

# Kasten K10 by Veeam on Red Hat OpenShift

## Solution Guide

Last Updated: 1/29/2021



# Contents

<b>1</b>	<b>OVERVIEW</b>	<b>5</b>
1.1	KASTEN K10 ON RED HAT OPENSIFT	5
1.2	K10 USE-CASES ON RED HAT OPENSIFT	6
1.3	COST	7
<b>2</b>	<b>ARCHITECTURE</b>	<b>8</b>
<b>3</b>	<b>PLANNING THE DEPLOYMENT</b>	<b>9</b>
3.1	TECHNICAL REQUIREMENTS	9
3.2	RESOURCE REQUIREMENTS	9
<b>4</b>	<b>DEPLOYMENT</b>	<b>10</b>
4.1	K10 PREREQUISITES	10
4.2	K10 DEPLOYMENT	10
4.2.1	<i>Step 1 - Install Helm</i>	10
4.2.2	<i>Step 2 - Configure Helm Repositories</i>	10
4.2.3	<i>Step 3 - Annotate Storage Classes</i>	10
4.2.4	<i>Step 4 - Verify prerequisites are satisfied</i>	11
4.2.5	<i>Step 5 - Verify Each Storage Class</i>	12
4.2.6	<i>Step 6 - Installing K10</i>	13
<b>5</b>	<b>K10 WORKFLOWS - BACKUP, RESTORE, AND DISASTER RECOVERY</b>	<b>16</b>
5.1	TEST APPLICATION PREREQUISITES	16
5.2	TEST APPLICATION DEPLOYMENT	19
5.2.1	<i>Step 1 - Deploy the application</i>	19
5.2.2	<i>Step 2 - Validate Application Is Running</i>	19
5.2.3	<i>Step 3 - Make Application Remotely Accessible</i>	20
5.2.4	<i>Step 4 - Verify Application Is Accessible</i>	20
5.2.5	<i>Step 5 - Finish Configuring WordPress</i>	20
5.3	TEST APPLICATION BACKUP	22
5.3.1	<i>Step 1 - Navigate Kasten Applications</i>	22
5.3.2	<i>Step 2 - Create Restore Point</i>	22
5.4	TEST APPLICATION RESTORE	24
5.4.1	<i>Step 1 - Delete a WordPress post</i>	24
5.4.2	<i>Step 2 - Navigate Kasten Applications</i>	25
5.4.3	<i>Step 3 - Restore from Restore Point</i>	25

## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

5.4.4	<i>Step 4 - Select the Restore Point</i>	26
5.4.5	<i>Step 5 - Select Namespace for Restore Operation</i>	27
5.4.6	<i>Step 6 - Confirm Restore Initiation</i>	27
5.4.7	<i>Step 7 - Monitor Restore Operation</i>	28
5.4.8	<i>Step 8 - Verify Data after Restore</i>	28
5.4.9	<i>Step 9 - Verify Application Component Restore</i>	29
5.5	<b>BACKUP POLICIES</b>	29
5.5.1	<i>Step 1 - Navigate Kasten Applications</i>	29
5.5.2	<i>Step 2 - Create Backup Policy</i>	30
5.5.3	<i>Step 3 - Specify Backup Policy Parameters</i>	31
5.5.4	<i>Step 4 - Backup Policy Summary</i>	31
5.5.5	<i>Step 5 - Successful Scheduled Backup</i>	32
5.6	<b>DISASTER RECOVERY CAPABLE BACKUP</b>	33
5.6.1	<i>Introduction</i>	33
5.6.2	<i>Step 1 - Access K10 Settings</i>	33
5.6.3	<i>Step 2 - Create Object Store Profile</i>	34
5.6.4	<i>Step 3 - Configure Object Store Profile</i>	34
5.6.5	<i>Step 4 - Configured Object Store Profile</i>	35
5.6.6	<i>Step 5 - Edit Existing Policy</i>	35
5.6.7	<i>Step 6 - Configure Existing Policy</i>	36
5.6.8	<i>Step 7 - Get Import Details</i>	37
5.6.9	<i>Step 8 - Copy Import Details</i>	37
5.6.10	<i>Step 9 - Verify Successful Export</i>	38
5.7	<b>DISASTER RECOVERY RESTORE</b>	39
5.7.1	<i>Introduction</i>	39
5.7.2	<i>Step 1 - Create Location Profile</i>	39
5.7.3	<i>Step 2 - Create Import Policy</i>	39
5.7.4	<i>Step 3 - Configure Import Policy</i>	40
5.7.5	<i>Step 4 - Wait for Import and Restore Completion</i>	41
<b>6</b>	<b>API AND CLI</b>	<b>43</b>
6.1	LIST BACKUP POLICIES	43
6.2	CREATE BACKUP POLICY	43
6.3	LIST BACKUP ACTIONS	44
6.4	TRIGGER BACKUP POLICY	44

## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

6.5	SUPPORT	45
6.5.1	<i>Kasten Team</i>	45
6.5.2	<i>Red Hat Team</i>	45

## 1 Overview

This Solution Guide provides step-by-step instructions for deploying the Kasten K10 by Veeam® data management platform on Red Hat® OpenShift® Container Platform. It was created by Kasten by Veeam in collaboration with the Red Hat Data Services team.

Solution Guides are reference deployments that can be used to deploy software end to end by following best practices.

### 1.1 Kasten K10 on Red Hat OpenShift

The K10 data management platform provides enterprise operations teams an easy-to-use, scalable, and secure system for backup/restore, disaster recovery, and mobility of Red Hat OpenShift applications.

K10's application-centric approach and deep integrations with relational and NoSQL databases, Kubernetes distributions, Red Hat OpenShift versions, and all clouds provides teams the freedom of infrastructure choice without sacrificing operational simplicity. Policy-driven and extensible, K10 provides a native Kubernetes API and includes features such as full-spectrum consistency, database integrations, automatic application discovery, multi-cloud mobility, and a powerful web-based user interface.

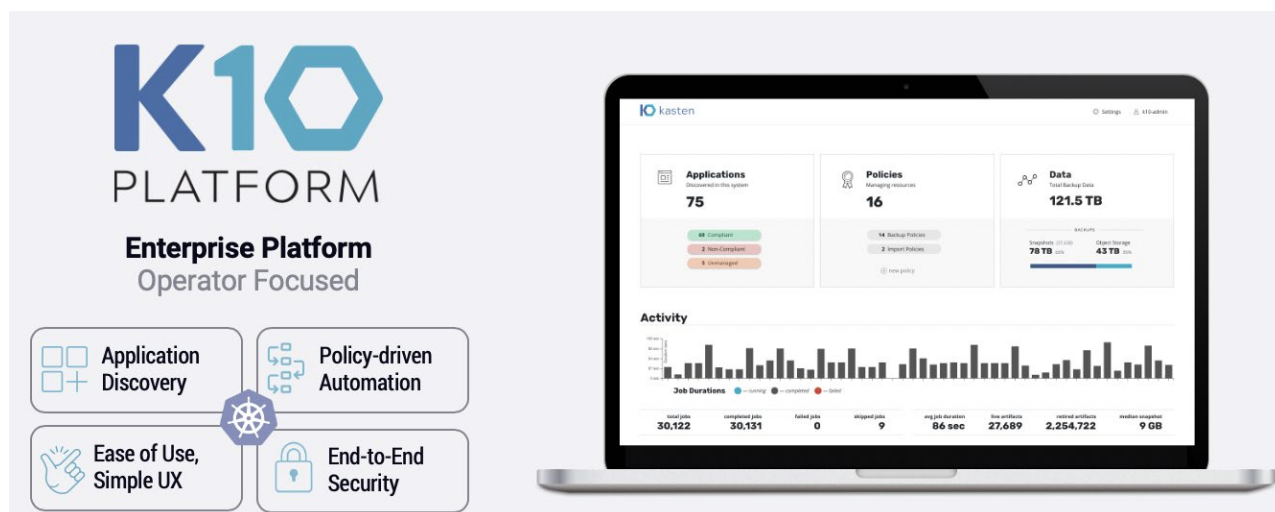


Figure 1 - K10 platform on OpenShift Container Platform

## 1.2 K10 use-cases on Red Hat OpenShift

Given K10's extensive ecosystem support, you have the flexibility to choose environments (public/ private/ hybrid cloud/ on-prem) and Red Hat OpenShift versions (cloud vendor managed or self-managed) in support of three principal use cases: Backup and Restore, Disaster Recovery, and Application Mobility.

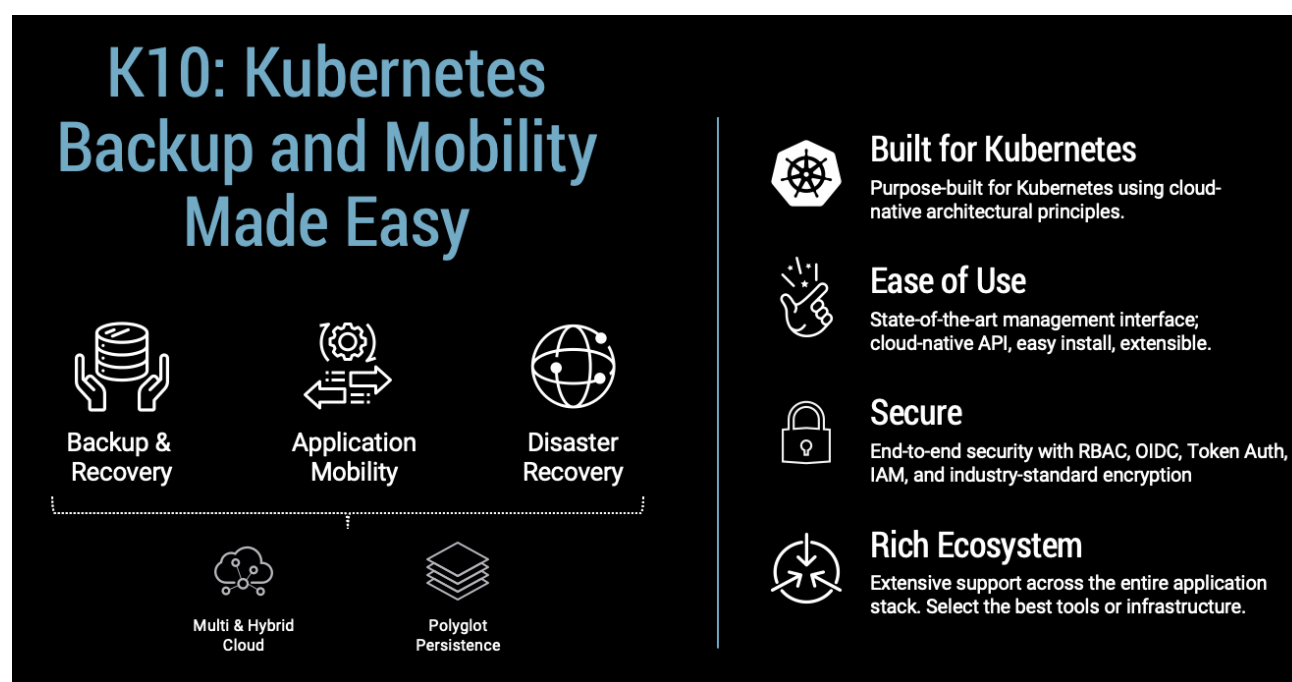


Figure 2 - K10 Use cases on Red Hat OpenShift Container Platform

- **Data Protection:** Snapshots are the basis of persistent data capture in K10. They are usually used in the context of disk volumes (PVC/PVs) used by the application but can also apply to application-level data capture (e.g., by leveraging Kasten's open and extensible application blueprints). Given the limitations of snapshots, it is often advisable to set up backups of your application stack. However, even if your snapshots are durable, backups might still be useful in a variety of use cases including lowering costs with K10's data deduplication or backing your snapshots up in a different infrastructure provider for cross-cloud delivery. Once applications have been protected via a policy or a manual action, it is possible to restore them in-place or clone them to a different namespace.
- **Seamless Mobility:** The ability to move an application across clusters is an extremely powerful feature that enables a variety of use cases including Disaster Recovery (DR), Test/Dev with realistic data sets, and performance testing in isolated environments. In particular, the K10 platform is built to support application migration and mobility in a variety of different and overlapping contexts:
  - Cross-Namespace
  - Cross-Cluster
  - Cross-Platform

