Technical white paper

Check if the document is available in the language of your choice.

BACKUP AND DISASTER RECOVERY WITH KASTEN K10 AND HPE EZMERAL FOR CLOUD NATIVE APPLICATIONS

CONTENTS

This white paper offers a step-by-step guide on how to configure and use Kasten K10 for data protection for cloud-native applications running on HPE Ezmeral Container Platform.

Companies are looking for ways to deploy and manage Kubernetes clusters in production and at scale. Hewlett Packard Enterprise and Kasten by Veeam are working together to make this possible. Used together, HPE Ezmeral Container Platform and Kasten K10 simplify and streamline data protection in Kubernetes with an integrated and easy-to-use solution.

HPE EZMERAL AND KASTEN K10 OVERVIEW

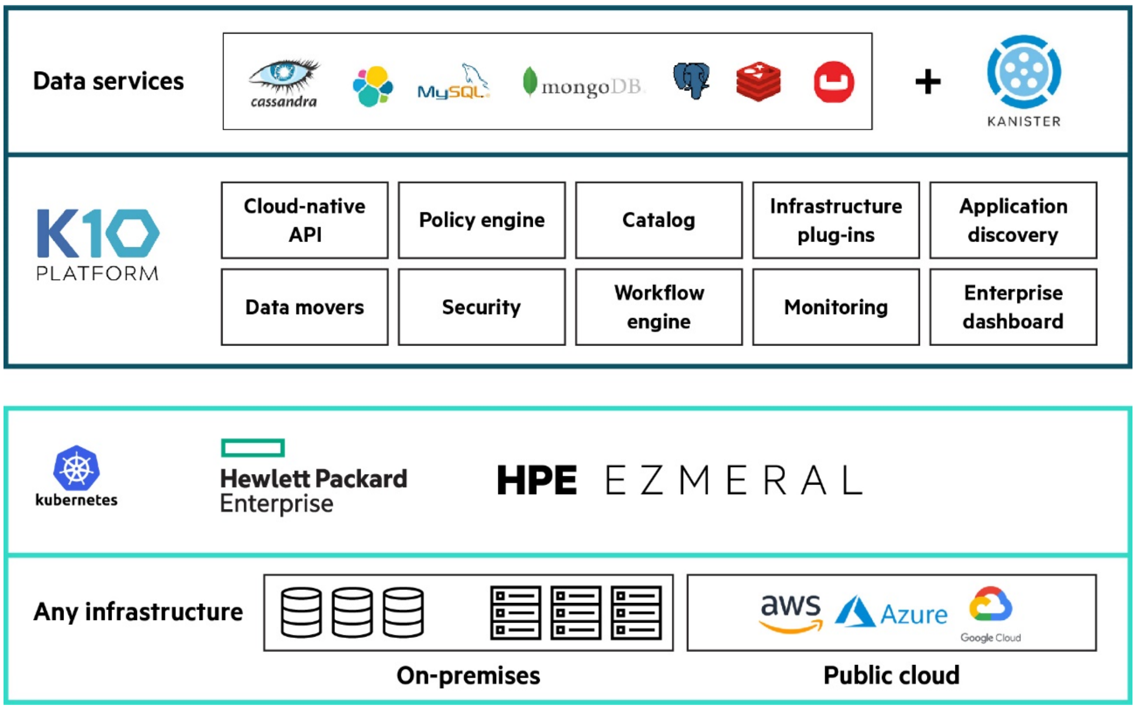


FIGURE 1. Kubernetes backup in HPE Ezmeral Container Platform with Kasten K10

An integrated component of HPE, HPE Ezmeral Container Platform enables IT operators to deliver and manage end-to-end, production-ready Kubernetes environments with push-button simplicity, all while preserving a native user experience. Every HPE Ezmeral cluster is deployed with HPE Ezmeral Data Fabric full-featured CSI driver, which natively integrates with HPE storage solutions, volumes, and file to deliver persistent storage for stateful containerized applications. S3-compatible storage is also easy to set up using HPE Ezmeral Data Fabric.

HPE Ezmeral Data Fabric is a software-defined storage solution that nondisruptively scales out while lowering overall storage costs. It's designed with an S3-compatible REST API interface to handle large amounts of unstructured data.

HPE Ezmeral Data Fabric is an ideal target for Kasten K10 backup export, as it provides long-term retention and archiving, as well as cross-region replication. As such, Kasten K10 is a perfect solution for managing protection and mobility of cloud-native applications on HPE Ezmeral Container Platform.

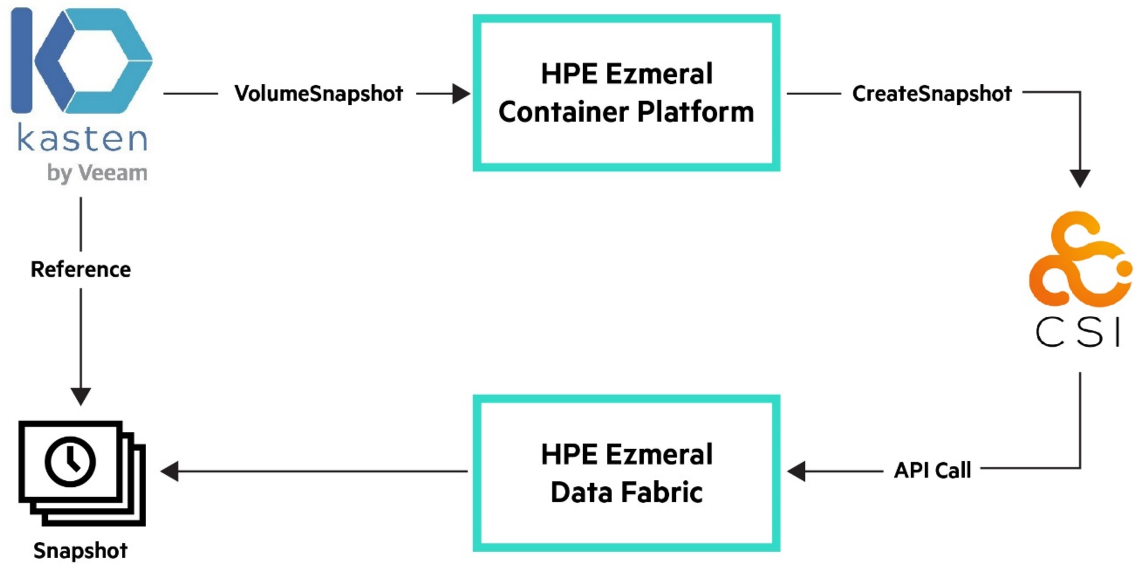


FIGURE 2. CSI interface with Kasten K10 and HPE Ezmeral

Purpose-built for Kubernetes, Kasten K10 is a data protection software platform that runs on a Kubernetes cluster in its own namespace and can protect single to multiple clusters. Kasten K10 provides secure multitenancy with fine-grained, role-based access control.

Kasten K10 offers:

- Prequalified integrations with leading data sources including relational and NoSQL data services
- Support for all major cloud-based managed Kubernetes offerings and all leading on-prem distributions
- Support for storage via Container Storage Interface (CSI) as well as direct storage integrations for efficiency

USING KASTEN K10 WITH HPE EZMERAL CONTAINER PLATFORM

Here's a summary of the steps we'll walk you through in demonstration of how to integrate HPE Ezmeral with Kasten K10:

1. Install Kasten K10.
2. Test the snapshot and recovery of a sample MySQL application.
3. Create an external object store with HPE Ezmeral Data Fabric.
4. Configure Kasten K10 to use the object store.
5. Export the snapshot to this object store and test the following three scenarios:
   a. Restoration of data to same namespace
   b. Restoration of a deleted namespace
   c. K10 Disaster Recovery

CREATING A KUBERNETES CLUSTER WITH HPE EZMERAL CONTAINER PLATFORM

Install and deploy a Kubernetes cluster using HPE Ezmeral Container Platform by following the instructions.

ADDING SNAPSHOT CAPACITY TO HPE EZMERAL CLUSTER WITH HPE EZMERAL DATA FABRIC CSI DRIVER

You can make use of the CSI snapshot API with Kasten as much as you can. This has numerous advantages.

- Kasten rely on the storage layer that in turn will make snapshot the most efficient way.
- A snapshot is crash consistent (all files state taken at the same time).
- Snapshots are local and that makes the restoration of an application quicker. You can also export the snapshot in a portable way to an object storage such as the HPE Ezmeral Data Fabric, and you can manage different retention policy between local and exported snapshot (because cost of the storage is much cheaper on object storage).

By default, HPE Ezmeral cluster does not come with a VolumeSnapshotClass deployed in it. You will be able to create the VolumeSnapshotClass with the following steps.

You need to have admin access to your cluster and a working kubeconfig/kubectl environment.

Create a VolumeSnapshotClass

Get the CSI secret name of the StorageClass to create your VolumeSnapshotClass with the following steps.

```
#Get the name of the secret that contains credentials for HPE Datafabric cluster
SECRETNAME=$(kubectl get sc -
o=jsonpath='{.items[?(@.metadata.annotations.storageclass\.kubernetes\.io\/is-default-
class=="true")].parameters.csi\.storage\.k8s\.io\/provisioner-secret-name}')

#Get the namespace in which the secret HPE Datafabric cluster is deployed
SECRETNAMESPACE=$(kubectl get sc -
o=jsonpath='{.items[?(@.metadata.annotations.storageclass\.kubernetes\.io\/is-default-
class=="true")].parameters.csi\.storage\.k8s\.io\/provisioner-secret-namespace}')

#Get the HPE datafabric cluster's rest server ip addresses
RESTSERVER=$(kubectl get sc -
o=jsonpath='{.items[?(@.metadata.annotations.storageclass\.kubernetes\.io\/is-default-
class=="true")].parameters.restServers}')

#Get the HPE datafabric cluster's name
CLUSTER=$(kubectl get sc -
o=jsonpath='{.items[?(@.metadata.annotations.storageclass\.kubernetes\.io\/is-default-
class=="true")].parameters.cluster}')

cat <<EOF | kubectl apply -f -
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
 name: mapr-snapshotclass
 namespace: $SECRETNAMESPACE
driver: com.mapr.csi-kdf
deletionPolicy: Delete
parameters:
 restServers: $RESTSERVER
 cluster: $CLUSTER
 csi.storage.k8s.io/snapshotter-secret-name: $SECRETNAME
 csi.storage.k8s.io/snapshotter-secret-namespace: $SECRETNAMESPACE
EOF
```

INSTALL KASTEN K10

Run the preflight script

To check if K10 is working fine, propose a preflight script. The following are the prerequisites to run the script.

