VMware Advanced Customer Engagements (ACE) Team

Quick Start Guide for Tanzu Kubernetes Grid Integrated (TKGI) Cluster Backup and Restore

July 2020

vmware

Table of Contents

Introduction

This document is a quick start guide for backing up a Tanzu Kubernetes Grid Integrated (TKGI, formerly known as Enterprise PKS) Kubernetes cluster and restoring it. This document will provide details on Valero backup software, installing Velero, backing up an existing cluster, and restoring to the same or another target cluster. The cluster backup will include all Kubernetes (K8) resources as well as persistent volumes.

Recoverability in Kubernetes

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services at scale and provides high availability that is declarative configuration and automation.

While Kuberentes provides high availability and zero downtime deployments, it does not provide data protection or migration solutions. We need to have a backup and recovery strategy that accounts for:

- Human errors.
- Bugs or vulnerable systems
- Natural disasters
- Legal Requirements for data retention
- Ability to migrate workloads from one cluster to another
    - To deal with changes with providers
    - Cost advantages
    - Compliance with standards.
- Ability to Archive data

We need to have a backup and recovery strategy that will help with the recoverability of Kubernetes. In this guide we will use Velero as the tool but there are several tools available that can be used to accomplish the same objective,

Velero

Running on Kubernetes clusters or on VMs, Velero gives you tools to back up and restore your Kubernetes cluster resources and persistent volumes. You can run Velero in Kubernetes clusters in a public cloud or on-premises. Velero lets you:

- Take backups of your cluster and restore in case of loss.
- Migrate cluster resources to other clusters.
- Replicate your production cluster to development and test clusters.

Velero consists of:

- A server that runs on your cluster
- A command-line client that runs locally

Disaster Recovery

If you periodically back up your cluster's resources, you can return to a previous state in case of some unexpected mishap, such as a service outage.

Cluster Migration

Velero can help you port your resources from one cluster to another, if you point each Velero instance to the same cloud object storage location.

Backup Reference

It is possible to exclude individual items from being backed up, even if they match the resource/namespace/label selectors defined in the backup spec.

Restore Reference

Velero can restore resources into a different namespace than the one they were backed up from.

How it Works

On-demand backups

- Uploads a tar ball of copied Kubernetes objects into cloud object storage.
- Calls the cloud provider API to make disk snapshots of persistent volumes, if specified.

Scheduled backups

- The **schedule** operation allows you to back up your data at recurring intervals

Restores

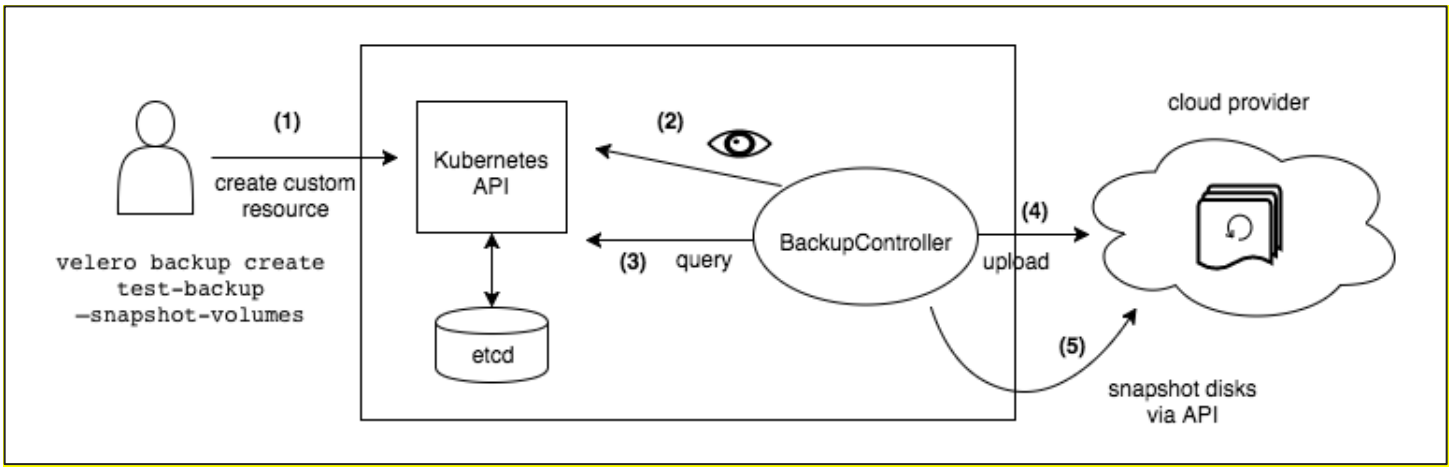
The **restore** operation allows you to restore all the objects and persistent volumes from a previously created backup. You can also restore only a filtered subset of objects and persistent volumes.

Backup workflow
When you run Velero backup,

- The Velero client makes a call to the Kubernetes API server to create a Backup object.
- The BackupController notices the new Backup object and performs validation.
- The BackupController begins the backup process. It collects the data to back up by querying the API server for resources.

The BackupController makes a call to the object storage service -- for example, AWS (Amazon Web Services) S3 -- to upload the backup.

By default, Velero backup create makes disk snapshots of any persistent volumes. You can adjust the snapshots by specifying additional flags. Run Velero backup create --help to see available flags. Snapshots can be disabled with the option --snapshot-volumes=false.



You can run Velero in Kubernetes clusters deployed in a public cloud provider or on-premises. For detailed information, see Compatible Storage Providers. Each Velero operation -- on-demand backup, scheduled backup, restore -- is a custom resource, defined with a Kubernetes Custom Resource Definition (CRD) and stored in etcd. Velero also includes controllers that process the custom resources to perform backup, restore, and all related operations. You can back up or restore all objects in your cluster, or you can filter objects by type, namespace, and/or label.

Restic inherently is a file-based backup. Currently, on a vSphere environment Velero uses Restic to backup Kubernetes Persistent Volumes (PV's) by taking the backup of all the files.

For more information go to *https://velero.io/docs/master/restic/*

Velero vSphere plugin

The Velero vSphere plugin enables Velero to take a crash-consistent VMware vSphere Storage APIs snapshot backup of a block Persistent Volume on vSphere storage, and store the backup on S3 compatible storage.

Use-case

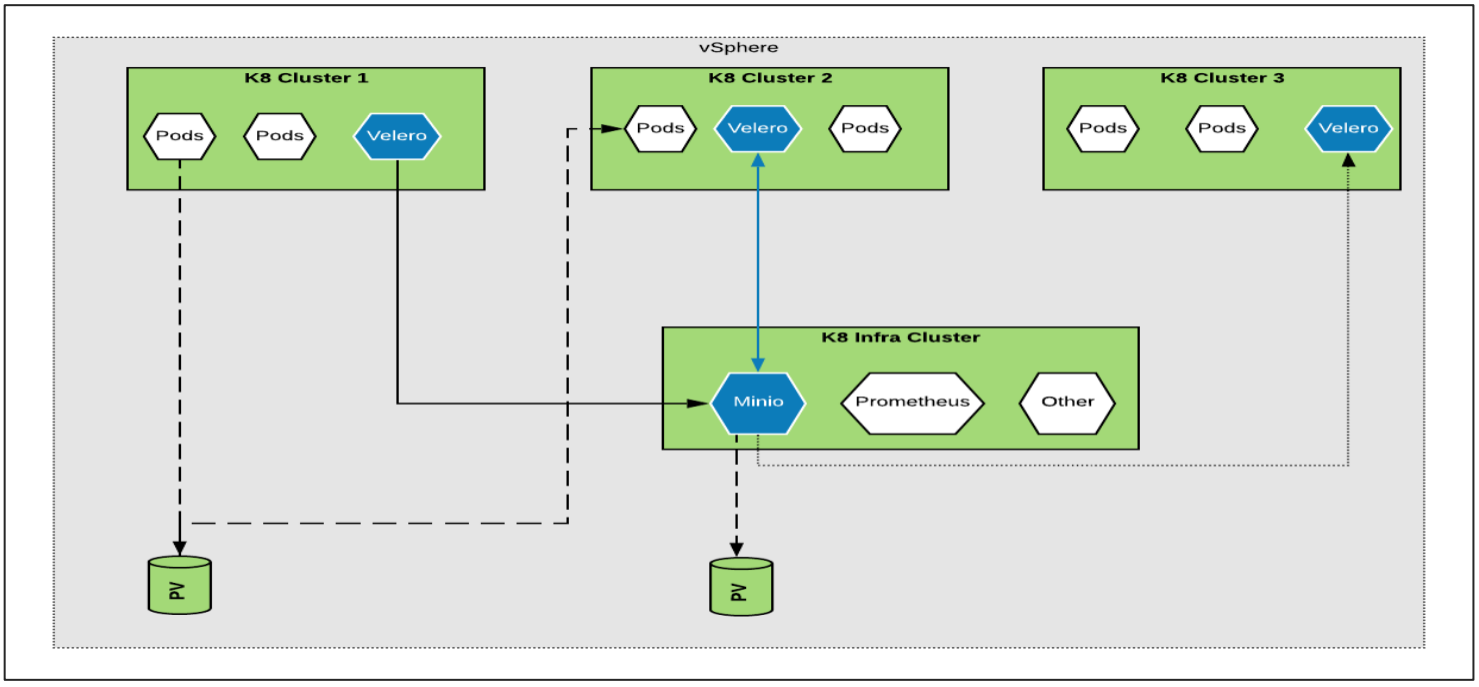
Stateless Applications : these applications do not store data or application state to the cluster or to persistent storage. Instead, data and application state stay with the client, which makes stateless applications more scalable. Those types of applications constitute about 70% of workloads deployed to Kubernetes today, E.g. Front-end applications.

Stateful Applications: these applications, for example databases, save data between sessions and require persistent storage to store the data. The retained data is called the application's state. You can later retrieve the data and use it in the next session. Kubernetes offers persistent volumes as objects capable of retaining their state and data. In the vSphere environment, the persistent volume objects are backed by virtual disks that reside on a datastore.

This section defines the common use cases where Velero would be applicable.

Cluster Migration**:** Migrate both stateful and stateless applications from one cluster to another

Disaster Recovery**:** Restore applications (both stateful and stateless) to a cluster from a backup in time (scheduled backup)



Assumptions

The following assumptions are made in the guide:

- TKGI is deployed
- The infrastructure team has setup 3 K8s clusters, the source cluster (where Velero backup is made from) , the target (where the Velero backup is restored to) cluster and an infra cluster where all infrastructure applications like Minio and Prometheus will be running .

  NOTE: This is not a fixed rule, Minio can run on any cluster, including the source and the target cluster, it could also be run on a standalone vm. For more information on Minio visit *https://docs.min.io/*

- The Minio backup endpoint is accessible from both the source and target clusters.
- A Linux/ubuntu machine is provisioned to install and use various software components
- The provisioned Linux/ubuntu machine meets the following
  - Can access all the 3 K8s clusters defined above
  - Has the appropriate kubectl cli installed
  - Has the appropriate pks cli installed
  - Has the latest version of *Helm* installed?
- In this document, we will be using ci-cluster as our source cluster and my-cluster as our target cluster.

Minio

Setup Minio

We will be setting up an open source version of Minio in the infrastructure cluster. We will be using the VMware Bitnami official opensource Minio Helm chart for K8s. The steps below describe how to setup Minio in a K8 cluster using the Bitnami distribution.

Step 1: ssh to the provisioned ubuntu vm (clivm)

Step 2: Get kube config for the infra cluster

```
pks login -a <pks api> -u <user> -p <password> -k
pks get-credentials <cluster>
```

Alternatively

```
pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k
```

E.g.

```
pks login -a pks.corp.local -u riaz -p VMware1! -k
pks get-credentials infra-cluster
```

Or

```
pks get-kubeconfig infra-cluster -a pks.corp.local -u riaz -p VMware1! -k
```

Step 3: Create as namespace to which Minio can be deployed

```
kubectl create ns minio
```

Step 4: Add Bitnami Helm repository

helm repo add bitnami *https://charts.bitnami.com/bitnami*

helm repo update

Step 5: Minio requires a backing store / K8s persistent volume. Create a storage-class on the infra cluster with the following storage class definition. Copy the contents of the file below to a file storage-class.yaml and create the storage class.