- **End-to-End Security:** K10 platform seamlessly integrates into a customer's environment with their authentication tool ([Red Hat OpenShift OAuth proxy](#), OIDC, Centrify, LDAP, SAML, Kerberos, etc.,) and offers a variety of different ways to secure access to its dashboard and [APIs](#). K10 supports a flexible permissions model which allows scoping of user permissions to perform K10 actions only within the context of specified applications. To facilitate role-based access for users, K10 leverages Kubernetes ClusterRoles and Bindings, which are user-configurable. K10 uses enterprise-grade AES-256 algorithm and TLS/SSL protocols to encrypt all data at-rest and in-flight.
- **Deep Kubernetes integration:** K10 platform can manage the entire stateful lifecycle of an application—from optimal scheduling decisions based on the proximity to data, to Kubernetes-driven backup, restore, and migration functionality.
- **Deep OCS integration:** K10 leverages the OpenShift Container Storage Interface API available through Red Hat OpenShift Container Storage 4.6 to provide a seamless experience when it comes to backup and restore operations. Through the CSI snapshot and clones capabilities provided by OpenShift Container Storage, Kasten K10 can perform backups of your data using OpenShift Container Storage storage classes (PVCs), your metadata (Kubernetes and OpenShift APIs such as namespaces and secrets), and provides local persistence of the backup for a minimal restore time and the ability to restore a running application namespace while also allowing restoring an application to a different namespace for test and QA purposes or to a different OpenShift Container Platform cluster.

This Solution Guide deploys K10 on a Red Hat OpenShift cluster within a few minutes to enable data protection.

### 1.3 Cost

The K10 platform has a Starter and Enterprise edition. The fully featured Starter edition is free up to a limited number of nodes. For more information on license types, features included with each type of license, and how to purchase, upgrade or transfer your license, visit [Kasten](#).

## 2 Architecture

Below is a high-level overview of the K10 architecture that can be deployed against several distributions and/or infrastructure types.

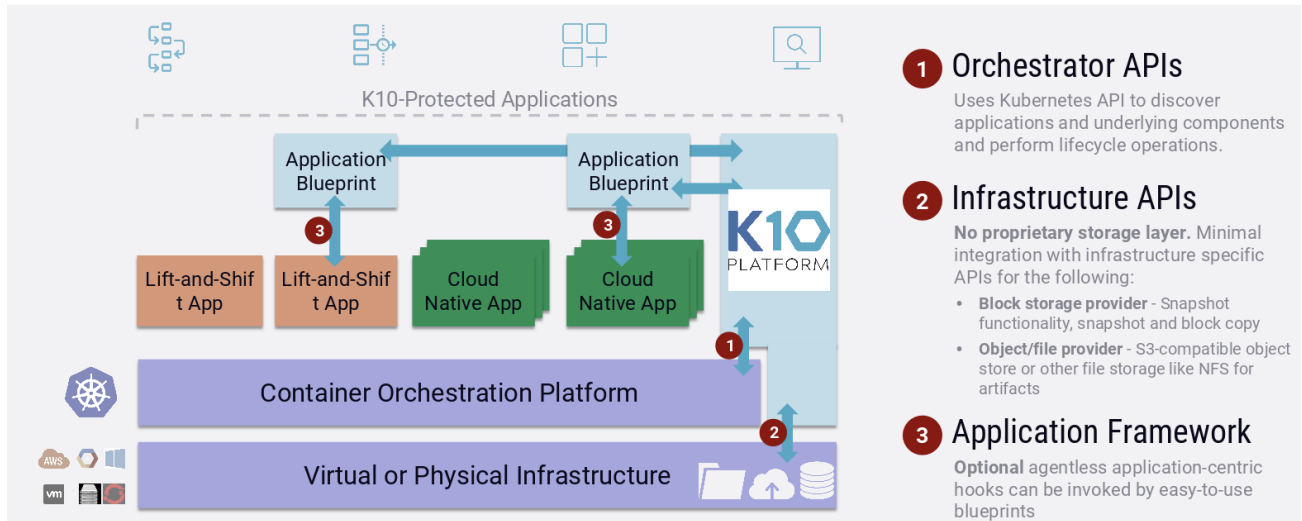


Figure 3 - High level overview of K10 architecture



## 3 Planning the Deployment

### 3.1 Technical requirements

Kasten K10 requires the following prerequisites:

- Red Hat OpenShift Container Platform 4.6
- Red Hat OpenShift Container Storage 4.6
- kubectl and oc client to communicate with the clusters
- Knowledge of Kubernetes
- OpenShift or Kubernetes Command Line Interface knowledge

### 3.2 Resource requirements

For a deployment in an OpenShift Container Platform environment, Kasten K10 will require the following resources. The numbers can vary based on the number of applications protected by K10.

Pod Count = 21\*  
CPU (cores) = 15m  
Memory = 750Mi

Note: 1m == 1millicore (1/1000 AWS CPU core)

**Note:** Subject to change as we continue to improve resource utilization

## 4 Deployment

### 4.1 K10 Prerequisites

This guide provides steps to ensure a smooth and successful K10 installation in your Red Hat OpenShift cluster while leveraging the OpenShift Container Storage snapshot and clone features for your backup and restore strategy.

- Pre-flight checks
- Install Helm Package Manager (v2.11.0+)
- Red Hat OpenShift Container Storage 4.6 or higher

### 4.2 K10 Deployment

We will also publish a version of this document that includes the Operator Hub based installation procedure in addition to the Helm based deployment enumerated below.

#### 4.2.1 Step 1 - Install Helm

Deploying Kasten K10 requires that your OpenShift Container Platform client machine has access to the helm command.

- On MacOS, we recommend using brew or pip to deploy the helm command. Alternatively, you may download the Helm binary from the same URL as for Linux client machines.
- On Linux, visit <https://github.com/helm/helm/releases> and download the appropriate version.

#### 4.2.2 Step 2 - Configure Helm Repositories

Deploying Kasten K10 requires a reconfiguration of your OpenShift Container Platform client machine from which you will deploy the software. To successfully deploy our solution, you must add the Kasten K10 Helm chart repository as well as client Configure and update the helm repo to download appropriate Kasten charts.

```
$ helm repo add kasten https://charts.kasten.io
```

#### 4.2.3 Step 3 - Annotate Storage Classes

Kasten K10 requires a special annotation on the storage classes that it needs to support as well as a retain policy. OpenShift Container Storage snapshot storage classes are not annotated by default and do not have a retain capability.

You can configure the correct retain policy as well as apply the correct annotation using the following commands for both OpenShift Container Storage storage classes.

## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

```
$ cat <<EOF | oc apply -f -
apiVersion: snapshot.storage.k8s.io/v1beta1
deletionPolicy: Retain
driver: openshift-storage.cephfs.csi.ceph.com
kind: VolumeSnapshotClass
metadata:
  name: ocs-storagecluster-cephfsplugin-snapclass-k10
  annotations:
    k10.kasten.io/is-snapshot-class: "true"
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/snapshotter-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
EOF
```

```
$ cat <<EOF | oc apply -f -
apiVersion: snapshot.storage.k8s.io/v1beta1
deletionPolicy: Retain
driver: openshift-storage.rbd.csi.ceph.com
kind: VolumeSnapshotClass
metadata:
  name: ocs-storagecluster-rbdplugin-snapclass-k10
  annotations:
    k10.kasten.io/is-snapshot-class: "true"
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/snapshotter-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
EOF
```

### 4.2.4 Step 4 - Verify prerequisites are satisfied

Before proceeding with the install we recommend you use the tool created by Kasten engineering to inspect your environment and detect any condition that could jeopardize the deployment of K10. This tool requires the kubectl command to be available on your client machine. The appropriate kubectl binary is available from the OpenShift client download package.

```
$ 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
[... truncated ...]
Validating Provisioners:
```

```
openshift-storage.rbd.csi.ceph.com:
Is a CSI Provisioner - OK
Storage Classes:
  ocs-storagecluster-ceph-rbd
  Valid Storage Class - OK
Volume Snapshot Classes:
  ocs-storagecluster-rbdplugin-snapclass
  ocs-storagecluster-rbdplugin-snapclass-k10
  Has k10.kasten.io/is-snapshot-class annotation set to true - OK
  Has deletionPolicy 'Retain' - OK

[... truncated ...]

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

Make sure both `ocs-storagecluster-ceph-rbd` and `ocs-storagecluster-cephfs` storage classes are detected and show an **OK** status when the script has completed.

### 4.2.5 Step 5 - Verify Each Storage Class

Before proceeding with the install we recommend you use the tool created by the Kasten engineering to functionally validate a storage class. The tool below will create a PVC using the desired storage, create a snapshot and perform a rollback/restore for an in-depth validation of the storage class.

```
$ curl -s https://docs.kasten.io/tools/k10_primer.sh | bash /dev/stdin \
                                                                    -s ocs-storagecluster-ceph-rbd

Namespace option not provided, using default namespace
Checking for tools
--> Found kubectl
--> Found helm
[... truncated ...]

Running K10Primer Job in cluster with command-
./k10primer storage csi-checker -s ocs-storagecluster-ceph-rbd
[... truncated ...]
Creating Snapshot
Validating snapshotter image version and the ability to create a snapshot from source
[... truncated ...]

CSI Snapshot Walkthrough:
[... truncated ...]
Successfully tested CSI Snapshot Restore - OK

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

### 4.2.6 Step 6 - Installing K10

You can install Kasten K10 on Red Hat OpenShift regardless of the infrastructure provider. Note that specific infrastructure providers will require you to provide access. More details can be found [here](#).

