



OpenShift  
Commons

kasten  
by Veeam

# Backup and DR for databases on OpenShift with Kasten by Veeam



Michael Courcy  
Soln. Arch., Kasten by Veeam



Gaurav Rishi  
VP Product, Kasten by Veeam

# Content

## Introducing Kasten by Veeam

- Use cases and architecture
- Customer benefits

## Demonstrating Kasten K10 on Red Hat OpenShift

- Protecting applications – including databases
- Backup and DR

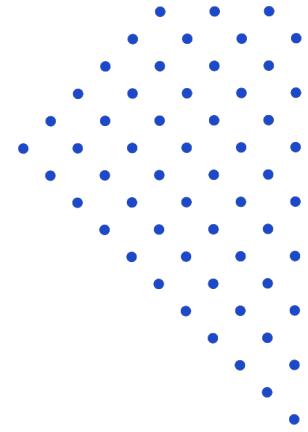
"We are pleased to collaborate with Kasten by Veeam so organizations across the globe can use Kasten K10 data management platform with Red Hat OpenShift to rollout, upgrade, and protect their cloud-native applications."

- Joe Fernandes, VP and GM, Core Cloud Platforms at Red Hat



# Kasten K10 by Veeam

## Use Cases



### Backup and Restore



Multi-tenancy, RBAC,  
Scale, Performance,  
Polyglot Persistence

### Disaster Recovery



Cross-AZ and Region,  
Multi and Hybrid Cloud,  
Storage Transforms

### Application Mobility



Cluster Upgrades,  
Application Transforms,  
Test/Dev Clusters

# Kasten: Kubernetes Data Protection Leader - Again!

## #1 Kubernetes Backup



### ANALYST REPORT

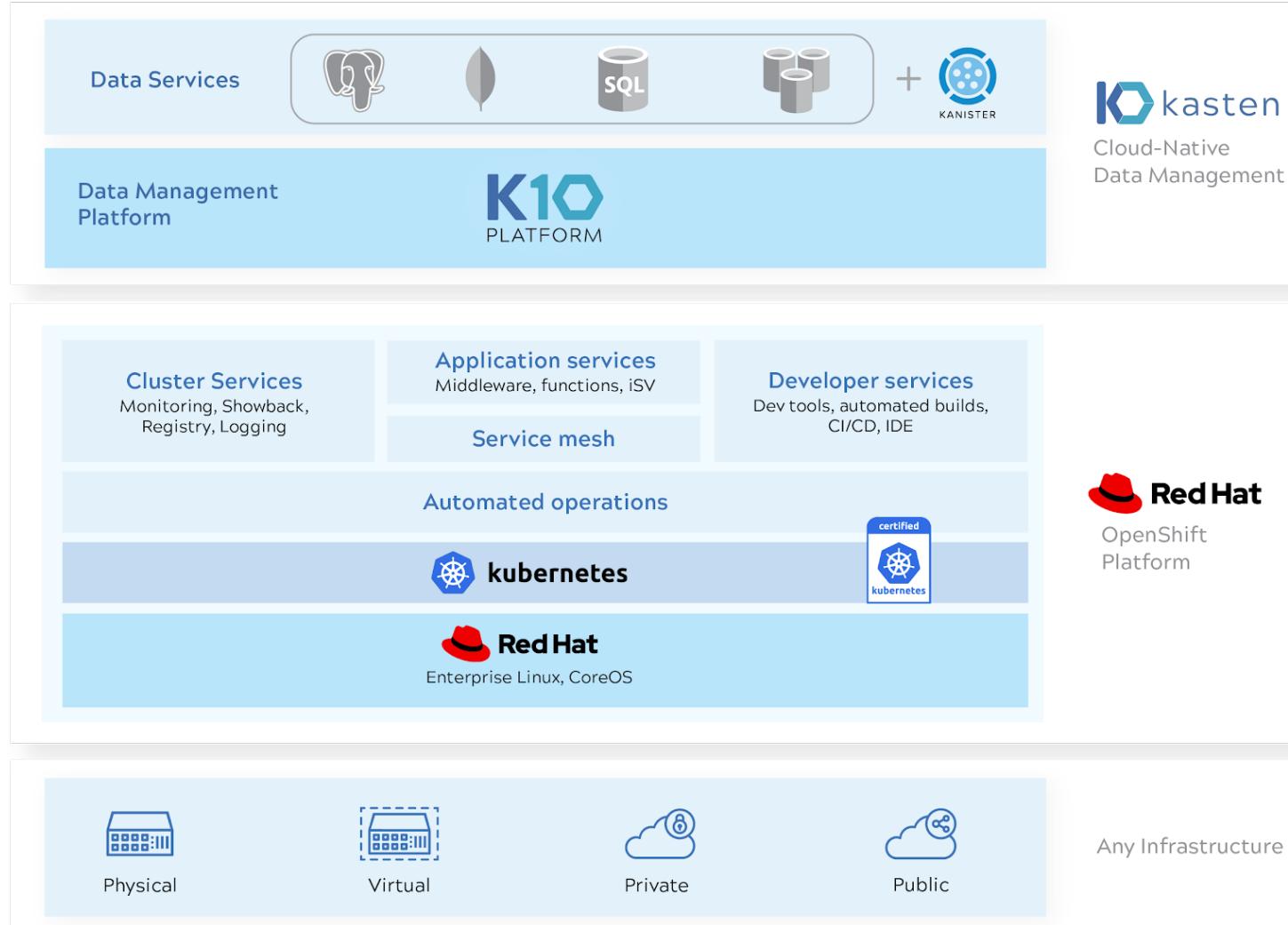


"Kasten is a Kubernetes-native, mature solution that's very suitable for self-hosted, self-managed use cases. Its architecture scales well and is especially well-suited for edge deployments. Its RBAC features and centrally managed policy model are well aligned with large enterprise and self-service requirements."