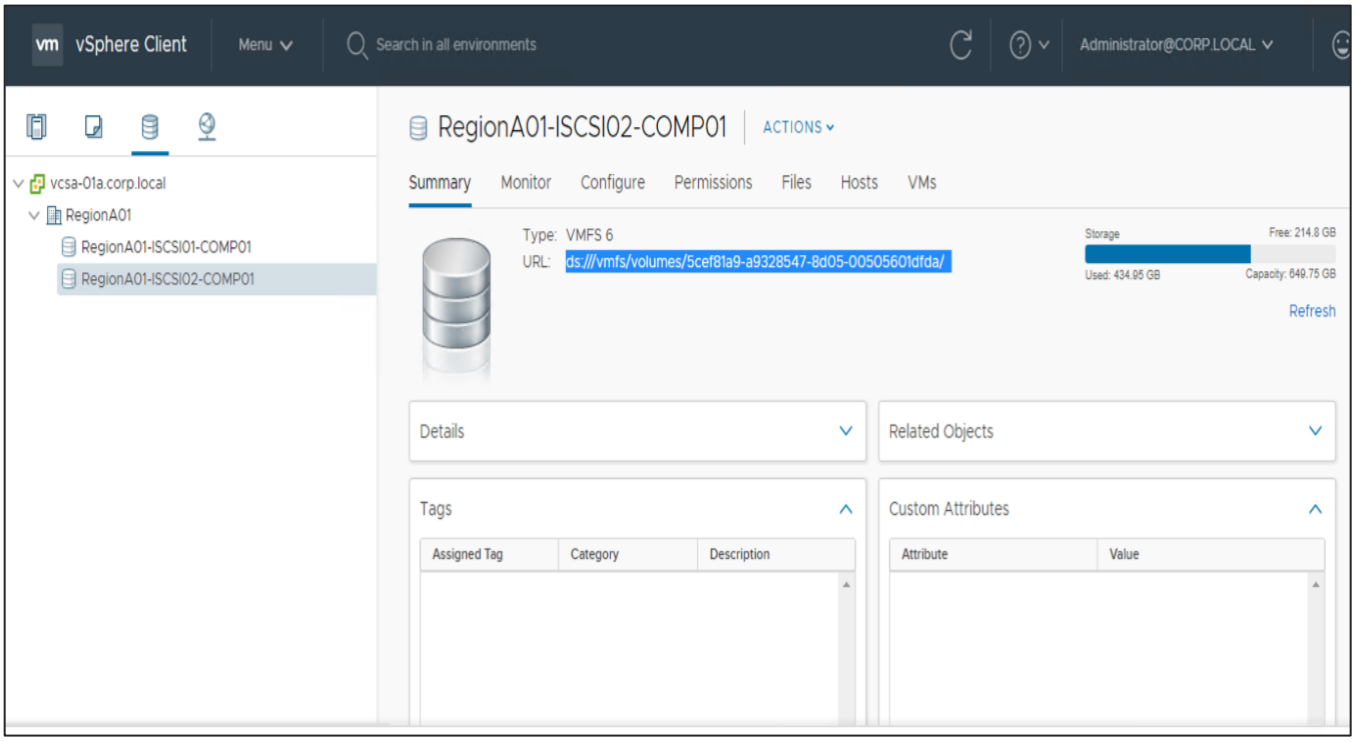
```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: minio-disk
provisioner: kubernetes.io/vsphere-volume
parameters:
    diskformat: thin
```

NOTE: If setting up the storage class using the CSI driver , follow the steps provided in the VMware Tanzu documentation to set up the CSI driver on the cluster you before creating the storage class. https://docs.pivotal.io/pks/1-7/vsphere-cns.html

Storage class definition when using a CSI driver

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: minio-disk
  annotations:
     storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/5cef81a9-a9328547-8d05-00505601dfda/"
```

Datastore url can be obtained from vCenter



kubectl apply -f storage-class.yaml

Step 6: Deploy the Bitnami Minio release. This will create the necessary resources to run Minio within the minio namespace

helm install minio-release -n minio --set accessKey.password=minio --set secretKey.password=minio123 --set persistence.storageClass=minio-disk --set persistence.size=128Gi bitnami/minio

NOTE: When sizing the storage for Minio, make sure there is about 3X times the storage the reason for which is minio stages the backup before commit. E.g. if a backup takes about 10Gi we would require about 30Gi ,. The extra storage is for the restore , velero uploads restore data to mino which also requires space.

Step 7: Check for all pods, deployments and services and make sure everything is created and the pods are running as expected. Also check if the PVC is created and bound

kubectl get all -n minio

kubectl get pvc -n minio

kubectl get deployment -n minio

```
ubuntu@cli-vm:~/velero$ kubectl get all -n minio
NAME                                    READY   STATUS    RESTARTS   AGE
pod/minio-release-d8b746dd8-lbsrw       1/1     Running   0          5m26s

NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/minio-release   ClusterIP   10.100.200.107   <none>        9000/TCP   5m27s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/minio-release      1/1     1            1           5m27s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/minio-release-d8b746dd8       1         1         1       5m27s
```

```
ubuntu@cli-vm:~/velero$ kubectl get pvc -n minio
NAME            STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
minio-release   Bound    pvc-66b6da36-e06e-4b0b-96c5-67efc3f2a1f8   8Gi        RWO            minio-disk     5m59s
```

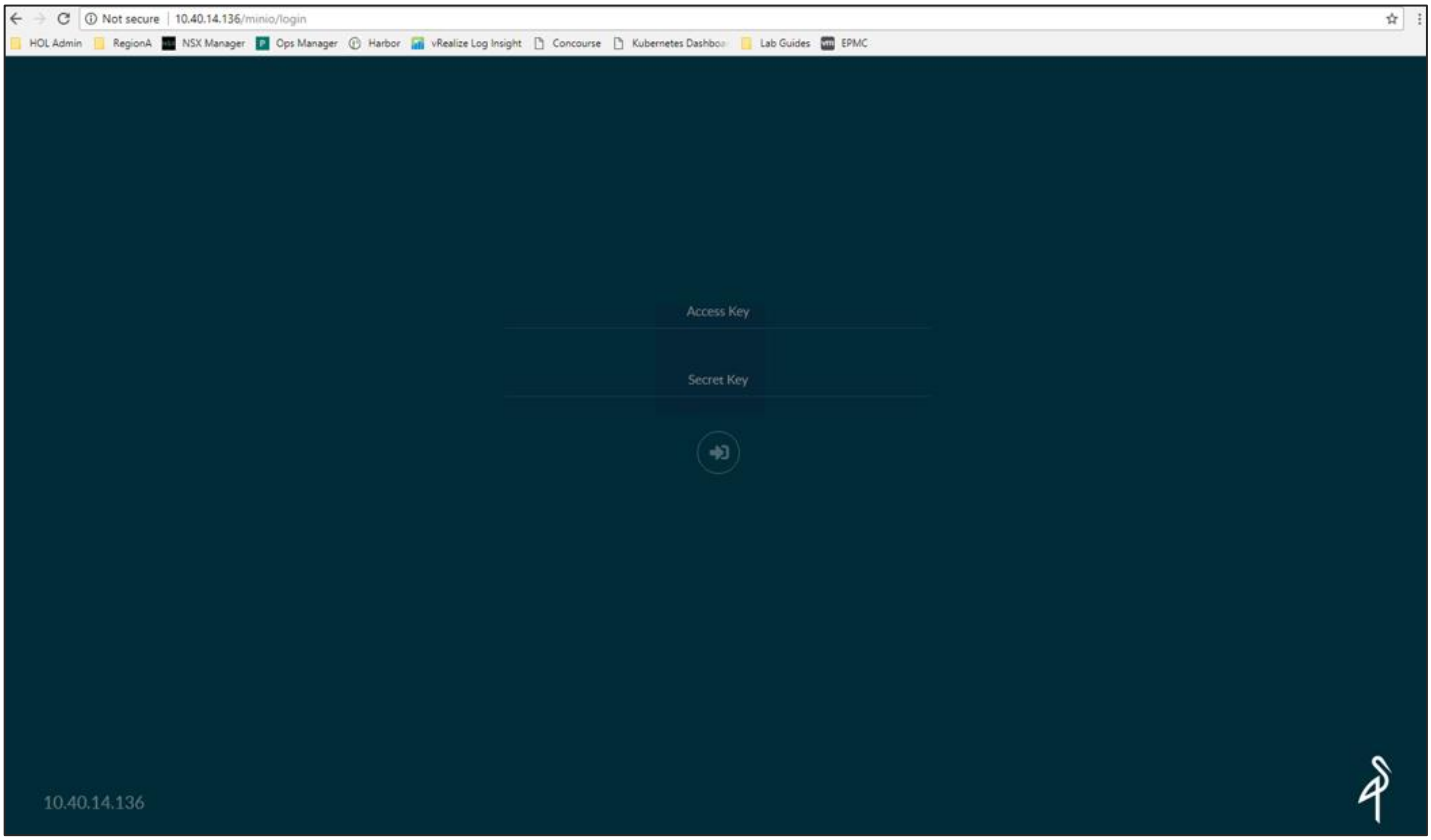
```
ubuntu@cli-vm:~/velero$ kubectl get deployment -n minio
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
minio-release   1/1     1            1           9m3s
```

Step 8: Expose the deployment as a Load Balancer. This will create a lb within NSX-T as well as an ingress.

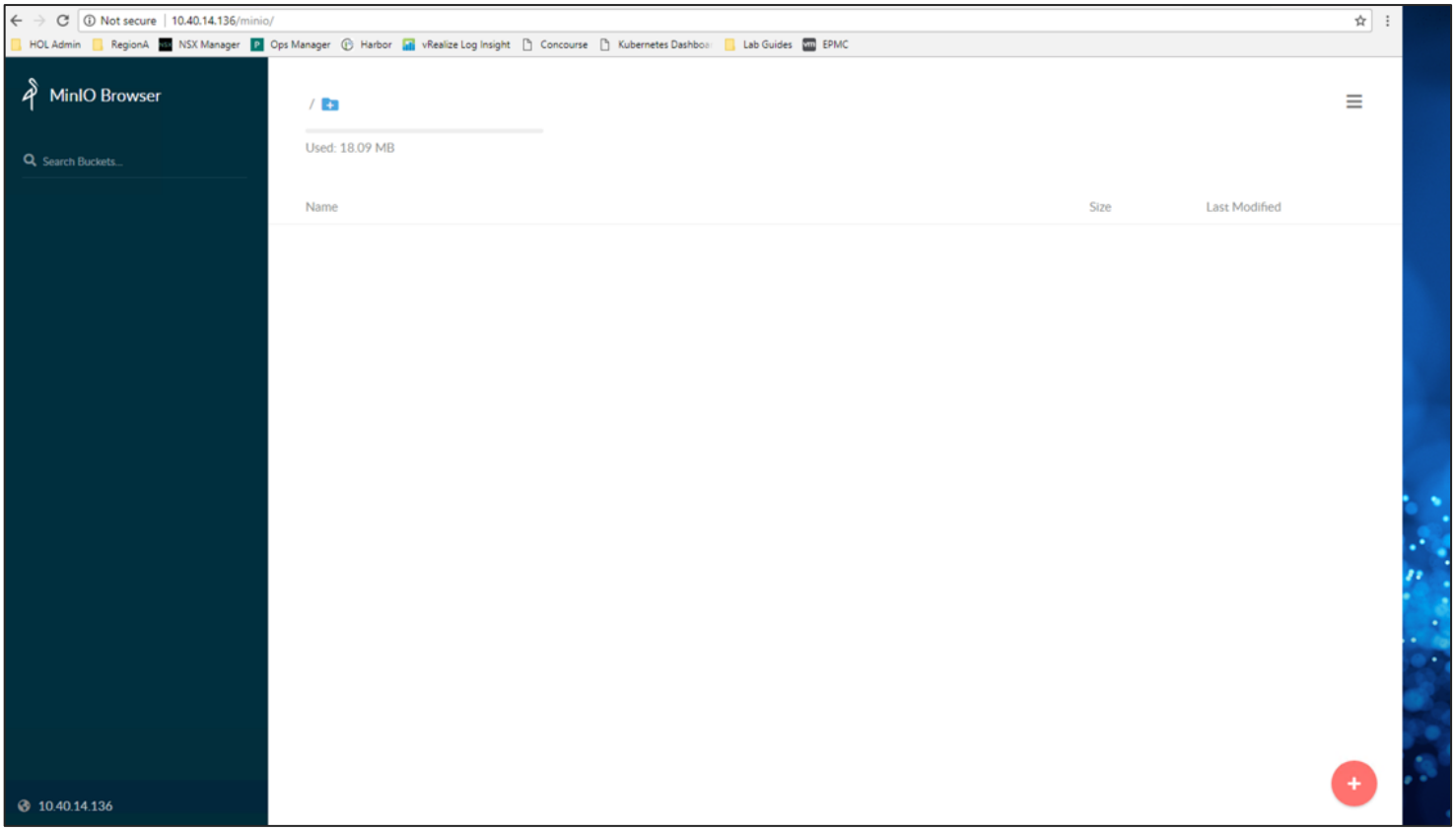
```
kubectl expose deployment minio-release --name=minio-frontend-lb --port=80 --target-port=9000 --type=LoadBalancer --namespace=minio
```

Step 9: Check the IP under the “External-IP” section, point your browser to that IP address:port . The Minio application should be accessible

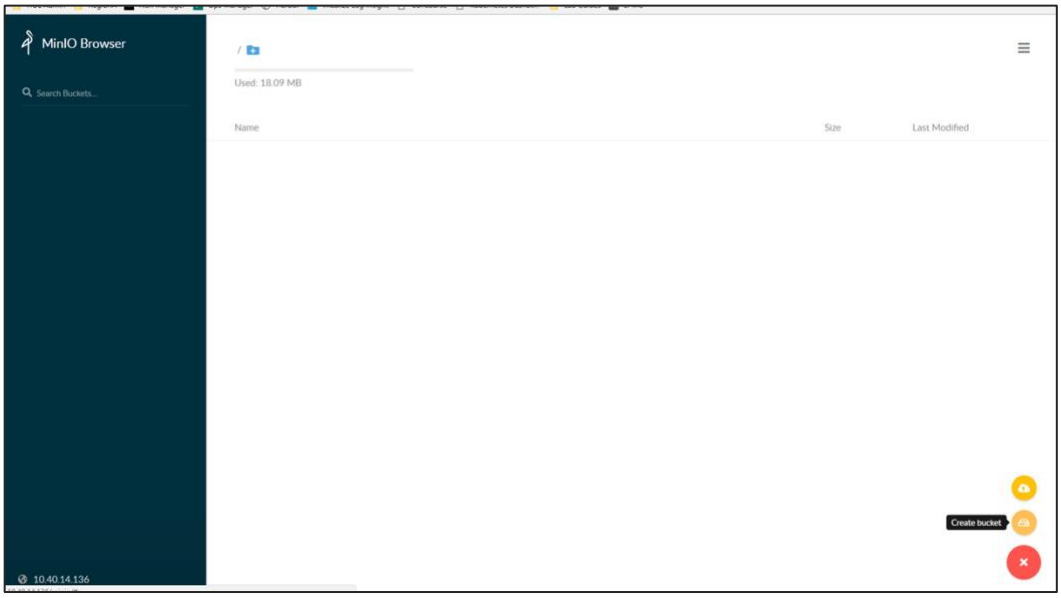
```
ubuntu@cli-vm:~/velero$ kubectl get svc -n minio
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
minio-frontend-lb   LoadBalancer   10.100.200.3     10.40.14.136   80:30568/TCP     18s
minio-release       ClusterIP      10.100.200.107   <none>         9000/TCP         13m
```

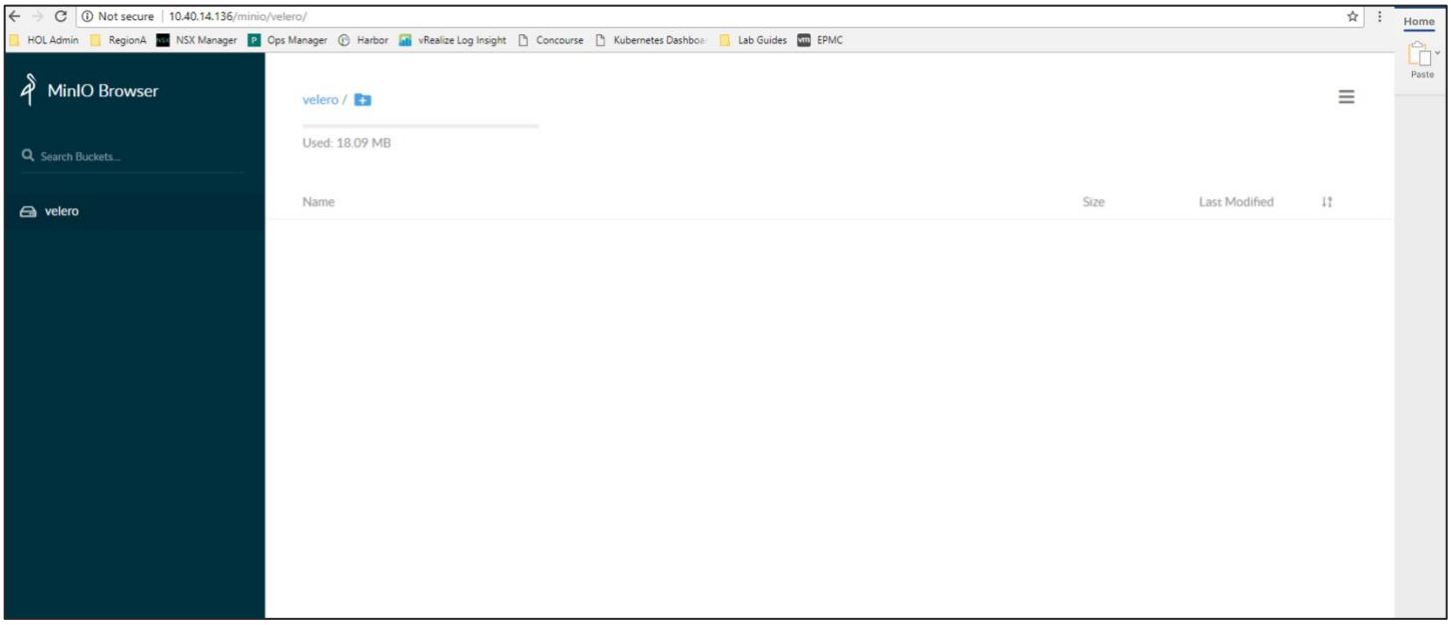


Step 10: Login with the credentials used in step 6. E.g. minio/minio123



Step 11: Create a bucket called Velero. We will be using this bucket when we install Velero to the clusters in the following steps:





Minio Cleanup

To clean up Minio, uninstall the deployed helm release

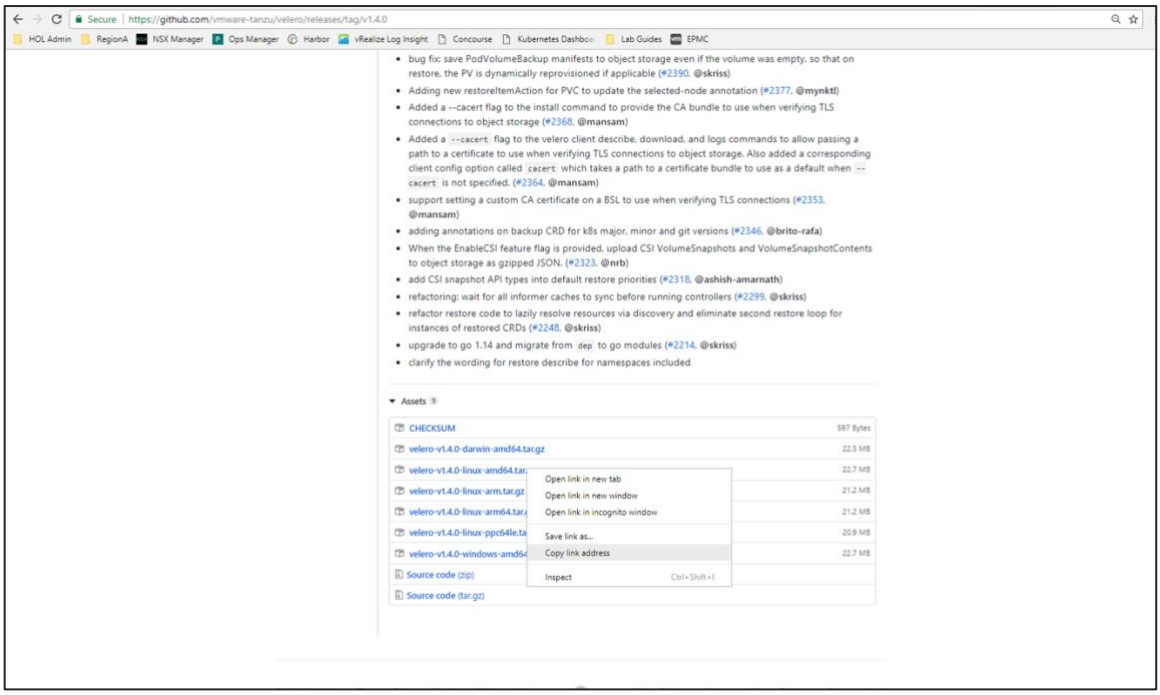
```
helm ininstall minio-release -n minio
```

Velero

This section goes through the steps to download Velero to the provisioned Ubuntu vm (cli vm) and install it on both the source and target clusters.

Velero Setup

Step 1: Navigate to the official page of Velero (*https://github.com/vmware-tanzu/velero/releases*) and copy the link for the target VM (Virtual Machine) OS (operating systems). (Eg. *https://github.com/vmware-tanzu/velero/releases/tag/v1.4.0*). At the bottom of the page the official releases are listed, Right clink on the release link 'Copy Link address'



Step 2: ssh to the provisioned ubuntu vm. (clivm)

Step 3: Download and uncompress the Velero distribution

mkdir velero

cd ~/velero

wget *https://github.com/vmware-tanzu/velero/releases/download/v1.4.0/velero-v1.4.0-linux-amd64.tar.gz*

tar xvf velero-v1.4.0-linux-amd64.tar.gz

Install Velero

This section describes the steps required to install Velero to both the source and target clusters. Any cluster from which a backup is taken or to which a backup is restored requires to have Velero deployed to it.

Source Cluster

Source cluster is the cluster from which a Velero backup will be taken from**.** As mentioned in the assumptions section we will be using the ci-cluster as our source cluster.

Step 1: ssh into the provisioned linux/ubuntu vm (clivm)

Step 2: Get kube config for the source cluster:

pks login -a <pks api> -u <user> -p <password> -k

pks get-credentials <cluster>

Alternatively

pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k

E.g.

pks login -a pks.corp.local -u riaz -p VMware1! -k

pks get-credentials ci-cluster

pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

