5 Kubernetes Backup Best Practices

Addressing Data Management Needs
for a Container-Native World

VEEAM kasten by Veeam

WHITE PAPER

CONTENTS

Executive Summary

Veeam®, the leader in backup solutions that deliver Cloud Data Management™, is focused on helping customers backup, secure and manage all of their cloud, virtual and physical workloads. Veeam's solution now extends to containerized workloads with Kasten. Kasten by Veeam, is the leader in Kubernetes application backup and mobility. Kasten's K10 Data Management Platform helps you achieve the best practices described in this white paper to adress cloud-native data protection needs for enterprises

Applications using microservice-based architectures have quickly gained traction in the enterprise – an evolution from earlier monolithic and virtualized approaches. Kubernetes has emerged as the dominant, indeed de-facto, container orchestration platform. In parallel with the adoption of containers, newer organizational approaches (usually termed DevOps or ITOps) are being rapidly adopted where software development and IT operations roles are being "combined" to deliver higher agility while maintaining quality. In this environment, developers have more latitude in their tool selection and exert a greater influence on operations.

Data, the most important asset in any enterprise, has seen its value increase further with the widespread adoption of Kubernetes given the artificial intelligence and machine learning stacks being deployed on it. However, this information and the associated software assets are subjected to hacking attacks, often to devastating effect. Privacy concerns have also led to stringent regulations. Taken together, these have made data management a "front and center" concern for IT teams spanning three distinct operational use cases: backup and restore, application mobility, and disaster recovery.

Operations teams, that manage infrastructure and applications, do not operate in a static world either. The cloud-native environments they are now managing are dynamic and complex. The teams need to support a varying mix of traditional, virtualized and containerized workloads based on the use of various data services, including relational and NoSQL databases. Their deployments can be spread across on-premises, and often, multiple cloud environments and often rely upon a range of storage solutions from multiple vendors. Any data management solution deployed in these environments will need to balance the needs of operators and developers by being both operations-focused and developer friendly.

Kubernetes further adds to the change in the IT landscape as it is fundamentally different from platforms based on earlier technologies. Accordingly, it requires a different approach to backup, one that we describe as a Kubernetes-native backup. There are seven reasons why Kubernetes-native backup is critical.

7 Reasons Why Kubernetes-Native Backup is Critical:

Kubernetes Deployment Patterns

The Kubernetes platform is fundamentally different from earlier compute infrastructures. There is no mapping of applications to servers or VMs. A backup solution needs to understand this Kubernetes-native architectural pattern, and be able to deal with continuous change.

DevOps and "ShiftLeft"

High-velocity application development and deployment cycles are the norm in Kubernetes environments. Consequently, this requires that backup solutions be application-centric and not infrastructure focused.

Kubernetes Operator Challenges

Operations requires ease of use to accelerate an IT team's production journey to Kubernetes deployments. Backup solution with CLI access and a clean API along with a powerful yet easy-to-use dashboard is critical.

Application Scale

Kubernetes-based microservices comprise hundreds of discrete components with independent lifecycles visible only to Kubernetes. A Kubernetes-native approach to backups, keeping applications as the unit of atomicity for consistent operations is an imperative.

Protection Gaps

Relying solely on high availability or replication capabilities can lead to data corruption or catastrophic data loss. A backup solution that works transparently against a wide range of Kubernetes application stacks and deployment methods is required.

Security

Kubernetes security features deny access to internal application components and their associated data services from not just outside the cluster but also to other untrusted applications. A well-architected Kubernetes-native backup solution that can embed itself into the Kubernetes control plane ensures consistent security operations.

Ecosystem Integration

Polyglot persistence, where multiple data services are used within the same application, has coincided with the growth of Kubernetes. A backup solution with workload knowledge to select the capture primitives best suited to the application's requirements as well as interoperation with the rest of the cloud-native infrastructure ecosystem is key.

“Containers in microservice architectures are foundational to many emerging cloud-native applications,” said Phil Goodwin, research director, IDC. “Protecting and recovering containerized environments has very different requirements from virtual infrastructure alone. The Veeam-Kasten combination could allow enterprises to protect and recover Kubernetes-based cloud-native container applications along with virtual and physical workloads from a single cloud data management framework. With both vendors being part of the Insight Partner’s portfolio, and Veeam already partnering with Kasten to enable application centric container data protection capabilities, this acquisition seems like a natural next step. Having Kasten now be a part of Veeam positions the combination very well to meet the majority of data protection and recovery needs for multi-cloud environments.”