- Enrico Signoretti, Senior Data Storage Analyst, GigaOm

# Kasten K10 on Red Hat OpenShift

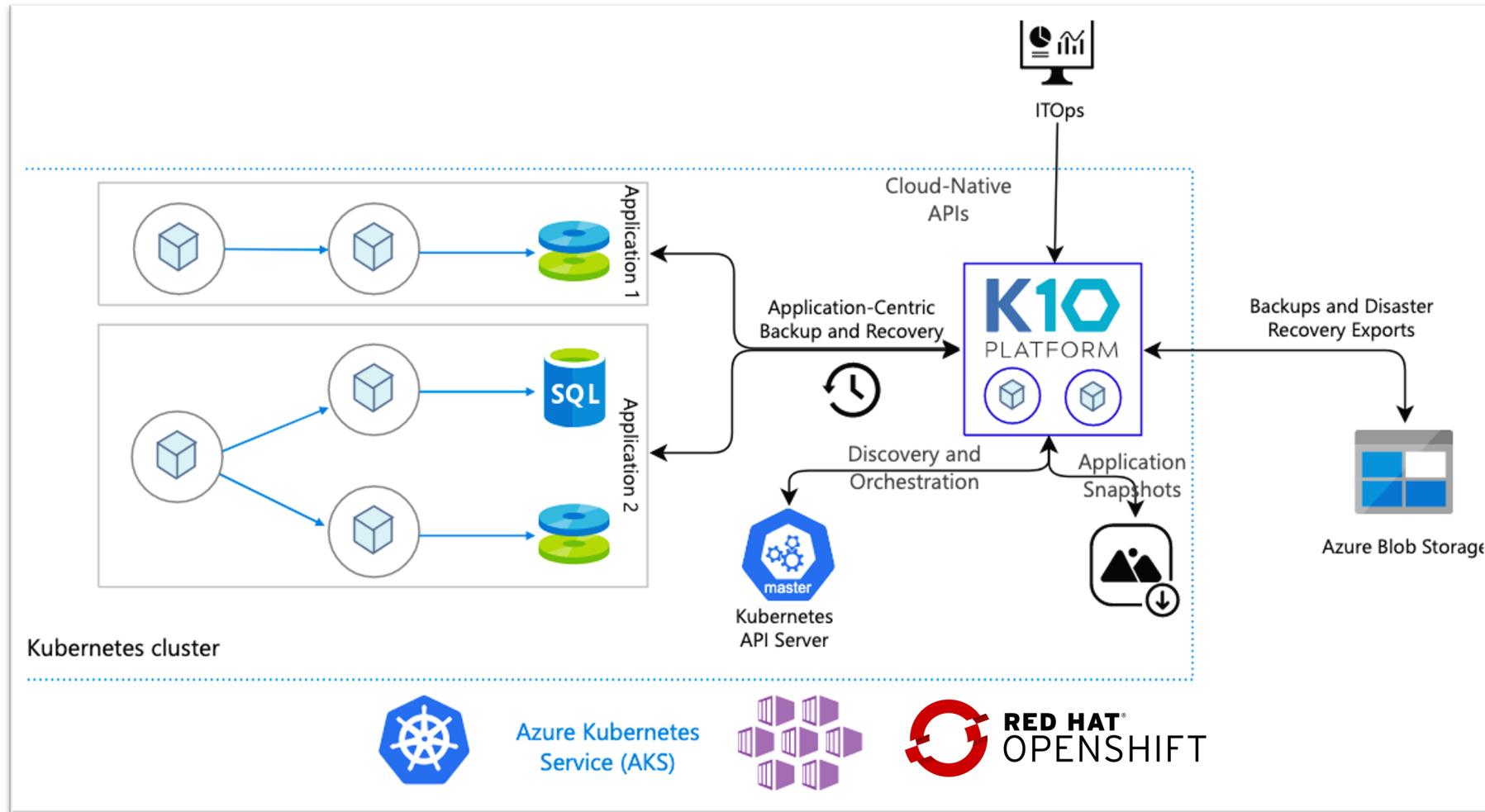
Kubernetes backup and mobility



- 1 Application-Centric
- 2 Backup, DR & Mobility
- 3 Red Hat Operator Certified
- 4 Red Hat Marketplace (including transactions)
- 5 Freedom of Choice (including OCS)

# e.g., Kasten K10 on Azure Red Hat OpenShift (ARO)

## Seamless operations



- Application Discovery
- Policy-driven Automation
- End-to-End Security
- Ease of Use, Simple UX



# Kasten K10 with Demo

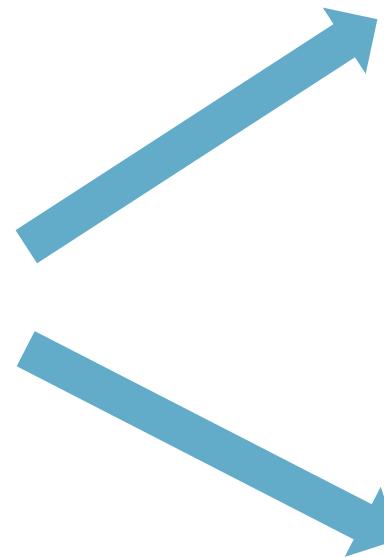
# Backing up MongoDB : a complete example

MongoDB deployed with an helm chart

2 Replicas, one Arbiter

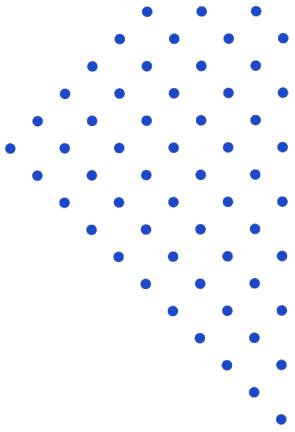
Openshift 4.8 on Azure

The screenshot shows the Red Hat OpenShift web interface. The left sidebar has a 'StatefulSets' tab selected. The main area displays a table for a StatefulSet named 'mongo-mongodb'. It shows two pods: 'mongo-mongodb-0' and 'mongo-mongodb-1', both in a 'Running' status. Below the table, there is another section for 'mongo-mongodb-arbiter' with '1 of 1 pods'.