- Add the Kasten's Helm chart repository.

```
helm repo add kasten  https://charts.kasten.io --force-update && helm repo update
```

- Create a namespace "kasten-io" where you will be installing K10 services.

```
kubectl create ns kasten-io
```

- To test the snapshot functionalities using the preflight script, you will be annotating the VolumeSnapshotClass as mentioned in the following. When K10 detects volumes that were provisioned via a CSI driver, it will look for a VolumeSnapshotClass with K10 annotation for the identified CSI driver and use it to create snapshots.

```
kubectl annotate volumesnapshotclass mapr-snapshotclass \
k10.kasten.io/is-snapshot-class=true
```

Now, you can run the preflight script and validate the output so that you are good to go.

```
curl -s https://docs.kasten.io/tools/k10_primer.sh  | bash

       Namespace option not provided, using default namespace
       Checking for tools
       --> Found kubectl
       --> Found helm
       Checking if the Kasten Helm repo is present
       --> The Kasten Helm repo was found
       Checking for required Helm Tiller version (>= v2.16.0)
       --> No Tiller needed with Helm v3.3.4
       K10Primer image
       --> Using Image (gcr.io/kasten-images/k10tools:3.0.8) to run test
       Checking access to the Kubernetes context kubernetes-admin@k8s-46
       --> Able to access the default Kubernetes namespace

       Running K10Primer Job in cluster with command-
                ./k10tools primer
       serviceaccount/k10-primer created
       clusterrolebinding.rbac.authorization.k8s.io/k10-primer created
       job.batch/k10primer created
       Waiting for pod k10primer-r446q to be ready - ContainerCreating
       Waiting for pod k10primer-r446q to be ready - ContainerCreating
       Waiting for pod k10primer-r446q to be ready -
       Pod Ready!

       Kubernetes Version Check:
       Valid Kubernetes version (v1.18.6)  -  OK

       RBAC Check:
       Kubernetes RBAC is enabled  -  OK

       Aggregated Layer Check:
       The Kubernetes Aggregated Layer is enabled  -  OK
```

```
        CSI Capabilities Check:
        Using CSI GroupVersion snapshot.storage.k8s.io/v1beta1  -  OK

        Validating Provisioners:
        com.mapr.csi-kdf:
        Is a CSI Provisioner  -  OK
        Missing/Failed to Fetch CSIDriver Object
        Storage Classes:
                hcp-mapr-cluster
                  Valid Storage Class  -  OK
        Volume Snapshot Classes:
         mapr-snapshotclass
           Has k10.kasten.io/is-snapshot-class annotation set to true  -  OK
                  Has deletionPolicy 'Delete'  -  OK

        Validate Generic Volume Snapshot:
        Pod Created successfully  -  OK
        GVS Backup command executed successfully  -  OK
        Pod deleted successfully  -  OK

        serviceaccount "k10-primer" deleted
        clusterrolebinding.rbac.authorization.k8s.io "k10-primer" deleted
        job.batch "k10primer" deleted
```

NOTE
Missing/Failed to Fetch CSIDriver Object error can be ignored as not all the CSI implementations have **CSIDriver Object**.

Preflight check for CSI snapshot validation

It is strongly recommended that the Primer tool be used to also perform a more complete CSI validation using the following command.

```
curl -s https://docs.kasten.io/tools/k10_primer.sh  | bash /dev/stdin -c "storage csi-
checker -s hcp-mapr-cluster --runAsUser=1000"

Namespace option not provided, using default namespace
Checking for tools
--> Found kubectl
--> Found helm
Checking if the Kasten Helm repo is present
--> The Kasten Helm repo was found
Checking for required Helm Tiller version (>= v2.16.0)
--> No Tiller needed with Helm v3.3.4
K10Primer image
--> Using Image (gcr.io/kasten-images/k10tools:3.0.8) to run test
Checking access to the Kubernetes context kubernetes-admin@k8s-46
--> Able to access the default Kubernetes namespace

Running K10Primer Job in cluster with command-
    ./k10tools primer storage csi-checker -s hcp-mapr-cluster

serviceaccount/k10-primer created
clusterrolebinding.rbac.authorization.k8s.io/k10-primer created
job.batch/k10primer created
```

```
Waiting for pod k10primer-hgvgw to be ready - ContainerCreating
Waiting for pod k10primer-hgvgw to be ready - ContainerCreating
Waiting for pod k10primer-hgvgw to be ready -
Pod Ready!

Starting CSI Checker. Could take up to 5 minutes

Failed to discover pod configuration for Pod (k8master1.test1.com): (pods
"k8master1.test1.com" not found)

I0304 00:29:22.508757    1874 request.go:655] Throttling request took 1.046455236s,
request: GET:https://gateway1.test1.com:10001/apis/autoscaling/v1?timeout=32s

Creating application
 -> Created pod (kubestr-csi-original-podj8k7s) and pvc (kubestr-csi-original-
pvcq52q8)

Taking a snapshot
 -> Created snapshot (kubestr-snapshot-20210304002923)

Restoring application
 -> Restored pod (kubestr-csi-cloned-podhxmsj) and pvc (kubestr-csi-cloned-pvc6rhzk)

Cleaning up resources
CSI Snapshot Walkthrough:
 Using annotated VolumeSnapshotClass (mapr-snapshotclass)
 Successfully tested snapshot restore functionality.  -  OK
```

Using the Helm chart to install Kasten K10

You can install Kasten with no options. In this tutorial, we mainly focus on creating policy for protecting namespace and we won't go on with authentication and authorization nor on how we are going to expose for long term the Kasten dashboard.

```
helm install k10 kasten/k10 --namespace=kasten-io
  NAME: k10
  LAST DEPLOYED: Thu Feb 18 02:06:30 2021
  NAMESPACE: kasten-io
  STATUS: deployed
  REVISION: 1
  TEST SUITE: None
  NOTES:
  Thank you for installing Kasten's K10 Data Management Platform!
  Documentation can be found at https://docs.kasten.io/.
  How to access the K10 Dashboard:
  The K10 dashboard is not exposed externally. To establish a connection to it
  use the following `kubectl` command:
  `kubectl --namespace kasten-io port-forward service/gateway 8080:8000`
  The Kasten dashboard will be available at: `http://127.0.0.1:8080/k10/#/`
```

Now, you will be able to verify if everything is working fine by checking that all the pods are up and running in the kasten-io namespace.

```
kubectl get pods -n kasten-io
NAME                               READY   STATUS       RESTARTS    AGE
aggregatedapis-svc-854548cd9b-vdz28   1/1       Running    0          3d14h
auth-svc-7bb46fdb95-p6xqw          1/1     Running    0          3d14h
catalog-svc-766898555f-r2jvs       2/2     Running    0          3d14h
config-svc-6cb68d6745-dgrg7        1/1     Running    0          3d14h
crypto-svc-7fc8cb8f57-m9289        1/1     Running    0          3d14h
dashboardbff-svc-85c47c85c9-k4zdv 1/1     Running    0          3d14h
executor-svc-78d6cfbf88-rlz4z      2/2     Running    0          3d14h
executor-svc-78d6cfbf88-sl2x6      2/2     Running    0          3d14h
executor-svc-78d6cfbf88-sr7cn      2/2     Running    0          3d14h
frontend-svc-7cbc66648f-lr7pb      1/1     Running    0          3d14h
gateway-69997d4768-xtzh9           1/1     Running    0          3d14h
jobs-svc-cf48db89-plbrz            1/1     Running    0          3d14h
kanister-svc-66f7d457c6-jf4dt      1/1     Running    0          3d14h
logging-svc-b5dc947cd-q7595        1/1     Running    0          3d14h
metering-svc-68797f4978-mlnqb      1/1     Running    0          3d14h
prometheus-server-78b94b85fb-w745z2/2     Running    0          3d14h
state-svc-78d847595f-9wfmq         1/1     Running    0          3d14h
```

TESTING THE SNAPSHOT BY BACKING UP AND RESTORING AN APPLICATION

Installing the test application

We're going to install a PostgreSQL application for testing out snapshot backups and restore on the cluster.

```
helm repo add bitnami https://charts.bitnami.com/bitnami --force-update && helm repo
update

kubectl create ns postgresql

helm install postgresql bitnami/postgresql --namespace=postgresql --set
volumePermissions.enabled=true
```

Let's insert some data into the PostgreSQL pod once it is ready.

```
#To get the password for "postgres" run:

       export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql
-o jsonpath="{.data.postgresql-password}" | base64 --decode)

#To connect to your database run the following command:

kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql -
-image docker.io/bitnami/postgresql:11.11.0-debian-10-r50 --
env="PGPASSWORD=$POSTGRES_PASSWORD" --command -- psql --host postgresql -U postgres -d
postgres -p 5432

#Insert data into the test database

CREATE DATABASE test;
\c test;
CREATE TABLE pets (name VARCHAR(20), owner VARCHAR(20), species VARCHAR(20),
sex CHAR(1), birth DATE, death DATE);
INSERT INTO pets VALUES ('Puffball','Diane','hamster','f','2010-03-30',NULL);
INSERT INTO pets VALUES ('Spike','Mike','pitbull','m','2011-04-28',NULL);
INSERT INTO pets VALUES ('Ashton','Varoon','German Sheperd','m','2014-06-15',NULL);
INSERT INTO pets VALUES ('Bear','Chris','Rottweiler','m','2013-10-10',NULL);
INSERT INTO pets VALUES ('Toby','Jenny','Golden Retriever','m','2019-03-19',NULL);

#Validate the data in the table PETS
test=# select * from pets;
   name    | owner  |       species        | sex |    birth      | death
----------+--------+-----------------+-----+----------+-------
 Puffball | Diane  | hamster       | f    | 2010-03-30 |
 Spike | Mike    | pitbull          | m    | 2011-04-28 |
 Ashton    | Varoon | German Shepherd    | m    | 2014-06-15 |
 Bear  | Chris  | Rottweiler       | m    | 2013-10-10 |
 Toby  | Jenny  | Golden Retriever | m    | 2019-03-19 |
(5 rows)
```

Accessing the dashboard

All the following operations will be done on the dashboard. Without the LoadBalancer/Ingress, we can have two options to access the dashboard:

1. Dashboard made available through port-forward

```
kubectl --namespace kasten-io port-forward service/gateway 8080:8000
```

The Kasten dashboard will be available at http://127.0.0.1:8080/k10/#/.