PHIL GOODWIN, RESEARCH DIRECTOR, IDC

Based on our customer experience and in light of the key considerations that warrant particular attention above we have arrived at following five best practices for Kubernetes backups:

Architecture 	The platform used to protect Kubernetes applications needs to automatically discover all the application components running on your cluster and treat the application as the unit of atomicity. The application must include the state that spans across storage volumes, databases (NoSQL/Relational) as well as configuration data included in Kubernetes objects such as configmaps and secrets.
Recoverability 	The data management platforms must allow you to restore the application components you want and where you want them. You should also have the granularity to restore only an application subset such as the data volume. The approach must make restoring simple and powerful by also allowing you to select the appropriate point of time copy of the application.
Operations 	It is important to ensure that a Kubernetes-native backup platform can be used at scale, provide operations teams with the workflow capabilities they require, and meets compliance and monitoring requirements. Operators should be able to give self-service capabilities to application developers without requiring application code or deployment changes.
Security 	Controls around identity and access management and role-based access control (RBAC) must be implemented. RBAC allows different personas in an operations team to adopt a least-privilege approach to common tasks such as monitoring. Encryption at rest and in transit must always be implemented to ensure that data is secure whenever it has left the compute environment.
Portability 	Living in a multi-hybrid-cloud world, a cloud-native data management platform needs to be able to be flexible in the support for multiple distributions and offer capabilities that allow for the portability of workloads and applications across all these diverse environments. Portability capabilities are required across multiple use cases including application restoring, cloning, and migration.

CHAPTER 1

Introduction

1

1.1 Modern IT Environments

Microservice architectures based on the use of containers are increasingly the norm in modern IT organizations. IT teams have worked diligently to both refactor existing applications and adopt cloud-native architectures as the default for new development. Kubernetes has emerged as the de-facto container orchestration platform and very few organizations would even consider using a different container platform today. The popularity of Kubernetes was demonstrated at the last KubeCon event (San Diego, 2019) which attracted nearly 12,000 attendees (up from around 8,000 the prior year in Seattle) representing more than 2,600 companies.

Along with this technology evolution, a change in the organizational and process approaches being adopted by IT teams has also been seen. These are typically described as DevOps or ITOps, and reflect a newer, more nimble and agile manner of working with developers exercising greater freedom in their selection of tools and playing a larger role in operational matters. As the mindset has evolved, so have the measures that the teams focus on – with greater attention being paid to metrics such as code release time, deployment frequency, time to restore and change fail rate.

[1] CNCF Survey 2019
[2] State of DevOps 2019

1.2 Information as a valuable asset

Not only have IT teams and their organizational philosophy evolved, but IT has become an even greater enabler of competitive advantage than in the past. Entirely new, and very successful, businesses have been spawned that rely on the effective and novel use of IT. This is driven by improved access to scalable computing, networking and storage resources; the growth of and access to data, and the adoption of advanced machine learning and artificial intelligence techniques. The unprecedented growth of data, a firm's ability to extract value from that data, relentless attacks on IT systems by state and non-state actors, and the need to comply with stringent privacy and data protection mandates has led to a clear recognition of corporate information as critical assets that must be protected. All underlying data assets and the associated code need to be protected against both malicious and accidental loss or corruption.

1.3 What the Operations Team needs to adress

Typical IT environments in a modern enterprise are not static entities with fixed, standardized single-vendor deployments. The push to continually improve the cost-structure and efficiency by adopting new technologies, coupled with corporate events such as mergers and acquisitions, and the need to comply with an ever-evolving regulatory compliance landscape result in a highly dynamic situation that operations teams need to not just address but embrace as the status quo. Any selected management solution will need to work with this diversity, scale, and rate of change.

1.3.1. Multi-workloads

Application architectures have undergone a rapid shift over the last few years and have gone from a monolithic design running on bare metal to virtualized applications running in hypervisors and now to containerized applications based on microservice architectures and often running itself on a virtualized infrastructure. Yet, for most enterprises with IT operating at significant scale, it is not reasonable to expect all applications to be on a single point on the evolution spectrum. Instead, the most common observed pattern is to find applications occupying the entire spectrum. The goal should be to deliver a data management solution that can work against this diversity, but still support refactoring and new development, delivering the benefits derived from cloud-native applications, providing an easy transition path for legacy applications, and reducing costs incurred to manage the diverse environment.

1.3.2. Multi-environments

As with applications, the deployment environments one finds are often a mix of on-premises and cloud. It is not uncommon to find enterprises taking a hybrid and multi-cloud approach to application deployment. When using managed Kubernetes or deploying Kubernetes workloads on different clouds, the Kubernetes distributions are also typically different in terms of proprietary extensions and supported features.

