Brought to you by:

veeam

Kubernetes Backup & Recovery

for dummies®
A Wiley Brand

Find out why Kubernetes native backup is important

Learn how to protect your Kubernetes applications

Create automated Kubernetes backup policies

Steve Kaelble

Kasten by Veeam®
2nd Special Edition

About Kasten by Veeam

Kasten by Veeam is the leader in Kubernetes backup and disaster recovery. Kasten K10 helps enterprises overcome Day 2 (live) operations backup management challenges to confidently run applications on Kubernetes. For additional information, visit http://kasten.io and explore the latest release of Kasten K10 backup and migration resources at http://docs.kasten.io. Follow @kastenhq on Twitter.

Kubernetes Backup & Recovery

Kasten by Veeam® 2nd Special Edition

by Steve Kaelble

for dummies®
A Wiley Brand

Kubernetes Backup & Recovery For Dummies®, Kasten by Veeam® 2nd Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-394-22648-1 (pbk); ISBN 978-1-394-22649-8 (ebk)

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Development Editor: Rebecca Senninger

Acquisition Editor: Traci Martin

Editorial Manager: Rev Mengle

Business Development Representative: Matt Cox

Production Editor: Saikarthick Kumarasamy

Table of Contents

Introduction

No one needs to tell you that the cloud is the place to be. Cloud-native adoption and usage is, well, skyrocketing (sorry, pun intended). Kubernetes is everywhere, well on its way to being the next enterprise platform of choice and the foundation for all kinds of applications, with very good reason. It's portable, it's agile, it's scalable, it's incredibly reliable, it's a developer's dream.

What it's *not,* however, is the be all and end all for the protection of your data. In fact, some of the architectural magic that makes it great also happens to create new challenges in data management and protection. The protections it offers your applications don't automatically extend to your data, and you can't just retrofit legacy backup architectures into a cloud-native ecosystem.

You need a truly cloud-native backup solution; one that not just speaks the language of Kubernetes but lives on this new and exciting planet. And you need insight into the best practices that ensure a smooth evolution to cloud-native adoption and usage.

About the Book

Kubernetes Backup & Recovery For Dummies, Kasten by Veeam Special Edition is your guide to the solution. It lays the groundwork with details on how Kubernetes came into being. It lets you know exactly why your backup tomorrow can't work the same way it did yesterday.

Get through this book and you'll gain a better handle on implementing and operating backup of your Kubernetes applications. Among other things, you'll find application-centric approaches for recoverability and ensuring security, operating in a world of multi-tenancy, transforming restore support, and really making the most of cloud-native application mobility.

This book is packed with actionable steps for protecting your data in the evolving ecosystem while making life easier for developers and operators. And it has insights for how you can simultaneously celebrate a revolution without rocking the boat in the wrong way.

Foolish Assumptions

In preparing this book, we've made some assumptions about you, the friend who has committed time to reading the pages.

- You may be a technical-type of person, involved in the DevOps of cloud-native applications.
- You may be on the business side but still intrigued about cloud-native opportunities or concerned about their risks.
- Either way, you're committed to ensuring that your organization succeeds in this ever-expanding frontier.

Icons Used in This Book

To help you navigate this book, we've included some icons in the margins. Think of them as alerts, letting you know which words are especially noteworthy.

REMEMBER

Skip around if you wish, but please pay special attention to these paragraphs, which include some of the most vital points.

TIP

We promised actionable information, and next to this icon you find some helpful ideas.

WARNING

Data backup can be challenging. This icon points out areas where special care is needed to avoid problems.

TECHNICAL STUFF

These particular paragraphs go further into the details. (If you like techie details, these paragraphs are for you!)

Beyond the Book

This book is intended as food for thought, an introduction to the concepts with a sampling of detail. If you devour the book and come away still hungry, go to `https://www.kasten.io/` for additional resources, background, and white papers.

IN THIS CHAPTER

- Choosing cloud-native and containerized applications
- Going with Kubernetes
- Spelling out the advantages
- Taking down the myths about Kubernetes

Chapter 1
Understanding Kubernetes and Cloud-Native Apps

Apple famously declared "there's an app for that" in a trademarked iPhone tagline that evolved into a popular meme. It couldn't have been more prescient — apps handle more and more daily tasks for consumers and businesses. Today, an increasing volume of app developers are adopting container-based architecture, with applications delivered in the Kubernetes environment.

This chapter discusses the increased use of container-based and cloud-native applications and outlines the rise of Kubernetes. It spells out the advantages of adopting a Kubernetes-native architecture and explodes some myths that surround what is quickly becoming the most effective way of deploying modern business applications.

Tracking the Rise of Cloud-Native Applications

Where do container-based applications fit into your organization right now? What does the future hold? Research suggests that we're well into the acceleration of a cloud-native and container-based boom.

Consider an April 2023 study from the Enterprise Strategy group, in which the firm provided an assessment of the current state of the Kubernetes market (`https://www.kasten.io/kubernetes/resources/analyst-reports/measuring-the-current-state-and-momentum-in-the-enterprise-market-for-kubernetes-protection`). They noted that market adoption for containers has risen rapidly. Almost half of organizations (47 percent) reported that they use containers today, and another 35 percent of organizations reported that they plan to start using containers. In another year, it expects that 82 percent of organizations will be using containers.

The Cloud Native Computing Foundation, meanwhile, found in its 2022 survey (`www.cncf.io/reports/cncf-annual-survey-2022`) that 44 percent of respondents are already using containers for nearly all applications and business segments and another 35 percent say containers are used for at least a few production applications.

Other key findings of this survey include

- **Lack of training is the biggest challenge.** In fact, lack of training is the most significant barrier inhibiting adoption. It is the top challenge cited by 44 percent that have yet to deploy containers in production, and 41 percent of those that use containers on a limited basis. Once containers are used for nearly all applications, then security becomes the top challenge.
- **Auxiliary workloads are starting to outnumber application workloads.** In 2021, in a typical Kubernetes cluster, application workloads accounted for most of the pods (59 percent). In contrast, all non-application workloads (system and auxiliary workloads) played a relatively smaller part.

  In 2022, this picture was reversed. Auxiliary workloads outnumbered application workloads (63 percent versus 37 percent) as organizations increasingly adopted advanced Kubernetes

platform technologies, such as security controls, service meshes, messaging systems, and observability tools. At the same time, organizations used Kubernetes for a broader range of use cases such as building pipelines and scheduled utility workloads. Kubernetes became the platform for running almost anything, emerging as the "operating system" of the cloud.

Running on Kubernetes

REMEMBER

Kubernetes, which made its open-source debut in 2015, has quickly become the most common platform for container scheduling and orchestration. It is, indeed, headed toward becoming the foundation for nearly all applications, wherever they might be deployed. It's on its way toward joining the ranks of Linux and vSphere as an enterprise platform of choice.

Kubernetes deploys and maintains applications and scales them based on various metrics such as CPU and memory. Its building blocks are known as *primitives*, and it defines compute and storage resources as objects. The most important Kubernetes API objects include:

- Clusters
- Nodes
- Labels and selectors
- Replication set
- Deployment

A QUICK KUBERNETES BACKSTORY

It's worth recounting how Kubernetes got to where it is today. The name Kubernetes is Greek for *helmsman* or *pilot,* and it has some etymology in common with the word cybernetics. It was the work of a group of Google engineers in 2014 and was originally known as Project 7.

(One more tidbit for sci-fi fans: That original codename honored the *Star Trek* character Seven of Nine, once part of the powerful Borg collective, tying into the fact that Kubernetes was influenced by an earlier Google system called Borg.)

Kubernetes' robustness makes it possible for users to deploy, scale, and manage containerized applications. Its extensibility and portability have earned it loads of popularity in the cloud-computing ecosystem. In addition, Kubernetes also gives users the flexibility of choosing which programming language, or framework, to use, and allows users to be able to monitor and log errors.