```
ubuntu@cli-vm:~/velero$ pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

Fetching kubeconfig for cluster ci-cluster and user riaz.
You can now use the kubeconfig for user riaz:
$kubectl config use-context ci-cluster
```

Step 3: Create a namespace for Velero, called velero:

kubectl create ns velero

Step 4: Change directory to the velero directory:

cd ~/velero/velero-v1.4.0-linux-amd64

Step 5: Create a credentials file, and name it credentials. This will contain the username and password used for Minio. The values would be the same as what was provided during the Minio setup.

[default]

aws_access_key_id = minio

aws_secret_access_key = minio123

NOTE: This file can be deleted once the Velero is installed to the cluster.

Step 6: Set kubectl context to the source cluster

kubectl config use-context <source-cluster>

E.g.

kubectl config use-context ci-cluster

Step 7: Install Velero to the source cluster.

```
./velero install  \
  --provider aws \
  --bucket velero \
  --secret-file ./credentials \
  --use-restic \
  --backup-location-config region=minio,s3ForcePathStyle="true",s3Url=http://<external-ip of minio>:<port> \
  --snapshot-location-config region=minio \
  --plugins velero/velero-plugin-for-aws:v1.1.0
```

Note: The secrets file points to location of the file credentials file we created above
use Restic to backup pv's the s3Url points to the Minio that was setup earlier.

E.g.

```
./velero install  \
  --provider aws \
  --bucket velero \
  --secret-file ./credentials \
  --use-restic \
  --backup-location-config region=minio,s3ForcePathStyle="true",s3Url=http://10.40.14.66 \
  --snapshot-location-config region=minio \
  --plugins velero/velero-plugin-for-aws:v1.1.0
```

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero install  \
>   --provider aws \
>   --bucket velero \
>   --secret-file ./credentials \
>   --backup-location-config region=minio,s3ForcePathStyle="true",s3Url=http://10.40.14.66 \
>   --snapshot-location-config region=minio \
>   --plugins velero/velero-plugin-for-aws:v1.1.0
CustomResourceDefinition/backups.velero.io: attempting to create resource
CustomResourceDefinition/backups.velero.io: created
CustomResourceDefinition/backupstoragelocations.velero.io: attempting to create resource
CustomResourceDefinition/backupstoragelocations.velero.io: created
CustomResourceDefinition/deletebackuprequests.velero.io: attempting to create resource
CustomResourceDefinition/deletebackuprequests.velero.io: created
CustomResourceDefinition/downloadrequests.velero.io: attempting to create resource
CustomResourceDefinition/downloadrequests.velero.io: created
CustomResourceDefinition/podvolumebackups.velero.io: attempting to create resource
CustomResourceDefinition/podvolumebackups.velero.io: created
CustomResourceDefinition/podvolumerestores.velero.io: attempting to create resource
CustomResourceDefinition/podvolumerestores.velero.io: created
CustomResourceDefinition/resticrepositories.velero.io: attempting to create resource
CustomResourceDefinition/resticrepositories.velero.io: created
CustomResourceDefinition/restores.velero.io: attempting to create resource
CustomResourceDefinition/restores.velero.io: created
CustomResourceDefinition/schedules.velero.io: attempting to create resource
CustomResourceDefinition/schedules.velero.io: created
CustomResourceDefinition/serverstatusrequests.velero.io: attempting to create resource
CustomResourceDefinition/serverstatusrequests.velero.io: created
CustomResourceDefinition/volumesnapshotlocations.velero.io: attempting to create resource
CustomResourceDefinition/volumesnapshotlocations.velero.io: created
Waiting for resources to be ready in cluster...
Namespace/velero: attempting to create resource
Namespace/velero: created
ClusterRoleBinding/velero: attempting to create resource
ClusterRoleBinding/velero: created
ServiceAccount/velero: attempting to create resource
ServiceAccount/velero: created
Secret/cloud-credentials: attempting to create resource
Secret/cloud-credentials: created
BackupStorageLocation/default: attempting to create resource
BackupStorageLocation/default: created
VolumeSnapshotLocation/default: attempting to create resource
VolumeSnapshotLocation/default: created
Deployment/velero: attempting to create resource
Deployment/velero: created
Velero is installed! ⛵ Use 'kubectl logs deployment/velero -n velero' to view the status.
```

Step 8: Get status of pods in the velero namespace

```
kubectl get po -n velero
```

Step 9: If the Restic pods fails to startup, we will need to edit the hostPath for the Restic pods.

```
kubectl edit daemonset restic -n velero
```

change hostPath from /var/lib/kubelet/pods to /var/vcap/data/kubelet/pods:

Which will look like below

```
-hostPath:
    path: /var/vcap/data/kubelet/pods
```

```
      - name: VELERO_NAMESPACE
        valueFrom:
          fieldRef:
            apiVersion: v1
            fieldPath: metadata.namespace
      - name: VELERO_SCRATCH_DIR
        value: /scratch
      - name: GOOGLE_APPLICATION_CREDENTIALS
        value: /credentials/cloud
      - name: AWS_SHARED_CREDENTIALS_FILE
        value: /credentials/cloud
      - name: AZURE_CREDENTIALS_FILE
        value: /credentials/cloud
      - name: ALIBABA_CLOUD_CREDENTIALS_FILE
        value: /credentials/cloud
      image: velero/velero:v1.4.0
      imagePullPolicy: IfNotPresent
      name: restic
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /host_pods
        mountPropagation: HostToContainer
        name: host-pods
      - mountPath: /scratch
        name: scratch
      - mountPath: /credentials
        name: cloud-credentials
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext:
      runAsUser: 0
    serviceAccount: velero
    serviceAccountName: velero
    terminationGracePeriodSeconds: 30
    volumes:
    - hostPath:
        path: /var/vcap/data/kubelet/pods
        type: ""
      name: host-pods
    - emptyDir: {}
      name: scratch
    - name: cloud-credentials
      secret:
        defaultMode: 420
        secretName: cloud-credentials
 templateGeneration: 1
 updateStrategy:
   rollingUpdate:
     maxUnavailable: 1
   type: RollingUpdate
tatus:
 currentNumberScheduled: 3
 desiredNumberScheduled: 3
 numberMisscheduled: 0
 numberReady: 0
 numberUnavailable: 3
 observedGeneration: 1
 updatedNumberScheduled: 3
-- INSERT --
```

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get po -n velero
NAME                      READY   STATUS    RESTARTS   AGE
restic-hmzfl              1/1     Running   0          43m
restic-jff8c              1/1     Running   0          43m
restic-s9flx              1/1     Running   0          43m
velero-84d944c59-2c9rv    1/1     Running   0          94s
```

Step 10: Get all the plugins in velero

./velero plugin get

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero plugin get
NAME                                KIND
velero.io/crd-remap-version         BackupItemAction
velero.io/pod                       BackupItemAction
velero.io/pv                        BackupItemAction
velero.io/service-account           BackupItemAction
velero.io/aws                       ObjectStore
velero.io/add-pv-from-pvc           RestoreItemAction
velero.io/add-pvc-from-pod          RestoreItemAction
velero.io/change-pvc-node-selector  RestoreItemAction
velero.io/change-storage-class      RestoreItemAction
velero.io/cluster-role-bindings     RestoreItemAction
velero.io/crd-preserve-fields       RestoreItemAction
velero.io/job                       RestoreItemAction
velero.io/pod                       RestoreItemAction
velero.io/restic                    RestoreItemAction
velero.io/role-bindings             RestoreItemAction
velero.io/service                   RestoreItemAction
velero.io/service-account           RestoreItemAction
velero.io/aws                       VolumeSnapshotter
```

Note: The AWS and restic plugins installed in the previous step is listed.