#### 1. Create a namespace

```
$ oc create namespace kasten-io
```

#### 2. Install K10

```
$ helm install k10 kasten/k10 --namespace=kasten-io --set scc.create=true --set route.enabled=true --set auth.tokenAuth.enabled=true
```

#### 3. Verify the installation

```
$ oc get pods --namespace=kasten-io
NAME                                READY STATUS  RESTARTS  AGE
aggregatedapis-svc-cff7bff46-tfxks  1/1   Running  0         69s
auth-svc-66fdbcb8c77-447gb         1/1   Running  0         69s
catalog-svc-59978f69dd-rpp79       2/2   Running  0         69s
config-svc-6b6b64dbff-q64k9        1/1   Running  0         69s
crypto-svc-766558cbc5-ct9l7        1/1   Running  0         69s
dashboardbff-svc-6b7cc6fb89-crkp5  1/1   Running  0         69s
executor-svc-55f6c4477b-frhdt      2/2   Running  0         69s
executor-svc-55f6c4477b-mwx5l      2/2   Running  0         69s
executor-svc-55f6c4477b-rg9zq      2/2   Running  0         69s
frontend-svc-dff69b7f5-ktcfh       1/1   Running  0         69s
gateway-67cdfc59cf-kds5t           1/1   Running  0         69s
jobs-svc-759bcf9fc-xdnp8           1/1   Running  0         69s
kanister-svc-9b8b757f-rljcx        1/1   Running  0         69s
logging-svc-668944bfb8-56j9r       1/1   Running  0         69s
metering-svc-cd85b7b69-dt485       1/1   Running  0         69s
prometheus-server-78b94b85fb-7mhv6 1/2   Running  0         68s
state-svc-9485d954f-5vcwf          1/1   Running  0         69s
```

**Note:** In the unlikely event scenario that pods are stuck in any other state, please follow the [support](#) documentation to debug further.

#### 4. Dashboard Access

```
$ oc get route -n kasten-io
NAME      HOST/PORT                                [truncated]
k10-route k10-route-kasten-io.apps.ocp45.ocstraining.com / [truncated]
```

Point your browser to the route name followed by /k10/# (e.g., <http://k10-route-kasten-io.apps.ocp45.ocstraining.com/k10/#>).

### 5. Dashboard Credentials

You will have to generate an access token to be able to satisfy the Kasten K10 dashboard authentication mechanism. From your Red Hat OpenShift client machine, once logged in the Red Hat OpenShift cluster, issue the following command:

```
$ oc whoami --show-token  
sha256~bywYS4f9HEWxyG-A6lo9Bro3JRyd1n3xCFBnvHsULF0
```

**Note:** Copy the entire output so it can be pasted in the Kasten K10 user interface to login.

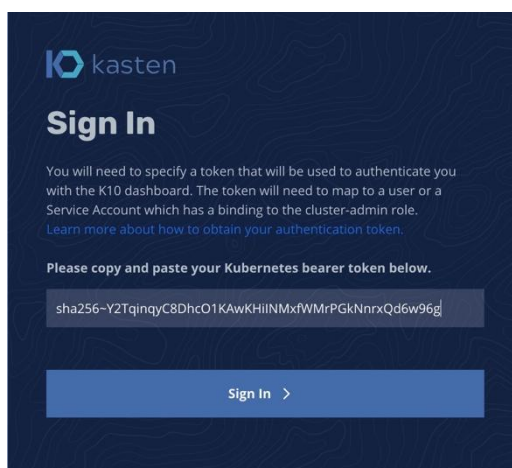


Figure 4 - Initial K10 dashboard login

When prompted, enter your company name, your email address and accept the license agreement to gain access to the dashboard as shown below.

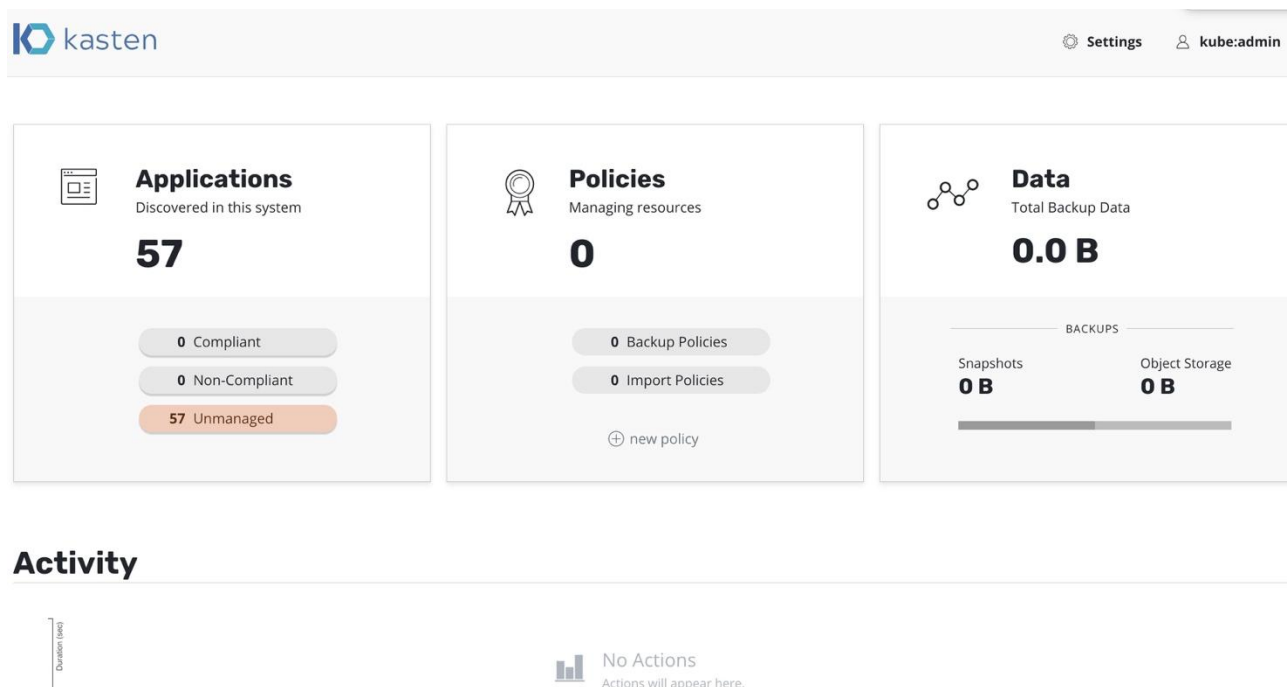


Figure 5 - K10 dashboard landing page

## 5 K10 Workflows – Backup, Restore, and Disaster Recovery

In this example, we will walk through how to use Kasten K10 to backup and restore a sample WordPress application. This application namespace has one MySQL pod, to provide data persistence, and a WordPress pod to provide an application web frontend.

The MySQL database pod is configured to use the `ocs-storagecluster-ceph-rbd` storage class.

- Deploy the sample application
- Backup and restore workflow using Kasten K10
- Disaster recovery workflow using Kasten K10 to a different cluster

### 5.1 Test Application Prerequisites

The test application we deployed was created from this [example project](#): **Deploying WordPress and MySQL with Persistent Volumes**.

MySQL Deployment file

```
$ cat kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=redhat
resources:
- mysql-deployment.yaml
- wordpress-deployment.yaml
(.venv) OH-iMac-JCL:wordpress-mysql-demo JCL$ cat mysql-deployment.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
  - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
  - ReadWriteOnce
```



## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

```
storageClassName: ocs-storagecluster-ceph-rbd
resources:
  requests:
    storage: 20Gi
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-pass
                  key: password
          ports:
            - containerPort: 3306
              name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-pv-claim
```

### WordPress Deployment file

```
$ cat wordpress-deployment.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
```

```
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
        - image: wordpress:4.8-apache
          name: wordpress
          env:
            - name: WORDPRESS_DB_HOST
              value: wordpress-mysql
            - name: WORDPRESS_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-pass
                  key: password
          ports:
            - containerPort: 80
```

```
name: wordpress
volumeMounts:
- name: wordpress-persistent-storage
  mountPath: /var/www/html
volumes:
- name: wordpress-persistent-storage
  persistentVolumeClaim:
    claimName: wp-pv-claim
```

### kustomization file

```
$ cat kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=redhat
resources:
- mysql-deployment.yaml
- wordpress-deployment.yaml
```

## 5.2 Test Application Deployment

### 5.2.1 Step 1 - Deploy the application

Use the commands below to create a namespace called wordpress and deploy the application.

```
$ cd wordpress-mysql-demo
$ oc create ns wordpress
namespace/wordpress created
$ oc project wordpress
$ oc adm policy add-cluster-role-to-user cluster-admin -z wordpress
Warning: ServiceAccount wordpress not found
clusterrole.rbac.authorization.k8s.io/cluster-admin added: "wordpress"
$ oc apply -k ./
secret/mysql-pass-ctm2f4889c created
service/wordpress-mysql created
service/wordpress created
deployment.apps/wordpress-mysql created
deployment.apps/wordpress created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
```

### 5.2.2 Step 2 - Validate Application Is Running

Use the commands below to verify the application is successfully deployed.

```
$ oc get pods
NAME                READY STATUS RESTARTS AGE
```

## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

```
wordpress-6c5d9c567c-lrhpq 1/1 Running 0 3m
wordpress-mysql-68f759bd8f-l97b7 1/1 Running 0 3m
$ oc get pvc
NAME          STATUS VOLUME          CAPACITY ACCESS MODES STORAGECLASS          AGE
mysql-pv-claim Bound  pvc-140a892d-32de-4ce2-ae39-ab84011caf6f 20Gi  RWO          ocs-storagecluster- 3m
ceph-rbd
wp-pv-claim   Bound  pvc-7dc440a6-c94b-48bd-9ed1-90bc3d834a4e 20Gi  RWO          ocs-storagecluster- 3m
ceph-rbd
```

### 5.2.3 Step 3 - Make Application Remotely Accessible

Use the commands below to make the application remotely accessible.

```
$ oc get svc -o name
service/wordpress
service/wordpress-mysql
$ oc expose svc/wordpress
route.route.openshift.io/wordpress exposed
```

### 5.2.4 Step 4 - Verify Application Is Accessible

Point your web browser to the URL indicated by the route.

```
$ oc get route
NAME    HOST/PORT          PATH SERVICES PORT TERMINATION WILDCARD
wordpress wordpress-wordpress.apps.ocp45.ocstraining.com wordpress 80      None
```

### 5.2.5 Step 5 - Finish Configuring WordPress

Follow the on-screen instruction to complete the setup.

1. First select the language
2. Choose a name for your WordPress site
3. Configure the admin username and password
4. Login WordPress
5. Add at least two sample posts

The final application should look similar to the screen capture below.

## Kasten K10 by Veeam on Red Hat OpenShift Container Storage 4.6

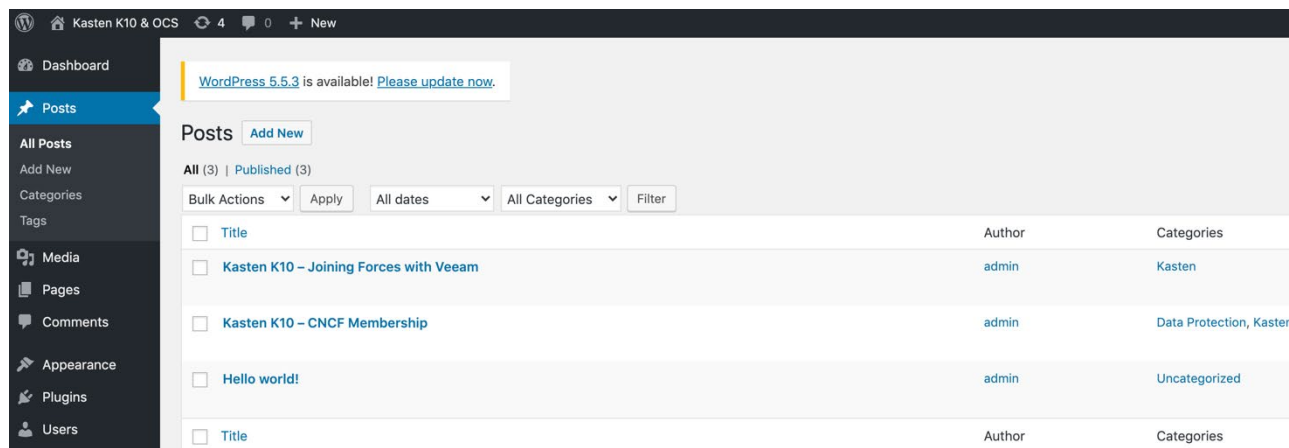


Figure 6 - WordPress Example

**Note:** K10 automatically discovers PostgreSQL and MongoDB instances. Following the successful deployment of your database instance, click on the Applications card on the K10 dashboard to see the discovered PostgreSQL instance.