The screenshot shows the Kasten by Veeam interface. On the left, a navigation sidebar lists 'Pods', 'Deployments', 'DeploymentConfigs', 'StatefulSets', 'Secrets', 'ConfigMaps', 'CronJobs', 'Jobs', 'DaemonSets', 'ReplicaSets', 'ReplicationControllers', 'HorizontalPodAutoscalers', and 'Networking'. The 'StatefulSets' item is highlighted. The main area shows a table of 'Pods' in the 'mongodb' project, with three entries: 'mongo-mongodb-0', 'mongo-mongodb-1', and 'mongo-mongodb-arbiter-0', all in 'Running' status. Below this, the 'PersistentVolumeClaims' section shows two entries: 'datadir-mongo-mongodb-0' and 'datadir-mongo-mongodb-1', both in 'Bound' status.

# Backing up MongoDB: Questions ??



**What's the best approach to backup this workload ?**

- Capture each PVC ?
- Capture a snapshot of each PVC ?
- Capture a mongodump against the mongo service ?
- Use fsyncLock() on all the replicas and capture a snapshot of each PVC ?

**What options can Kasten support?**

# Demo ...

The screenshot shows the Kasten by Veeam web interface. At the top, there's a navigation bar with the Kasten logo, a "Docs" link, a user dropdown for "kube:admin", and a notification bell icon with a "1" badge.

The main content area shows a "Clusters < primary-aro" path. A ribbon icon is displayed next to the title "Policy Run Running". The title is "mongodb-backup" and the ID is "policy-run-x7rsg".

Below the title, there are three status indicators:

- 2 Total Actions
- 1 Running Action
- 1 Successful Action

A table provides details about the run:

Start	End	Duration
Today, 11:33am	--	--

The "Applications (1)" section lists "mongodb".

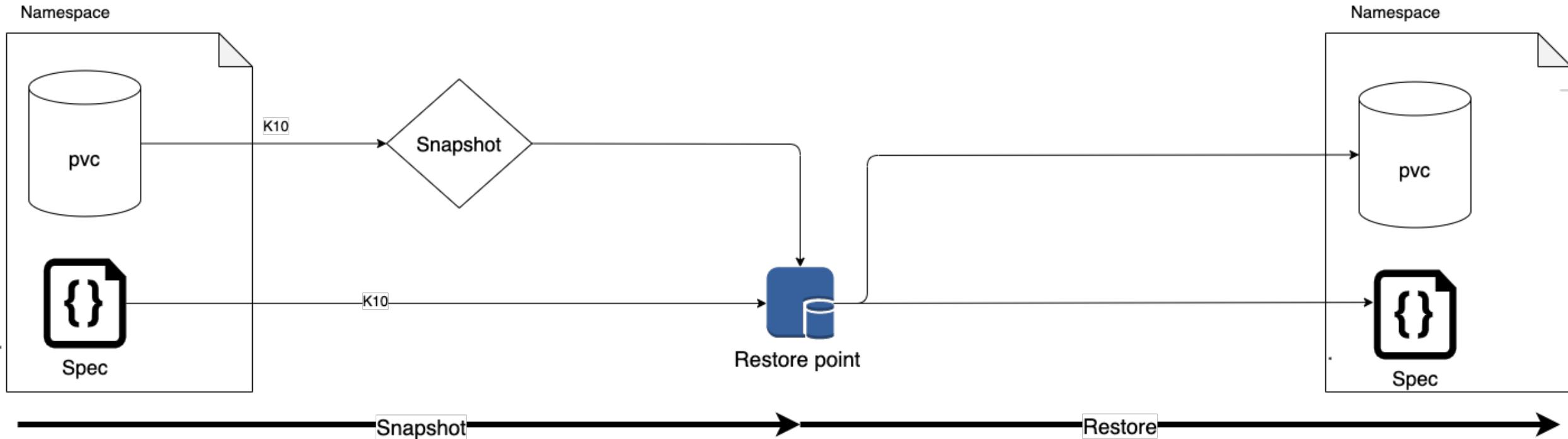
The "Actions (2)" section contains two entries:

Action Type	Description	Protected Object	Originating Policy	Artifacts	Start Time
Export	Exporting RestorePoint scheduled-bls4kv...	mongodb	mongodb-backup	none	Today, 11:34am
Backup	Backuping Application Components Snapshotting Workload mongo-mongodb Snapshotting Application configuration Snapshotting Workload mongo-mongodb-arbiter All phases completed successfully.	mongodb	mongodb-backup	2 snapshot + 16 GiB 36 spec	Today, 11:33am 37 secs