1.3.3. Multi-data Services

When it comes to data services underpinning applications, they too typically demonstrate significant variety. Developers are selecting data services that best meet the needs of the task at hand and often multiple data services within the same application. This is resulting in a polyglot mix drawing from Software as a Service (SaaS) offerings, managed services, relational databases, NoSQL systems, message queues and more.

1.3.4. Multi-storage Vendors

The storage infrastructure layer is yet another dimension that shows significant variety. Not only are teams in hybrid environments using storage on-premises and from cloud vendors but depending on their needs, they use different types of storage and potentially within a single Kubernetes cluster. Further, even on-prem, the storage tends to be sourced from different vendors (Dell EMC, HP, etc.), each bringing their own management tools and frameworks.

CHAPTER 2

Kubernetes-Native Backup is Critical

2. Kubernetes-Native Backup is Critical

Backup is a long-established discipline with multiple solutions serving the needs for users large and small. Yet, when it comes to backing up and protecting Kubernetes-based applications, there are certain reasons that employing a Kubernetes-native backup solution is critical. The ebook **"7 Critical Reasons for Kubernetes-Native Backup"** covers this aspect in detail and we have included a summary here for context.

2.1. Kubernetes Deployment Patterns

The Kubernetes platform is fundamentally different from earlier compute infrastructures. There is no mapping of applications to servers or VMs – Kubernetes manages the distribution of application components across servers potentially co-locating applications on servers. Traditional backup systems are challenged to cleanly capture a given application's state.

2.2 DevOps and "ShiftLeft"

The DevOps philosophy adopted in parallel with Kubernetes cedes control over both infrastructure and deployments to the developer (known as "shift left"). Backup systems must not only integrate with the CI/CD tools the developers use, they must automatically detect and protect applications coming online. They should do this in a manner transparent to the developers and employ Kubernetes-native APIs that the developers are familiar with. Consequently, this requires that backup solutions be application-centric and not infrastructure-focused.

2.3 Kubernetes Operator Challenges

Teams moving from a vSphere or Linux background to supporting Kubernetes will benefit from a platform that is deeply integrated with Kubernetes and masks its inherent complexity. Ignoring the multitude of resources is not an option because limiting the backup to just infrastructure disks and volumes will result in error-prone recovery and extended recovery times.

2.4 Application Scale

In the container paradigm, a single application that comprised of just a few VMs may now correspond to hundreds of distinct Kubernetes resources. When considered across all applications in a cluster, this can represent an overwhelming number of

components to manage without a Kubernetes-native backup platform. Further, Kubernetes and cloud-native applications are designed to scale in response to load. The backup platform must be able to respond effectively to this application scaling across the multi-cluster environments typically being used.

2.5. Protection Gaps

Whether running in the cloud or on-prem, the underlying storage is not failure-proof: even AWS's battle-hardened EBS advertises a non-zero failure rate and on-prem volume snapshots may not be resistant to hardware failures, and deletion of a volume usually results in simultaneous and automatic deletion of all related snapshots.

2.6. Security

Kubernetes offers several security features, and to avoid compromising their effectiveness, it is critical that a backup solution be Kubernetes-native and embed within the Kubernetes control plane. Also, with developers taking on more of the infrastructure responsibilities ("ITOps" model), it is important to be able to provide fine-grained, role-based and scoped access using the same roles and tools used by Kubernetes instead of succumbing to the use of additional role management systems and associated increased complexity. Further, to work well with Kubernetes' approach of delegating encryption to storage and backup platforms, the backup system needs to understand Kubernetes certificate management, work with storage-integrated Key Management Systems (KMSs), and support Customer Managed Encryption Keys (CMEKs) through the Kubernetes Secrets interface.

2.7. Ecosystem Integration

The use of polyglot persistence prevalent within Kubernetes environments requires the use of a backup platform that can derive the relationships between the various data services using the Kubernetes metadata. Such a backup solution can then use these relationships and its understanding of the workloads to capture a consistent copy of the entire application stack. The backup solution also needs to fit well with the rest of the Kubernetes cloud-native tools set that the operations team uses e.g., for monitoring, alerting, access control (Kubernetes APIs), logging and auditing.

CHAPTER 3

Data Management Use Cases

3. Data Management Use Cases

Having touched on the reasons why a Kubernetes-native backup is critical and before we dive into Kubernetes backup best practices, let's briefly discuss the key use cases associated with backup.