## 5.3 Test Application Backup

### 5.3.1 Step 1 - Navigate Kasten Applications

In the Application tile click 'Unmanaged' as illustrated below.

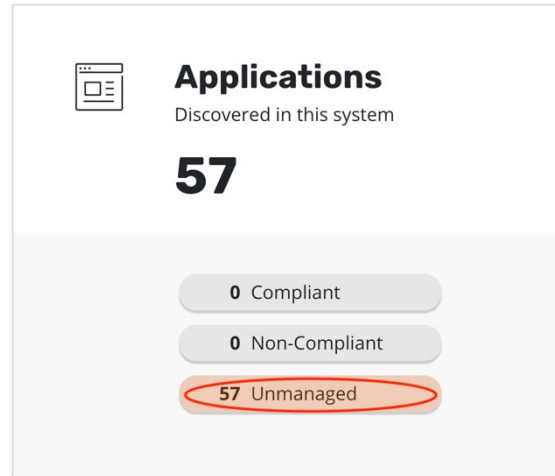


Figure 7 - Application Tile

### 5.3.2 Step 2 - Create Restore Point

In this section, we will use K10's default backup mechanism which relies on taking volume snapshots. Click on the Applications card in the K10 dashboard as illustrated in 5.3.1, locate your application using the filter and create a Restore Point to perform a full manual backup as illustrated below.

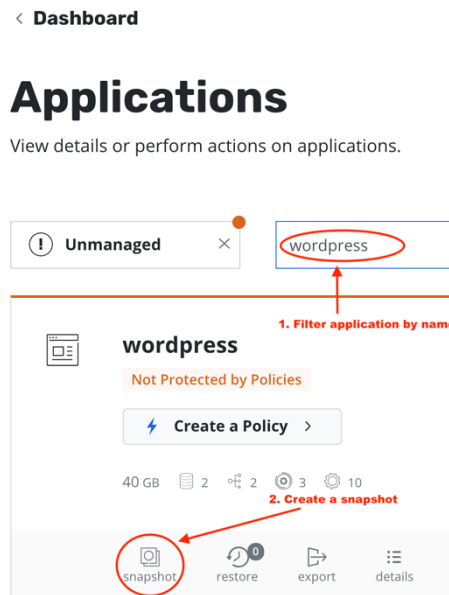


Figure 8 - Create Manual Restore Point

In the pane that opens on the right-hand side of your browser, keep the default options selected and simply click 'Snapshot Application' as illustrated below.



Figure 9 - Confirm Manual Snapshot

Check the progress of the backup action in the main K10 dashboard.

## Activity

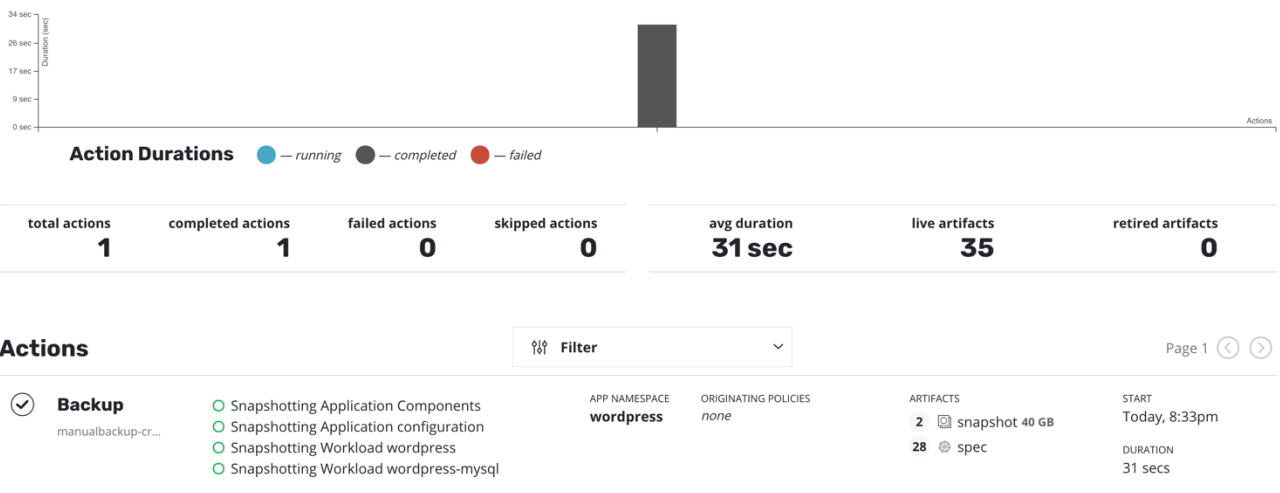


Figure 10 - Activity Dashboard

The completion of the backup process will result in the creation of a Restore Point (a set of configuration and data artifacts) which can be used for a restore operation.

## 5.4 Test Application Restore

### 5.4.1 Step 1 - Delete a WordPress post

To make sure the restore is operational, make sure you delete a post from the test WordPress application. Here are the existing posts.

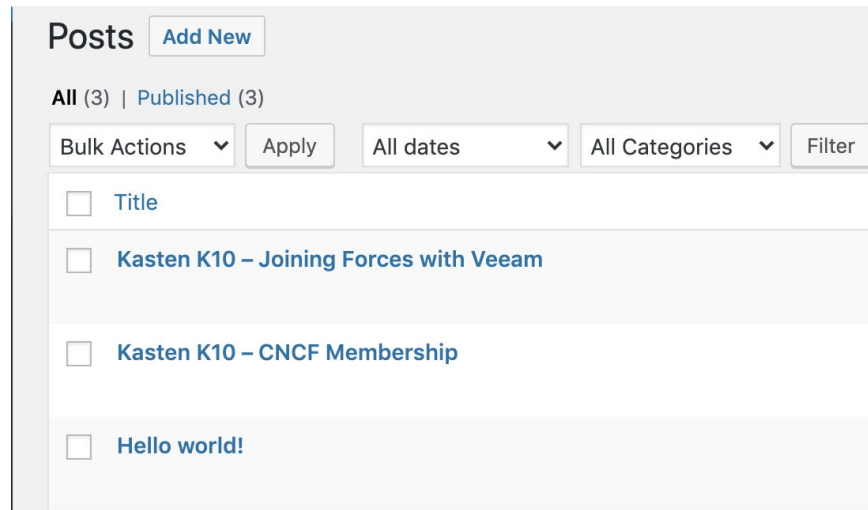


Figure 11 - Existing WordPress Posts

Delete the post title 'Kasten K10 - CNCF Membership' so we can verify the successful completion of the restore process we are about to trigger. This will prove we can perform a restore in-place for a running application.

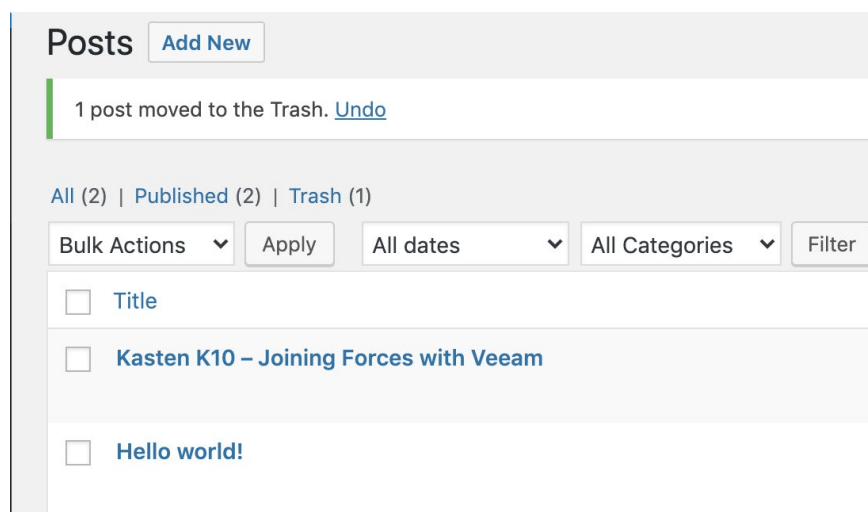


Figure 12 - After WordPress Post Deletion



### 5.4.2 Step 2 - Navigate Kasten Applications

In the Application tile click 'Unmanaged' as illustrated below.

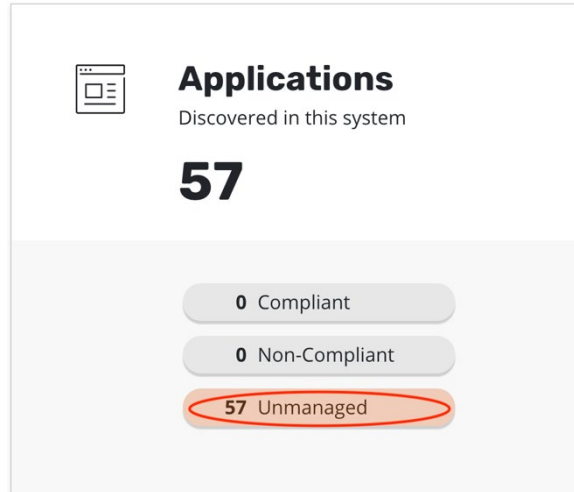


Figure 13 - Navigate to Applications

### 5.4.3 Step 3 - Restore from Restore Point

In the list of Unmanaged applications, use the filter to find the WordPress application and select the 'Restore' function as illustrated below.

## Applications

View details or perform actions on applications.

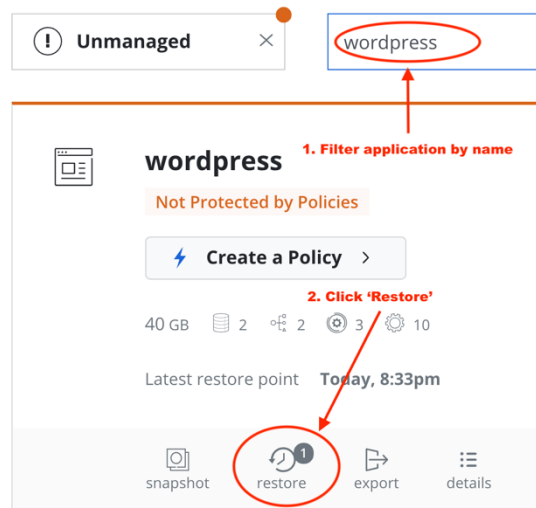


Figure 14 - Restore from Snapshot

#### 5.4.4 Step 4 - Select the Restore Point

We have performed a single manual backup so you will only be able to choose from a single Restore Point at this particular point in time.