REMEMBER

Ultimately, going down this path boosts productivity, gets applications into production more quickly, and reduces costs. Kubernetes automates many processes for deploying, managing, and scaling. You can tap into the magic of Kubernetes to create clusters of hosts running containers, and manage those clusters across public, private, or hybrid clouds. And you can expect it all to work exceptionally well, with few worries about application downtime.

It's no wonder that Kubernetes has been adopted so quickly. Check the next section for a broader outline of some of the advantages. But as with anything that gains popularity quickly, some people jump in without a full understanding of what they're getting into.

In the case of Kubernetes, that has led to what you might call "Day 2" production challenges — in particular, data management, security, and observability. You've removed a lot of the pain related to achieving high availability and scalability of application services. But you can't automatically stretch these benefits to cover your data, which means you need to make a priority of Kubernetes application data management.

Seeing the Advantages


REMEMBER

Development teams love the increased agility, portability, and reliability they gain through Kubernetes. No surprise, then, that there has been a rapid influx of applications onto the platform. Not just stateless applications but stateful applications, including applications powered by a NoSQL database and applications using a relational database for their backend.

Here are some of the advantages to be gained through the use of cloud-native infrastructure and Kubernetes:

- You can easily access the necessary computer, storage, and networking you need to support rapid growth.

- Storage is easy to use, and self-service is a snap. With Kubernetes, relational and NoSQL databases can be built in easily and smoothly.
- Changes to containerized applications can be easily deployed. Improvements and updates to applications, even complex ones, can happen quickly.
- The platform can enable scaling requirements almost instantaneously.

Check Figure 1-1 for a glimpse of the Kubernetes ecosystem.

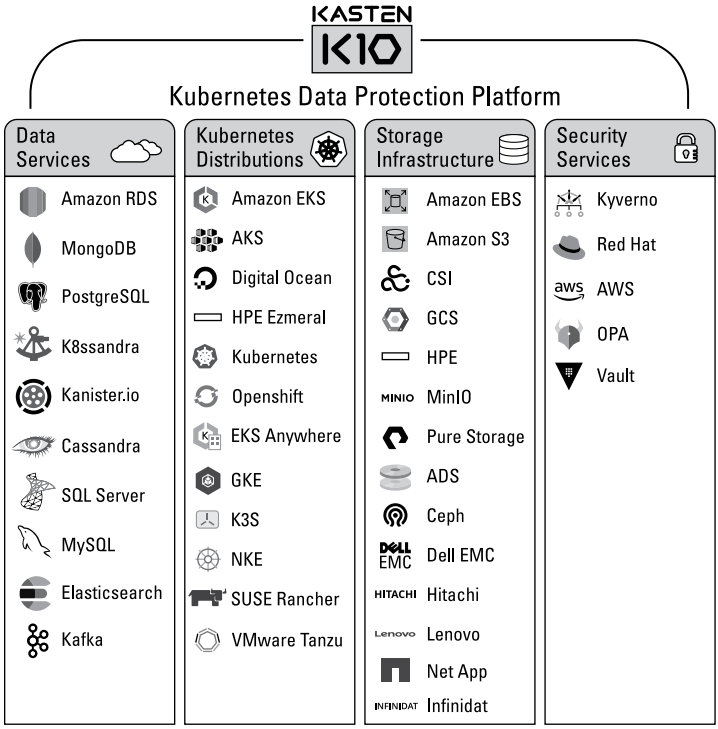

FIGURE 1-1: The Kubernetes ecosystem.

Exploring the Myths

Many people will tell you that applications running on the Kubernetes platform should be architected to be stateless. And in the infancy of Kubernetes, there was some truth to this belief. But as

with everything else about technology, things change quickly, and that one-time reality is now a myth.

Today's reality is that support for storage and stateful applications has reached maturity. Kubernetes has become an ideal platform for both stateless and stateful applications — and to protect your stateful Kubernetes application, you need a copy of the data in a completely different, totally independent location.

WARNING

And therein lies the potential problem. Application developers can access low-friction and self-service dynamic storage provisioning, which is great. But it's not uncommon to find extensive storage use in Kubernetes clusters that's not necessarily built for stateful applications.

What makes that problematic is that widespread use of relational and nonrelational databases on Kubernetes can leave your data exposed if you don't have the right data management systems in place. And that spells R-I-S-K to the business.

This risk is not obvious to everyone, and it's easy enough to see why. After all, one of the best things about running applications on Kubernetes is high availability in the face of software problems, server issues, or region failures. If you're running a stateful application, that makes it way easier to run replicated data services across failure domains.

REMEMBER

The bottom line is that high availability is not the same thing as backup. Just because you have achieved high availability doesn't mean you have a backup of your workload in a different location. Replication improves data availability and protects against partial infrastructure failure, but doesn't protect against data loss or corruption, whether malicious or accidental. Gaining that protection is what this whole book is about.

IN THIS CHAPTER

- Understanding why native backup matters
- Taking new deployment patterns into account
- Fitting into the DevOps world
- Easing stress for operators
- Addressing the gaps in data protection
- Plugging into multiple data services

Chapter 2
Building Native Data Protection for Kubernetes

Your organization relies on application uptime, and Kubernetes is fantastic at achieving that. Even so, where would you be if you lost all the data in your business?

This chapter spells out exactly why you need Kubernetes-native backup solutions. It explores many of the factors that make it absolutely essential, from the different deployment patterns to the practices known as DevOps. Read on to find out how cloud-native backup can ease stress for operators and give them more bandwidth for innovation, fill in the data protection gaps, and ensure that you're in good shape even as your apps tap into multiple different data services.

Recognizing the Need

No one needs to remind you about the importance of backup. Regardless of how you fit into the technology picture, you've probably at some point or another woken up from a bad dream involving a catastrophic loss of data.

WARNING

Numerous failure scenarios could turn that bad dream into a real-life nightmare. There's accidental deletion, general misunderstanding of the platform, ransomware, and other malicious activity. Many perils threaten your data, and how would your business survive without that data?

REMEMBER

Not to trigger any more restless nights, but there's no escaping the fact that Kubernetes is a whole new world, and for a number of reasons it can exacerbate existing reasons for data loss in the cloud. It's complex, it's still unfamiliar to a lot of people, and administrative responsibilities are not as centralized. That makes it all the more possible for accidents to happen.

WARNING

Great tools are available for virtualization-based infrastructure, but you'll need a new set of tools for non-virtualized environments. It's tempting to assign responsibility to each application team, but splitting up the backup responsibility increases both the risk and recovery time. Time is, of course, money. In this case, the average cost per hour of critical application failure is half a million dollars.

You're far better off, then, with a cloud-native backup solution in place. To back up and protect Kubernetes-based applications, you need a Kubernetes-native backup solution.

Dealing with Different Deployment Patterns

No one needs to tell you that Kubernetes is a game-changer. It's revolutionary and popular, a different way of architecting that has moved the abstraction layer, increasing flexibility for improving the running of workloads. There are fundamental differences between the Kubernetes platform and just about all the compute infrastructure before it.

What has not changed, though, are the requirements around backup. In every platform, not just Kubernetes, every admin should have a backup plan.

One of the timeless rules that can effectively address any failure scenario is called the 3-2-1 backup rule. This approach helps to answer two important questions: How many backup files should I have, and where should I store them? The 3-2-1 backup rule provides the answers:

- **3:** Have at least three copies of your data.
- **2:** Store the copies on two different media.
- **1:** Keep one backup copy offsite.

REMEMBER

Begin with the fact that you don't map containerized applications to servers or virtual machines (VMs). In contrast to VMs, all a container requires is enough of an operating system, supporting programs and libraries, and system resources to run a specific program.

Given that, you can put two or three times as many applications on a single server with containers than you can with a VM. Containers also allow you to create a portable, consistent operating environment for development, testing, and deployment. Kubernetes distributes application components across all servers using its own placement policy to boost performance and fault tolerance.

Approach these situations with a traditional data management system and you're likely to fail. You may be able to achieve backup, but if you're using tools that aren't designed in a cloud-native way, things will start to get difficult when it comes to recovery.

