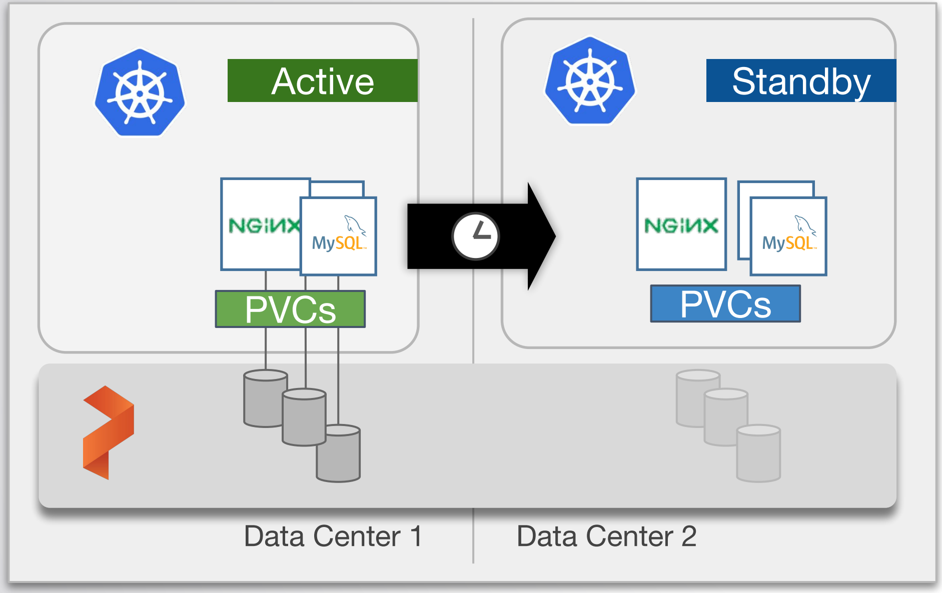
Just data is not enough. Neither is just app config.

✓ **Optimized for multi-cloud world**

Options for local and WAN-based DR

Instant failover protection

for Kubernetes across data centers in a Campus



Campus

1. **Portworx Cluster Stretches**
   Spans datacenters 1, 2 with synchronous data protection and central management

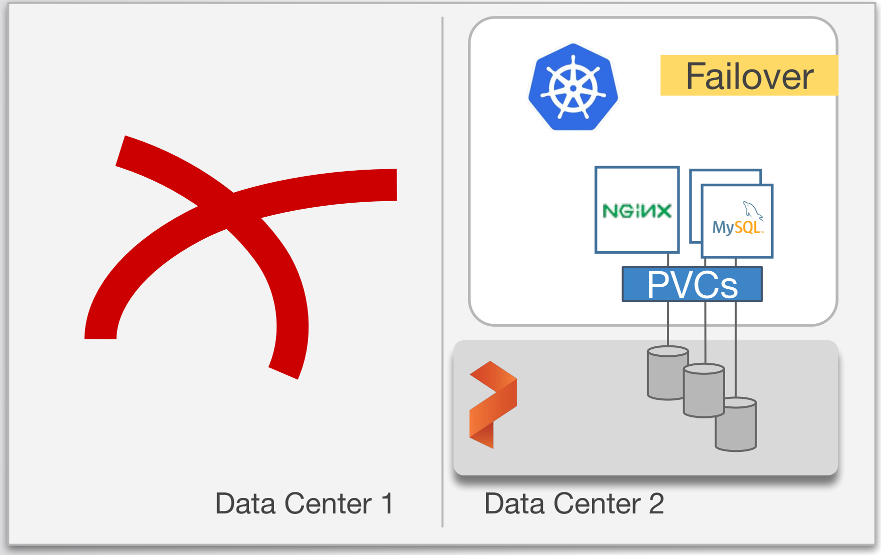
2. **Kubernetes Clusters Pair**
   Active cluster continuously or periodically backs-up apps, configuration to a Standby

3. **Volumes Placed per Fault-Domain**
   Fault-domains (fd) detection is integrated with K8s. Volumes place replicas in each (fd).