### Restore application *wordpress*

Restore an application to a previous state. Restore points are shown and ordered based on scheduled execution time which may be different from the actual creation time. During a restore, the existing application is deleted and then recreated with the data artifacts restored from backups.

*Select a restore point for details.*

#### Past day



< Dashboard < Applications

### Restore application *wordpress*

Restore an application to a previous state. Restore points are shown and ordered based on scheduled execution time which may be different from the actual creation time. During a restore, the existing application is deleted and then recreated with the data artifacts restored from backups.

*Select a restore point for details.*

#### Past day

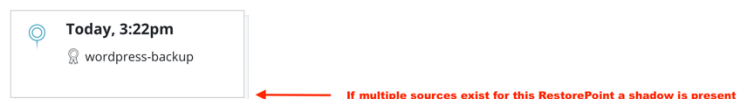


Figure 15 - Select Restore Point

**Note:** The type of the Restore Point is 'Manual Protect' indicating the Restore Point was manually triggered and not the result of a scheduled backup policy.

**Note:** If you are selecting a Restore Point that was exported, you will see a shadow under the actual tile to highlight the presence of multiple source (lower image). If so, an extra dialog will appear on the screen to select the appropriate data source.

## 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.

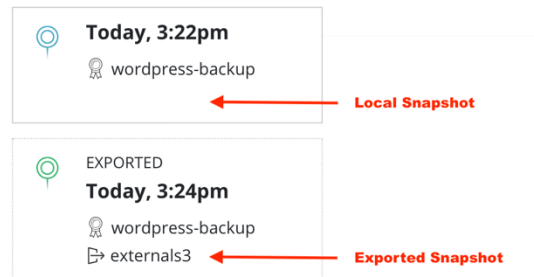


Figure 16 - Select Data Source

### 5.4.5 Step 5 - Select Namespace for Restore Operation

When performing a restore you have the ability to select either the original namespace or a new namespace. For this example, we will restore to the same namespace so we can verify the data has been restored and the application route is left untouched.

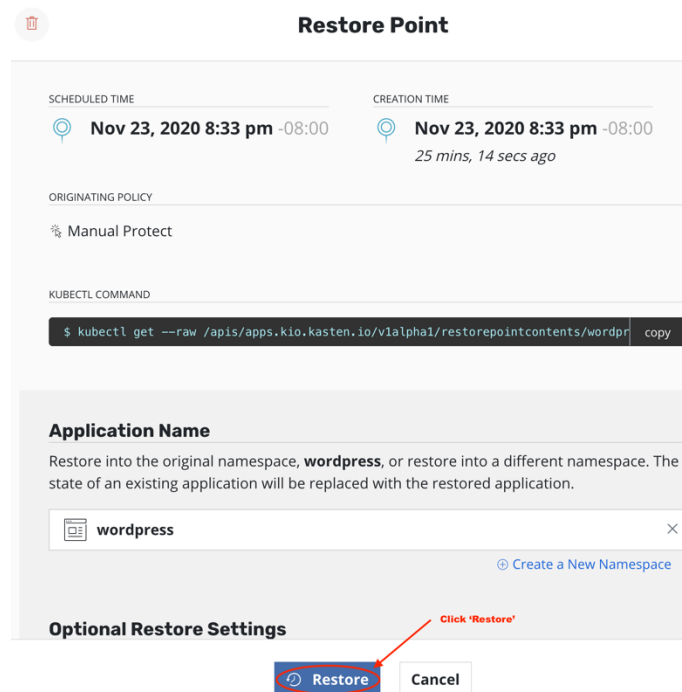


Figure 17 - Start Restore Process

### 5.4.6 Step 6 - Confirm Restore Initiation

In the dialog box confirm you want to initiate the restore process.

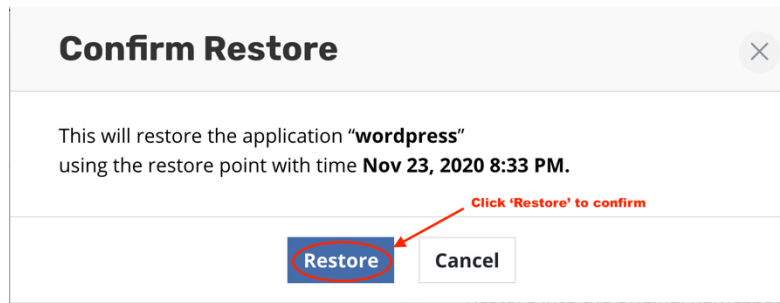


Figure 18 - Restore Confirmation

### 5.4.7 Step 7 - Monitor Restore Operation

On the main dashboard page, we can track the activity performed by K10. As illustrated below the restore operation is now completed.

#### Activity

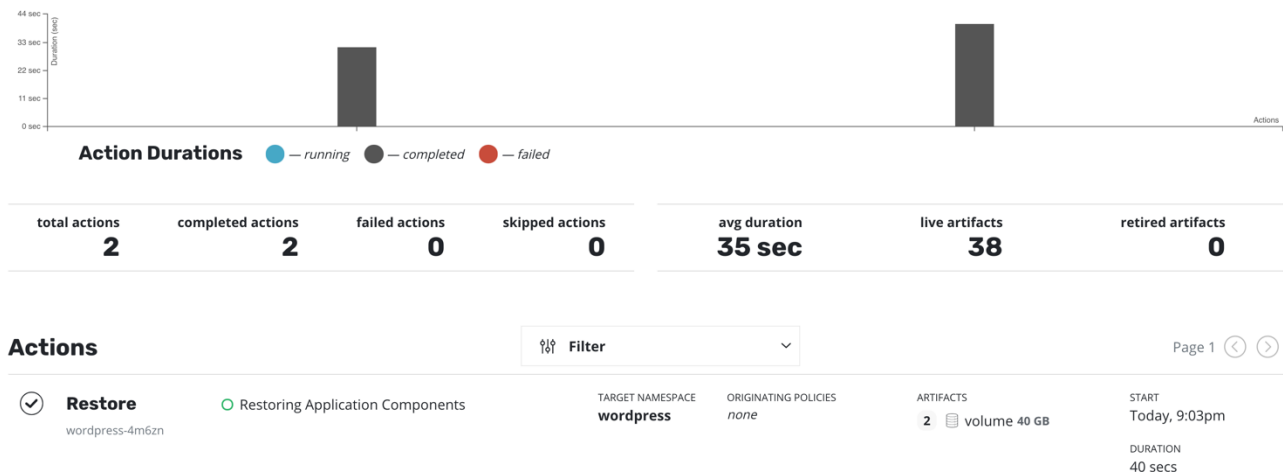


Figure 19 - Restore completed

### 5.4.8 Step 8 - Verify Data after Restore

To make sure the restore is successful, point your browser to your WordPress application to verify the list of posts available is back to its original 3 entry list.

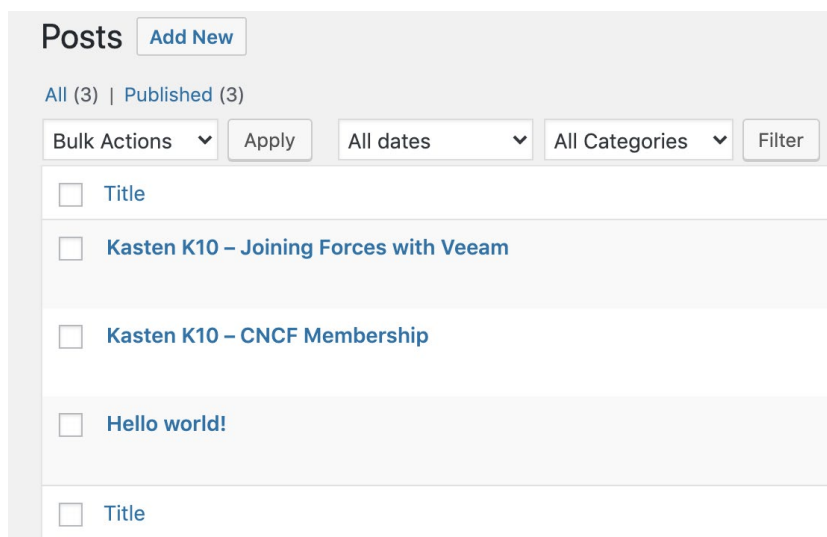


Figure 20 - Posts after Restore

As you can see the posts are identical to the posts illustrated in 5.4.1 before the deletion of a post.

## 5.4.9 Step 9 - Verify Application Component Restore

We can also go and check the age of the pods and PVCs in the project. As you can see all resources have been redeployed after the snapshot has been rolled back.

```

$ oc get pods
NAME                                READY STATUS RESTARTS AGE
wordpress-6c5d9c567c-zbmdn        1/1   Running 0      35s
wordpress-mysql-68f759bd8f-qjjft  1/1   Running 0      35s
$ oc get pvc
NAME          STATUS VOLUME                                     CAPACITY ACCESS MODES STORAGECLASS      AGE
mysql-pv-claim Bound  pvc-da2d847f-54ba-4534-b6f2-91600fd0a432  20Gi     RWO             ocs-storagecluster- 39s
ceph-rbd
wp-pv-claim  Bound  pvc-3aa44772-e650-4530-9a90-dcd1d87e2a6e  20Gi     RWO             ocs-storagecluster- 39s
ceph-rbd
    
```

## 5.5 Backup policies

In the previous chapters we used a manual backup and restore process. You can also create policies to automatically schedule backups and define the retention characteristics that will fit your application as well as any regulation that might be observed. Policies are extremely configurable for fine-grained.

### 5.5.1 Step 1 - Navigate Kasten Applications

In the Application tile click 'Unmanaged' as illustrated below.

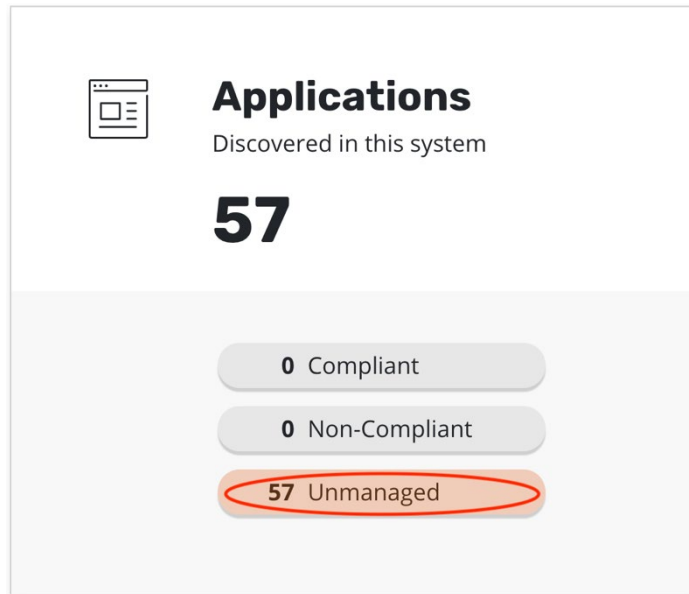


Figure 21 - Application Tile

### 5.5.2 Step 2 - Create Backup Policy

In the list of Unmanaged applications, use the filter to find the WordPress application and then click on 'Create a Policy' as illustrated below.