Add in the fact that cloud-native applications benefit from their environment's dynamic nature. To improve load balancing, containers can be rescheduled on the fly or scaled on different nodes. New deployments happen all the time, with components added and removed continually.

REMEMBER

In other words, the application is shifting constantly. What you need is a backup solution that understands the cloud-native architectural patterns, one that's fine without stability in IP addresses — one that loves change as much as your cloud-native Kubernetes application does.

Traditional backup solutions that function well in an environment of servers and VMs may be like freshwater fish in a salty ocean if you toss them into the Kubernetes environment. To meet such requirements as dynamic application discovery, instantaneous backup, platform-integration recovery, and the ability to capture all application context, what you need is Kubernetes-native backup.


TECHNICAL STUFF

To take in a little more background on this evolution, consider that physical systems required an agent-based approach to protect the data and operating system. When virtualization came along, a lot of those same backup vendors simply moved their agents to the virtual world. This created an overhead on VMs to perform the backup because it treated them like physical machines in their new surroundings.

It became clear that the best way to protect these virtual workloads was at the virtualization layer through APIs. That allowed creation of application-consistent backups in a fast and efficient manner without impacting the performance of the virtual machine.

Simply put, the same story is unfolding once again in the Kubernetes world. In theory, you could take your virtualization or physical backup process and protect some of your Kubernetes environment and data. Some, but not all could be protected, which makes things rather tricky when it comes to recovery.

Shifting to the Left

Many visualize the DevOps philosophy as an infinity symbol, the sideways "8" that creates an infinite loop of movement from left to right to left to right, and so on. You can also think of it almost as a racetrack, which is fitting in that DevOps is all about high-velocity application development cycles.


REMEMBER

When visualizing DevOps with infinity imagery, it's common to think of development on the left and operations on the right. The DevOps philosophy, when used in the world of Kubernetes, brings the needs and requirements of developers and operations together in a way that traditionally has not been the case. More is happening on the left side of that loop, hence the term "shift left."

Kubernetes is, indeed, driven by a focus on developers and their applications — and that ever-running race through development cycles. Because of the platform's design, backup solutions really must be application-centric rather than infrastructure-focused.

Developers in this environment define — as code — both application components and infrastructure requirements such as storage and load balancers. The data protection requirement needs to be integrated and speak the same language as their code. That lets them define protection schedules rather than the traditional practice of handing over to the infrastructure or backup admin.


TIP

You need a way to introduce data management tasks into the day-to-day development process. That way it's no longer just the responsibility of the operations team. A great way to achieve this is through an API approach. The backup platform must be API-first, with a cloud-native API.

We're talking Kubernetes-native API as opposed to the older REST or SOAP APIs. Authentication and authorization become seamless, and you achieve simpler application and workflow integration. Developers and operators are able to use tools they know well, such as kubecti.

Resolving Operator Challenges

Virtualization focuses on abstracting the physical hardware and leveraging that to build multiple virtual machines to house monolithic workloads. Cloud-native workloads, on the other hand, are completely focused on the application rather than the VM. This simplifies management operations by focusing on applications as the operational unit, abstracting away infrastructure or data stores.


TIP

It's essential to have a backup tool that allows flexibility and choice of approach. Not every organization will need an API to drive it's backup workflows. Some will need a dashboard to navigate the journey into Kubernetes.

Deep Kubernetes integration can hide the complexity of the underlying platform. You can eliminate or reduce manual or integration work by gaining a sharp focus on the user experience and revisiting backup workflows for cloud-native applications.

Consider that in the past, a single application might have involved a few virtual machines, whereas today's containerized applications are typically composed of hundreds of distinct Kubernetes resources, including configuration, disks, and secrets.

That's just one application. Now think about all the applications in a cluster, and your operator must understand and protect millions of components. That is, unless the application is the operational unit for backup.


WARNING

A legacy backup system is likely to pay attention to infrastructure such as disks and volumes while ignoring Kubernetes resources. That's a recipe for lots of errors in recovery playbooks with missing relationships that result in super-high recovery times.

What would be required in this unfortunate situation is a manual process to figure out which backups are required for restore, and then another complex manual process to connect restored volumes back into Kubernetes applications. That would be a huge burden for operations, even if there hasn't been any drift in Kubernetes objects between backup and restore (and that's not likely).

Filling in the Gaps

It's hard to beat Kubernetes when it comes to keeping your applications running even if there are partial infrastructure outages. Fault tolerance is a strong selling point, but it's important not to get a false sense of security and brush off the need for backup, disaster recovery, and application mobility.


REMEMBER

Again, high availability and replication are not the same thing as backup. You're still at risk for data corruption or deletion, either by accident or through a malicious act. If that happens, it can spread to all replicas and result in catastrophic data loss.

Doesn't the fact that Kubernetes is often running on public clouds make the storage failure-proof? Nope, that's a myth. The most reliable cloud storage solutions promise super-reliable uptime, but protecting data is your responsibility.

How about on-premises storage vendors — they can provide volume snapshots, right? Yes, but these snapshots are often still vulnerable to hardware failure. And if a volume gets deleted, related snapshots are usually automatically deleted.

This is where the 3-2-1 backup rule outlined earlier in the "Dealing with Different Deployment Patterns" section comes into play. Following the rule will protect you against most, if not all, failure scenarios, especially when you start adding immutability protection to your offsite copies.

Without elevated Kubernetes security privileges, you won't normally have access to such actions as quiescing file system activity. But if you have a Kubernetes-native backup system with well-defined permissions and role-based access control, you can gain database and Kubernetes workload quiescing hooks, allowing the same result with no security compromise.

What's important is that developers and operations are working together, or at least more closely than they have in the past. The common denominator linking their interests is doing what's right for the data.

Disaster recovery is another key area of consideration. Backups are fundamental to failure scenarios that can be recovered from within the same cluster, but disaster recovery involves bringing your workloads up in a completely different location.

Integrating the Ecosystems

Polyglot persistence is a term referring to using multiple data storage technologies for varying data storage needs across an application or within smaller components of an application. And we're not just talking about relational and nonrelational databases but also such storage areas as batch/data streaming and message queues.

Such varying data storage needs could arise in an enterprise with multiple applications, or within singular components of an application that need to store data differently from one another. Polyglot persistence is on the rise as Kubernetes becomes more prevalent.

The good news is that, despite this complexity, you can get richer backups for these workloads by integrating against Kubernetes for automated workload discovery. That means the backup solution can consider the application's requirements and pick the capture primitive (such as volume snapshots, application-consistent backups, and logical dumps).

With a single data service becoming a thing of the past, Kubernetes metadata can allow the backup solution to automatically discern the relationship between multiple independent data services.


REMEMBER

Getting a handle on application topology allows the Kubernetes-native backup solution to capture a consistent copy of the whole application stack, both within and across services. That lets it identify and gather data from replicas to reduce the application impact. Performance and efficiency improve, and leveraging Kubernetes parallelism optimizes restore performance.

Another angle on the complexity: More organizations are running lots of Kubernetes clusters across different environments. The backup platform must, therefore, interoperate with the rest of the cloud-native infrastructure ecosystem.


REMEMBER

Ultimately, you'll gain a better user experience, help ops teams be more efficient, and cut costs, all at the same time. One element of that improved user experience is the fact that developers and operators can keep using cloud-native tools to which they've become accustomed. For example, integrate with Prometheus to monitor and alert, and integrate with Kubernetes APIs for RBAC, logging, and the auditing you need to perform root-cause analyses.

IN THIS CHAPTER

- Focusing on the application as a whole
- Exploring and scaling the architecture
- Ensuring recoverability
- Easing operations
- Maintaining tight security in a multi-tenant environment
- Succeeding at restore while keeping it portable

Chapter 3
Following Best Practices for Kubernetes Backup

As outlined in Chapters 1 and 2, the Kubernetes environment is a world unto itself. What you used to do in terms of backup isn't enough and may not even work at all.