2. Dashboard is made available using NodePort service.

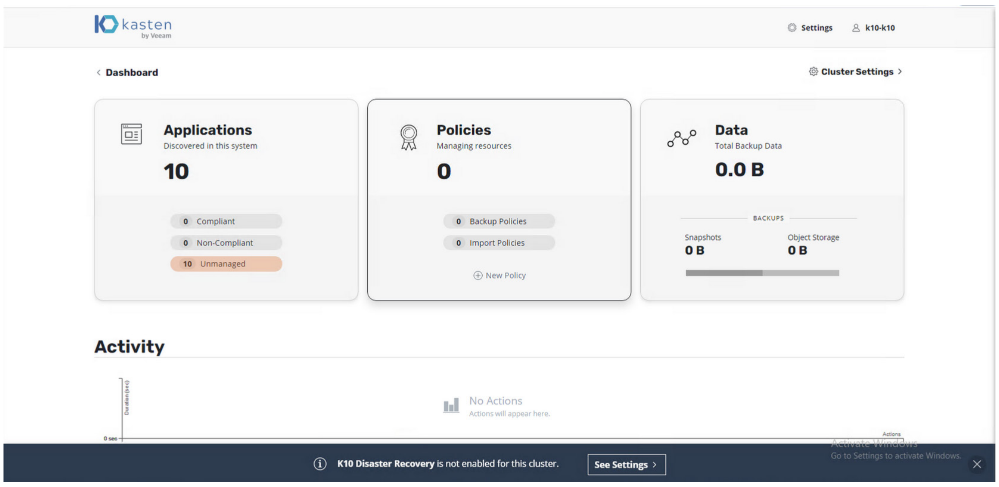
```
kubectl expose service gateway -n kasten-io --type=NodePort --name=gateway-nodeport


# Validate the service and get the nodeport details
kubectl get svc -n  kasten-io gateway-nodeport
NAME                  TYPE            CLUSTER-IP     EXTERNAL-IP    PORT(S)       gateway-
nodeport    NodePort    10.100.127.231    <none>          8000:32161/TCP
```

The Kasten dashboard will be available at http://<worker-node-ip>:<NodePort> where the NodePort in this example is 32161.

Once you have access to the dashboard, fill out and accept the end-user license.

The following is the Dashboard page of K10.



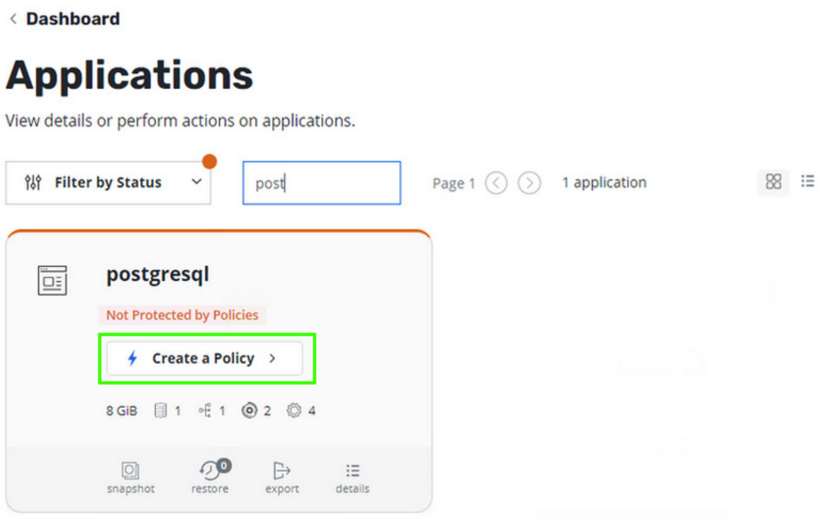
K10 Disaster Recovery is not enabled for this cluster.

FIGURE 3. Kasten K10 dashboard

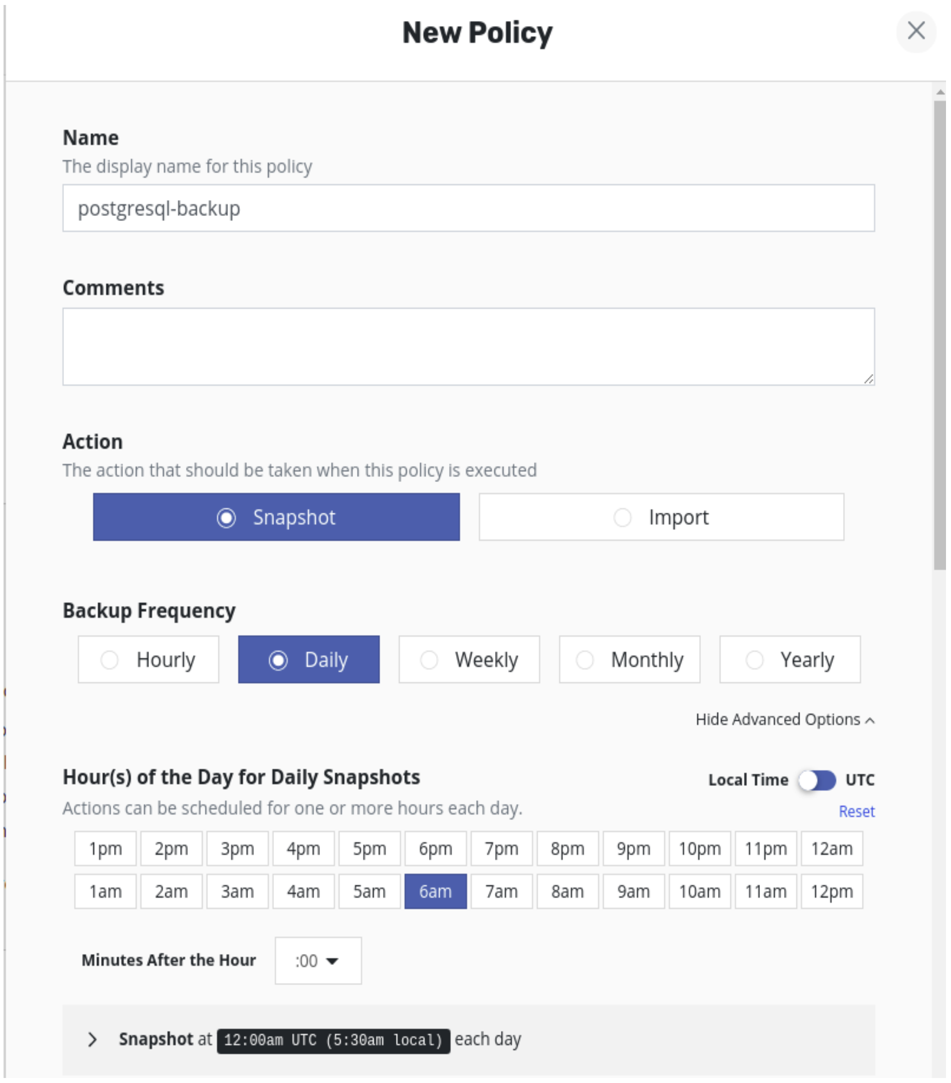
Snapshot the application

You are going to snapshot the application. In other words, you are going to take the complete state of the application without exporting it to an external storage.

Go to Applications --> postgresql --> Create a Policy



Provide the policy a name and select the action as Snapshot. You will be able to select the backup frequency in the Advanced options.



New Policy

Name
The display name for this policy
postgresql-backup

Comments

Action
The action that should be taken when this policy is executed
[✓ Snapshot] Import