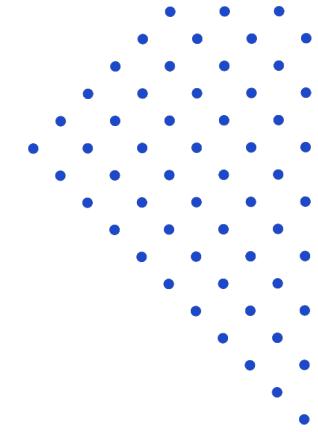
A "Filter Actions" dropdown is located in the top right of the actions table.

# Just capture a snapshot

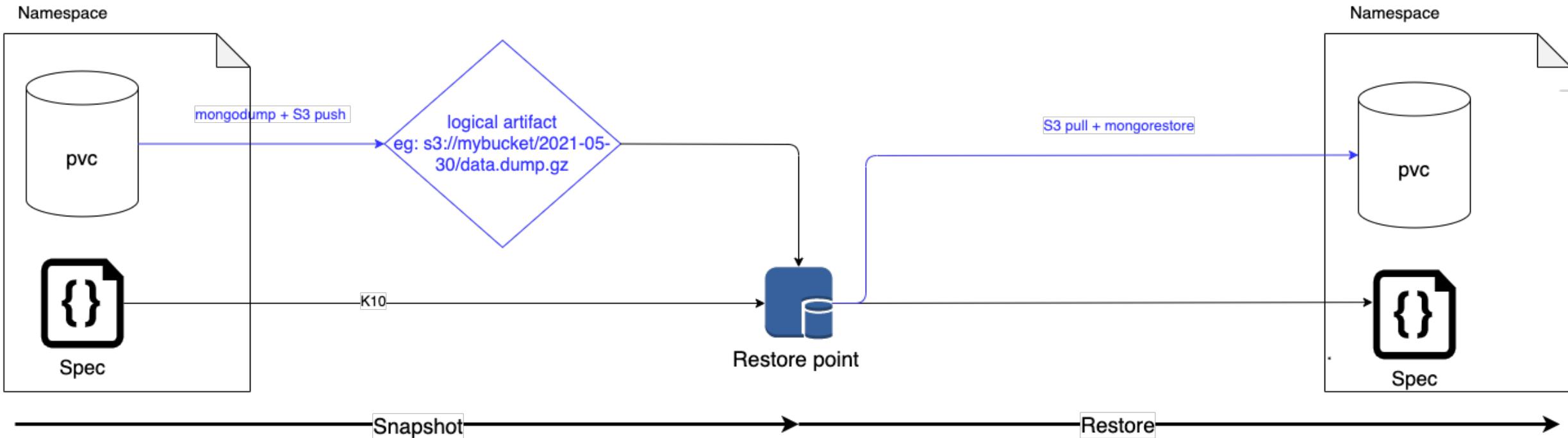
- Snapshot are « crash consistent » and Kasten leverage cloud API or CSI API
- All the objects in the namespace are also captured, it's a namespace snap
- However storage snapshot are not « application consistent »