3.1. Backup and Restore

Naturally, the first use case when one thinks about backup is protecting against accidental or malicious loss or corruption. This involves regularly storing copies of the relevant information so that, in case of need, one can restore from an appropriate copy. Traditionally, IT teams have adopted an approach of performing a mix of full backups (perhaps weekly) interspersed with more frequent partial ones. Besides caring that a backup was successfully created, other factors of interest include how long the backup took (this may affect the application's performance while the backup is in progress) and how much storage the backup consumes (solutions with effective deduplication and compression result in lower storage costs).

In addition to the protection aspect, backups can be used for testing and development purposes i.e., one may take a backup of production data and restore it to a separate environment to support development or performance testing needs.

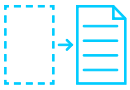
Not all approaches to backup are created equal. In particular, when considering implementing a backup strategy, it is useful to also bear in mind the notion of consistency levels. The different approaches that should be considered include storage-centric, storage-centric but with data service hooks, data services-centric and application-centric. More details on these consistency levels can be found in the Flavors of Data Management in Kubernetes article.

3.2 Application Mobility

Application mobility is the ability to move the application from one environment to another, including across clusters, regions, Kubernetes distributions, from on-premises to cloud, or even from one cloud to another. Such moves may be necessitated by changes in an organization's preferred technology platform, cost optimization, or new business requirements. This use case also includes the requirements originating from the need to upgrade from one stack to another (e.g., upgrading from OpenShift 3.x to OpenShift 4.x). Addressing the mobility use case requires strong support to handle the many differences between the source and target environments – that the destination may not have the same storage system is just one example. Teams choosing solutions for this use case are mindful of the ability to provide data and application

transformations from source to the target when underlying Kubernetes specifications, computer and storage infrastructure can change (e.g., the platform provides policy-based operations to transparently modify application specifications on the fly).

3.3. Disaster Recovery



Disaster recovery is the third related use case and refers to scenarios where there is a significant-enough local failure (e.g. due to a fire or a flood, or extended power or networking outages) that operations need to be continued at a different location altogether. The entire application stack as well as the associated data need to be made available for use at an alternate location. Besides cost, the key considerations for disaster recovery are usually measured in terms of Recovery Point Objective (RPO), how fresh the data is and how much data loss can be tolerated) and Recovery Time Objective (RTO), how long it takes to get the recovery environment operational). Teams need to take these considerations into account when deciding what kind of data management platform will be appropriate for their business requirements.

CHAPTER 4

Best Practices

4

4. Best Practices

When we think about backups in a traditional sense, we would expect that in a Kubernetes environment, we would want to be able to back up and restore containers, but this is not so sufficient as containers are immutable. Protecting the entire application state is the actual goal of a backup in a Kubernetes environment. The goal can be visualized as creating a recipe describing, in exquisite detail, the components and services the application needs in order to run, and then taking a snapshot of the persistent data and storing it all together, ideally in a different fault domain than the Kubernetes cluster. In this case, a restore would involve following the recipe to restore the persistent data and then deploying and configuring the application and associated services to resume normal operations.

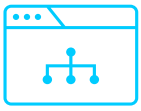
As traditional enterprises start to increasingly adopt containers in production and at scale, traditional monolithic applications are also being "containerized" or refactored to give them a more manageable deployment mechanism. The transition of these traditional applications brings in new considerations around how these workloads are protected and recovered in the event of disaster. When we think about these applications, they traditionally have contained stateful configurations that need to be recovered. Stateful data doesn't just include the data being stored by the application (e.g., tables in a database). It also has to include data related to application configuration (secrets, TLS certificates, etc.) that in the event of a crash, can't be recovered by simply re-deploying the application. These requirements bring in unique challenges for backing up and restoring workloads. When container platforms were primarily used with stateless workloads, applications could typically be spun up quickly and then destroyed and re-deployed as needed. As the adoption of containers is now encompassing stateful as well as stateless applications, the need for a robust backup solution has never been greater.

With most considerations around protecting business critical data, ensuring that data is captured with the correct consistency level is important. Just because an application has been containerized doesn't mean this requirement changes. From a technical perspective, considerations of crash-consistent and application-aware backups are still very much at the forefront. For most workloads, a crash-consistent backup or snapshot is enough, but for workloads that have regular changes like databases, we need to make sure that we can take a consistent point-in-time backup to avoid data corruption, even in the containerized world.

Finally, with the adoption of Kubernetes focusing on scale and ease of deployment, automation is a critical component for any management system deployed in the container environment. When looking at backup solutions, traditional backup products have struggled to work alongside or even integrate with the true dynamic scale of Kubernetes infrastructure. Having a backup solution that not only integrates and leverages the Kubernetes APIs but can also extend these APIs and provide greater automation integration is key. Deploying a backup platform as a native containerized system that runs within Kubernetes would the ideal option, rather than having a traditional "backup server" running in a different environment that needs to be separately managed.