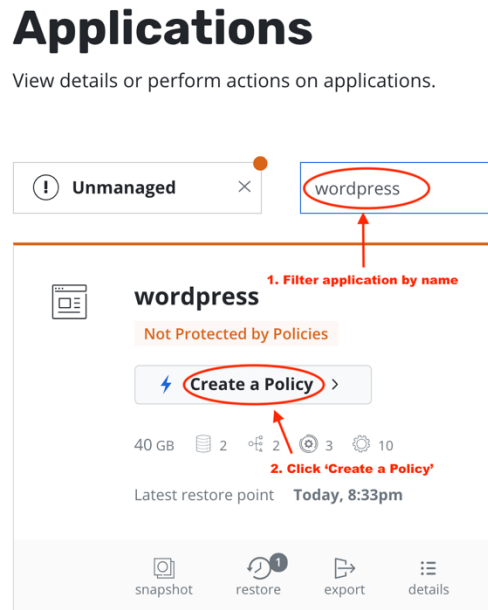


Figure 22 - Initiate backup policy creation

### 5.5.3 Step 3 - Specify Backup Policy Parameters

In the pane that opens on the right-hand side of your browser window choose the characteristics desired for your scheduled backup. When done, click the 'Create Policy' button.

Figure 23 - Customize Backup Policy Parameters

### 5.5.4 Step 4 - Backup Policy Summary

Once you have confirmed the creation of the backup policy, a summary of this policy will be displayed on screen as illustrated below.

Figure 24 - Backup Policy Summary

Until the backup policy is successfully executed the application for which it was created will show in the K10 dashboard as 'Non-Compliant' as illustrated below.

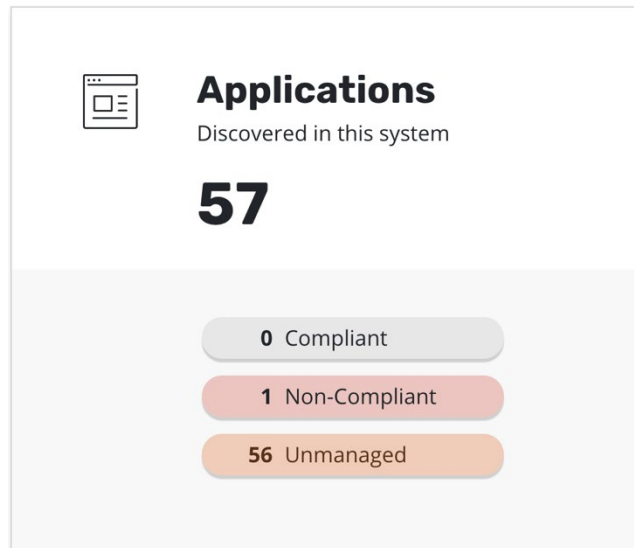


Figure 25 – Non-Compliant Applications

### 5.5.5 Step 5 - Successful Scheduled Backup

Upon successful scheduling and completion of the backup for the application, the compliance of the application with its backup policy will be reported and displayed in the application card. In the screenshot below, we can see that our WordPress application is now compliant with its policy as illustrated below.

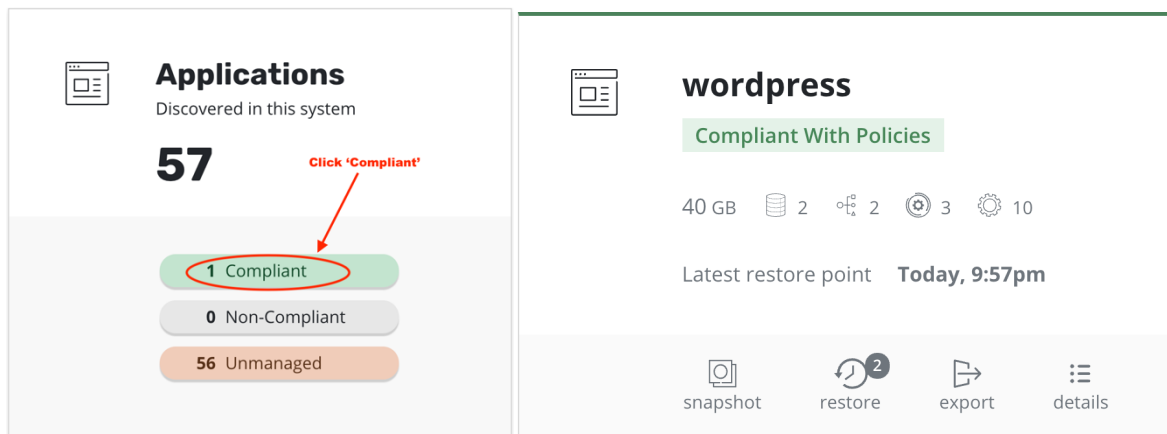


Figure 26 - Application Backup Policy Compliance



## 5.6 Disaster Recovery Capable Backup

### 5.6.1 Introduction

We can use an existing Restore Point to enable Disaster Recovery capabilities through the export of backup data and metadata using a S3 object store. Kasten K10 does support Red Hat OpenShift Container Storage, Red Hat Ceph® Storage, Google Cloud Storage, Amazon S3, Azure Storage and any S3 compatible endpoint.

The process involves the following steps:

- Backup the data via snapshot and the metadata
- Export the data and the metadata to an object store
- Create an import profile on the destination cluster
- Restore the data and the metadata in the destination cluster
- Start the application in the destination cluster

**Note:** The S3 endpoint must be available from the source cluster to perform the backup and the export functions as well as from the remote cluster at the time of the restore in case the source cluster has failed.

**Note:** We can leverage policies to automate this workflow at a user-defined schedule and retention.

### 5.6.2 Step 1 - Access K10 Settings

In the top right-hand corner of your browser window click on 'Settings' as illustrated below.

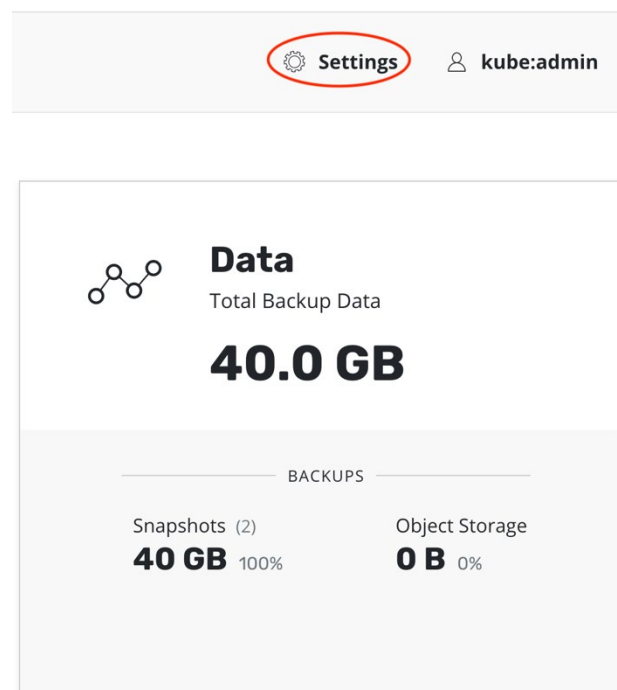


Figure 27 - Access K10 Settings

### 5.6.3 Step 2 - Create Object Store Profile

In the new browser page that displays, click on 'New Profile' as illustrated below.

## 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.

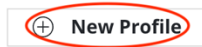


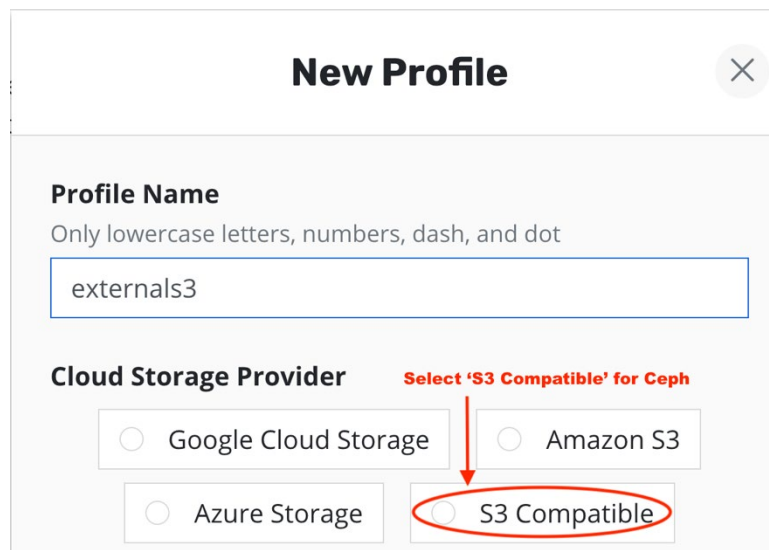
Figure 28 - Create New S3 Profile

### 5.6.4 Step 3 - Configure Object Store Profile

In the new browser page that displays, name your object store profile and enter the appropriate information to grant access to the object store. For our example, we will use a S3 compatible object store provided by an OpenShift Container Storage RADOS Gateway.

The parameters include:

- Access key
- Secret key
- Endpoint URL
- SSL override
- Bucket name



**New Profile** [X]

**Profile Name**  
Only lowercase letters, numbers, dash, and dot  
externals3

**Cloud Storage Provider** Select 'S3 Compatible' for Ceph

Google Cloud Storage  Amazon S3  
 Azure Storage  S3 Compatible

Figure 29 - Name Object Store Profile

**Note:** The access key, secret key and bucket must be created in the Red Hat Ceph Storage or OpenShift Container Storage cluster.

## 5.6.5 Step 4 - Configured Object Store Profile

If the set of parameters is correct and K10 is able to connect to the object store the profile will appear on the Location Profile page as illustrated below.

### 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.

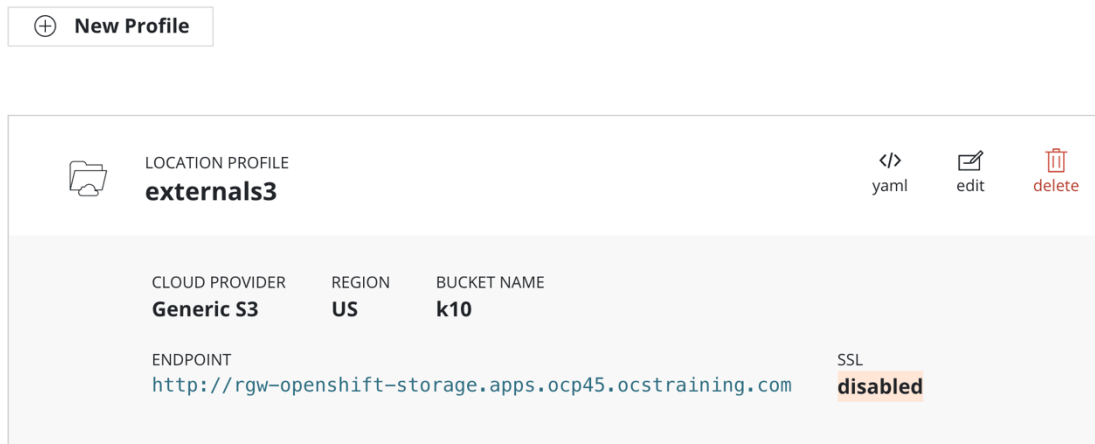


Figure 30 - Successful Object Store Creation

## 5.6.6 Step 5 - Edit Existing Policy

Go back to the Dashboard, click on the Policies card, click on the policy created earlier, and click the 'Edit' icon as illustrated below.