Backup Frequency
Hourly [✓ Daily] Weekly Monthly Yearly

Hide Advanced Options

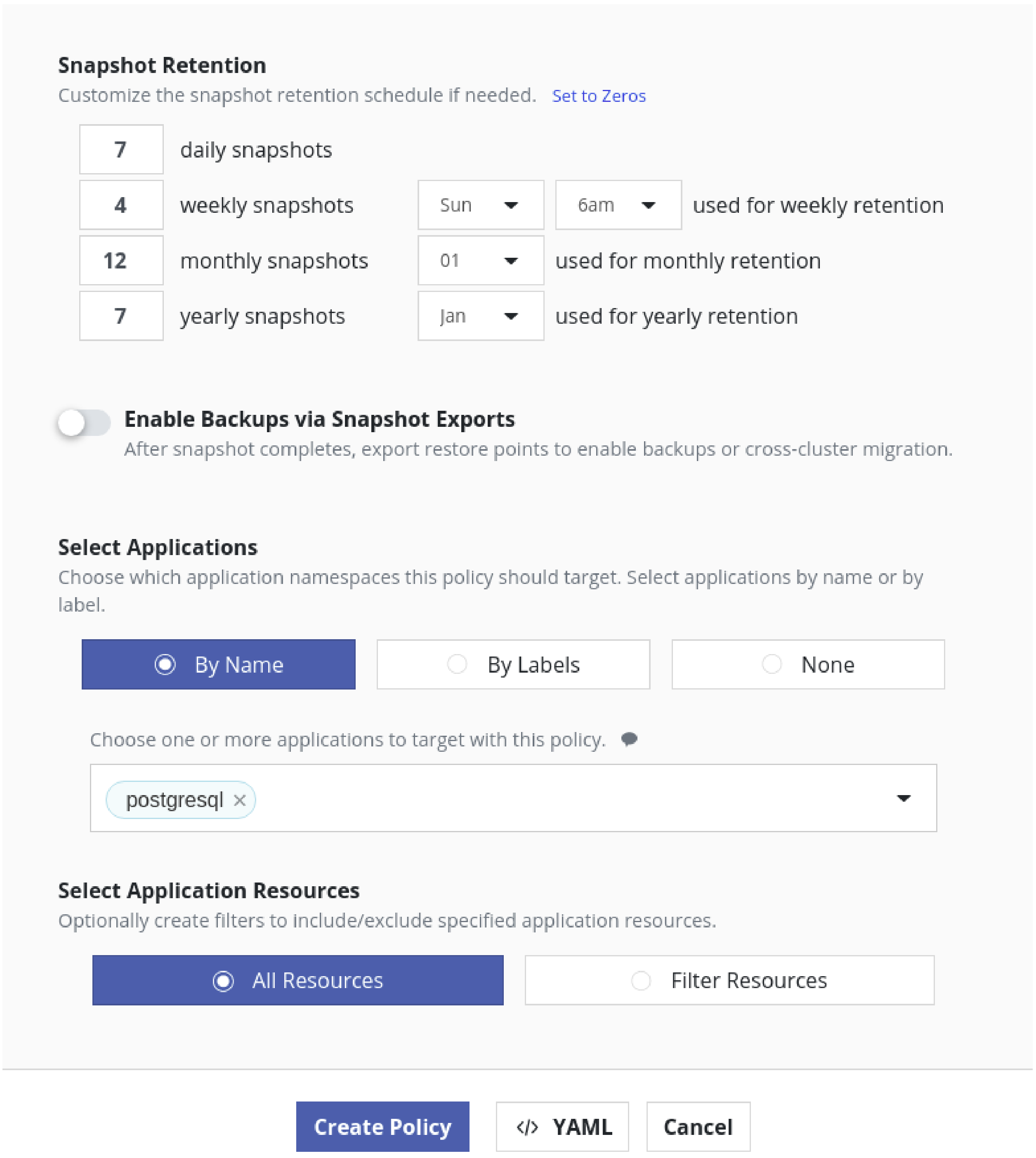
Hour(s) of the Day for Daily Snapshots Local Time / UTC
Actions can be scheduled for one or more hours each day. Reset

1pm	2pm	3pm	4pm	5pm	6pm	7pm	8pm	9pm	10pm	11pm	12am
1am	2am	3am	4am	5am	[✓ 6am]	7am	8am	9am	10am	11am	12pm

Minutes After the Hour :00

Snapshot at `12:00am UTC (5:30am local)` each day

You will be able to select the retention of the snapshot and select/filter resources by name/labels to snapshot based on the requirement. Once done, click the **Create Policy** button.



Snapshot Retention

Customize the snapshot retention schedule if needed. Set to Zeros

7	daily snapshots			
4	weekly snapshots	Sun	6am	used for weekly retention
12	monthly snapshots	01	used for monthly retention	
7	yearly snapshots	Jan	used for yearly retention	

Enable Backups via Snapshot Exports

After snapshot completes, export restore points to enable backups or cross-cluster migration.

Select Applications

Choose which application namespaces this policy should target. Select applications by name or by label.

[✓ By Name] By Labels None

Choose one or more applications to target with this policy.

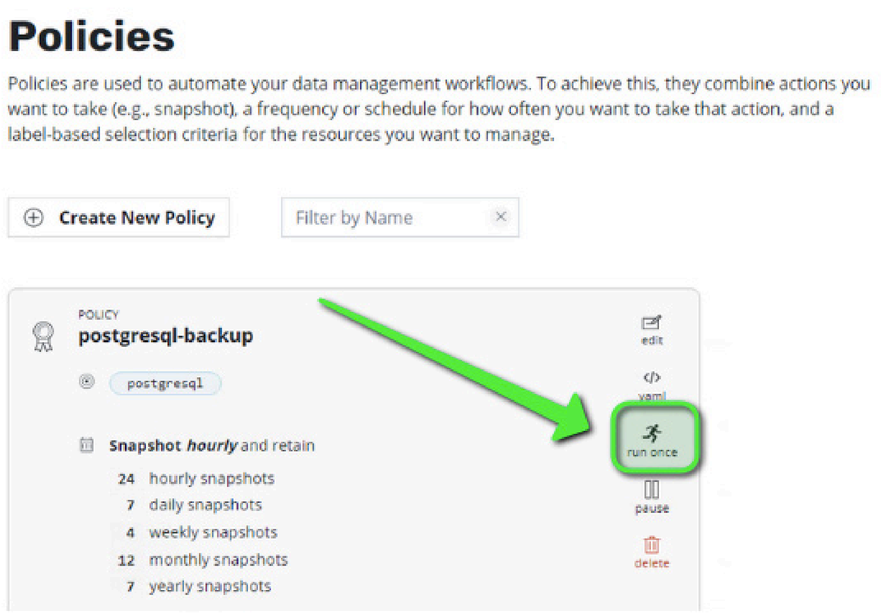
postgresql ×

Select Application Resources

Optionally create filters to include/exclude specified application resources.

[✓ All Resources] Filter Resources

Click **run once** from the policy menu to run an ad hoc snapshot manually.



Policies

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

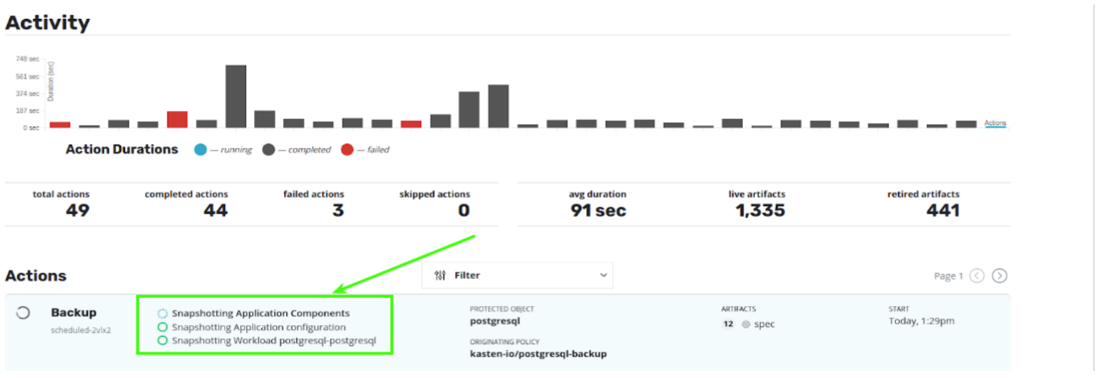
Create New Policy | Filter by Name

POLICY
postgresql-backup
postgresql

Snapshot *hourly* and retain

24	hourly snapshots
7	daily snapshots
4	weekly snapshots
12	monthly snapshots
7	yearly snapshots

Go to the main dashboard and scroll down under Actions to view progress.



Activity

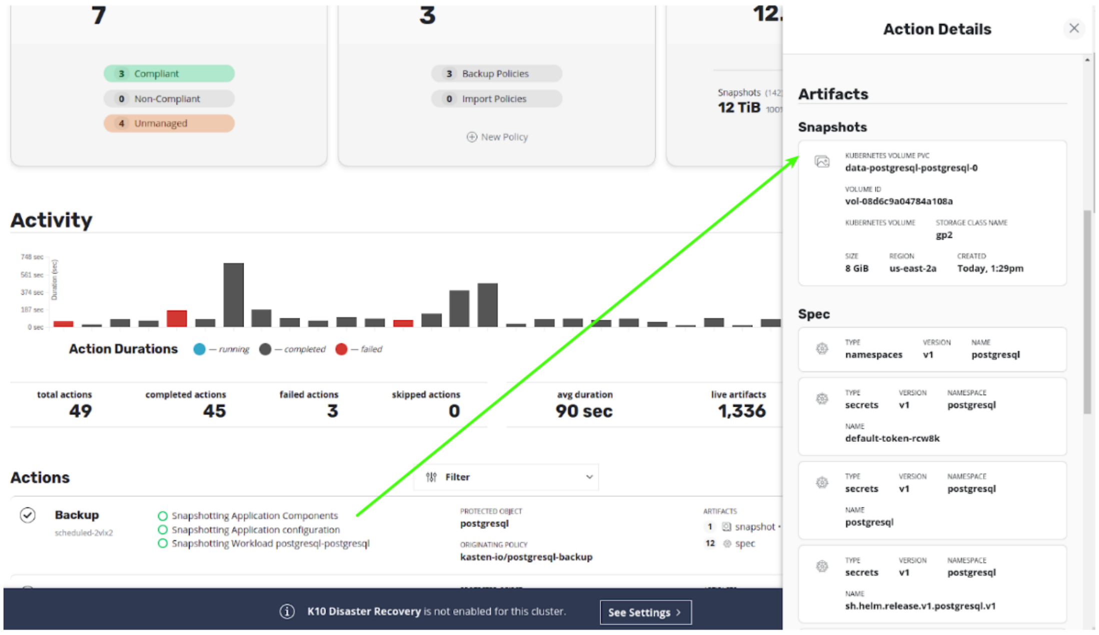
total actions	completed actions	failed actions	skipped actions	avg duration	live artifacts	retired artifacts
49	44	3	0	91 sec	1,335	441

Actions

Backup scheduled-2vlx2
- Snapshotting Application Components
- Snapshotting Application configuration
- Snapshotting Workload postgresql-postgresql

PROTECTED OBJECT postgresql
ORIGINATING POLICY kasten-io/postgresql-backup
ARTIFACTS 12 spec
START Today, 1:29pm

When the circle turns solid green, click the job to open a details pane and view all artifacts that were captured.