Throughout this section, we will discuss the best practices and recommendations for implementing a Kubernetes-aware backup solution and the requirements for a successful backup strategy while also making it simple and flexible to adapt to this rapidly evolving cloud-native ecosystem.

4.1 Architecture



When implementing a backup strategy to protect Kubernetes workloads, a deep understanding of how Kubernetes works is critical. The purpose of this whitepaper is not to describe the Kubernetes architecture in depth, but to better understand how a backup strategy should be implemented. A few components need to be discussed.

In the diagram above, we see an example of a typical Kubernetes application. It is made up of pods, services, certificates, secrets, persistent volumes, and other components. On average, we observe production applications to be composed of hundreds of these components. With these considerations in mind, it is important to find the correct solutions to be able to not just protect and restore data, but also be able to do the same with all these internal components and at scale.

Once we deploy a backup platform into Kubernetes, the solution can then automatically interface with the Kubernetes control plane via the API server. This integration can be used to not just discover the Kubernetes applications running on the cluster but also integrate with the underlying compute, network and storage infrastructure.

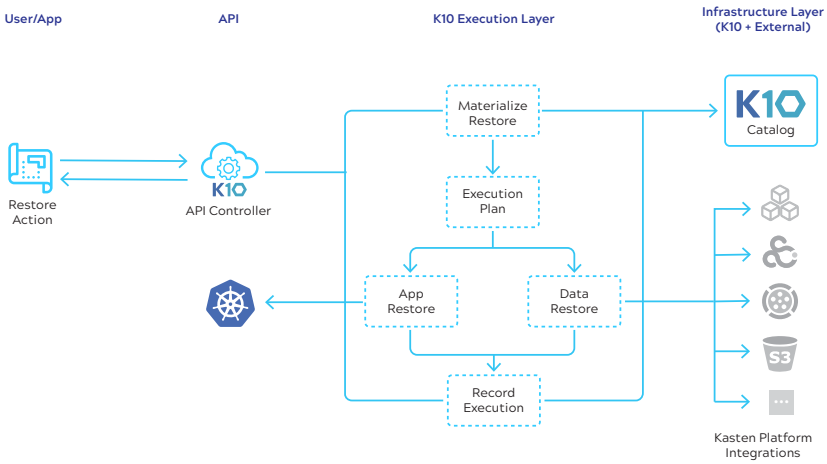
As a first step, the integration is used to discover the relationship between storage and applications and then determine the best (efficient, performant, consistent) way to capture the application data stored on persistent volumes along with the related application resources. The next consideration is the backup data location including within the storage system for fast recovery or, when running on the major cloud providers, depending on durable snapshots. For most cases though, backup data would be stored data in an object storage system in a different fault domain that could extend all the way to geo-replication for disaster recovery.

When it comes to the storage integrations with Kubernetes, there are several key areas that need to be considered. Storage in Kubernetes is represented as persistent volumes that are made available for use to the application containers. Apart from application configuration, this is the key business data that needs to be protected. Another consideration is where to keep that data. Is it going to be kept on local block storage? If Kubernetes is running in an on-premises environment or going to be kept off site, maybe use an object storage platform like Amazon S3 or Microsoft Azure blob storage. Retaining flexibility, choice and ease-of-use in the selection of secondary storage for backup data will be a critical component of implementing a successful data protection strategy.

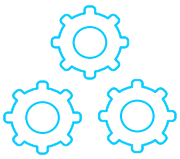
4.2 Recoverability

Recovery is not as simple as recreating Kubernetes objects and storage volumes. Given the number of components and Kubernetes' complexity, an execution plan needs to be created that first verifies cluster dependencies, creates new Kubernetes views of data that will get restored, and determines the compute infrastructure and Kubernetes cluster where the recovery needs to be initiated (e.g., a cross-availability zone recovery). Once the recovery execution plan is in place, the backup data sources (object storage, snapshots, backups) have to be identified, and the destination (storage class remapping, storage platform changes, etc.) storage prepared. Finally, the plan needs to be transformed if needed (e.g., regeneration of TLS certifications, DNS changes, editing stale secrets, etc.). Finally, Kubernetes applications components need to be updated to reflect the new storage resources that will be created as a part of the recovery.

Once this execution plan is in place, the backup platform needs to be able to translate it into relevant Kubernetes API calls to create the required resources (e.g., create a load balancer or recreate a secret). This process ensures that all required Kubernetes resources and microservices that make up a cloud-native application are redeployed with the correct configuration. The diagram below outlines this involved restore process.