Step 11: Install the Velero Plugin for vSphere. This plugin is an alternate to Restic and enables Velero to take crash-consistent snapshot backup of a block Persistent Volume on vSphere storage and backup of volume data into S3 compatible storage. The Velero vSphere plugin has a few prerequisites:

- Velero - Version 1.3.2 or above
- vSphere - Version 6.7U3 or above
- vSphere CSI/CNS driver 1.0.2 or above
- Kubernetes 1.14 or above (note: the Velero Plug-in for vSphere does not support Guest or Supervisor clusters on vSphere yet)

Make sure these are met before you proceed to the next step

Step 12: Install the vSphere CSI driver . Follow the steps provided in the VMware Tanzu documentation to set up the CSI driver on the cluster you before creating the storage class. https://docs.pivotal.io/pks/1-7/vsphere-cns.html

NOTE: Make sure that the csi-vsphere.conf file has the correct values

```
[Global]
cluster-id = <cluster-name>

[VirtualCenter "<vcenter-ip>"]
insecure-flag = "true"
user = "<vcenter-username>"
password = "vcenter-password>”
port = "443"
datacenters = "<vcenter-datacenter>"
```

Eg.

```
[Global]
cluster-id = ci-cluster

[VirtualCenter "192.168.110.22"]
insecure-flag = "true"
user = "administrator@corp.local"
password = "VMware1!"
```

```
port = "443"

datacenters = "RegionA01"
```

Step 13: Install the velero vsphere plugin

./velero plugin add vsphereveleroplugin/velero-plugin-for-vsphere:1.0.1

Step 14: Verify if the velero vsphere plugin has been installed

./velero plugin get

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero plugin get
NAME                                KIND
velero.io/crd-remap-version         BackupItemAction
velero.io/pod                       BackupItemAction
velero.io/pv                        BackupItemAction
velero.io/service-account           BackupItemAction
velero.io/aws                       ObjectStore
velero.io/add-pv-from-pvc           RestoreItemAction
velero.io/add-pvc-from-pod          RestoreItemAction
velero.io/change-pvc-node-selector  RestoreItemAction
velero.io/change-storage-class      RestoreItemAction
velero.io/cluster-role-bindings     RestoreItemAction
velero.io/crd-preserve-fields       RestoreItemAction
velero.io/job                       RestoreItemAction
velero.io/pod                       RestoreItemAction
velero.io/restic                    RestoreItemAction
velero.io/role-bindings             RestoreItemAction
velero.io/service                   RestoreItemAction
velero.io/service-account           RestoreItemAction
velero.io/aws                       VolumeSnapshotter
velero.io/vsphere                   VolumeSnapshotter
```

Note : The vsphere plugin is listed along with AWS and restic.

Target Cluster

A target cluster is the cluster to which a Velero backup is to be restored**.** As mentioned in the assumptions section we will be using 'my-cluster' as our source cluster

Step 1: ssh into the provisioned linux/ubuntu vm (clivm)

Step 2: Get kube config for the target cluster

pks login -a <pks api> -u <user> -p <password> -k

pks get-credentials <cluster>

Alternatively

pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k

E.g.

pks login -a pks.corp.local -u riaz -p VMware1! -k

pks get-credentials my-cluster

pks get-kubeconfig my-cluster -a pks.corp.local -u riaz -p VMware1! -k

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ pks get-kubeconfig my-cluster -a pks.corp.local -u riaz -p VMware1! -k

Fetching kubeconfig for cluster my-cluster and user riaz.
You can now use the kubeconfig for user riaz:
$kubectl config use-context my-cluster
```

Step 3: Follow steps 3 to 14 in the previous section.

Uninstall Velero

To uninstall Velero we will need to delete all the resources associated with its install

```
kubectl delete namespace/velero clusterrolebinding/velero

kubectl delete crds -l component=velero
```

Yelb Application

In this section we will be deploying the Yelb (https://github.com/mreferre/yelb) application to the source cluster. Yelb allows users to vote on a set of alternatives (restaurants) and dynamically updates pie charts based on number of votes received.

The Yelb application will help us validate stateful pods and persistent data is being backed up by Velero and the Velero restore operation keeps the state of the data intact.

Deploy Yelb

Step 1: Get kube config for the source cluster

pks login -a <pks api> -u <user> -p <password> -k

pks get-credentials <cluster>

Alternatively

pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k

E.g.

pks login -a pks.corp.local -u riaz -p VMware1! -k

pks get-credentials ci-cluster

pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

```
ubuntu@cli-vm:~/velero$ pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

Fetching kubeconfig for cluster ci-cluster and user riaz.
You can now use the kubeconfig for user riaz:
$kubectl config use-context ci-cluster
```

Step 2: Set kubectl context to the source cluster

kubectl config use-context <source-cluster>

E.g.

kubectl config use-context ci-cluster

Step 3: Yelb requires a backing store / K8s persistent volume. Create a storage-class on the infra cluster with the following storage class definition. Copy the contents of the file below to a file storage-class.yaml and create the storage class.

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: thin-disk
 annotations:
     storageclass.kubernetes.io/is-default-class: "true"
provisioner: kubernetes.io/vsphere-volume
parameters:
   diskformat: thin
```

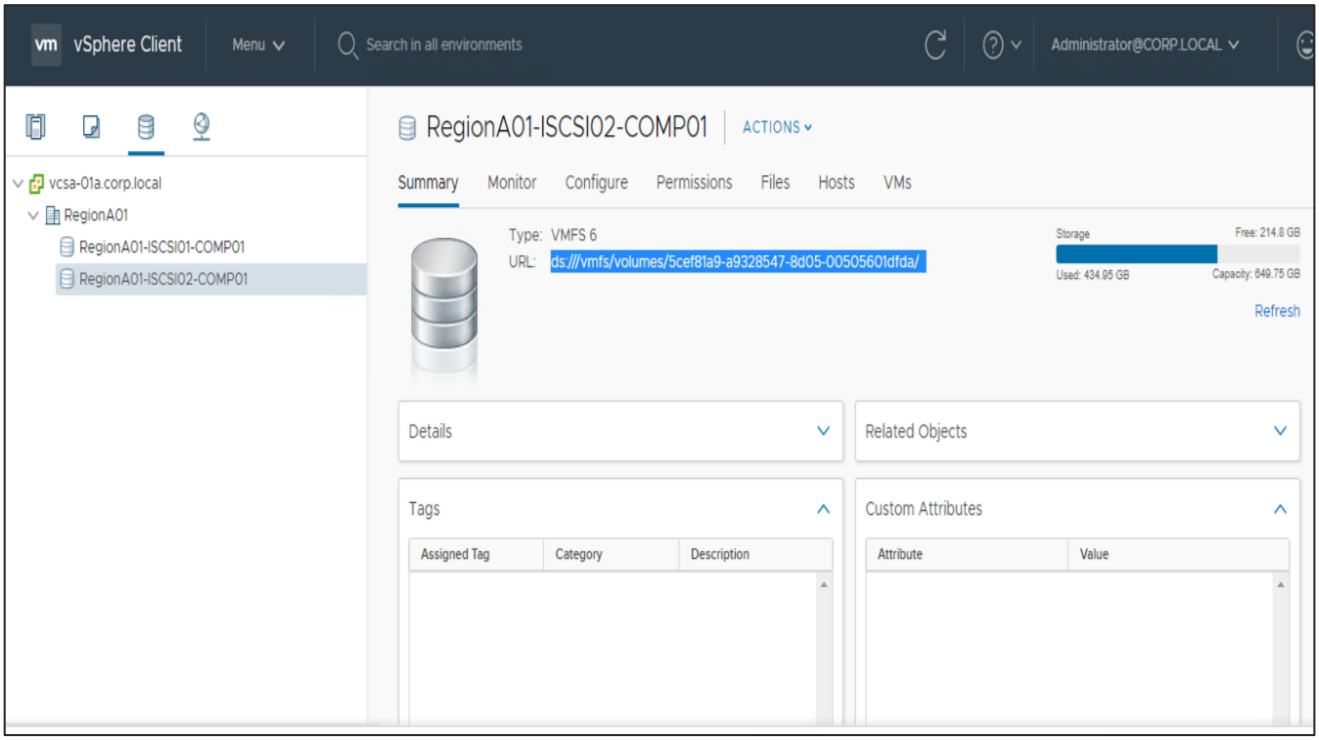
NOTE: If setting up the storage class using the csi driver , follow the steps provided in the VMware Tanzu documentation to set up the CSI driver on the cluster you before creating the storage class. https://docs.pivotal.io/pks/1-7/vsphere-cns.html

Storage class definition when using a CSI driver

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sc
  annotations:
     storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.vsphere.vmware.com
parameters:
```

```
datastoreurl: "ds:///vmfs/volumes/5cef81a9-a9328547-8d05-00505601dfda/"
```

Datastore url can be obtained from vcenter



kubectl apply -f storage-class.yaml

Step 3: Create a yelb namespace

kubectl create ns yelb

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl create ns yelb
namespace/yelb created
```

Step 4: Create the persistent volume claims required for this application. This will create a PVC for the redis-server and the yelb-db database.

Copy the contents of the file
https://github.com/riazvm/TKGIClusterBackupAndRestore/blob/master/yelb/yelb-pvc.yaml

to a local file for e.g. yelb-pvc.yaml

kubectl apply -f yelb-pvc.yaml

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl apply -f yelb-pvc.yaml
persistentvolumeclaim/redis-pv-claim created
persistentvolumeclaim/db-pv-claim created
```

Step 5: Deploy the Yelb application, this will create the necessary deployments, pods and services and expose the yelb-ui deployment as an ingress of type loadbalancer

Copy the contents of the file
https://github.com/riazvm/TKGIClusterBackupAndRestore/blob/master/yelb/yelb.yaml

file to a local file for e.g. yelb.yaml

kubectl apply -f yelb.yaml

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl apply -f yelb.yaml
service/redis-server created
service/yelb-db created
service/yelb-appserver created
service/yelb-ui created
deployment.extensions/yelb-ui created
statefulset.apps/redis-server created
statefulset.apps/yelb-db created
deployment.extensions/yelb-appserver created
```

Step 6: Verify if all the pods are running on the yelb name namespace

kubectl get po -n yelb --watch

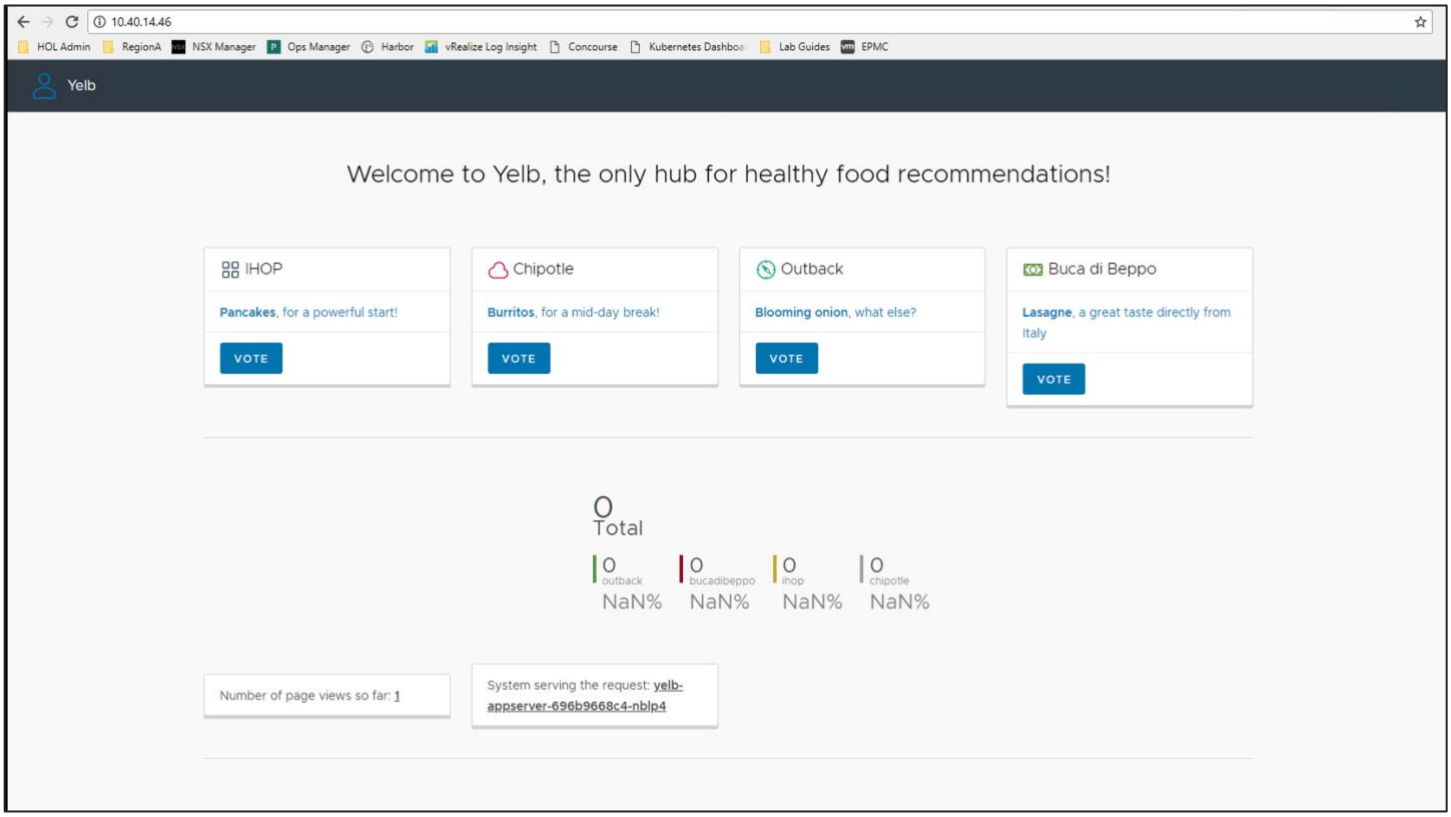
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get po -n yelb --watch
NAME                              READY   STATUS    RESTARTS   AGE
redis-server-0                    1/1     Running   0          69s
yelb-appserver-696b9668c4-7mrhs   1/1     Running   0          69s
yelb-db-0                         1/1     Running   0          69s
yelb-ui-6665575695-2lqzw          1/1     Running   0          68s
```