Activity

total actions	completed actions	failed actions	skipped actions	avg duration	live artifacts
49	45	3	0	90 sec	1,336

Actions

Backup scheduled-2vlx2
- Snapshotting Application Components
- Snapshotting Application configuration
- Snapshotting Workload postgresql-postgresql

PROTECTED OBJECT postgresql
ORIGINATING POLICY kasten-io/postgresql-backup
ARTIFACTS 1 snapshot · 12 spec

K10 Disaster Recovery is not enabled for this cluster. See Settings >

Action Details

Artifacts

Snapshots

KUBERNETES VOLUME PVC data-postgresql-postgresql-0
VOLUME ID vol-08d6c9a04784a108a
KUBERNETES VOLUME
STORAGE CLASS NAME gp2
SIZE 8 GiB
REGION us-east-2a
CREATED Today, 1:29pm

Spec

TYPE namespaces VERSION v1 NAME postgresql

TYPE secrets VERSION v1 NAMESPACE postgresql
NAME default-token-rcw8k

TYPE secrets VERSION v1 NAMESPACE postgresql
NAME postgresql

TYPE secrets VERSION v1 NAMESPACE postgresql
NAME sh.helm.release.v1.postgresql.v1

BACKING UP THE APPLICATION

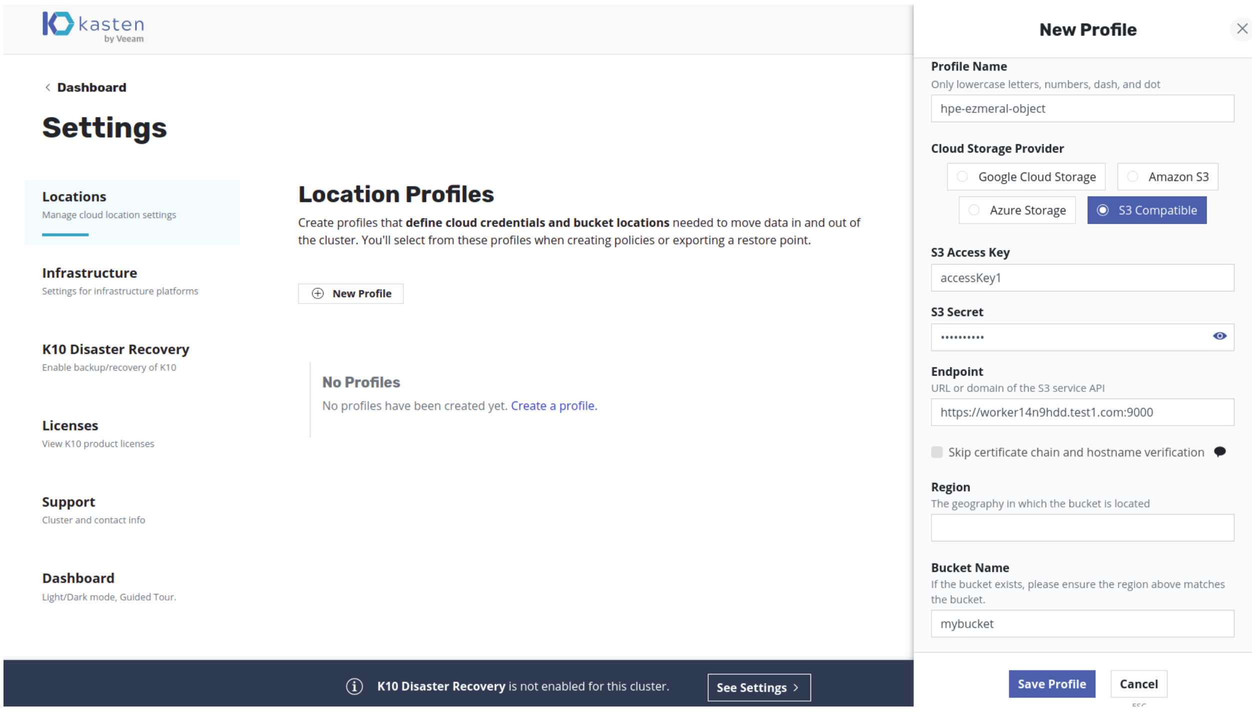
Backing up an application involves exporting the snapshot to an external backup target, which is recommended to be an S3-API-compatible object storage. For this use case, you can use HPE Ezmeral Data Fabric as the object storage backup target.

Creating an external object storage with HPE Ezmeral Data Fabric

Follow the instructions at https://docs.datafabric.hpe.com/61/MapRObjectStore/Configuring-MapR-Object-Store-with-S3-Compatible-API.html to configure HPE Ezmeral Data Fabric Object Store and then the instructions at https://docs.datafabric.hpe.com/61/MapRObjectStore/maprObjectStore-AWS-CLI.html to create an S3 bucket on HPE Ezmeral Data Fabric.

Setting up the location profile in Kasten Dashboard

Go to settings -> Location Profiles -> New Profile -> S3 Compatible


kasten by Veeam

‹ Dashboard

Settings

Locations
Manage cloud location settings

Infrastructure
Settings for infrastructure platforms

K10 Disaster Recovery
Enable backup/recovery of K10

Licenses
View K10 product licenses

Support
Cluster and contact info

Dashboard
Light/Dark mode, Guided Tour.

Location Profiles

Create profiles that **define cloud credentials and bucket locations** needed to move data in and out of the cluster. You'll select from these profiles when creating policies or exporting a restore point.

New Profile

No Profiles
No profiles have been created yet. Create a profile.

K10 Disaster Recovery is not enabled for this cluster. See Settings ›

New Profile

Profile Name
Only lowercase letters, numbers, dash, and dot
hpe-ezmeral-object

Cloud Storage Provider
Google Cloud Storage Amazon S3 Azure Storage [✓ S3 Compatible]

S3 Access Key
accessKey1

S3 Secret
••••••••••

Endpoint
URL or domain of the S3 service API
https://worker14n9hdd.test1.com:9000

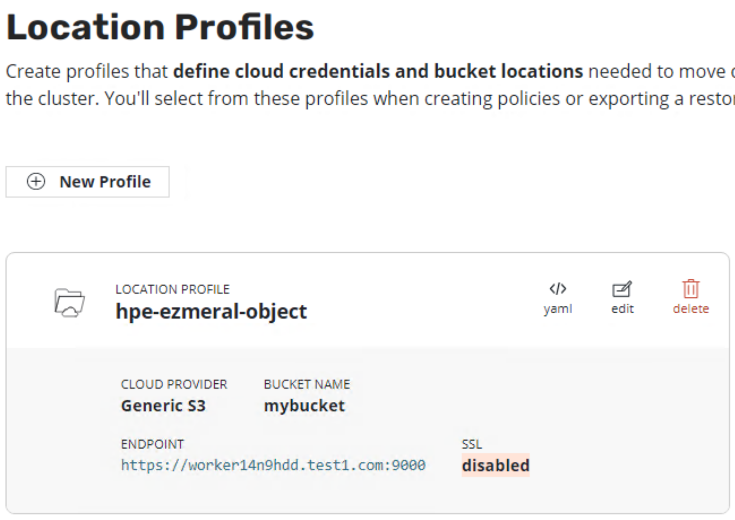
Skip certificate chain and hostname verification

Region
The geography in which the bucket is located

Bucket Name
If the bucket exists, please ensure the region above matches the bucket.
mybucket

Save Profile Cancel

Name the profile and fill in the details of the endpoint, access key, secret key, and name of the bucket created using the previous steps. Click **Save Profile** once done.


Location Profiles

Create profiles that **define cloud credentials and bucket locations** needed to move data in and out of the cluster. You'll select from these profiles when creating policies or exporting a restore point.

New Profile

LOCATION PROFILE
hpe-ezmeral-object

yaml edit delete

CLOUD PROVIDER
Generic S3

BUCKET NAME
mybucket

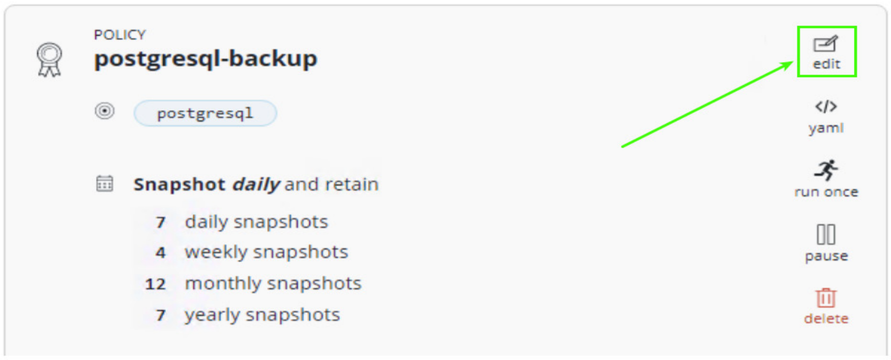
ENDPOINT
https://worker14n9hdd.test1.com:9000

SSL
disabled

Change the policy to add an export profile

You can now change the **postgresql-backup** policy to include an export to the object store.

Find the appropriate policy under Dashboard --> Policies, and then click edit.



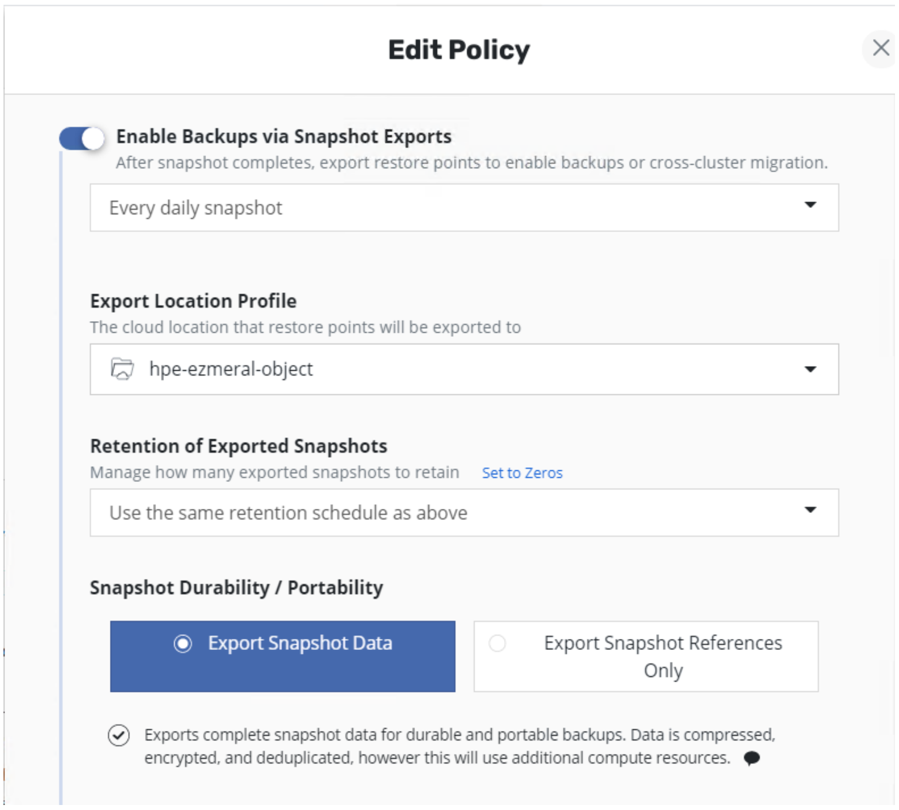
POLICY
postgresql-backup

postgresql

Snapshot *daily* and retain

7 daily snapshots
4 weekly snapshots
12 monthly snapshots
7 yearly snapshots

Select Enable Backups via Snapshot Exports and then select the appropriate profile. You will be able to provide separate retention for exported backups as well.