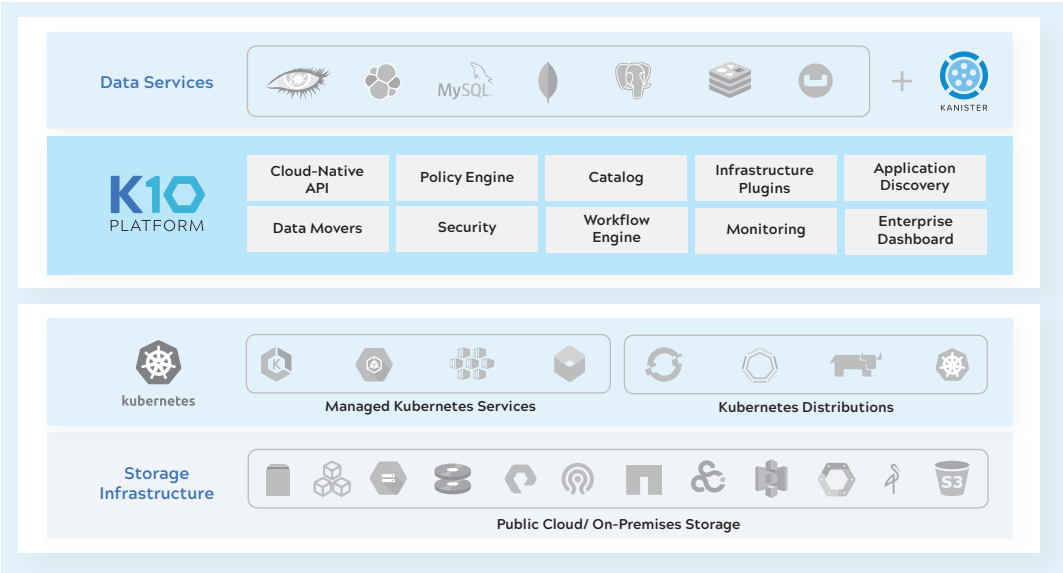


4.3 Operations



Operational best practices are typically the biggest challenge for enterprises, especially when implementing new tools, services and capabilities for an extremely dynamic infrastructure. It is important to ensure that a Kubernetes-native backup platform can be used at scale, provides operations teams with the workflow capabilities they require, and meets compliance and monitoring requirements. Another important aspect is the impact, or lack thereof, on developers. One of the greatest benefits of Kubernetes is providing developers with a quick and easy way to deploy applications, roll out upgrades, and the ability to do it at scale. If a backup platform hinders those use cases, developers will find ways to circumvent any processes put in place. There should be zero code, packaging, toolchain, or deployment changes required for developers. At the same time, operators should be able to give self-service capabilities to application developers such as the ability to restore their own application or the option to customize and extend backup operations for their data services (e.g., use of custom or database-vendor tools, cross-service coordination and quiescing, etc.). Further, the ability for all the developer interactions with the backup platform to be API drive is also a must-have requirement. It is therefore essential that any backup platform deployed can meet the needs of both container platform operations teams and developers.

From the operator's perspective, they should not need to focus on the hundreds of Kubernetes components that make up the application. Backup policies need to not only be completely automated, but they also need to focus on the application and not individual resources or storage infrastructure. Materialization of the policy to concretely define the application components that need to be protected should only happen at policy execution time to ensure that all components in a rapidly changing application are captured without requiring manual policy updates. Similarly, backup policies need to be broad and label-based so that they automatically pick up new applications as soon as they are deployed (e.g., policy that matches all applications that use MongoDB or are deployed via the Helm package management tool). This will not only save the operations teams from having to build out manual change control processes but will also ensure that, as applications are created and destroyed at scale, they never fall out of compliance with backup SLAs.



4.4 Security

Security is at the forefront of every product deployed in an enterprise production environment, regardless of whether it is deployed in a public cloud or using an on-premises infrastructure. Controls around identity and access management and role-based access control (RBAC) must be implemented. RBAC gives users and groups specific, and often restricted, user privileges or access privileges into the actual backup platform. This allows different personas in an operations team to adopt a least-privilege approach to common tasks such as monitoring backups, verifying backup success and integrity, and performing requested restores. RBAC also allows for use cases such as granting developers permissions for fast restore and clones from snapshots, but only grants certain team members access to backups stored in off-site secondary storage locations.

In a public cloud, apart from the security requirements described above, a cloud-native backup platform also needs deep integration into the cloud's Identity and Access Management (IAM) systems, Key Management Systems (KMSs) and certificate management.