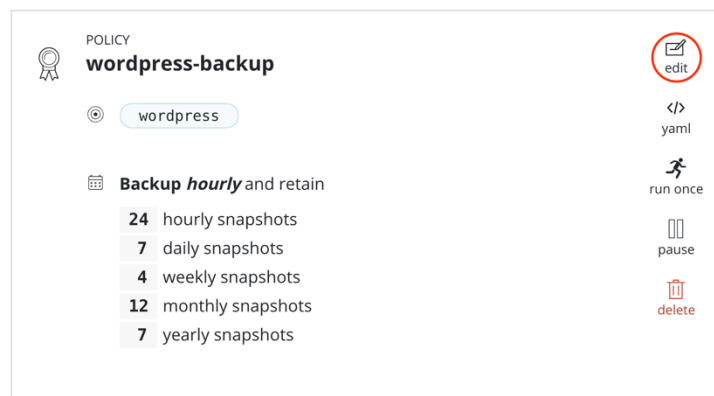


Figure 31 - Edit Backup Policy

### 5.6.7 Step 6 - Configure Existing Policy

Change the following parameters for the backup policy:

- Toggle the 'Enable Backups via Snapshot Exports'
- Choose which backup you want exported (e.g., Every snapshot, Daily snapshot)
- Choose the 'Export Location Profile' for your external object store
- Select the proper retention for your exports

When done click 'Edit Policy' at the bottom of the pane to confirm changes.

The screenshot shows the 'Edit Policy' dialog box with the following settings:

- Enable Backups via Snapshot Exports:** Toggled on. Description: "After snapshot completes, export restore points to enable backups or cross-cluster migration." The dropdown menu is set to "Every snapshot".
- Export Location Profile:** Description: "The cloud location that restore points will be exported to". The dropdown menu is set to "externals3".
- Retention of Exported Snapshots:** Description: "Manage how many exported snapshots to retain" with a link "Set to Zeros". The dropdown menu is set to "Use the same retention schedule as above".
- Snapshot Durability / Portability:** Two radio buttons are present: "Export Snapshot Data" (selected) and "Export Snapshot References Only".
- A checked checkbox below the radio buttons reads: "Exports complete snapshot data for durable and portable backups. Data is compressed, encrypted, and deduplicated, however this will use additional compute resources."
- At the bottom, there are three buttons: "Edit Policy", "</> YAML", and "Cancel".

Figure 32 - Export Specific Parameters

### 5.6.8 Step 7 - Get Import Details

In the policy page click on `Show import details` as illustrated below.

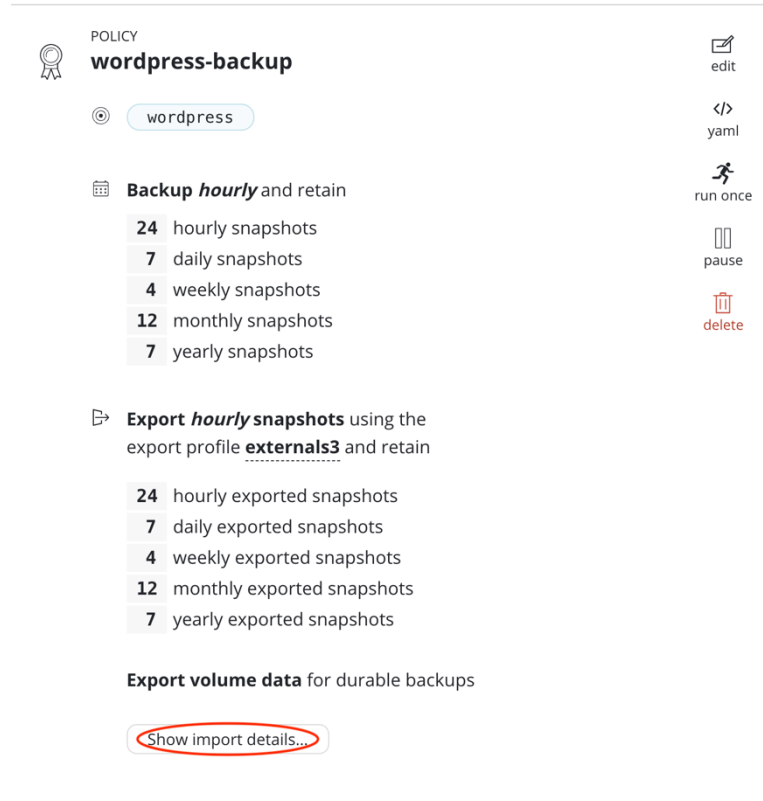


Figure 33 - Import Details

### 5.6.9 Step 8 - Copy Import Details

Copy the character string. It is used as a handshake on the destination cluster to import from the appropriate backup.

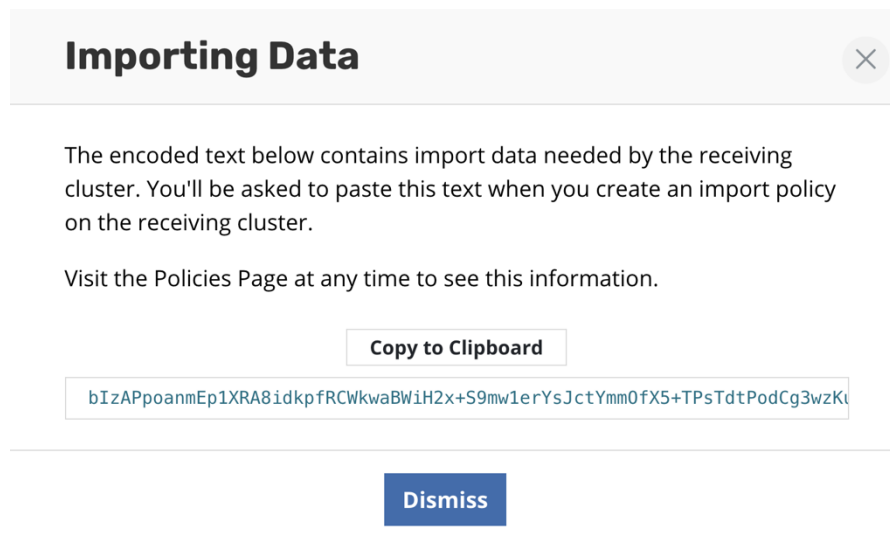


Figure 34 - Copy Import Details

### 5.6.10 Step 9 - Verify Successful Export

Go back to the Dashboard and you should see an Export job with a successful upload to the S3 endpoint.

#### Activity

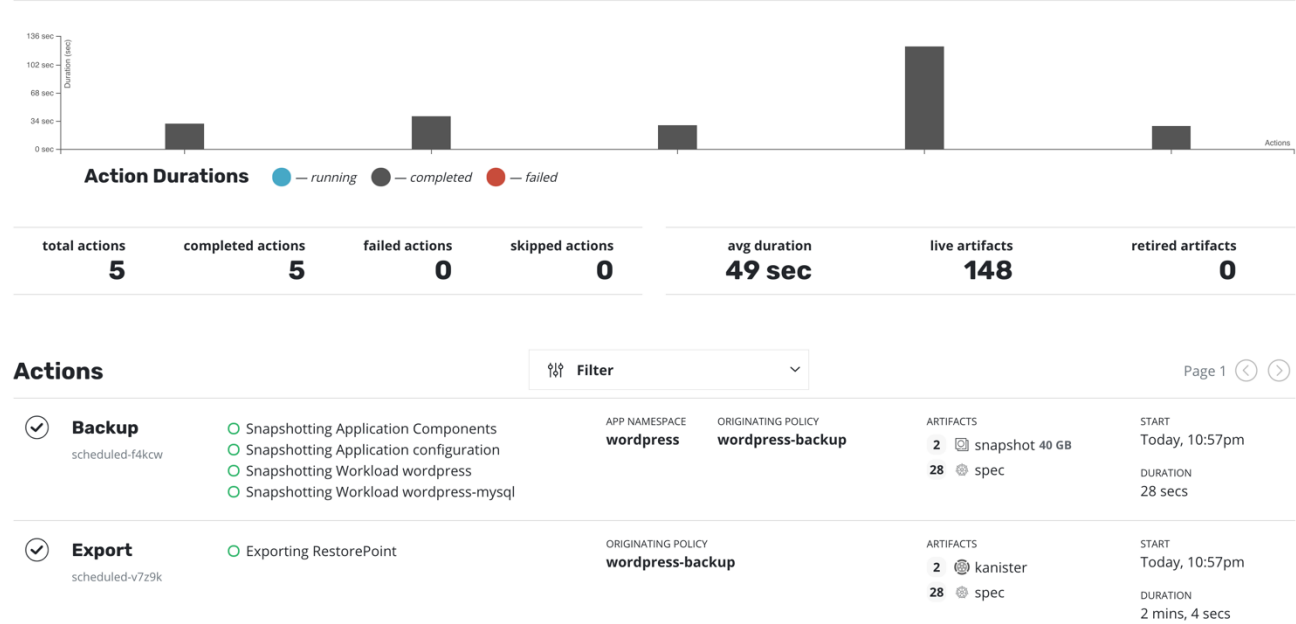


Figure 35 - Export Successful

## 5.7 Disaster Recovery Restore

### 5.7.1 Introduction

This section will cover how an application can be recovered using the data and metadata store in a Kasten K10 compatible external object store.

The target cluster must have access to the external object store configured on the source cluster.

### 5.7.2 Step 1 - Create Location Profile

In the destination cluster, follow steps described in 5.6.2 through 5.6.5 to configure the external object store profile.

### 5.7.3 Step 2 - Create Import Policy

In the destination cluster, locate the 'Policies' tile from the main K10 dashboard and select 'new policy' as illustrated below.

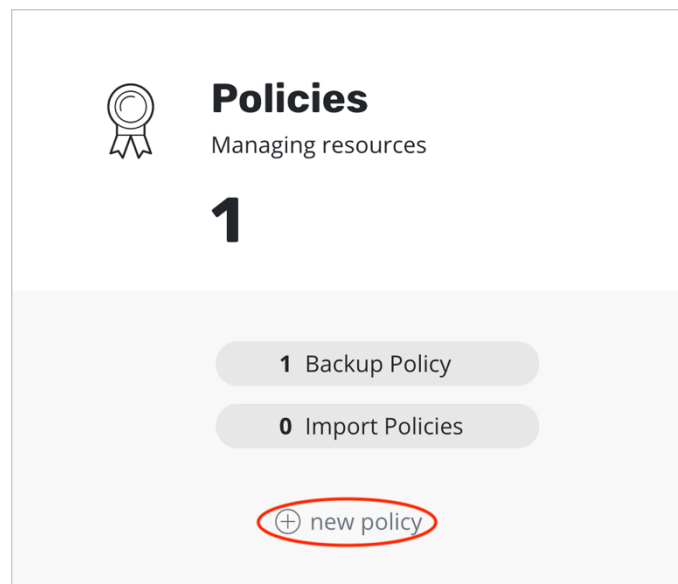


Figure 36 - Create Import Policy

## 5.7.4 Step 3 - Configure Import Policy

In the policy dialog, provide the following information as illustrated below:

- Policy name
- Select 'Import'
- Check the 'Restore After Import' box
- Select the frequency for the restore
- Paste the import character string from 5.6.9
- Select the external object store you want to pull the data from
- Click 'Create Policy' to validate

The figure shows two side-by-side screenshots of the 'New Policy' configuration dialog. The left screenshot displays the 'Name' field with the value 'restorefromobjectstore', the 'Action' set to 'Import', and the 'Restore After Import' checkbox checked. The right screenshot displays the 'Action Frequency' set to 'Hourly', the 'Config Data for Import' field containing a long alphanumeric string, and the 'Profile for Import' dropdown set to 'externals3'. Both screenshots have red circles highlighting the specific fields mentioned in the instructions.