Step 7: Get the loadbalancer ip for the ingress to access the Yelb application. The EXTERNAL-IP is the loadbalancer's ip.

kubectl get svc -n yelb

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n yelb
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
redis-server     ClusterIP      10.100.200.113   <none>        6379/TCP       2m12s
yelb-appserver   ClusterIP      10.100.200.155   <none>        4567/TCP       2m11s
yelb-db          ClusterIP      10.100.200.158   <none>        5432/TCP       2m12s
yelb-ui          LoadBalancer   10.100.200.3     10.40.14.46   80:30561/TCP   2m11s
```

Step 8: Access the application by pointing the browser to the ip from the previous step



Unistall Yelb

To delete the Yelb application and all its associated resources, delete the yelb namespace

```
kubectl delete ns yelb
```

On-demand Backups

This section describes steps to backup resources deployed to a source cluster. Before backing up an application , namespace, or a cluster there are a few considerations to take into account

- Schedule a downtime before taking a backup, this would help with data consistency for stateful applications
- For stateless applications built on the 12-factor principles and if application require to be highly available, consider having a global load balancer to route traffic to an active-active cluster and block traffic to the source cluster.
- Consider how pod to pod routing of services is designed, if a service mesh is being considered in the design as well
- For stateless applications built on the 12-factor principles check if a backup is necessary or if the application can be deployed to the target cluster with a pipeline in place
- Consider application retry mechanisms in place for events , message bus etc.

The steps give an overview of backing up all the resources in a cluster as well as backing up just a namespace in a cluster. This document does not go through the steps required to be considered before backing up a cluster.

The typical uses cases to perform a K8s resource backup is to facilitate application migrations from one cluster to another or between namespaces etc.

Step 1: Get kube config for the source cluster

```
pks login -a <pks api> -u <user> -p <password> -k
pks get-credentials <cluster>
```

Alternatively

```
pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k
```

E.g.

pks login -a pks.corp.local -u riaz -p VMware1! -k

pks get-credentials ci-cluster

pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

```
ubuntu@cli-vm:~/velero$ pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

Fetching kubeconfig for cluster ci-cluster and user riaz.
You can now use the kubeconfig for user riaz:
$kubectl config use-context ci-cluster
```

Step 2: Set kubectl context to the source cluster

kubectl config use-context <source-cluster>

E.g.

kubectl config use-context ci-cluster

Step 3: Check all resources running on the source cluster

kubectl get ns

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS   AGE
default           Active   32d
kube-node-lease   Active   32d
kube-public       Active   32d
kube-system       Active   32d
pks-system        Active   32d
velero            Active   6d22h
x1                Active   20d
y1                Active   20d
yelb              Active   3d19h
z1                Active   20d
```

NOTE: apart from the default and system namespaces, yelb, x1, y1 and z1 exist

kubectl get po --all-namespaces

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get po --all-namespaces
NAMESPACE     NAME                                                READY   STATUS             RESTARTS   AGE
default       busybox-7b87695f88-jgc4f                            0/1     CrashLoopBackOff   6010       21d
kube-system   coredns-6f9bcd8956-6znfz                            1/1     Running            1          32d
kube-system   coredns-6f9bcd8956-cg14n                            1/1     Running            1          32d
kube-system   coredns-6f9bcd8956-kmcpc                            1/1     Running            1          32d
kube-system   kubernetes-dashboard-5fc4ccc79f-csmb5               1/1     Running            1          32d
kube-system   metrics-server-7f85c59675-sjc2r                     1/1     Running            1          32d
pks-system    cert-generator-11b35c51b71ea3086396a780dbf20b5cd695b25d-dbxp7   0/1   Completed   0          32d
pks-system    event-controller-7b96987577-kjczf                   2/2     Running            0          21d
pks-system    fluent-bit-hxh69                                    2/2     Running            0          21d
pks-system    fluent-bit-whqxw                                    2/2     Running            0          21d
pks-system    fluent-bit-xrbzx                                    2/2     Running            0          21d
pks-system    metric-controller-66b8b66498-f9x8h                  1/1     Running            0          21d
pks-system    observability-manager-7dd6c4c6d-gq4q9               1/1     Running            1          32d
pks-system    sink-controller-5d76d8d546-sbghh                    1/1     Running            0          21d
pks-system    telegraf-9cmrm                                      1/1     Running            0          21d
pks-system    telegraf-gnj7j                                      1/1     Running            0          21d
pks-system    telegraf-zmvxc                                      1/1     Running            0          21d
pks-system    telemetry-agent-58797bf64d-7skz5                    2/2     Running            2          32d
pks-system    validator-847cb99cc-pzhrl                           1/1     Running            0          21d
pks-system    vrops-cadvisor-6kj4p                                1/1     Running            1          32d
pks-system    vrops-cadvisor-r4jcp                                1/1     Running            1          32d
pks-system    vrops-cadvisor-wfw4h                                1/1     Running            1          32d
velero        restic-hmzfl                                        1/1     Running            0          6d22h
velero        restic-jff8c                                        1/1     Running            0          6d22h
velero        restic-s9flx                                        1/1     Running            0          6d22h
velero        velero-84d944c59-2c9rv                              1/1     Running            0          6d21h
x1            service-a-84965f57cc-nhbrx                          2/2     Running            187        20d
x1            service-a-84965f57cc-q72fh                          2/2     Running            187        20d
y1            service-b-86d8c888c7-6r9kp                          2/2     Running            187        20d
y1            service-b-86d8c888c7-jpwf9                          2/2     Running            187        20d
yelb          redis-server-0                                      1/1     Running            0          3d19h
yelb          yelb-appserver-696b9668c4-7mrhs                     1/1     Running            0          3d19h
yelb          yelb-db-0                                           1/1     Running            0          3d19h
yelb          yelb-ui-6665575695-2lqzw                            1/1     Running            0          3d19h
z1            service-c-5c5fc5c857-d9sqm                          2/2     Running            187        20d
z1            service-c-5c5fc5c857-jxqbh                          2/2     Running            187        20d
z1            service-d-69f59f4cb9-f94rs                          2/2     Running            187        20d
z1            service-d-69f59f4cb9-pkplt                          2/2     Running            187        20d
```

NOTE: Note the pods running in the yelb , x1, y1 and z1 namespaces

kubectl get pv --all-namespaces

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get pv --all-namespaces
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
pvc-443e4fd9-a024-485d-b0fb-2273b668f908   2Gi        RWO            Delete           Bound    yelb/redis-pv-claim   thin-disk               3d19h
pvc-ed1b4b11-661a-450f-9d11-84c0acbbf1a0   5Gi        RWO            Delete           Bound    yelb/db-pv-claim      thin-disk               3d19h
```

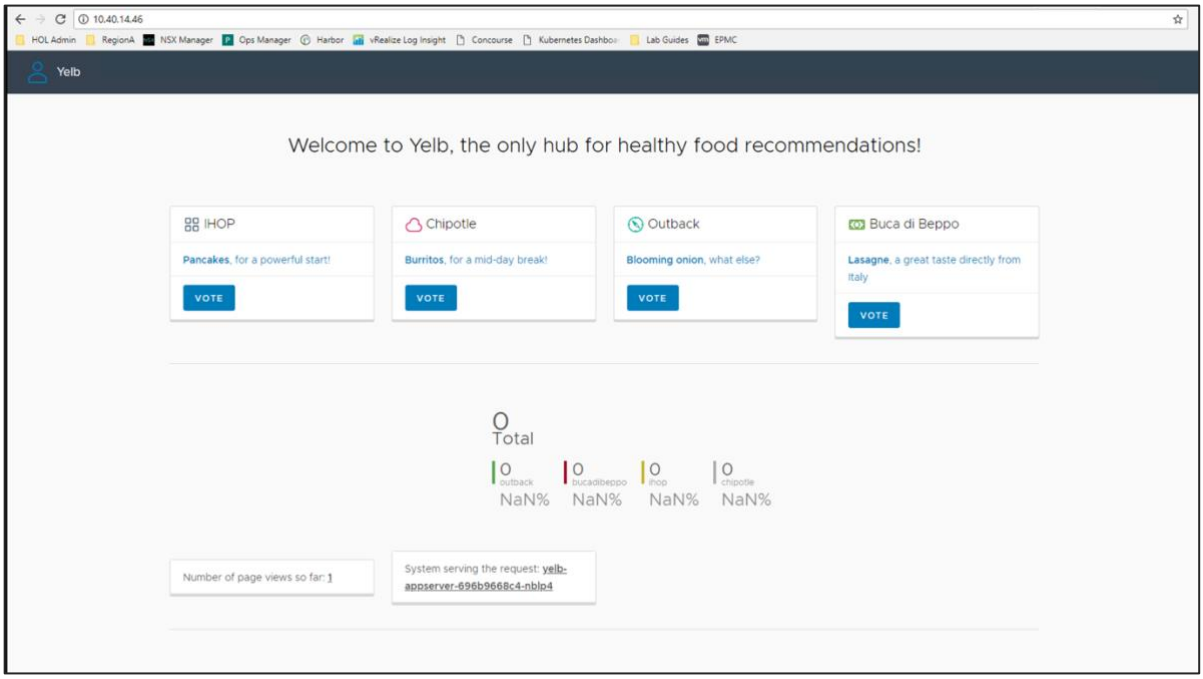
NOTE: There are a couple of PVs in the yelb namespace (redis and yelb database)

Step 4: Login to the Yelb application and make sure that the application is reachable

kubectl get svc -n yelb

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n yelb
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
redis-server     ClusterIP      10.100.200.113   <none>        6379/TCP         2m12s
yelb-appserver   ClusterIP      10.100.200.155   <none>        4567/TCP         2m11s
yelb-db          ClusterIP      10.100.200.158   <none>        5432/TCP         2m12s
yelb-ui          LoadBalancer   10.100.200.3     10.40.14.46   80:30561/TCP     2m11s
```

Point the browser to the external-ip of yelb-ui service



Step 5: The Yelb application runs a yelb-db and the redis-server pods which are both stateful. All stateful pods need to be annotated.

NOTE: when using the velero vsphere plugin annotation is not required (Skip Step 5, Step 6 and Step 7)

Run the following to annotate each pod that contains a volume to back up

```
kubectl -n YOUR_POD_NAMESPACE annotate pod/YOUR_POD_NAME backup.velero.io/backup-volumes=YOUR_VOLUME_NAME_1,YOUR_VOLUME_NAME_2,...
```

Step 6: To find the volumes for the stateful pod, identify the stateful pod and describe it. For eg. in the Yelb deployment is the stateful pod

kubectl get po -n yelb

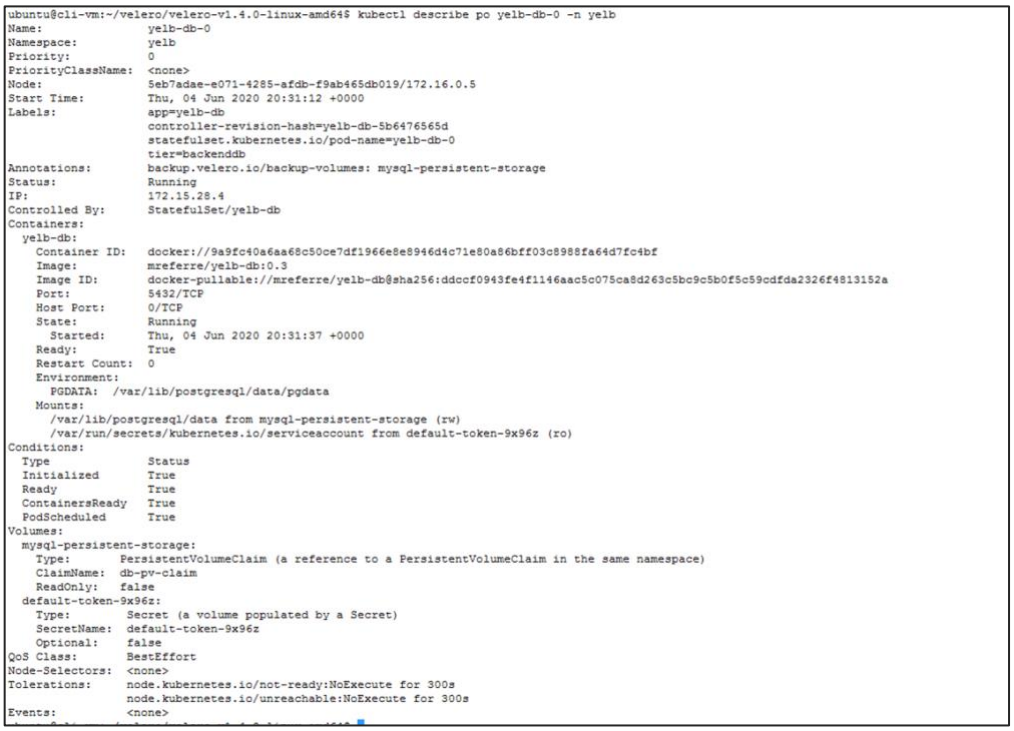
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get po -n yelb
NAME                              READY   STATUS    RESTARTS   AGE
redis-server-0                    1/1     Running   0          3d19h
yelb-appserver-696b9668c4-7mrhs   1/1     Running   0          3d19h
yelb-db-0                         1/1     Running   0          3d19h
yelb-ui-6665575695-2lqzw          1/1     Running   0          3d19h
```

The yelb application has two stateful pods (yelb-db and the redis-server)

kubectl describe po yelb-db-0 -n yelb

The volumes associated with this pod is

Volumes:
mysql-persistent-storage:
Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
ClaimName: db-pv-claim
ReadOnly: false



```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl describe po yelb-db-0 -n yelb
Name:               yelb-db-0
Namespace:          yelb
Priority:           0
PriorityClassName:  <none>
Node:               5eb7adae-e071-4285-afdb-f9ab465db019/172.16.0.5
Start Time:         Thu, 04 Jun 2020 20:31:12 +0000
Labels:             app=yelb-db
                    controller-revision-hash=yelb-db-5b6476565d
                    statefulset.kubernetes.io/pod-name=yelb-db-0
                    tier=backenddb
Annotations:        backup.velero.io/backup-volumes: mysql-persistent-storage
Status:             Running
IP:                 172.15.28.4
Controlled By:      StatefulSet/yelb-db
Containers:
  yelb-db:
    Container ID:   docker://9a9fc40a6aa68c50ce7df1966e8e8946d4c71e80a86bff03c8988fa64d7fc4bf
    Image:          mreferre/yelb-db:0.3
    Image ID:       docker-pullable://mreferre/yelb-db@sha256:ddccf0943fe4f1146aac5c075ca8d263c5bc9c5b0f5c59cdfda2326f4813152a
    Port:           5432/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 04 Jun 2020 20:31:37 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      PGDATA:  /var/lib/postgresql/data/pgdata
    Mounts:
      /var/lib/postgresql/data from mysql-persistent-storage (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9x96z (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  mysql-persistent-storage:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  db-pv-claim
    ReadOnly:   false
  default-token-9x96z:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-9x96z
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:          <none>
```

kubectl describe po redis-server-0 -n yelb

The volumes associated with this pod is

Volumes:
redis-persistent-storage:
Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
ClaimName: redis-pv-claim
ReadOnly: false