This chapter focuses on some of the best practices for handling backup of your Kubernetes applications. It discusses the differences in architecture that require an application-centric focus, dives into the ways to ensure recoverability, and explores how a native backup system makes life easier for operations as the application scales up and down. It highlights the importance of security considerations, even as multi-tenancy complicates matters. And it touches on the challenges of ever-evolving objects and APIs, and the importance of portability.

Capturing the Application

In the big picture, the application is the real focus of attention. The infrastructure is vital but it's there to make the application available and flexible. That has always been true, but that fact takes on special meaning in the context of Kubernetes with its keen focus on developers, the applications they create, and the speed at which they develop and upgrade them.

A platform so focused on the developer and the application requires a backup solution that also is application-centric. As put forth in Chapters 1 and 2, your backup must be Kubernetes-native to properly serve containerized applications in the Kubernetes world.


TIP

Being application-centric means understanding Kubernetes constructs rather than being focused on the infrastructure. It requires full application capture, protecting all components and not missing any resources, filters, or labels.

Tapping into the Architecture

This book focuses specifically on Kubernetes backup, an essential part of the big picture of success on this platform. There's no need for a super-deep dive on Kubernetes architecture for this purpose, but it's helpful to spend a little time on the topic of architecture to really get a sense for how to implement a good backup strategy.

Connecting the components

Figure 3-1 shows some of the main components of a Kubernetes application. Among other things, you'll see pods, services, certificates, secrets, and persistent volumes.

An application in production will consist of hundreds of components like these. The question becomes, how do you best protect and restore data as well as all the other internal components? How do you do that at scale?

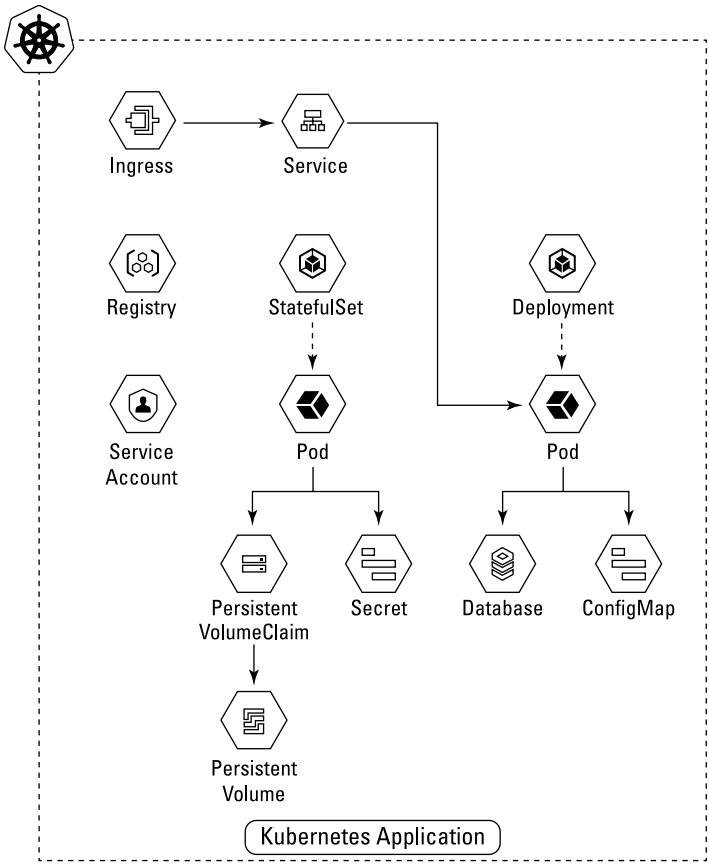

FIGURE 3-1: A basic Kubernetes application is a collection of interrelated components that all must interface with the backup and recovery system.

TIP

You'll be pleased to know that it's not all up to you. Choose the right backup platform and it automatically interfaces with the Kubernetes control plane by way of the API server. Through this integration, the backup solution will discover Kubernetes applications running on the cluster. Then, it integrates with the underlying compute, network, and storage infrastructure.

The first order of business is to discover the relationship between storage and applications. Then it's a matter of figuring out the best way to capture application data that's stored on persistent volumes, along with all the related application resources. This needs to be handled efficiently and consistently.

Next up is determining the backup data location. That can be within the storage system to enable fast recovery, or if you're running on one of the major cloud providers, going with durable snapshots. In most cases, though, it's better to store backup data in an object storage system located in a different fault domain, one that could allow geo-replication for disaster recovery. "Better safe than sorry" is a simple but applicable way to roll.


TIP

The 3-2-1 backup rule discussed in Chapter 2 is a good rule to follow when it comes to planning your backup. You need at least three copies of your data, they should be stored on two different media, and one copy should be stored offsite. Your data is at significantly greater risk if you don't follow this basic approach.

You have a couple more things to think about when it comes to storage integration in Kubernetes:

- Storage is represented as persistent volumes made available for use by the containers. You need to protect this key business data.
- Where are you going to keep that data? Local block storage? You might consider an object storage platform, such as Amazon S3 or Microsoft Azure blob storage, if you're running Kubernetes on-premises or offsite. Your consideration of secondary storage for backup data needs to reflect flexibility, choice, and ease of use.

So, your basic to-do list in this area includes:

- Discover the relationship between applications and their storage.
- Determine how you'll capture application data on persistent volumes.
- Decide where you'll keep the backup to ensure you're in compliance with the 3-2-1 guidance.


REMEMBER

Whatever platform you use to protect Kubernetes applications, it must automatically discover all the components running on the cluster and treat that application as the unit of atomicity. It's vital for the application to include the state spanning across all storage volumes and databases, as well as the configurable data in Kubernetes objects, such as configmaps and secrets.

Scaling up

There's so much under-the-hood in an application running on Kubernetes. Applications are broken down into hundreds of different components with their own lifecycles, thanks to microservices and Kubernetes support for everything from configuration to secret handling. This complexity is largely visible only to Kubernetes.

It takes a cloud-native backup solution to handle all the millions of components in large clusters and understand the relationships between applications, the data they use, and the related Kubernetes state. Only a cloud-native solution can capture all of this at scale.

REMEMBER

Just as Kubernetes and cloud-native applications are built to easily scale up and down in response to load, backup solutions must be able to follow suit. Here's what you should expect from your backup solution:

- Adopt the same cloud-native architectural pattern so that it can scale with application and cluster changes.
- Scale down to zero when not in use.
- Do these things automatically, with no need for manual operator input.

REMEMBER

With a backup platform that grows along with the cluster, you achieve the best performance. But you also save money, because its resource footprint is tied to current, instantaneously variable requirements rather than peak load. Also, because the backup system scales up in linear fashion with the application and cluster growth, it's a whole lot smoother. No step-function jumps like you'd see with an appliance-based model.

The scale challenges get all the more complex with Kubernetes multi-cluster use increasing. There may be thousands of name spaces per cluster and hundreds of Kubernetes resources per name space.

You see multiple clusters across environments, such as dev, staging, and prod. But there are also splits across the boundaries of applications, security, and teams. And clusters are deployed across multiple availability zones, regions, clouds, and on-premises data centers.

That's just a crazy potential operational burden if it all had to be handled manually. It's really not even possible, to be honest, unless you have a cloud-native backup platform that can handle multi-cluster operations and offer global visibility across it all.

Here are your essential steps related to scaling:

- Ensure that your backup solution can scale up and down in concert with the applications it is protecting.
- Determine how your solution will respond to the challenges of Kubernetes multi-clustering.

Planning for Recoverability

Recoverability is the Holy Grail that this book is all about. It takes significant planning and execution; it's far more than just re-creating Kubernetes objects and storage volumes.


TIP

After all, your container-based application is complex, with lots of Kubernetes components. The first step is to create an execution plan that takes care of the following:

- Verify cluster dependencies.
- Create new Kubernetes views of the data that must be restored.
- Determine the compute infrastructure and Kubernetes cluster where recovery will be initiated — such as a cross-availability zone recovery.