Edit Policy

Enable Backups via Snapshot Exports
After snapshot completes, export restore points to enable backups or cross-cluster migration.
Every daily snapshot

Export Location Profile
The cloud location that restore points will be exported to
hpe-ezmeral-object

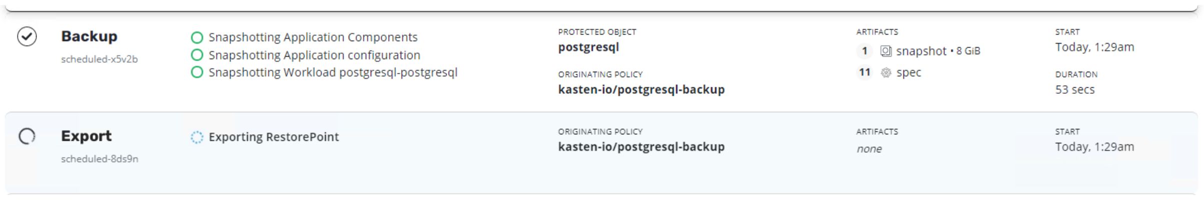
Retention of Exported Snapshots
Manage how many exported snapshots to retain   Set to Zeros
Use the same retention schedule as above

Snapshot Durability / Portability
[✓ Export Snapshot Data]   Export Snapshot References Only

Exports complete snapshot data for durable and portable backups. Data is compressed, encrypted, and deduplicated, however this will use additional compute resources.

Click Edit Policy and click run once again.

Go to the main dashboard and scroll down under Actions to view progress.



Backup scheduled-x5v2b	Snapshotting Application Components; Snapshotting Application configuration; Snapshotting Workload postgresql-postgresql	PROTECTED OBJECT **postgresql**; ORIGINATING POLICY **kasten-io/postgresql-backup**	ARTIFACTS 1 snapshot • 8 GiB; 11 spec	START Today, 1:29am; DURATION 53 secs
Export scheduled-8ds9n	Exporting RestorePoint	ORIGINATING POLICY **kasten-io/postgresql-backup**	ARTIFACTS *none*	START Today, 1:29am

Export will start as soon as the snapshot action finishes. When the circle turns solid green, click the job to open a details pane and view all artifacts that were captured in the previous step.

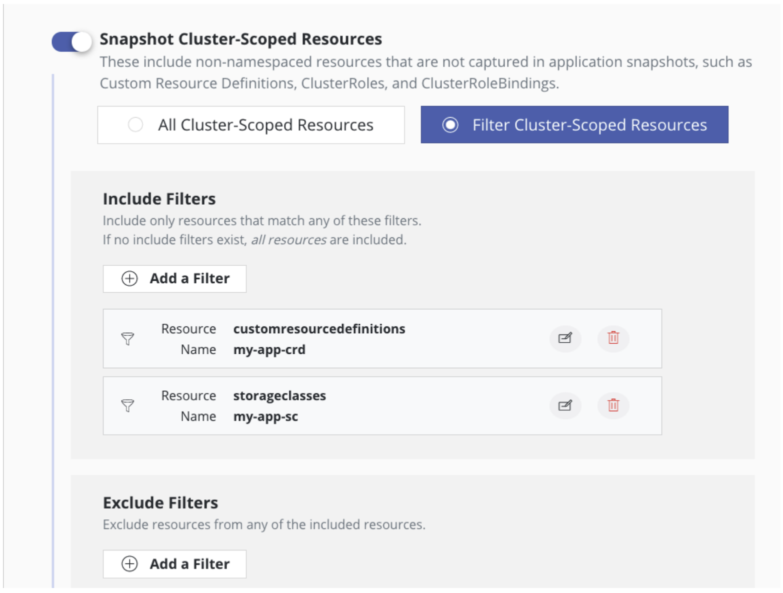
Backing up and restoring applications with cluster-scoped resources

K10 protects cluster-scoped resources in the same way that it protects applications with snapshot policies, backups, and manual snapshots.

Some applications have cluster-scoped resources such as StorageClasses, CustomResourceDefinitions or Cluster Role as well as namespaced components such as StatefulSets. To create a policy that protects the entire application, create a policy that protects both the application and its associated cluster-scoped resources.

Use filters to include and exclude cluster-scoped resources while creating a policy.

When this policy runs, it will create both a restore point for the application and a cluster restore point with artifacts that capture the application's cluster-scoped resources.



Snapshot Cluster-Scoped Resources

These include non-namespaced resources that are not captured in application snapshots, such as Custom Resource Definitions, ClusterRoles, and ClusterRoleBindings.

All Cluster-Scoped Resources [✓ Filter Cluster-Scoped Resources]

Include Filters

Include only resources that match any of these filters.
If no include filters exist, *all resources* are included.

Add a Filter

Resource	customresourcedefinitions
Name	my-app-crd

Resource	storageclasses
Name	my-app-sc

Exclude Filters

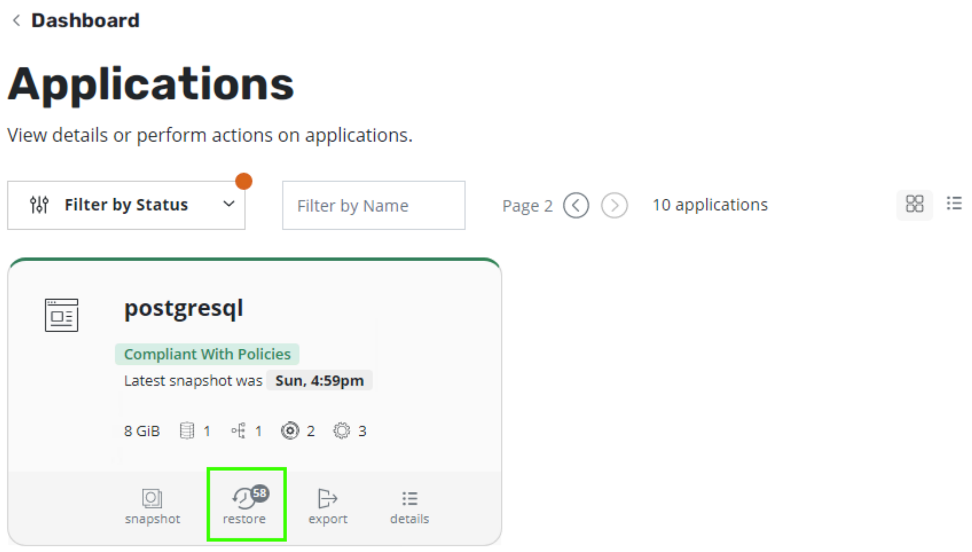
Exclude resources from any of the included resources.

Add a Filter

FIGURE 4. How to use filters to include and exclude cluster-scoped resources while creating a policy

RESTORING THE APPLICATION

Select the PostgreSQL in the applications menu and click the restore button.



Dashboard

Applications

View details or perform actions on applications.

Filter by Status | Filter by Name | Page 2 | 10 applications

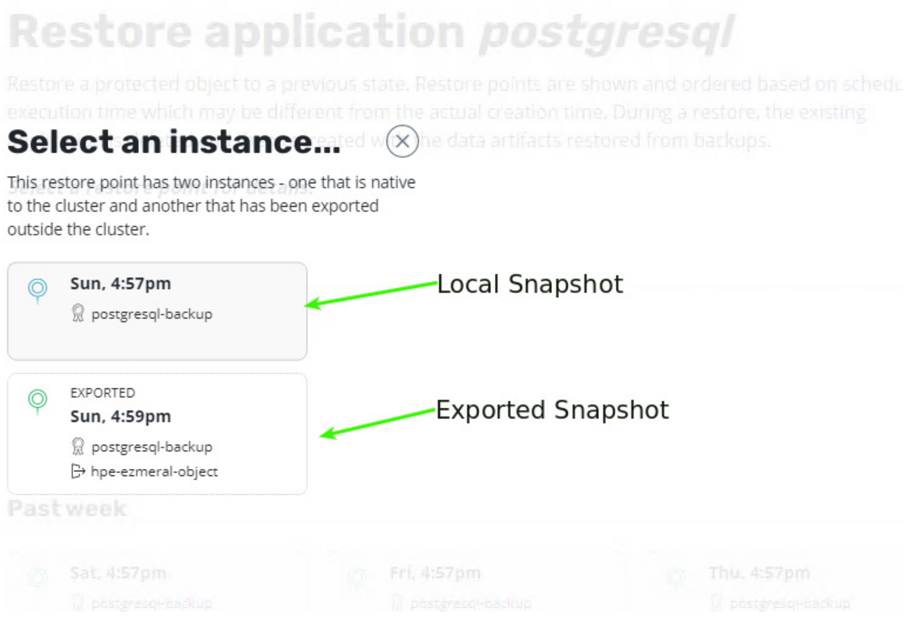
postgresql

Compliant With Policies

Latest snapshot was Sun, 4:59pm

8 GiB 1 1 2 3

You can see that now we have two restore points. A local restore point and an exported restore point.