Step 7: Annotate the yelb-db-0 and redis-server-0 pods

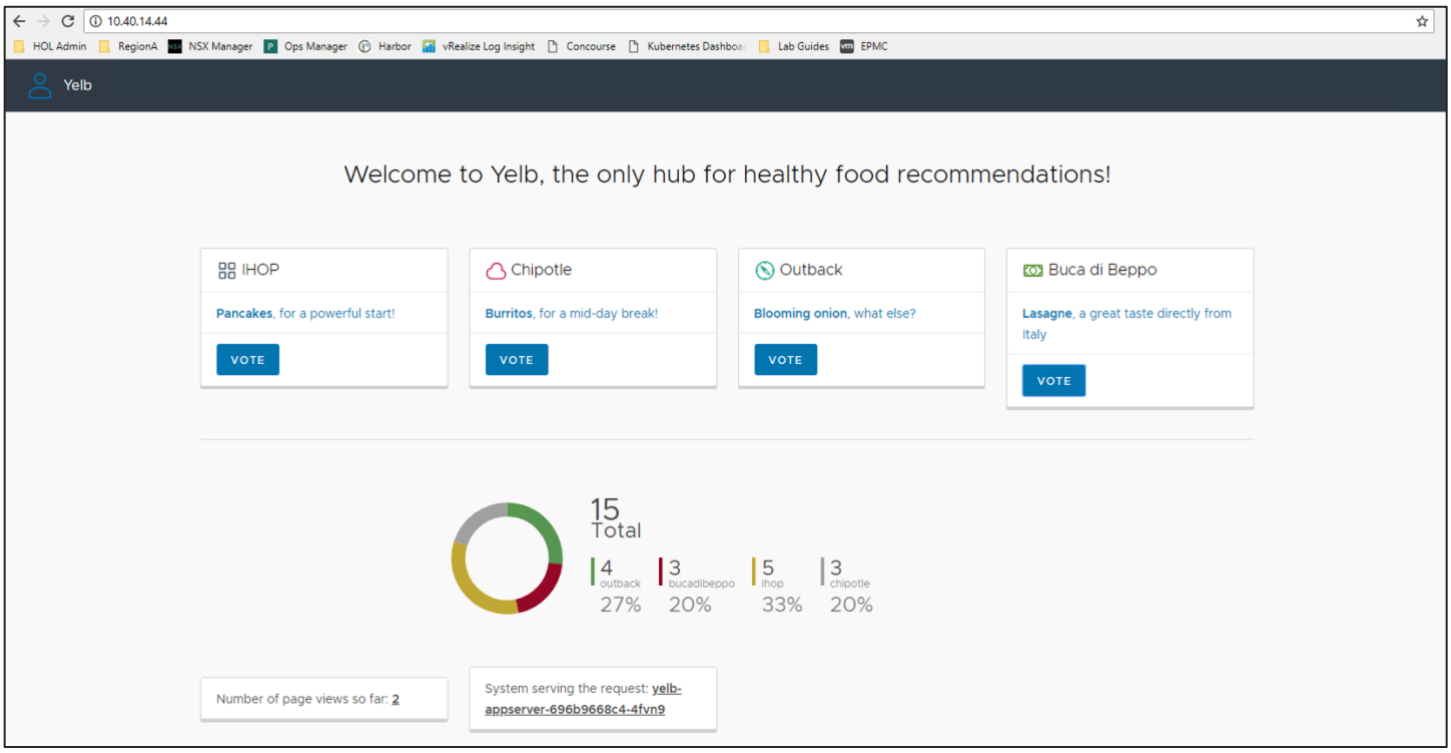
kubectl -n yelb annotate pod/< yelb db name for eg. yelb-db-0 > backup.velero.io/backup-volumes=<volume name>

e.g.

kubectl -n yelb annotate pod/yelb-db-0 backup.velero.io/backup-volumes=mysql-persistent-storage

kubectl -n yelb annotate pod/redis-server-0 backup.velero.io/backup-volumes=redis-persistent-storage

Step 8: Login to the Yelb application and vote for some of the restaurants listed. The voting data is persisted in the Yelb database, this data will validate the state of the application when restored. Keep a tab on the votes.



Cluster Backup Using the Restic Plugin

Step 9: Create a backup all the resources in a cluster

```
cd ~/velero/velero-v1.4.0-linux-amd64
./velero create backup <BACKUP NAME>
E.g.
./velero create backup sourceclusterbk
```

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero create backup sourceclusterbk
Backup request "sourceclusterbk" submitted successfully.
Run `velero backup describe sourceclusterbk` or `velero backup logs sourceclusterbk` for more details.
```

Step 10: Check status of the backup. The output describes the status of the backup. We have taken a complete backup and hence all resources and PVs are backed up.

./velero backup describe <BACKUP NAME>

E.g.

./velero backup describe sourceclusterbk

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero backup describe sourceclusterbk
Name:         sourceclusterbk
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/source-cluster-k8s-gitversion=v1.15.5
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=15

Phase:  Completed

Namespaces:
  Included:  *
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        <none>
  Cluster-scoped:  auto

Label selector:  <none>

Storage Location:  default

Velero-Native Snapshot PVs:  auto

TTL:  720h0m0s

Hooks:  <none>

Backup Format Version:  1

Started:    2020-06-09 04:20:33 +0000 UTC
Completed:  2020-06-09 04:20:49 +0000 UTC

Expiration:  2020-07-09 04:20:33 +0000 UTC

Total items to be backed up:  570
Items backed up:              570

Velero-Native Snapshots: <none included>

Restic Backups (specify --details for more information):
  Completed:  2
```

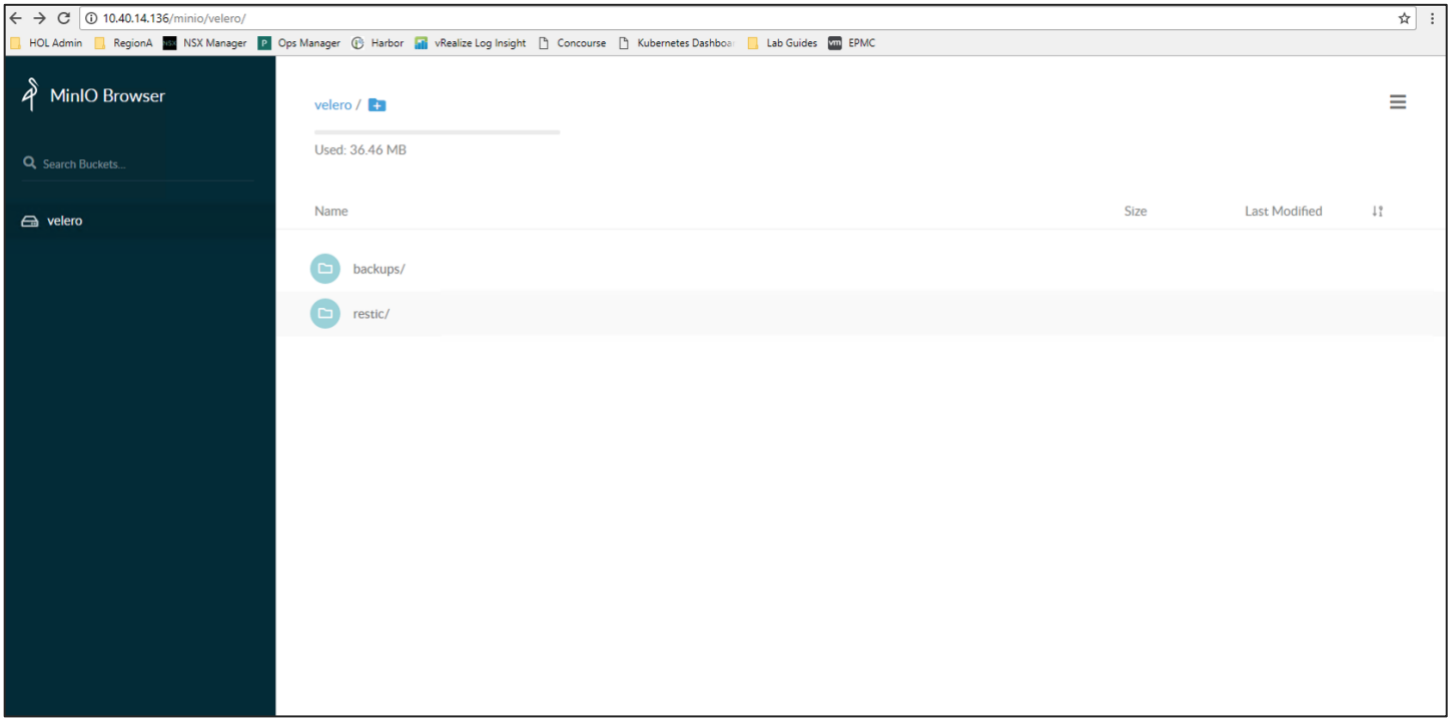
Note: The restic backups – these are backups of the pv's in this case the yelb-db and redis server

Step 11: Login to Minio and check if the backup and restic folders have been created. Use *http://<minio-service-external-ip*>, e.g. *http://10.40.14.136*.

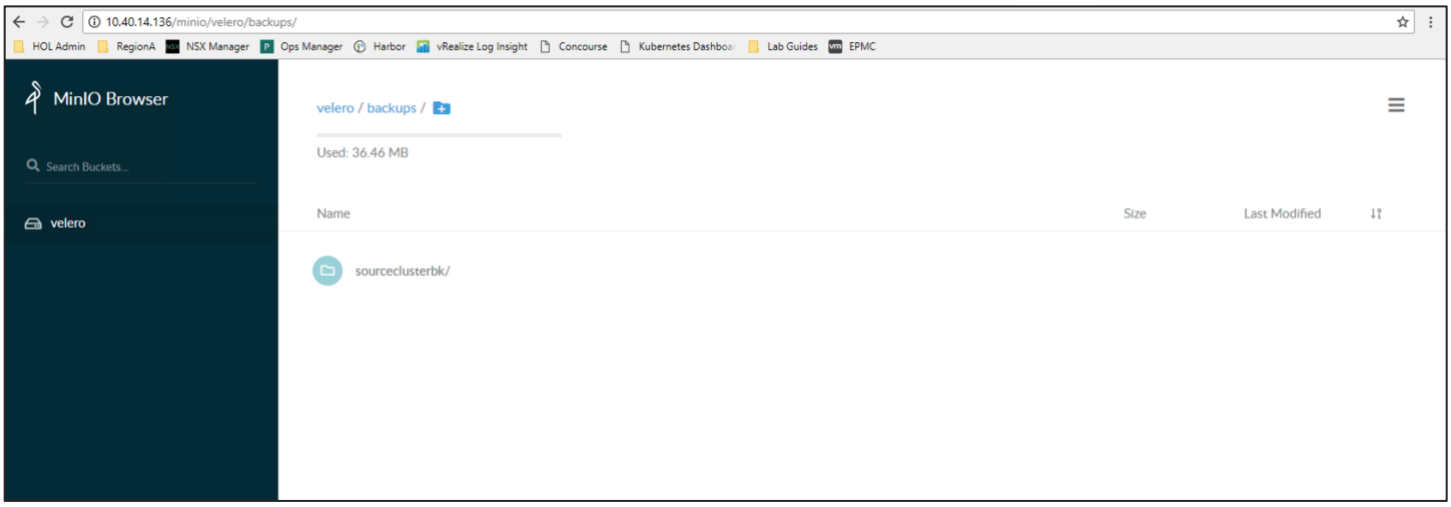
To find the ip of minio check the IP under the "External-IP" section, point your browser to the location <external-ip>. The Minio application should be accessible