With that plan in place, you'll have to identify the backup data sources, such as object storage, snapshots, and backups. You must prepare the destination storage, which involves storage class remapping and storage platform changes, among other things.


TIP

Determine whether the plan needs to be transformed. Consider such things as regeneration of TLS certifications, DNS changes, and editing stale secrets. And then you must update Kubernetes application components so that they reflect the new storage resources the recovery will create.

With all this planning completed, the backup platform must translate the plan into the appropriate Kubernetes API calls to

create the resources that will be needed (examples include calls to create a load balancer or re-create a secret). All resources and microservices that are part of a cloud-native application must be redeployed with the correct configuration. See Figure 3-2 for a visualization of this process.

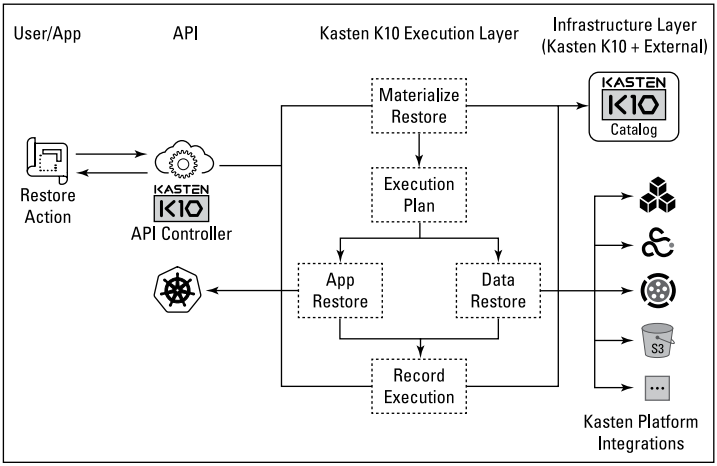

FIGURE 3-2: A look at the restore process, including the API calls that create all necessary resources.

The bottom line is to be able to restore all the application components wherever you want them, and you should be afforded the granularity needed so you can restore only an application subset if desired — the data volume, for example. Be sure your backup solution gives you the power to pick the appropriate point-of-time copy of the application. And is it too much to ask that all this be made as simple as possible? Not at all.

In short, disaster recovery requires careful planning combined with the right tool for the execution:

- Build your execution plan focused on cluster dependencies, views of the data, and where the recovery will be initiated.
- Identify data sources and destination storage.
- Determine whether transformation is necessary and how components will be updated.
- Turn it all into Kubernetes API calls.

Focusing on Operations

Remember some of the things that make Kubernetes so popular: It's quick and easy for developers to deploy applications, upgrade them, and do it all at scale. The last thing you want to do is have a backup platform that gets in the way of that efficiency. That would be frustrating at best, and at worst might inspire developers to find their way around best-practice processes.

TIP

Indeed, ensuring that everyone adheres to operational best practices is always a challenge, particularly when you're rolling out new tools and services and capabilities in a dynamic infrastructure. Your Kubernetes-native backup platform must:

- Be able to be used at scale
- Give operations teams the workflow capabilities they need
- Meet all the various requirements for compliance and monitoring
- Not give developers headaches

From the developers' perspective, there shouldn't be any requirements for code, packaging, toolchain, or deployment changes. Operators should also be able to give developers self-service capabilities.

REMEMBER

For example, developers ought to be able to restore their own applications and customize and extend backup operations for their data services. They should have control over cross-service coordination and quiescing and the use of custom or database-vendor tools. Also, developer interactions with the backup platform must be API-driven.

As for operators, they'll be pleased if the backup solution frees them from focusing on all the hundreds of Kubernetes components that are part of the application. They want backup policies that are totally automated and allow them to pay attention to the big picture of the application rather than individual resources and storage infrastructure.

When it comes to the minute details of the policy and all the application components that must be protected, that work should need to happen only at the time of executing the policy. From that point on, there should be no need for manual updates to capture

all the components as the application changes (which, of course, it always does).

TIP

Here's another best practice point to help keep things as simple as possible. Create broad, label-based backup policies that are able to automatically pick up new applications as soon as they come along.

For example, have a policy that matches all applications that use MongoDB, or a policy governing apps deployed via the Helm tool. Play your cards right and you won't force operations teams to build manual change control processes. And equally important, you'll be sure that applications never fall out of compliance with backup service level agreements as they are created or retired.

To put it all in the simplest terms, aim for a backup solution that gives everyone what they want and need. It's, indeed, possible to deploy a platform that meets the needs of both container operations teams and developers, so don't settle for less.

BACKUP JOBS VERSUS INTELLIGENT POLICIES

What are the benefits of establishing intelligent backup policies, rather than setting up specific backup jobs? Consider the differences and see for yourself.

Backup jobs describe each step to the software stack, right down to the time to run the job. A backup job is largely ignorant of the data center architecture and needs you to hold its hand so it won't clobber workloads or impact network capacity.

Policies, on the other hand, are more outcome-focused. Rather than dictating schedules or imperative changes, you define a recovery point objective or a desired outcome. Once established, the policy is always running, always watching for issues and needs, and ready to ensure the desired outcome.

You can see which option is more automated and which one needs more human attention. And not only does that human attention take time, but it also introduces that unfortunate tendency of humans to make mistakes when they have lots to configure and remember.

Regarding operations, here are your key to-dos:

- Ensure that your backup solution really is a solution and not a problem. It needs to boost efficiency, meet workflow needs, and fulfill requirements.
- Determine where intelligent policies might ease pain and avoid the need to create specific backup jobs.
- Build policies that discover on their own new applications that fit their focus.

Ensuring Security

Whether you're deploying in a public cloud, using on-premises infrastructure or any kind of hybrid environment, security has to be front and center. A glance at the headlines just about any day makes it clear that security is more vital than ever. And the increase in multi-tenancy gives you that much more reason to keep an eye on security.

Layering in protection

On the plus side, Kubernetes incorporates lots of security features, including network policies that protect internal application components and the data services associated with them. Kubernetes not only blocks access to these components from outside the cluster, but it also puts up a stop sign to untrusted applications running in the same cluster.

REMEMBER

That's a good thing, but it also means you're not going to be able to run backup solutions outside your Kubernetes clusters. They won't be able to discover applications — nor access them, such as to quiesce — without weakening isolation policies. The answer is a well-architected Kubernetes-native solution that's able to embed itself into the control plane.

Another challenge you might run into is the need for self-service capabilities, as developers take on more and more infrastructure responsibilities. Your backup system deployment must include controls around identity and access management, and you can't go without role-based access control (RBAC). That's how you govern which users and groups get which specific levels of access, restrictions, or user privileges with regard to the backup platform.

Your scoped access, in fact, should be granted using the same roles and tools as defined by Kubernetes. That's better than introducing more role management systems for your teams to learn.

With respect to the operations team, you want a least-privilege approach for common tasks, such as monitoring backups, verifying the success and integrity of backups, and handling requested restores. You can create specific use cases as needed. You may allow developers permission for fast restore and clones from snapshots, while access to backups in offsite secondary storage is reserved for only certain people.

Beyond that, your cloud-native backup platform has to have deep integration into the cloud's identity and access management systems, key management systems, and certificate management. Also, a data management system that is truly Kubernetes-native will integrate into the cloud provider's authentication solution without the need for extra user or group management, nor new tools or APIs for RBAC policies.


TIP

Keep in mind that Kubernetes delegates data encryption to the underlying storage system and backup platform. You need to be certain that your application data is never stored or transferred in plain text. For that reason, the backup platform must:

- Understand Kubernetes certificate management
- Work with storage-integrated key management systems
- Support customer-managed encryption keys through the Kubernetes Secrets interface

To cite object storage as an example, if you have an on-premises Kubernetes application deployment that offloads backups to AWS S3, your data will get from here to there through an external Internet connection. The backup platform must ensure that data is encrypted through a protocol such as TLS.