# Capture a mongodump

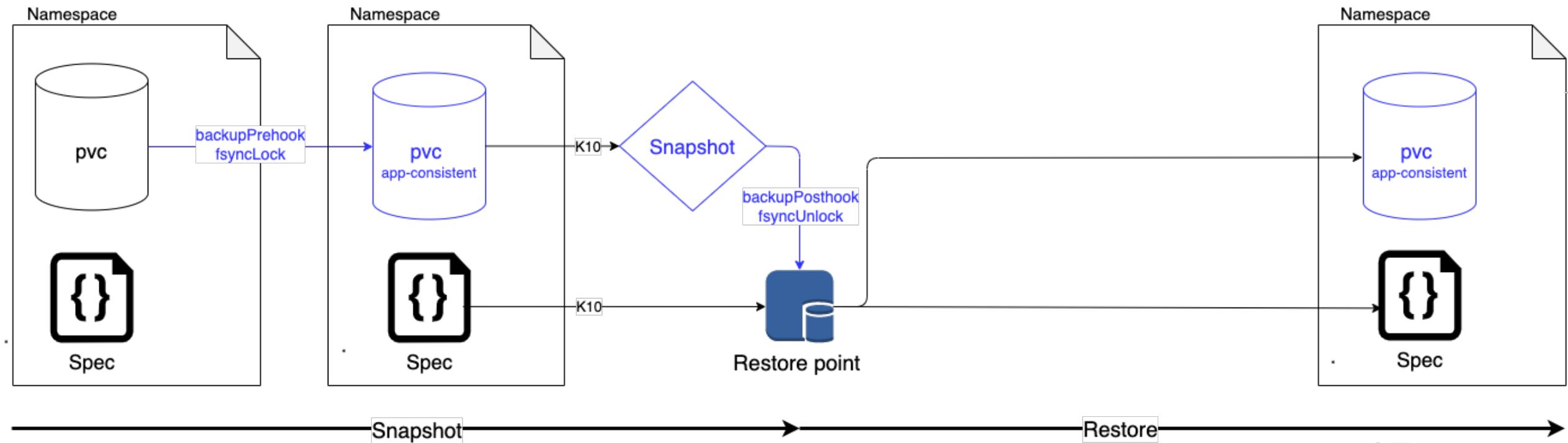


- We don't capture the PVC anymore
- When we backup we call mongodump
- When we restore we call mongorestore
- We keep on capturing all the objects in the namespace
- The backup is « app consistent » but is not incremental ...



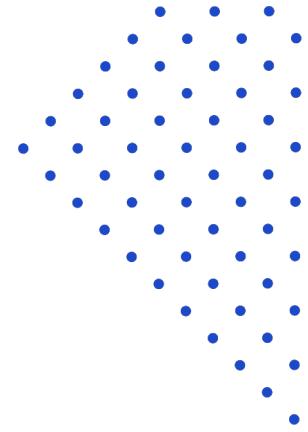
# Use fsynclock and capture a snapshot

- Before capturing the snapshot we call fsyncLock
- After capturing we call fsyncUnlock
- Now backup are « app consistent » and incremental !!





# About blueprints and database operators ...



- Blueprints are an extension mechanism
  - Gets you that remaining functionality to make it a perfect backup!
- Can be shared across all the databases you deployed a particular way
  - i.e. With the same operator or with the same helm chart
- Blueprint can also deal with **database operators**
  - By calling « backup api » exposed by the operator
- Let's see another example with the well known « Etcd Operator »