4. **PX-DR Readies for Outage**
   Standby K8s cluster has running controllers, configuration and PVCs map to the same volumes (and replicas)

Failover protects apps + data

with Zero RPO



Campus

1. **Datacenter Loss Detection**
   Detection is integrated with your infra or a dedicated Portworx cloud service

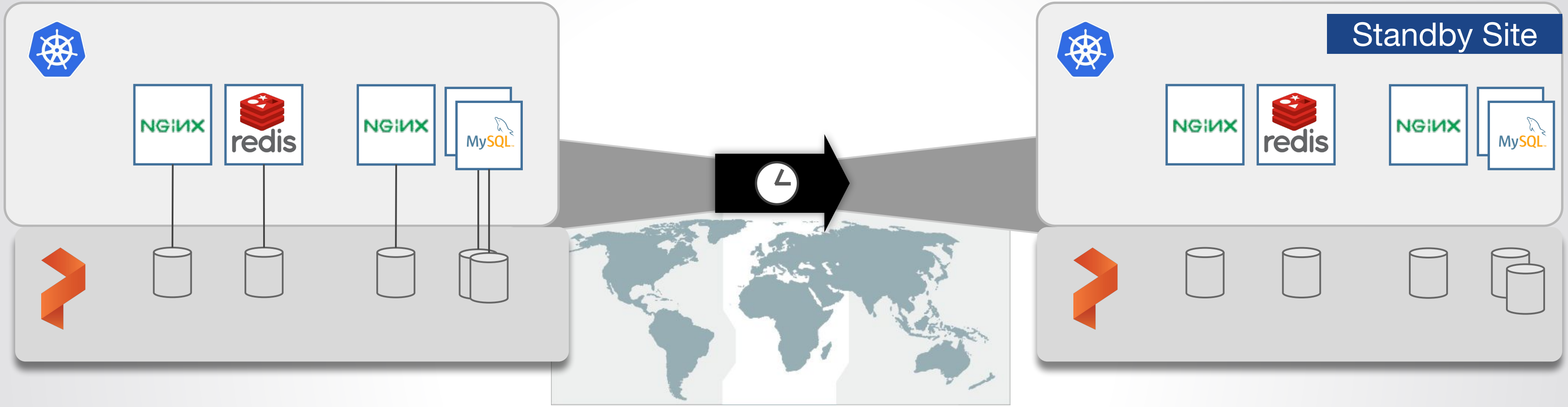
2. **Kubernetes Failover**
   Standby promoted automatically to serving as already running controllers spin-up Pods

3. **Data Available with Zero RPO**
   Volumes were sync replicating data, Standby PVCs were ready, and RTO is near zero

4. **After, Promote either to Active**
   Once DC 1 is ready, Portworx resyncs data, backs-up K8s to new DC1, and you promote either to Active

Asynchronously replicate

Kubernetes App + Data across the WAN

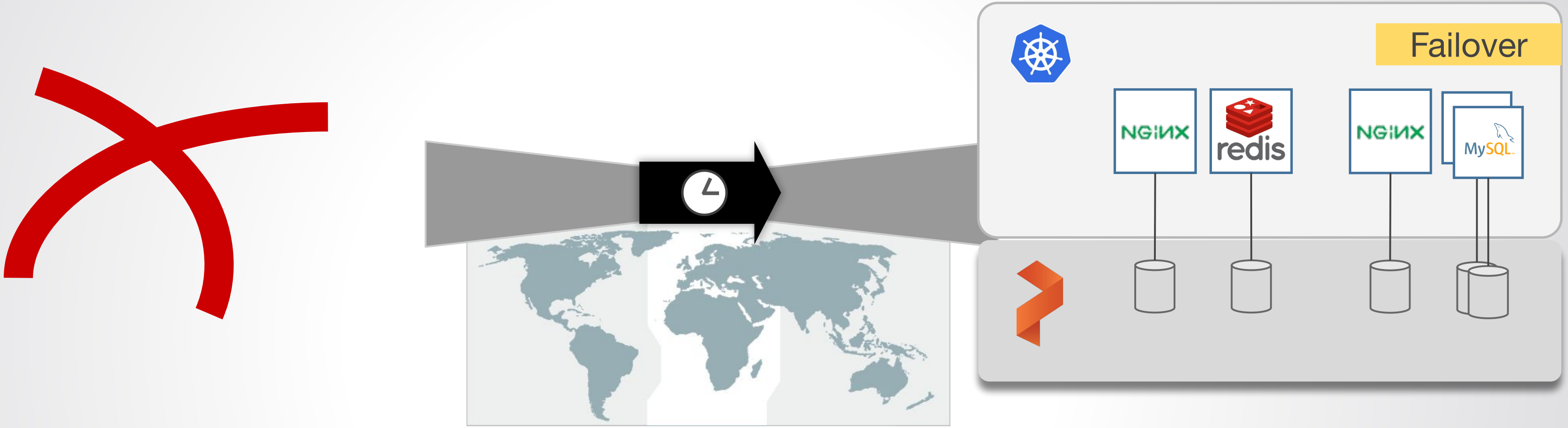


- **Active Site**
  PX-DR continuously sends the incremental changes in Kubernetes applications and Portworx data

- **Standby Site**
  PX-DR receives data, maps Kubernetes objects/PVCs to infrastructure, and keeps K8s controllers running and ready