The encryption consideration doesn't stop once the data gets to the place where it's ultimately stored. Your system must ensure it's encrypted there, too, because if it's not encrypted at rest, it's not safe there. Putting in place RBAC and related policies won't suffice — you also need proven encryption algorithms such as AES-256-GCM with per-application encryption keys. Otherwise, you're at risk for accidental data leaks, or even malicious copying.

WARNING

And speaking of malicious risks, your system must always be on the lookout for Kubernetes-targeted malware and ransomware attacks. The threat grows all the time as so many Kubernetes applications are external customer-facing.

With that in mind, check out the protection known as *immutability*. This protects against any modifications and deletions of your backups in object storage — including malicious encryption or deletion. That helps neutralize the threat of ransomware.

Add it all up, and you can see the need for a backup solution that is Kubernetes-native but can also create reliable backups that are independent of Kubernetes and the storage system. It must have deep integrations to allow for fast and automated restores.

Security is a critical job, and here are some key considerations:

- Embed a Kubernetes-native backup solution into the control plane.
- Create self-service capabilities that don't compromise security.
- Ensure that your solution always provides encryption and other layers of protection.

Living with multi-tenancy

When you think of legacy products, backup systems are usually admin-specific, and there could be just a few people who are users of the entire system. There could be view-only users, with much fewer having full permissions. For restores, you might have to file a ticket or ask someone.

Kubernetes clusters tend to be multi-tenant, and you're always having developers and developer teams dynamically added and removed from the system. They have their own spaces and their own applications, and everyone needs to basically mind their own business.

TIP

Any backup system in this environment needs to have self-service, with developers having control over network, firewall, storage provisioning, and restores — but only for their own applications. The system must be scoped so that developers only have visibility and access into the apps that belong to them.

For example, if one developer owns application X, they should see only application X, and definitely not application Y, which belongs to a different developer. And that developer of application Y should not be able to have any visibility or access into application X.

How do you enable this? Among other things, you have access control based on role (as you have already seen earlier in this chapter). Some developers need permission for backup and restore, while others may have access only to restore, with backup controlled by someone else. Some just have view access. The idea is to establish a variety of different groups, each with appropriate access.

How can this be automated? Methods such as Open Policy Agent, or OPA, allow you to essentially express policy as code.


TIP

The key is that the approach has to be Kubernetes-native. You don't want separate systems for managing users, such as an RBAC system outside of the platform. The backup system needs to inherit the APIs so that there aren't extra tools or user management. Anything else would be too cumbersome.

Your key considerations with respect to multi-tenancy:

- Ensure that your self-service options limit developers to only their own applications.
- Strive for consistency so you don't have to maintain separate systems for managing users.

Transforming Restore Support

Kubernetes is certainly a fast-moving world. There's a public release about every three months, which means objects can change quickly. Add in the fact that organizations may jump ahead multiple versions at a time when they upgrade; they may not have kept up with every upgrade in the interim. And also consider the portability that goes hand-in-hand with Kubernetes container-based environments. How do you make it all work?

Keeping up with change

WARNING

The tendency for release cycles to get shorter and shorter creates situations in which you may have a backup from, say, a month ago or a year ago, and it doesn't recognize objects that have changed. And it's not just Kubernetes APIs but also custom resources or other kinds of components. When that happens and you try to restore with an API version that the system no longer supports, the restore will fail.

TIP

That's why the system needs the ability to transform characteristics and descriptors of system components from older versions to updated ones. For example, it must be able to evolve the object's API from the old version to the most recent version. Or deal with a certificate that is short-lived. Or secrets that need to be updated.

Transformation is also key as Kubernetes clusters hit the road and move from one place to another (as discussed in the following section on portability). Moving from on-premises to a public cloud? Transform allows the restore process to be simple and portable.

In this fast-paced and dynamic environment, you can't get past the fact that things get outdated or expired, or API versions change, or some other thing is updated, or your cluster takes advantage of Kubernetes portability. When you restore, you also must be able to transform.

So, with regard to transform, keep these keys in mind:

- Ensure that your solution embraces transform to keep on top of the ever-accelerating pace of releases.
- Use transform to help enable portability.

Enabling portability

Portability is one of the many great capabilities provided by Kubernetes, and a backup platform can use this powerful feature to unlock a lot of use cases. See Figure 3-3 for some of the possibilities.

Here are some of the potential portability use cases:

- Across namespaces in the same cluster
- Across storage systems

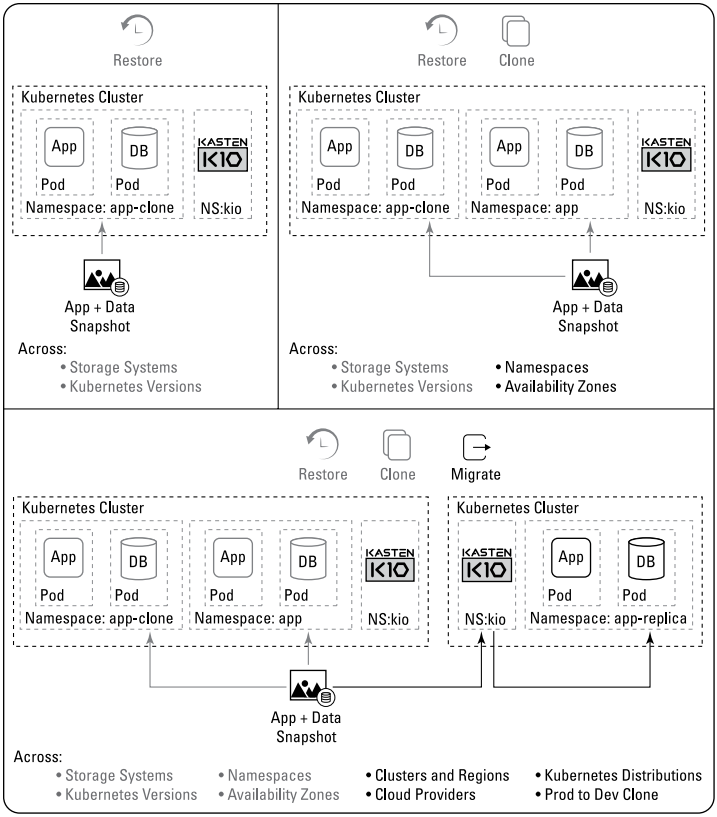

FIGURE 3-3: The power of portability in Kubernetes-native backup.

- Across Kubernetes clusters, distributions, and versions
- Across availability zones in the same region
- Across regions in the same cloud
- Across cloud or hybrid environments
- Across test and development environments

REMEMBER

As you know, you've got a lot of ecosystem diversity in Kubernetes, both on-premises and in-the-cloud. Also, more and more enterprises use a hybrid cloud model to run container based applications. With that in mind, your backup and data management platform must be able to migrate applications across source and destination clusters that could be running on different infrastructures.

Check out Figure 3-4 to see what you're up against when migrating a workload from Amazon Elastic Kubernetes Service over to Microsoft Azure Kubernetes Service.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
      name: gp2
provisioner: kubernetes.io/aws-ebs
parameters:
      type: gp2
      fsType: ext4
```

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
      name: managed-premium-retain
provisioner: kubernetes.io/azure-disk
reclaimPolicy: Retain
parameters:
      storageaccounttype: Premium_LRS
      kind: Managed
```

FIGURE 3-4: Migration from EKS to AKS and how the terminology must transform.

Difference in storage classes is just the beginning with regard to different distributions, even if those distributions are based on the same underlying version of Kubernetes. It's critical that your backup platform can perform restores across these varying distributions and infrastructure options, automatically transforming the application backup to fit into the restore environment.

REMEMBER

It's a heavy lift but a vital one. The backup platform has to be able to understand all the application dependencies and successfully translate them across environments.

To ensure that your migration execution is successful, you must have a migration plan to be certain your infrastructure, cluster-wide, and application dependencies are available or are transformed to an equivalent resource. Keep in mind that you must not just migrate containers and storage volumes but also handle in-flight modification of such things as FQDNs, secrets, and DNS addresses.