Further, a truly Kubernetes-native data management system will integrate not just into the cloud provider's authentication solution (e.g., ODIC, OAuth, Token-based auth, etc.) but will also do so without requiring any extra user or group management, new tools, or new APIs for RBAC policies. All of these features will be exposed via a Kubernetes-native CRD-based API for a well-architected, cloud-native backup platform.

The final security aspect that must always be implemented is encryption. Protecting data, whether in transit or at rest, is critical to ensuring that data is secure whenever it has left the compute environment.

4.4.1. Encryption in Motion

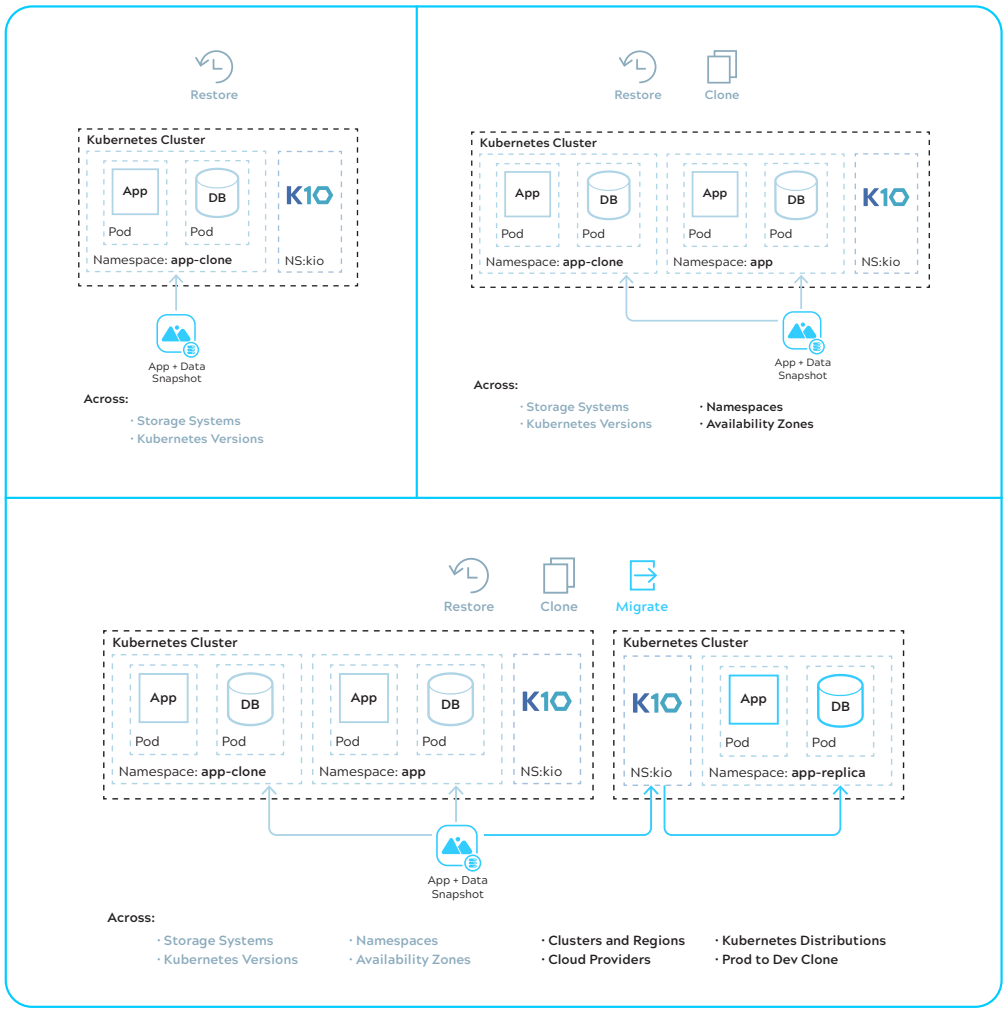
When moving or copying data between Kubernetes clusters or storage environments, making sure that the data is encrypted as it leaves one end point and arrives in another is key. Using object storage as an example, an on-premises Kubernetes application deployment that needs to offload backups to AWS S3 will typically transfer data over an external internet connection. The backup platform must always ensure that must ensure that the data is encrypted using well-known protocols such as TLS when being copied over the internet.

4.4.2. Encryption at Rest

Continuing with the example above, when the data is finally stored in a secondary location, ensuring that it is encrypted is a critical security consideration. Simply implementing RBAC and related security policies at the control layer is insufficient if the data is not encrypted at rest. Using well-proven encryption algorithms such as AES-256-GCM with per-application encryption keys will prevent accidental data leaks or even malicious copying by rogue infrastructure operators or malicious external entities.