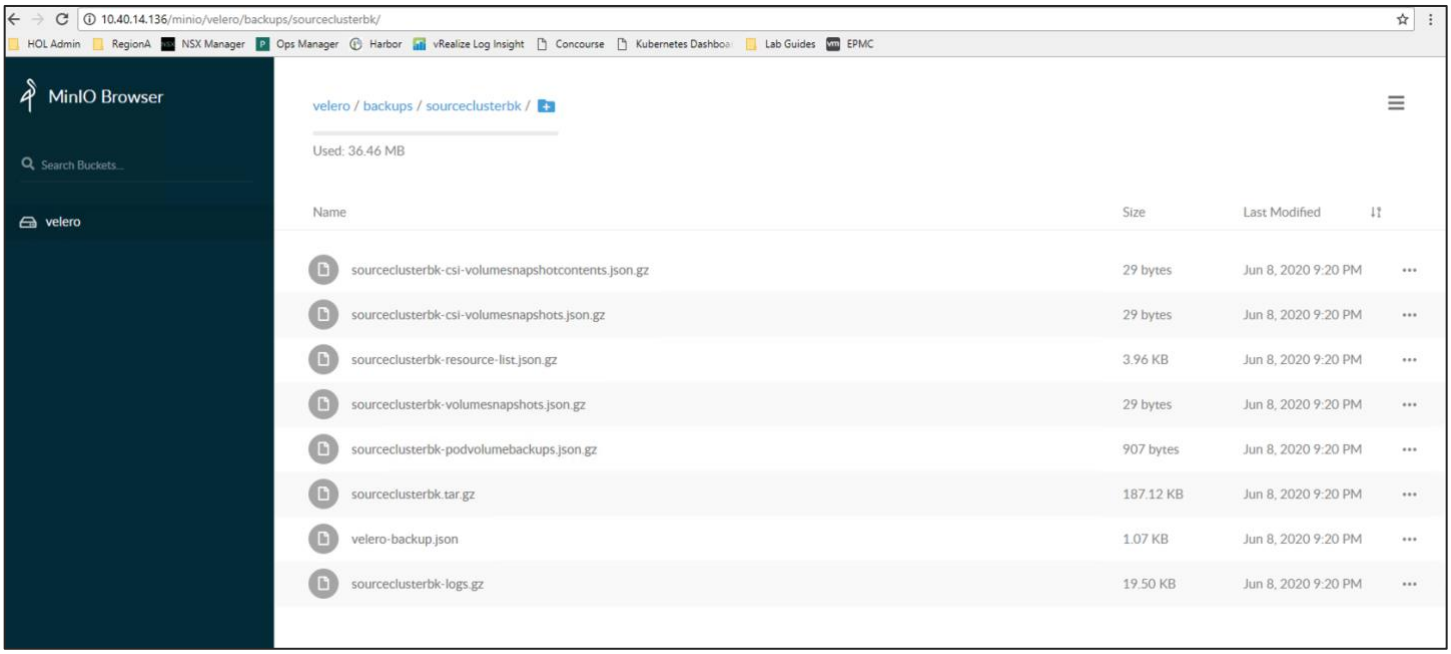
kubectl get svc -n minio

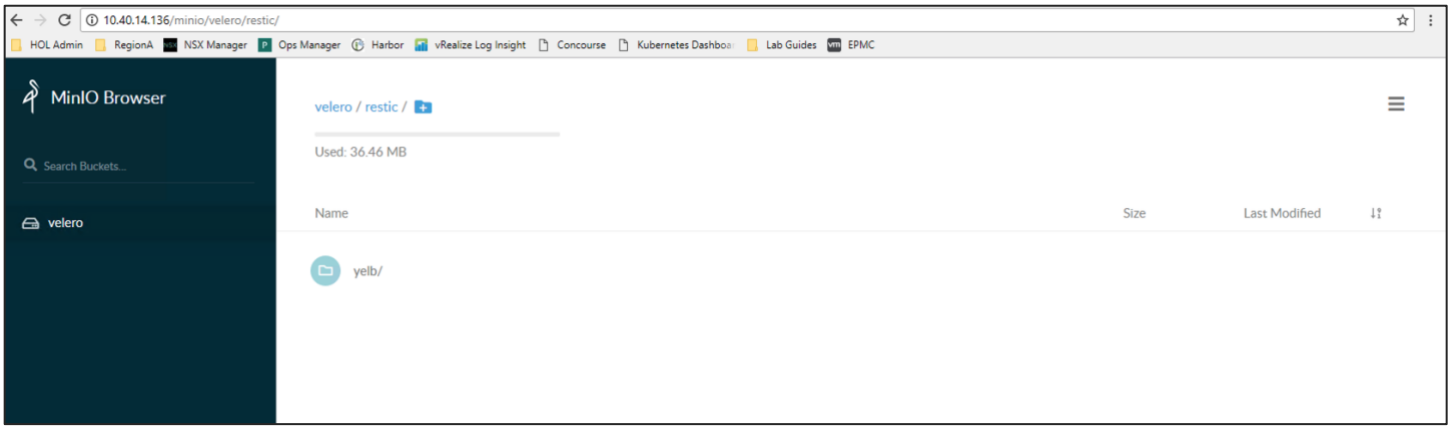
```
ubuntu@cli-vm:~/velero$ kubectl get svc -n minio
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
minio-frontend-lb   LoadBalancer   10.100.200.3     10.40.14.136   80:30568/TCP     18s
minio-release       ClusterIP      10.100.200.107   <none>         9000/TCP         13m
```

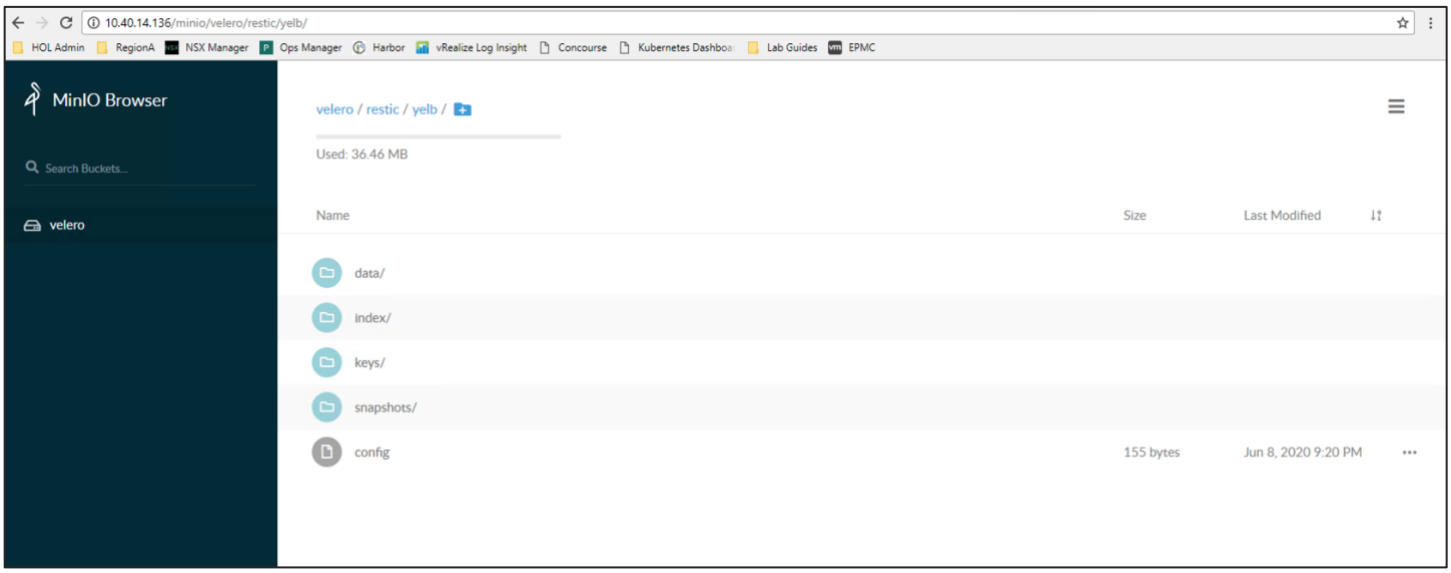


The backup folder contains the resource backup's and the restic contains the persistent volume backup's, the pv backups are referenced within the backup.

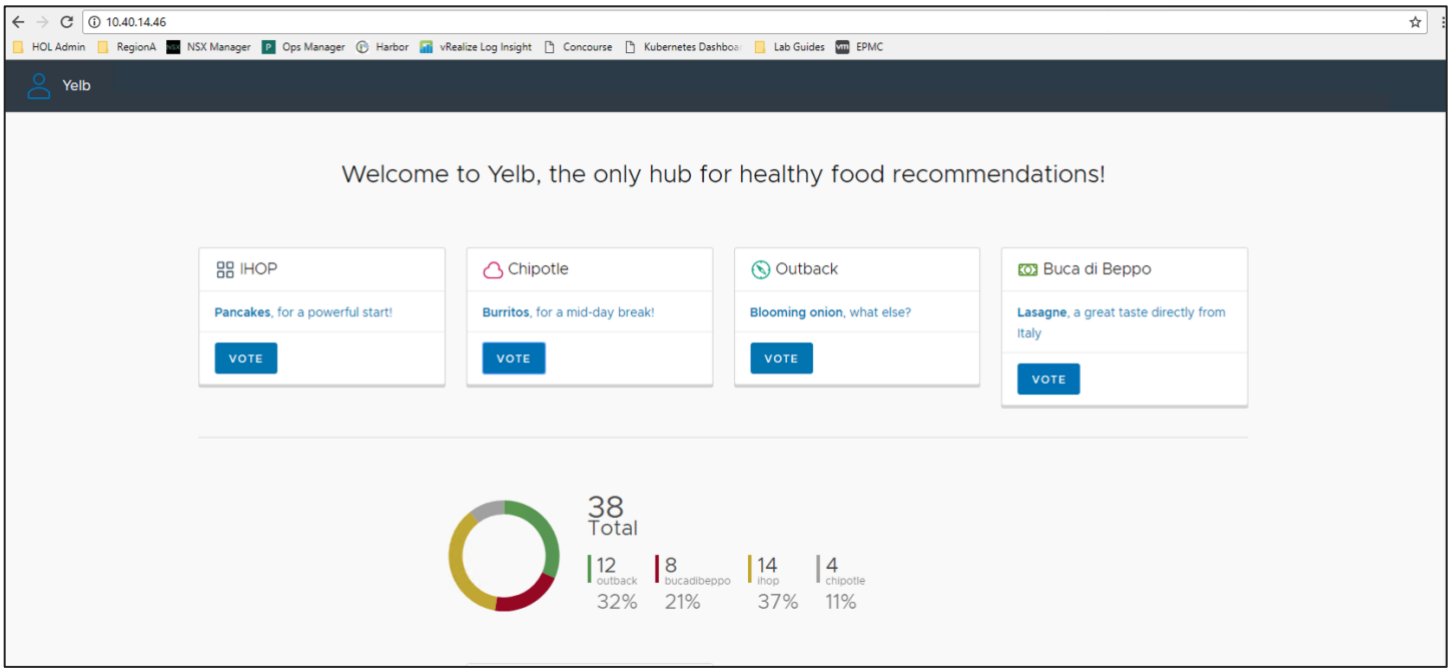

10.40.14.136/minio/velero/backups/
HOL Admin
RegionA
NSX Manager
Ops Manager
Harbor
vRealize Log Insight
Concourse
Kubernetes Dashboa
Lab Guides
EPMC
MinIO Browser
Search Buckets...
velero
velero / backups /
Used: 36.46 MB
Name
Size
Last Modified
sourceclusterbk/


10.40.14.136/minio/velero/backups/sourceclusterbk/
HOL Admin
RegionA
NSX Manager
Ops Manager
Harbor
vRealize Log Insight
Concourse
Kubernetes Dashboa
Lab Guides
EPMC
MinIO Browser
Search Buckets...
velero
velero / backups / sourceclusterbk /
Used: 36.46 MB
Name
Size
Last Modified
sourceclusterbk-csi-volumesnapshotcontents.json.gz
29 bytes
Jun 8, 2020 9:20 PM
sourceclusterbk-csi-volumesnapshots.json.gz
29 bytes
Jun 8, 2020 9:20 PM
sourceclusterbk-resource-list.json.gz
3.96 KB
Jun 8, 2020 9:20 PM
sourceclusterbk-volumesnapshots.json.gz
29 bytes
Jun 8, 2020 9:20 PM
sourceclusterbk-podvolumebackups.json.gz
907 bytes
Jun 8, 2020 9:20 PM
sourceclusterbk.tar.gz
187.12 KB
Jun 8, 2020 9:20 PM
velero-backup.json
1.07 KB
Jun 8, 2020 9:20 PM
sourceclusterbk-logs.gz
19.50 KB
Jun 8, 2020 9:20 PM


10.40.14.136/minio/velero/restic/
HOL Admin
RegionA
NSX Manager
Ops Manager
Harbor
vRealize Log Insight
Concourse
Kubernetes Dashboa
Lab Guides
EPMC
MinIO Browser
Search Buckets...
velero
velero / restic /
Used: 36.46 MB
Name
Size
Last Modified
yelb/



Step 12: Login to the Yelb application and more votes to the restaurants listed. The voting data is persisted in the Yelb database, this data will validate the state of the application when restored.



Cluster Backup Using the vSphere plugin

Step 1: Create a Volume snapshot location. This Volume Snapshot location is referenced when a backup is taken

cd ~/velero/velero-v1.4.0-linux-amd64

./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere
Snapshot volume location "vsphere-snap-loc" configured successfully.
```

Step 2: Create a cluster backup

./velero backup create srccluster-snap-backup --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero backup create srccluster-snap-backup --snapshot-volumes --volume-snapshot-locations v
Backup request "srccluster-snap-backup" submitted successfully.
Run `velero backup describe srccluster-snap-backup` or `velero backup logs srccluster-snap-backup` for more details.
```

Step 3: Check status of the backup. The output describes the status of the backup. We have taken a complete backup and hence all resources and PVs are backed up.

./velero backup describe <BACKUP NAME>

For more details on the backup

./velero backup describe <BACKUP NAME> --details

E.g.