In short, portability depends on considerations such as these:

- Ensuring that your backup and data management solution can migrate applications across clusters operating on different infrastructures
- Having a migration plan to ensure all dependencies are available or transformed to an equivalent resource

IN THIS CHAPTER

- Becoming familiar with the cloud environment
- Uniting backup and disaster recovery
- Thriving in a multi-cluster environment

Chapter 4
Attaining Cloud-Native Application Mobility

You could be forgiven if you worked your way through the previous three chapters and decided that achieving efficient backup and disaster recovery in today's container-based world sounds incredibly complicated. Certainly, it's sophisticated work, but with the right tools, it doesn't feel daunting. What you need is a centralized place to manage the work.

This chapter explores your ultimate desire to achieve application mobility with solid backup and successful disaster recovery always in place. It discusses the blurring lines between backup and recovery, and the challenges you're overcoming as the numbers of clusters and resources grow exponentially.

Getting Going in the Cloud

TIP

For those relatively new to Kubernetes, the easiest way to get acclimated is to run it in the public cloud. You'll find Kubernetes available as a service from such providers as Google, Microsoft, and AWS.

Public cloud providers keep the service maintained and keep tabs on such details as certificate rotation, version, and patching. They also provide options for storage and will manage everything under the cluster.

Get an even deeper dive into the makeup and architecture by deploying your own Kubernetes cluster. Bootstrapping from scratch is certainly an option, but keep in mind that deploying your own cluster means all the things the service provider would handle for you in the public cloud become your responsibility.

Blurring Lines

You would not have opened this book if you didn't already know just how vital backup and disaster recovery are to the success — and even survival — of your organization. That's certainly not breaking news.

You also probably already know that backup and recovery have not always been a piece of cake. In years past, and certainly before Kubernetes arrived on the scene in 2015, there often were separate platforms for different functions, and more than one copy of data.

REMEMBER

In the Kubernetes world, today's user wants to capture data once. There should be one set of data that can be used in multiple contexts; for backup, disaster recovery, application mobility, moving across different clusters, and the like. The lines between backup and disaster recovery are blurring, but the tools at your disposal will help ease the way.

These concepts speak to the need to offer self-service to developers and operators performing multiple tasks. It's important because of the multiple use cases involved, and none of this should be an afterthought. Portability and application mobility shouldn't be extremely complicated, which is why support for transformation (as discussed in Chapter 3) is so vital.

When you talk about moving across different clouds or across regions, you're talking about transforming things. How do you transform things when it comes to the features required for mobility? It's one of the basic capabilities required from a data management platform.

Adding Clusters

A lot of organizations are running many, many clusters. It's not uncommon to see 50-plus clusters in production. These organizations likely started small and began thinking bigger, with larger clusters or multiple clusters for different events, such as test, dev, staging, and production.

Clusters may run workloads based on a variety of attributes. Examples include specific applications, security domains, and deployment readiness, by individual business unit and perhaps by geographic divisions.

There are many good reasons to take a multi-cluster approach. Benefits include customizing clusters to fit workload requirements (think node size), and perhaps reducing the blast radius. The default these days is multi-cluster, and even a lot of small clusters have three, five, or ten nodes.

REMEMBER

Do the multiplication and the scope of the task of managing this becomes astonishing. Factor in the number of clusters and applications and the number of resources and volumes per application, and pretty soon it's clear that you can't possibly manage them all independently.

TIP

For a series of instructive case studies showing implementation details for Kubernetes backup and recovery, go to `kasten.io/kubernetes/resources/customer-case-studies`.

That's because you're managing backups and managing permissions tied back to RBACs mentioned in earlier chapters. Multitenancy and security aspects complicate matters all the more.

Your backup and disaster recovery system should provide a centralized place to manage all this complexity, offering a big-picture view of what's happening. You need to be able to drill down into different contexts and support mobility across these clusters.

It's important to make it easy for operators to handle all this from one place. It's vital to make developers' jobs simpler, too. Your goal is to simplify by not just adding clusters but enabling multi-cluster app mobility.

Here's a look at some of the data management requirements for seamless operations in multi-cluster environments:

- **Security:** Not just for data but for operations, offering appropriate visibility to the right users and systems.
- **Easy setup:** You must be able to establish multi-cluster operations without complicated installations, and it should be simple to set up and manage policies and resources.
- **Automated discovery:** This includes discovery of Kubernetes applications, backup policies, and any changes across all clusters.
- **Global view:** Insist on a single pane of glass for a real-time look at global status.
- **Global policies and resources:** Teams should be able to define global policies such as backup frequency, as well as resources such as target storage location.
- **Flexible cluster grouping:** Let users easily define flexible and arbitrary logical grouping of clusters for distribution of global policies and resources.
- **Single cluster drill-down:** Within this big-picture capability, you must be able to take a close look at any individual cluster.

REMEMBER

Here's the moral of this particular story. You need a Kubernetes-native backup and data management system that unwinds all this complexity.

IN THIS CHAPTER

- Getting to know the ecosystem
- Fitting in with the way operators work
- Enabling auditing and security policies
- Making logging happen and increasing visibility
- Keeping the backup system current

Chapter 5
Settling into the Cloud-Native Ecosystem

An effective backup system needs to align lots of stars to keep everyone happy and your data secure. The system needs to fit into the organization's workflows and preferences and policies. It also needs to keep itself on the cutting edge in an ecosystem that changes at blinding speed.

This chapter discusses the need to integrate the system with the tools your operators already prefer to use and with the security policies and auditing needs already in place. It explains how the system must facilitate logging and give operators visibility and observability into how it's functioning. It also describes why new releases must happen with astonishing frequency.

Understanding the Data Management Ecosystem

As shown in Figure 5-1, the Kubernetes data management ecosystem consists of four important subcomponents that each play a pivotal role in an overall solution:

- **Data services:** For a complete solution, data services should have prequalified integrations with leading data sources, including relational and NoSQL systems.
- **Kubernetes distributions:** The distributions should have support for all major cloud-based, managed Kubernetes offerings and all leading on-premises distributions.
- **Storage infrastructure support:** This should be provided for Container Storage Interfaces as well as direct storage integrations for best efficiency.

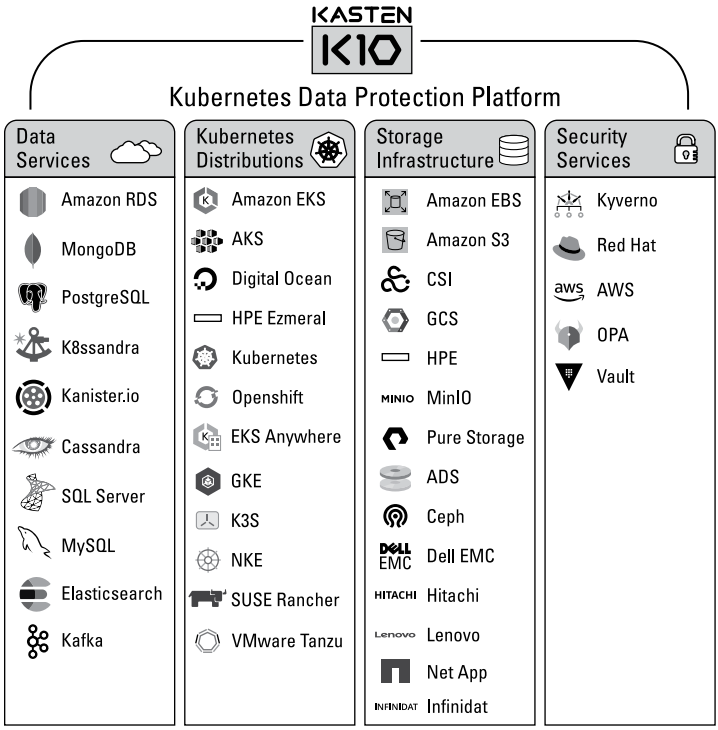

FIGURE 5-1: The four important parts of the Kubernetes data management ecosystem.