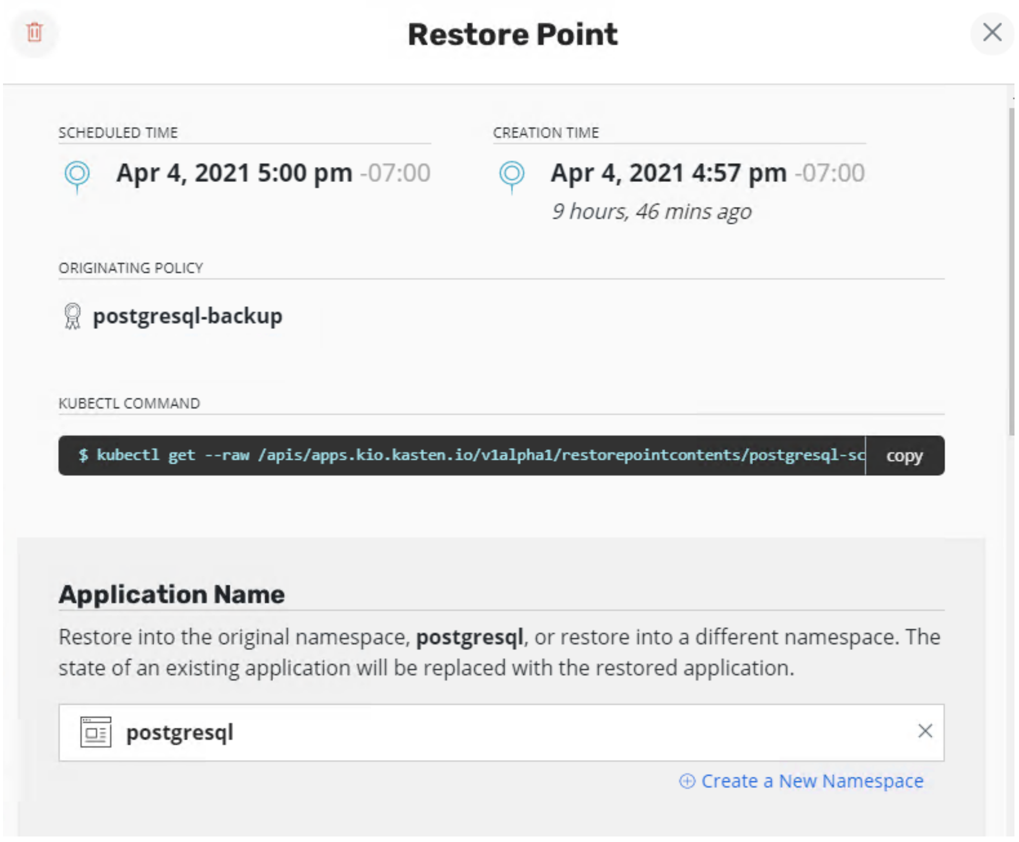
Select an instance...

This restore point has two instances - one that is native to the cluster and another that has been exported outside the cluster.

Sun, 4:57pm
postgresql-backup

EXPORTED
Sun, 4:59pm
postgresql-backup
hpe-ezmeral-object

You can select the local snapshot to restore the application and all the objects that are backed up as a part of the namespace. It can be restored to overwrite the objects in the same namespace or cloned to a new namespace.



Restore Point

SCHEDULED TIME
Apr 4, 2021 5:00 pm -07:00

CREATION TIME
Apr 4, 2021 4:57 pm -07:00
9 hours, 46 mins ago

ORIGINATING POLICY
postgresql-backup

KUBECTL COMMAND

```
$ kubectl get --raw /apis/apps.kio.kasten.io/v1alpha1/restorepointcontents/postgresql-sc
```

Application Name

Restore into the original namespace, **postgresql**, or restore into a different namespace. The state of an existing application will be replaced with the restored application.

postgresql

FIGURE 5. Exemplification of a Restore Point

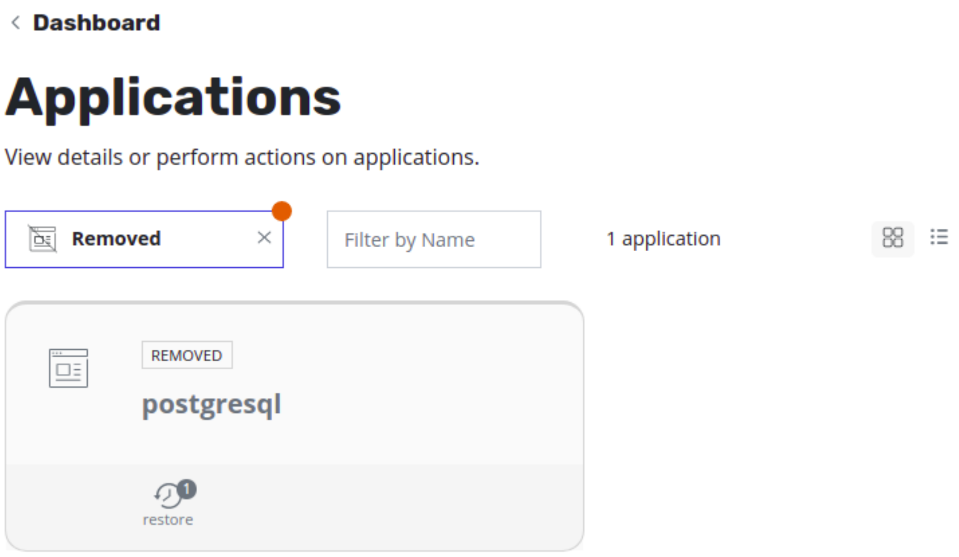
Restoring a deleted namespace

The exported restore point can be used even if the namespace is deleted or even if the cluster is deleted. It can also be used to restore a namespace to another cluster where Kasten is installed.

Let's delete the postgresql namespace and restore it from the exported snapshot.

```
kubectl delete ns postgresql
```

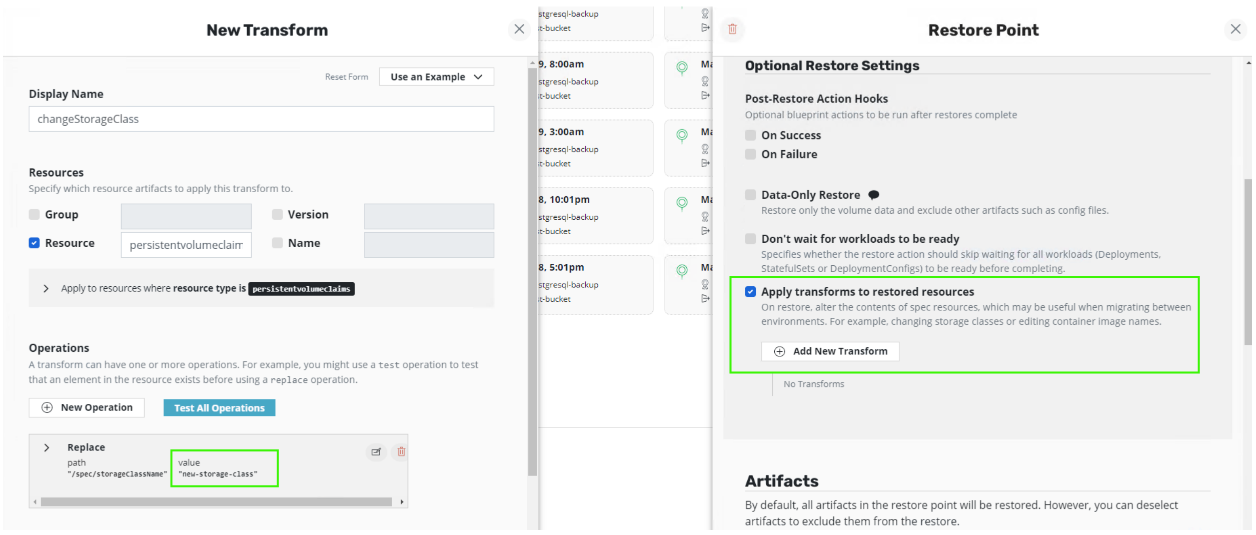
Once the namespace is deleted, you will not be able to see it in the application tab by default. You will have to select the filter to list the removed applications.



You will be able to select the exported restore point and restore as we did in the previous step.

Also, you will be able to apply **Transforms** to modify the manifest file if there are any requirements. For example, if you are going to restore the application to a different cluster with a different StorageClass, you will be able to replace the StorageClass spec using the Transforms before the restoration.

You will also be able to test out the Transforms applied to see the rendered manifests and validate before the restoration.



Display Name: changeStorageClass

Resources
Specify which resource artifacts to apply this transform to.

- [ ] Group
- [x] Resource: persistentvolumeclaim
- [ ] Version
- [ ] Name

Apply to resources where resource type is `persistentvolumeclaims`

Operations
A transform can have one or more operations. For example, you might use a `test` operation to test that an element in the resource exists before using a `replace` operation.

Replace
path "/spec/storageClassName"
value "new-storage-class"

Optional Restore Settings

Post-Restore Action Hooks
Optional blueprint actions to be run after restores complete
- [ ] On Success
- [ ] On Failure

- [ ] **Data-Only Restore**
  Restore only the volume data and exclude other artifacts such as config files.
- [ ] **Don't wait for workloads to be ready**
  Specifies whether the restore action should skip waiting for all workloads (Deployments, StatefulSets or DeploymentConfigs) to be ready before completing.
- [x] **Apply transforms to restored resources**
  On restore, alter the contents of spec resources, which may be useful when migrating between environments. For example, changing storage classes or editing container image names.

No Transforms

Artifacts
By default, all artifacts in the restore point will be restored. However, you can deselect artifacts to exclude them from the restore.

There can be other use cases such as moving an application to a faster storage or adding/removing/replacing specific annotations to the specs where you can leverage **Transforms** feature.