4.5 Portability

As illustrated in the diagrams below, portability provided by Kubernetes is a very powerful feature and can be used by a backup platform to enable a large number of use cases, including across namespaces in the same cluster, across storage systems, across Kubernetes clusters, distributions and versions, across availability zones in the same region, across regions in the same cloud, across cloud or hybrid environments, and even across test and development environments.



With the ecosystem diversity in Kubernetes offerings available on-premises and in-the-cloud, it is also critical that a backup and data management platform can migrate Kubernetes applications across arbitrary source and destination clusters that could be running on wildly heterogeneous infrastructures. For example, when migrating a workload from Amazon Elastic Kubernetes Service (Amazon EKS) to Microsoft Azure Kubernetes Service (AKS), you will see the following on each cluster:

```
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
```

These different storage classes are just the tip of the iceberg as far as the differences between different distributions go, even though these distributions might be based on the same underlying Kubernetes version. Ensuring that a backup platform can reliably perform restores across these different distributions and infrastructure options while being able to automatically transform the application backup to fit the new restore environment is critical.

Ultimately, when performing migrations of workloads across namespaces, clusters, regions and even Kubernetes distributions, any reliable backup platform being used must be able to understand all application dependencies and translate them across environments. Similar to how a restore is planned and executed in a cloud-native environment, a migration plan is needed to ensure that infrastructure (e.g., load balancers), cluster-wide and application dependencies are available or transformed to an equivalent resource for a successful migration execution. It is not only containers and storage volumes that must be migrated, but also FQDNs, secrets and DNS addresses that must be modified in-flight during a migration.

CHAPTER 5

Conclusion

5

5. Conclusion

In the previous section, we discussed the critical requirements and recommendations on how to implement a Kubernetes backup solution. They can be summarized into the following five best practices:

Architecture

The platform used to protect Kubernetes applications needs to automatically discover all the application components running on your cluster and treat the application as the unit of atomicity. The application must include the state that spans across storage volumes, databases (NoSQL/Relational) as well as configuration data included in Kubernetes objects such as configmaps and secrets.

Recoverability

The data management platforms must allow you to restore the application components you want and where you want them. You should also have the granularity to restore only an application subset such as the data volume. The approach must make restoring simple and powerful by also allowing you to select the appropriate point of time copy of the application.

Operations

It is important to ensure that a Kubernetes-native backup platform can be used at scale, provide operations teams with the workflow capabilities they require, and meets compliance and monitoring requirements. Operators should be able to give self-service capabilities to application developers without requiring application code or deployment changes.

Security

Controls around identity and access management and role-based access control (RBAC) must be implemented. RBAC allows different personas in an operations team to adopt a least-privilege approach to common tasks such as monitoring. Encryption at rest and in transit must always be implemented to ensure that data is secure whenever it has left the compute environment..

Application Portability

Living in a multi-hybrid-cloud world, a cloud-native data management platform needs to be able to be flexible in the support for multiple distributions and offer capabilities that allow for the portability of workloads and applications across all these diverse environments. Portability capabilities are required across multiple use cases including application restoring, cloning, and migration.

Ensuring you adhere to the common best practices found in this guide will help you provide a consistent and reliable offering if you face data loss or corruption, or even a complete outage.

Veeam is a recognized leader in backup solutions for virtualized workloads. With the accelerated pace of Kubernetes applications and deployments, Veeam has partnered with Kasten to address the cloud-native data protection needs for enterprises. The Kasten K10 data management software platform has been purpose-built for Kubernetes and provides for the backup, restore and mobility of your entire Kubernetes application while keeping the best practices highlighted above.

About Veeam

Veeam® is the leader in backup solutions that deliver Cloud Data Management™. Veeam provides a single platform for modernizing backup, accelerating hybrid cloud and securing your data. Our solutions are simple to install and run, flexible enough to fit into any environment and always reliable.

About Kasten

Kasten, the leader in Kubernetes Backup and Disaster Recovery, helps enterprises overcome Day 2 data management challenges to confidently run applications on Kubernetes. Kasten K10, a data management platform purpose-built for Kubernetes, provides enterprise operations teams an easy-to-use, scalable, and secure system for backup/restore, disaster recovery, and application mobility with unparalleled operational simplicity. Kasten, an independent Kubernetes Business Unit within Veeam, has offices in the San Francisco Bay Area and Salt Lake City, Utah. For more information, visit www.kasten.io or follow @kastenhq on Twitter

Contacts

Kasten

289 S. San Antonio Road, #204
Los Altos, CA, 94022
info@kasten.io
www.kasten.io

Veeam Software

8800 Lyra Dr #350
Columbus, OH 43240
www.veeam.com