PX-DR: asynchronously replicate

Kubernetes App + Data across the WAN



- **Failover Site**
  Detects outage based on integration with your infrastructure
- **Standby Site**
  Standby promoted automatically to serving as already running controllers spin-up Pods

Case Study: Multi-Cloud DR for Truly Mission Critical App

CHALLENGE

- ▶ Zero downtime for Kubernetes apps even in face of entire data center loss

SOLUTION

- ▶ Two Kubernetes clusters- Production and DR site
- ▶ PX-DR provides cross data center replication of data and app config

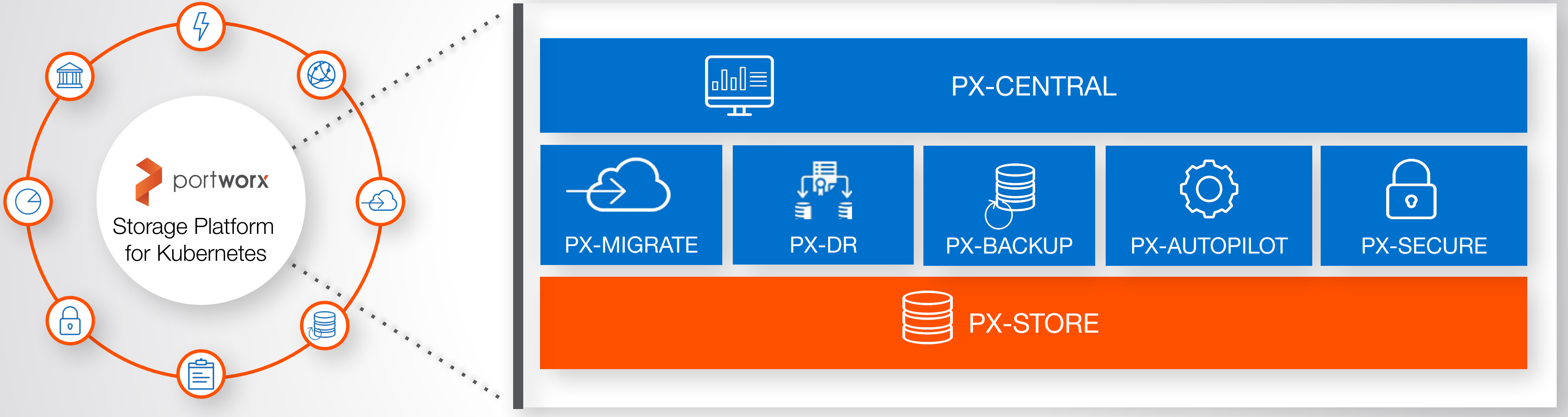
RESULTS

- ▶ Zero RPO failover with < 2 minute RTO in event of primary data center loss

Data services used: Couchbase MySQL

PORTWORK RUNS ON ANY KUBERNETES PLATFORM

RED HAT OPENSHIFT
Amazon EKS
Azure Kubernetes Service (AKS)
Google Kubernetes Engine
Rancher Kubernetes Engine
IBM Cloud Container Service



portworx
Storage Platform for Kubernetes

PX-CENTRAL

PX-MIGRATE
PX-DR
PX-BACKUP
PX-AUTOPILOT
PX-SECURE

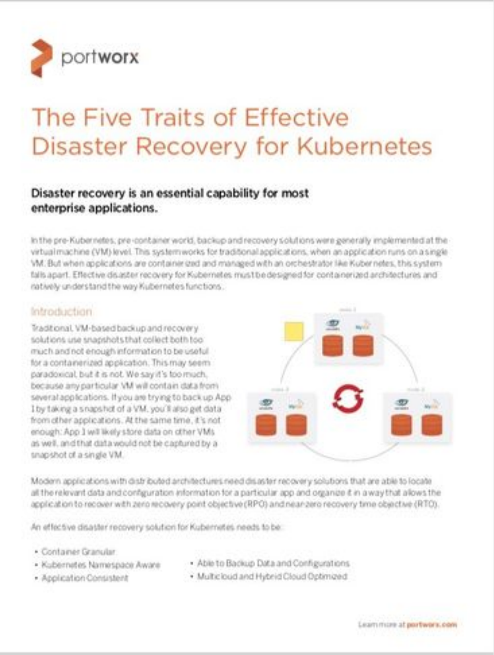
PX-STORE

aws
Google Cloud
Azure
vmware
vSAN™
NetApp
PURE STORAGE

PORTWORK RUNS ON ANY STORAGE HARDWARE

Want to learn more? Download free whitepaper on DR

ask.portworx.com/data-disaster-recovery-kubernetes/



portworx

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

Learn more at portworx.com

Q&A