- **Security services** Kubernetes security should address securing the cluster's infrastructure, managing access control, protecting sensitive data and monitoring for vulnerabilities, and ransomware attacks.

Full-spectrum consistency, database integrations, automatic application discovery, multi-cloud mobility, and a powerful web-based user interface round out the needed capabilities of a strong solution.

Integrating with Prometheus and Grafana

Older backup systems have introduced lots of useful features, including logging, reporting, alerting, and auditing, but they have not always made life easier for operators. Now, the measure of success of a good backup system should not be just how it enables backup but how it achieves zero friction for those in ops. How can it fit into their workflow and live happily in the infrastructure to which they are accustomed?


REMEMBER

In the Kubernetes ecosystem, Prometheus is the gold standard for monitoring. Grafana is an open-source visualization tool often used with Prometheus. It makes sense for your Kubernetes-native backup platform to integrate with both. Operators will have the right data for creating dashboards, building triggers, setting up alerts, and gaining more visual insights to overall system health.

Learning through Auditing

With something as critical as data, you always want to know exactly who did what with it, where, and when. Your backup system needs to give you that visibility for auditing purposes down the road.


REMEMBER

Rather than provide its own separate auditing system, the Kubernetes backup solution should accomplish this by hooking into Kubernetes auditing transparently. This works well in the multi-tenant environment and allows user permission and identity to go all the way down into the infrastructure.

A backup system must be in tune with the user so that the right audit logs show up. It's not enough to know that "the backup system did X." You need to know that "the backup system did X on behalf of user Bob." That ability must be present for the most effective and accurate logging capabilities.

Tying into Network Policies and Security

It's important to never weaken the security posture of an application. For example, a container running a database on an SQL system must not be exposed to the outside world just for backup.


TIP

Even if the database is internal to the application, a backup system that is embedded in the Kubernetes control plane will be able to get at the database without exposing it to the wider world. That lessens the dangers of ransomware and other malicious behavior.

Advancing Logging

All users will have their own log system. The backup system will need to be tunable with respect to logs, and able to be altered to fit the organization's needs.

There also should be the ability to extract logs, and most definitely not lose them. You need a mature interface with whatever third-party logging system you have in your environment.

Improving Observability

Observability involves understanding the monitoring aspects of the backup system and having the data to correlate events. Your organization needs full details about applications that aren't behaving properly, or which applications were busy and caused the backup to take longer.

Also key is observability of the backend aspect of the backup operations when pushing data to an object store. You want to be able to monitor that and track the flow of data.

Keeping Up with Kubernetes Release Cycles

Kubernetes updates every three months, and that drives many downstream system changes including with backup and recovery. The backup system needs to keep up so that the organization is always protected.

All the commercial distributions complicate matters, as they have their own release cycles. Customers upgrade very often, too, because they know that within three to four release cycles, if not sooner, their version will become obsolete.

REMEMBER

The backup, then, needs to ensure it's always on top of these changes so that it can do the job of protecting. Given the rapid pace of change swirling all around this environment, updates once or twice a year will simply not be sufficient.

Indeed, expect multiple releases a month. Kasten by Veeam, for example, typically releases updates every two weeks, and in some cases even more often than that. That's the only way to be responsive to the way the rest of the ecosystem is changing.

Chapter 6
Ten Key Takeaways about Kubernetes Backup

There's plenty to learn about Kubernetes backup, and the preceding 40-something pages cover a lot of detail. But if you want to stick with the most crucial points, read on, as this chapter spells out some of the key takeaways.

Understanding the Architecture

» **Keeping up with Kubernetes:** In the world known as Kubernetes, you can blink a few times and end up missing a lot of action. For example, when Kubernetes first emerged in 2015, applications running on the Kubernetes platform were often architected to be stateless, but not anymore. This control plane is an ideal vehicle for both stateless and stateful applications. That's a powerful development, but it means you need to mind your data.

- **Empowering developers:** Kubernetes is all about making apps work well, and developers will tell you that's what they're all about, too. But while they're in the driver's seat, they're steering down a new road with some curves they've never navigated before. They're designing application components but also defining infrastructure requirements in ways they may not have done in the past. There's some added risk there, which is one reason why you need robust backup and recovery made for this ecosystem.

Focusing on Operations

- **Differing on deployment:** The Kubernetes compute infrastructure is wildly different from everything that has come before. You're on it to build containerized applications, the components of which Kubernetes will distribute across various nodes to improve performance and fault tolerance. What you're not doing is mapping these containerized applications to servers or virtual machines. Your backup solution needs to understand these cloud-native architectural patterns to know how to do its job effectively.
- **Growing and shrinking:** Containerized apps in the Kubernetes environment can scale up and down as the load dictates. So, too, must the backup solution. It needs to have the same cloud-native architectural pattern so that it can scale with the changes in the application and clusters. That ensures its effective functioning, and also happens to be the most cost-effective way to go.

Upping the Backup Game

- **Choosing a native system:** You know how important backup and disaster recovery are, but don't expect to plug your shiny new stateful Kubernetes-based applications into a legacy backup tool created for virtualization-based infrastructure. That's just a whole different kind of technology from what's happening in Kubernetes. Your only safe bet for Kubernetes backup is a Kubernetes-native backup system that's an apples-to-apples fit.

- **Confusing availability with backup:** Downtime can be incredibly expensive. That's one reason why so many people love running containerized applications on Kubernetes. But the replication and assorted other magic that makes this happen is not the same thing as backup. Your data is still very much at risk unless you set out quite deliberately to protect it.
- **Keeping it recoverable:** Kubernetes-native backup and disaster recovery pays close attention to the relationships and dependencies between the many components in a given container. Kubernetes is going to move things around all the time to keep the lights on and maintain scalability (think about the burst capacity needed by a retail application on Black Friday). The backup needs to know how to put it all together. You'll need a detailed recovery plan that includes views of all data that'll need to be restored. You'll have to identify backup data sources, as well as destination storage. And you'll need to know what must be transformed in the recovery.

Staying Secure

- **Plugging into native security measures:** Kubernetes has smart security measures built right in, and that helps keep the right people in and the wrong people out. Not just people but also components of applications. If your backup and recovery solution is running outside your Kubernetes clusters, it really can't do its job of discovering and accessing applications.
- **Loving your neighbors:** Ever live in an apartment complex, sharing walls with neighbors? You learn to balance mutual trust with a healthy level of self-protection. That's the way Kubernetes clusters tend to be, with applications sharing spaces through multi-tenancy. Developers need access and visibility into their own apps but not into other people's. And the backup system must follow the same rules, employing the same kinds of access controls to ensure everyone and everything stays in its rightful place.

Changing the Speed of Improvement

- **Keeping up to date:** Some people drive their cars until they've racked up 150,000 or 200,000 miles, while others like to sign a new car lease every couple years. You won't find computer code with high mileage in the Kubernetes world. Inhabitants here definitely keep up with the latest trends and technologies. So must your backup solution. Expect new releases on a super-regular basis, because that's the only way you can ensure ongoing protection in a world where updates happen at a dizzying pace.

Kubernetes native backup and restore, DR, and application mobility

Kubernetes is the fastest-growing infrastructure software, well on track to become the leading enterprise application platform. While Kubernetes provides high availability and scalability of application services, these benefits do not extend to your data. As a result, data protection is a critical priority for Kubernetes workloads. This book provides you with the information and tools you need to properly protect your Kubernetes applications.

Inside...

- Understanding Kubernetes and cloud native applications
- Building Kubernetes native data protection
- Best practices for Kubernetes backup and recovery
- Attaining cloud native application mobility
- The cloud native ecosystem

Steve Kaelble is the author of many books in the *For Dummies* series, and his writing has also been published in magazines, newspapers, and corporate annual reports. When not immersed in the *For Dummies* world or writing articles, he engages in healthcare communications.

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-394-22648-1
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.