Figure 37 - Import Policy Parameters



## 5.7.5 Step 4 - Wait for Import and Restore Completion

Once you go back to the dashboard, you will see an Import job followed by a Restore job to simulate a successful disaster recovery. Simply monitor the completion of the import until the import and the restore complete successfully, as illustrated below.

### Activity

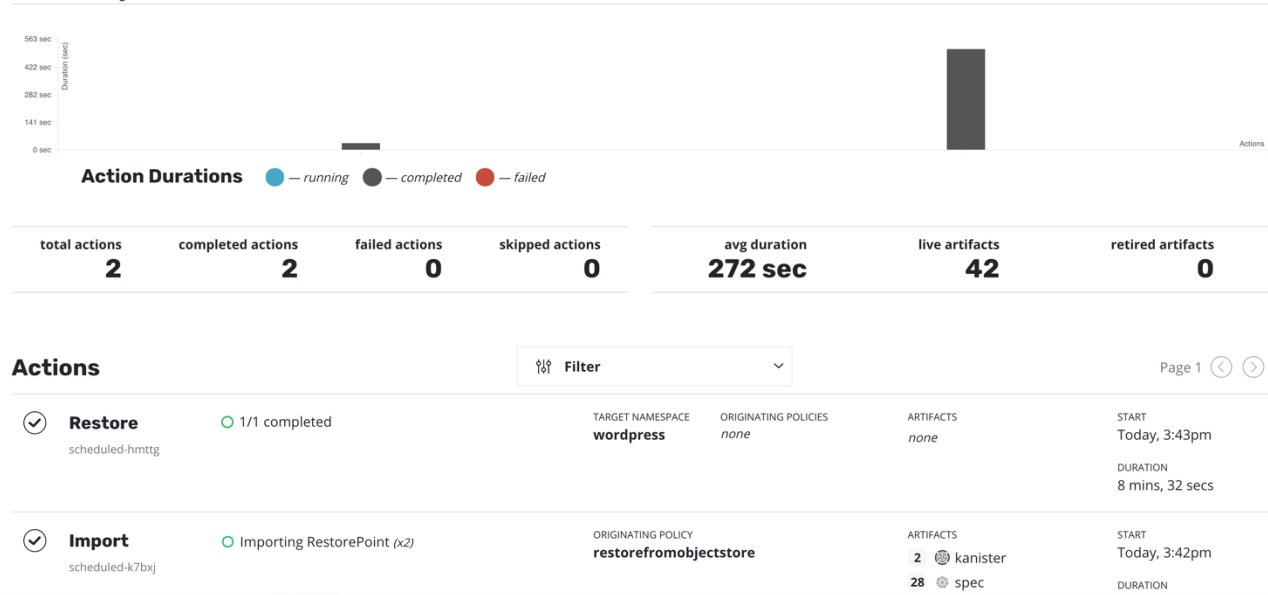


Figure 38 - Import and Restore Complete

In this test scenario, we restored to a live cluster with a different cluster name and base domain. This required delete the existing route that was restored with the same base domain and to expose the WordPress service again using the target cluster base domain to visualize the application user interface as illustrated below.

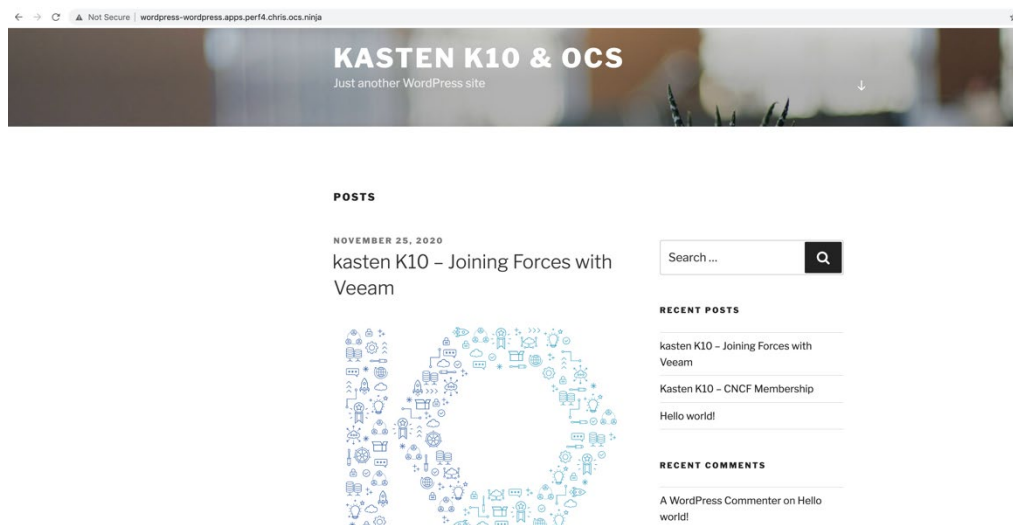


Figure 39 - Restore WordPress Application UI

**Note:** Pay attention to the browser URL we used to reach out to the test DR cluster (original base domain was ocp45.ocstraining.com while the disaster recovery base domain is perf4.chris.ocs.ninja).

## 6 API and CLI

Kasten K10 is fully compatible with the Kubernetes, OpenShift Container Platform and OpenShift Container Storage CSI APIs. This allows for a full automation of tasks via configuration file in YAML format.

### 6.1 List Backup Policies

```
$ oc get policies.config.kio.kasten.io -n kasten-io
NAME          STATUS
wordpress-backup Success
```

### 6.2 Create Backup Policy

```
$ cat k10-wordpress-cli-policy.yaml
apiVersion: config.kio.kasten.io/v1alpha1
kind: Policy
metadata:
  name: wordpress-cli-backup
  namespace: kasten-io
spec:
  comment: "This is a backup policy created via a yaml resource file"
  frequency: "@daily"
  actions:
    - action: backup
      backupParameters:
        profile:
          namespace: kasten-io
          name: externals3
    - action: export
      exportParameters:
        frequency: "@daily"
        profile:
          name: externals3
          namespace: kasten-io
      exportData:
        enabled: true
  retention:
    daily: 0
    weekly: 0
    monthly: 0
    yearly: 0
  retention:
    daily: 7
    weekly: 4
    monthly: 12
    yearly: 7
  selector:
    matchExpressions:
      - key: k10.kasten.io/appNamespace
        operator: In
        values:
```

```
- wordpress
$ cat k10-wordpress-cli-policy.yaml | oc create -f -
policy.config.kio.kasten.io/wordpress-cli-backup created
$ oc get policies.config.kio.kasten.io -n kasten-io
NAME          STATUS
wordpress-backup  Success
wordpress-cli-backup  Success
```

### 6.3 List Backup Actions

```
$ oc get backupactions.actions.kio.kasten.io -n wordpress
NAME          CREATED AT
scheduled-qbr24  2020-11-20T23:57:04Z
scheduled-pdph4  2020-11-20T21:57:02Z
manualbackup-7jvlt  2020-11-20T19:57:48Z
scheduled-7pncf  2020-11-20T22:57:03Z
```

### 6.4 Trigger Backup Policy

```
$ cat k10-wordpress-cli-policy-trigger.yaml
apiVersion: actions.kio.kasten.io/v1alpha1
kind: BackupAction
metadata:
  name: backup-wordpress
  namespace: wordpress
labels:
  # These labels are required for on-demand actions so that
  # actions can be filtered.
  # Label presence is validated.
  k10.kasten.io/appName: "wordpress"
  k10.kasten.io/appNamespace: "wordpress"
  k10.kasten.io/policyName: "wordpress-cli-backup"
  k10.kasten.io/policyNamespace: "kasten-io"
spec:
  subject:
    # Reference to the K10App CR for the application
    name: wordpress
    namespace: wordpress
$ cat k10-wordpress-cli-policy-trigger.yaml | oc create -f -
backupaction.actions.kio.kasten.io/backup-wordpress created
$ oc get backupactions.actions.kio.kasten.io -n wordpress
NAME          CREATED AT
scheduled-qbr24  2020-11-20T23:57:04Z
scheduled-pdph4  2020-11-20T21:57:02Z
scheduled-4w4dq  2020-11-21T00:57:04Z
manualbackup-7jvlt  2020-11-20T19:57:48Z
scheduled-7pncf  2020-11-20T22:57:03Z
backup-wordpress  2020-11-21T01:00:18Z <-- Triggered by previous command
```

For additional resource file configuration please visit the official [Kasten K10 documentation](#).

## Appendix

### 6.5 Support

Any Red Hat OpenShift version that provides Kubernetes v1.15 or higher is supported by the current K10 release. Please refer to [this docs](#) page for information.

#### 6.5.1 Kasten Team

If you have questions, need support, or would like an invite to our support Slack channel, please email our support team ([support@kasten.io](mailto:support@kasten.io)) or visit our [contact page](#).

#### 6.5.2 Red Hat Team

If you have questions, need support feel free to visit our [contact page](#) and select your favorite method of contact as well as the type of contact you would need (talk to a Red Hatter, phone contact or support).

## Illustrations

Figure 1 - K10 platform on OpenShift Container Platform	4
Figure 2 - K10 Use cases on OpenShift Container Platform	5
Figure 3 - High level overview of K10 architecture	7
Figure 4 - Initial K10 dashboard login	12
Figure 5 - K10 dashboard landing page	13
Figure 6 - WordPress Example	18
Figure 7 - Application Tile	19
Figure 8 - Create Manual Restore Point	19
Figure 9 - Confirm Manual Snapshot	20
Figure 10 - Activity Dashboard	20
Figure 11 - Existing WordPress Posts	21
Figure 12 - After WordPress Post Deletion	21
Figure 13 - Navigate to Applications	22
Figure 14 - Restore from Snapshot	22
Figure 15 - Select Restore Point	23
Figure 16 - Select Data Source	23
Figure 17 - Start Restore Process	24
Figure 18 - Restore Confirmation	24
Figure 19 - Restore completed	25
Figure 20 - Posts after Restore	25
Figure 21 - Application Tile	26
Figure 22 - Initiate backup policy creation	27
Figure 23 - Customize Backup Policy Parameters	27
Figure 24 - Backup Policy Summary	28
Figure 25 - Non-Compliant Applications	28
Figure 26 - Application Backup Policy Compliance	29
Figure 27 - Access K10 Settings	30
Figure 28 - Create New S3 Profile	31

Figure 29 - Name Object Store Profile	31
Figure 30 - Successful Object Store Creation	32
Figure 31 - Edit Backup Policy	32
Figure 32 - Export Specific Parameters	33
Figure 33 - Import Details	34
Figure 34 - Copy Import Details	34
Figure 35 - Export Successful	35
Figure 36 - Create Import Policy	36
Figure 37 - Import Policy Parameters	37
Figure 38 - Import and Restore Complete	38
Figure 39 - Restore WordPress Application UI	38