# Blueprint can leverage database operator for backup

Example : Etcd operator

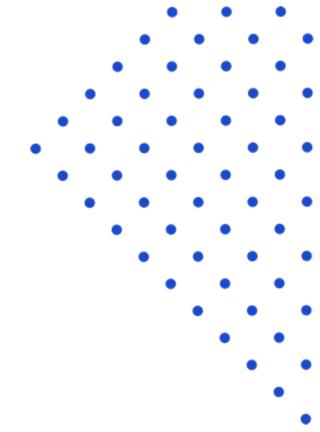
Backup API : EtcdBackup

Triggered by annotating the Etcd CR

```
! backup.yaml × ! etcd-bp.yaml
Users > sysmic > Dropbox > pro > kasten.io > architecture > olm > etcd > ! backup.yaml
1 apiVersion: "etcd.database.coreos.com/v1beta2"
2 kind: "EtcdBackup"
3 metadata:
4   name: example-etcd-cluster-backup
5 spec:
6   etcdEndpoints:
7     - "http://example-client:2379"
8   storageType: S3
9   s3:
10     path: "michael-bucket/demo-cluster/etcd-example/backup-test"
11     awsSecret: aws
```

```
! backup.yaml × ! etcd-bp.yaml × assignment.md .../01-configure-keycloak U assignment.md .../02
Users > sysmic > Dropbox > pro > kasten.io > architecture > olm > etcd > ! etcd-bp.yaml
1 apiVersion: cr.kanister.io/v1alpha1
2 kind: Blueprint
3 metadata:
4   name: etcd-bp
5   namespace: kasten-io
6 actions:
7   backup: ←
8     outputArtifacts:
9       cloudObject:
10         keyValue:
11           backupLocation: "{{ .Phases.takeEtcdBackup.Output.backupLocation }}"
12     phases:
13       - func: KubeTask
14         name: takeEtcdBackup
15         args:
16           image: ghcr.io/kanisterio/kanister-kubectl:1.18
17           namespace: '{{ index .Object.metadata "namespace" | toString }}'
18           command:
19             - sh
20             - -o
21             - errexit
22             - -o
23             - pipefail
24             - -c
25             - |
26               BUCKET="{{ index .Object.metadata.annotations "bucket" | toString }}"
27               ETCD_CLUSTER_NAME="{{ index .Object.metadata "name" | toString }}"
28               BACKUP_TIME="{{ toDate "2006-01-02T15:04:05.9999999Z07:00" .Time | date "2006-01-NAMESPACE="{{ index .Object.metadata "namespace" | toString }}"
29               BACKUP_LOCATION=${BUCKET}/${NAMESPACE}/${ETCD_CLUSTER_NAME}/${BACKUP_TIME}/backup
30               BACKUP_NAME=${ETCD_CLUSTER_NAME}-${BACKUP_TIME}
31               cat <<EOF | kubectl create -f -
32               apiVersion: "etcd.database.coreos.com/v1beta2"
33               kind: "EtcdBackup"
34               metadata:
35                 name: ${BACKUP_NAME}
36                 namespace: ${NAMESPACE}
37               spec:
38                 etcdEndpoints:
39                   - "http://${ETCD_CLUSTER_NAME}:2379"
40                 storageType: S3
41                 s3:
42                   path: "${BACKUP_LOCATION}"
43                   awsSecret: aws
44               EOF
45               kando output backupLocation ${BACKUP_LOCATION}
46
47 restore:
```

# Kasten K10 – Get started today!



Red Hat Marketplace | Learn more | Sell with us | Blog | Docs | Support | Log in



Certified enterprise ready  
[About certification](#)

## Kasten K10 by Veeam

By Kasten

Purchase

Kubernetes Backup and Mobility 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 mobility.

Delivery method  
**Operator**

Rating  
☆☆☆☆ Not rated

Overview Documentation Pricing Help

Kasten K10 is extensible and in addition to deep OpenShift integration is also pre-integrated with popular relational and NoSQL data services. This provides operations teams the capabilities to create policy-based automation to protect Kubernetes applications at scale. An extremely easy-to-use user interface along with a Kubernetes-integrated API, integrated observability and monitoring, and support for enterprise authentication and authorization schemes.

### Automatic Application Discovery

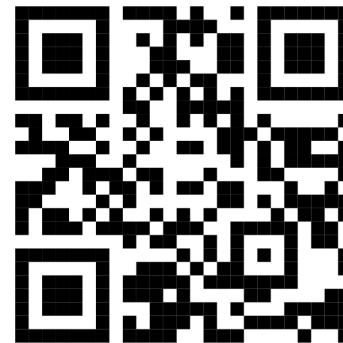
K10 is natively integrated into Kubernetes to automatically discover all the application components running on your cluster and treating the application as the unit of atomicity. The application includes the state that spans across storage volumes, databases (NoSQL/Relational) as well as configuration data included in Kubernetes objects such as configmaps and secrets.

### Integrated Security

We know security is top of mind for you and it definitely is for us! K10 supports Kubernetes ransomware protection along with user authentication mechanisms such as OIDC and Role Based Access Control (RBAC). Seamless integration with authenticating using the OpenShift OAuth proxy. Backed up data is encrypted both at rest and in motion with the ability to bring your own encryption keys. K10 runs on your cluster and you can assign fine-tuned access control permissions to K10 itself.

### Rich Backup Policies

Policies are used to automate your data management workflows. To achieve this, policies combine actions you want to take (e.g., snapshot), a frequency or schedule for how often you want to take that action, and label-based



[kasten.io/try-kasten-k10](https://kasten.io/try-kasten-k10)

<https://marketplace.redhat.com/en-us/products/kasten-k10>