Verify if the restored pod has all the data.

```
# To get the password for "postgres" run:
       export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql
-o jsonpath="{.data.postgresql-password}" | base64 --decode)
# To connect to your database run the following command:
       kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace
postgresql --image docker.io/bitnami/postgresql:11.11.0-debian-10-r50 --
env="PGPASSWORD=$POSTGRES_PASSWORD" --command -- psql --host postgresql -U postgres -d
postgres -p 5432

postgres=# \c test
You are now connected to database "test" as user "postgres".
test=# select * from pets
test-# ;
   name    | owner   |       species       | sex |    birth      | death
----------+--------+-----------------+-----+-----------+-------
 Puffball | Diane  | hamster       | f   | 2010-03-30 |
 Spike | Mike    | pitbull       | m   | 2011-04-28 |
 Ashton    | Varoon | German Shepherd   | m    | 2014-06-15 |
 Bear  | Chris   | Rottweiler      | m   | 2013-10-10 |
 Toby  | Jenny   | Golden Retriever | m    | 2019-03-19 |
(5 rows)
```

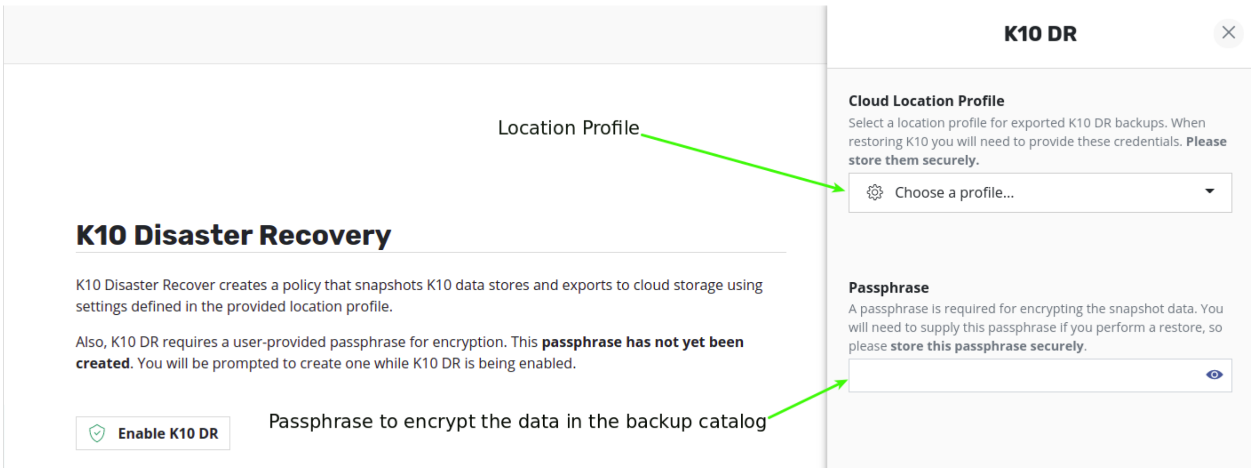
K10 DISASTER RECOVERY

K10 Disaster Recovery (DR) aims to protect K10 from the underlying infrastructure failures. In particular, this feature provides the ability to recover the K10 platform in case of a variety of disasters such as the accidental deletion of K10, failure of underlying storage that K10 uses for its catalog, or even the accidental destruction of the Kubernetes cluster on which K10 is deployed.

K10 enables DR with the help of an internal policy to back up its own datastores and store these in an object storage bucket or an NFS file storage location configured using a Location Profiles.

K10 DR settings can be accessed from the **Settings** icon in the top-right corner of the dashboard.

On the Settings page, select **K10 Disaster Recovery** and then click the **Enable K10 DR** button to enable disaster recovery.



Location Profile

K10 Disaster Recovery

K10 Disaster Recover creates a policy that snapshots K10 data stores and exports to cloud storage using settings defined in the provided location profile.

Also, K10 DR requires a user-provided passphrase for encryption. This **passphrase has not yet been created**. You will be prompted to create one while K10 DR is being enabled.

Enable K10 DR

Passphrase to encrypt the data in the backup catalog

K10 DR

Cloud Location Profile

Select a location profile for exported K10 DR backups. When restoring K10 you will need to provide these credentials. **Please store them securely.**

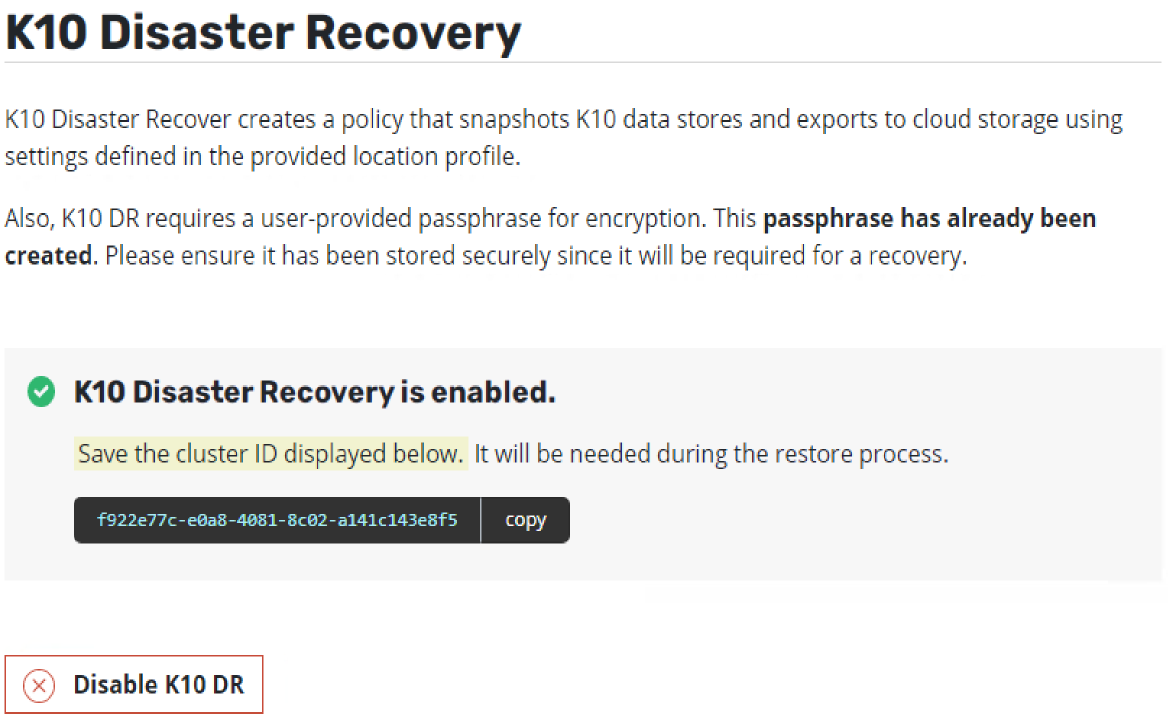
Choose a profile...

Passphrase

A passphrase is required for encrypting the snapshot data. You will need to supply this passphrase if you perform a restore, so please **store this passphrase securely**.

Passphrase provided will be saved as a secret k10-dr-secret in kasten-io namespace.

This creates a policy for K10 Disaster Recovery and its schedule/retention can be modified as per the requirement.



K10 Disaster Recovery

K10 Disaster Recover creates a policy that snapshots K10 data stores and exports to cloud storage using settings defined in the provided location profile.

Also, K10 DR requires a user-provided passphrase for encryption. This **passphrase has already been created**. Please ensure it has been stored securely since it will be required for a recovery.

K10 Disaster Recovery is enabled.

Save the cluster ID displayed below. It will be needed during the restore process.

```
f922e77c-e0a8-4081-8c02-a141c143e8f5
```

NOTE
After enabling K10 DR, it is **essential** that you copy and save the following to successfully recover K10 from a disaster:

1. The cluster ID displayed on the disaster recovery page.
2. The DR passphrase entered previously.
3. The credentials and object storage bucket or the NFS file storage information (used in the location profile configuration previously).

Without this information, K10 Disaster Recovery will not be possible.

Restoring K10 to a new cluster

Recovering from a K10 backup involves the following sequence of actions:

1. Create a Kubernetes Secret, k10-dr-secret, using the passphrase provided while enabling DR.

```
kubectl create secret generic k10-dr-secret \
   --namespace kasten-io \
   --from-literal key=<passphrase>
```

2. Install K10 instance in the new cluster by following Using the Helm chart to install Kasten.
3. Configure the same location profile in the new cluster where the K10 DR backups are stored by following Setting up the location profile in Kasten Dashboard.

4. Restoring the K10 backup using the following Helm command.

```
#Install the helm chart that creates the K10 restore job and wait for completion of
the `k10-restore` job

#Assumes that K10 is installed in 'kasten-io' namespace.

helm install k10-restore kasten/k10restore --namespace=kasten-io \
    --set sourceClusterID=<source-clusterID> \
    --set profile.name=<location-profile-name>
```

After the K10 restore job is completed, you can restore cluster-scoped resources and application resources.

NOTE
Prior to recovering applications, it may be desirable to restore cluster-scoped resources. Cluster-scoped resources may be needed for cluster configuration or as part of application recovery.

CONCLUSION

The HPE and Kasten by Veeam partnership helps companies confidently run and protect cloud-native applications on Kubernetes, across private, hybrid, and multi-cloud environments. Kasten K10 can restore applications on HPE Ezmeral Container Platform to a known state with granular control of backup and restore capabilities for the entire application stack. These container applications can also be seamlessly recovered and redeployed across clusters and sites to facilitate effective disaster recovery strategies.

You can get started in a few minutes with a free and full featured version of Kasten K10 by going to kasten.io/try-kasten-k10. Also, you can get additional information from:

- Access and Authentication to the dashboard
- Logical and application consistent backups
- Multi-tenancies and RBAC
- Multicluster management
- HPE and Kasten by Veeam partnership
- HPE Ezmeral Container Platform website and technical paper
- HPE Ezmeral Data Fabric website and technical paper
- HPE Ezmeral Marketplace

LEARN MORE AT

hpe.com/ezmeral

Make the right purchase decision.
Contact our presales specialists.

Chat Email Call

Get updates

a50004221ENW, June 2021

Hewlett Packard
Enterprise