./velero backup describe srccluster-snap-backup

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero backup describe srccluster-snap-backup
Name:         srccluster-snap-backup
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/source-cluster-k8s-gitversion=v1.16.7+vmware.1
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=16

Phase:  Completed

Namespaces:
  Included:  *
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        <none>
  Cluster-scoped:  auto

Label selector:  <none>

Storage Location:  default

Velero-Native Snapshot PVs:  true

TTL:  720h0m0s

Hooks:  <none>

Backup Format Version:  1

Started:    2020-07-08 17:00:11 +0000 UTC
Completed:  2020-07-08 17:00:46 +0000 UTC

Expiration:  2020-08-07 17:00:11 +0000 UTC

Total items to be backed up:  575
Items backed up:              575

Velero-Native Snapshots:  2 of 2 snapshots completed successfully (specify --details for more information)
```

NOTE: Event if the velero backup status shows complete the backup would not have been complete. When the vsphere plugin is used there is a lot of operations that happen in the background. During the backup there are a number of operations that occur in Vsphere.

Step 4: Monitor the uploads

kubectl -n velero get uploads

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl -n velero get uploads
NAME                                          AGE
upload-2d9203ed-ac81-4e98-b468-15ca21658766   10m
upload-2dcc8590-d02b-4bf2-9222-f51562c971c5   19m
upload-75423580-9dcc-405d-8e34-7b324a226022   19m
upload-864510ed-73f1-4afd-ba23-5be409eb6b50   19m
upload-8671ebeb-f703-4525-8797-81d3e0e0fd9f   19m
upload-c6d903d2-eda7-41b2-b6de-6f2b339bf009   10m
upload-df77bd51-56db-45e7-9ae6-225e8fb6a3f7   10m
upload-fd1460d6-710a-4c0b-9b1b-422fa5373078   10m
```

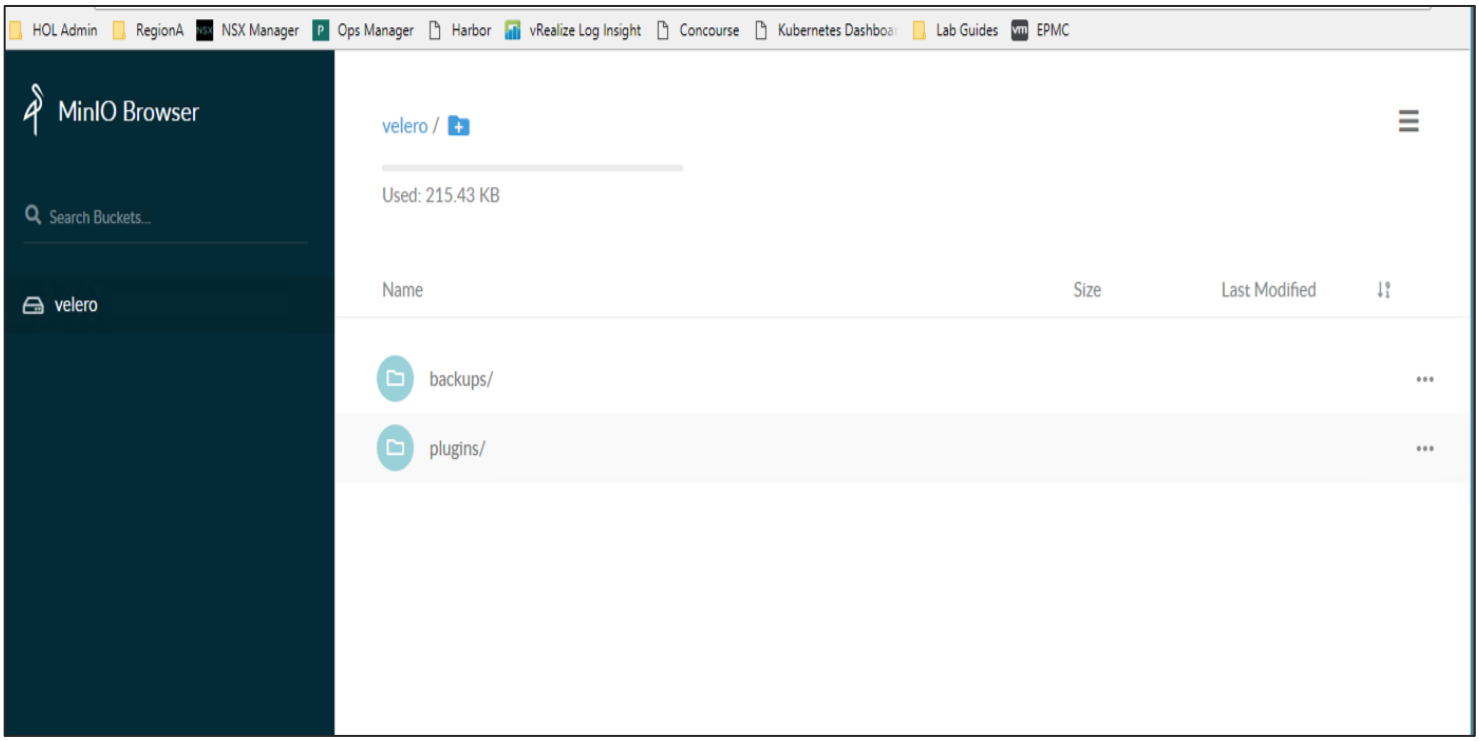
NOTE: Until the uploads are complete do not perform a restore operation

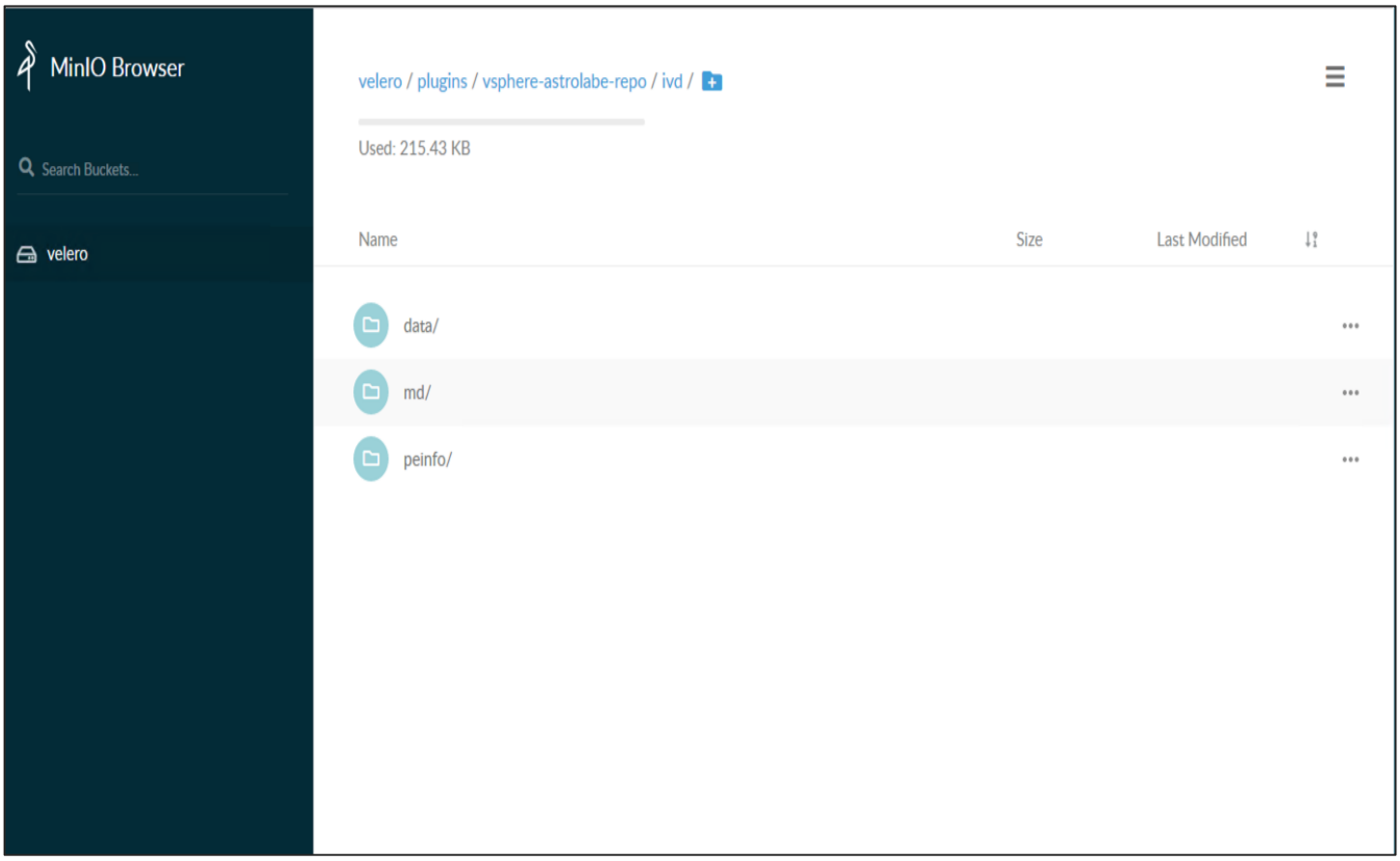
Step 5: Login to Minio and check if the backup and plugins folders have been created.

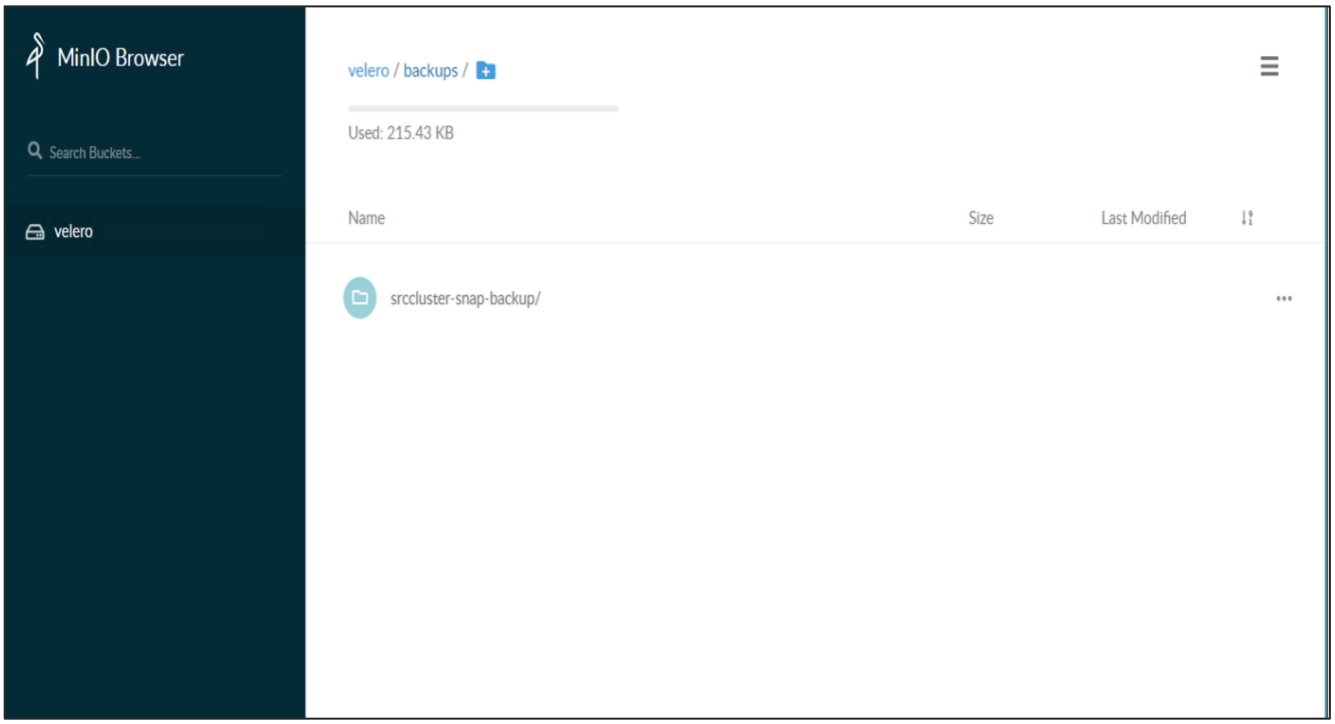
To find the ip of minio check the IP under the “External-IP” section, point your browser to the location <external-ip>. The Minio application should be accessible

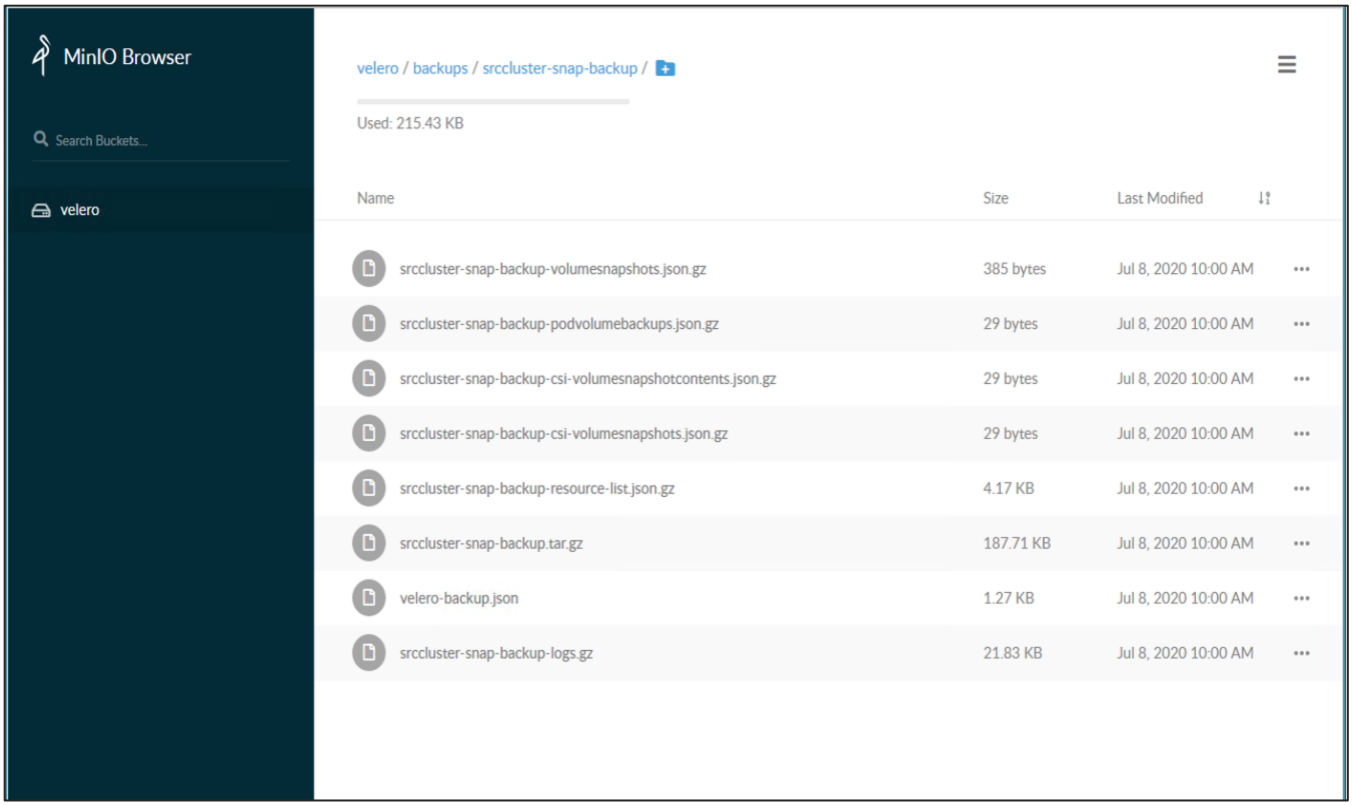
kubectl get svc -n minio

```
ubuntu@cli-vm:~/velero$ kubectl get svc -n minio
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
minio-frontend-lb   LoadBalancer   10.100.200.3     10.40.14.136   80:30568/TCP     18s
minio-release       ClusterIP      10.100.200.107   <none>         9000/TCP         13m
```

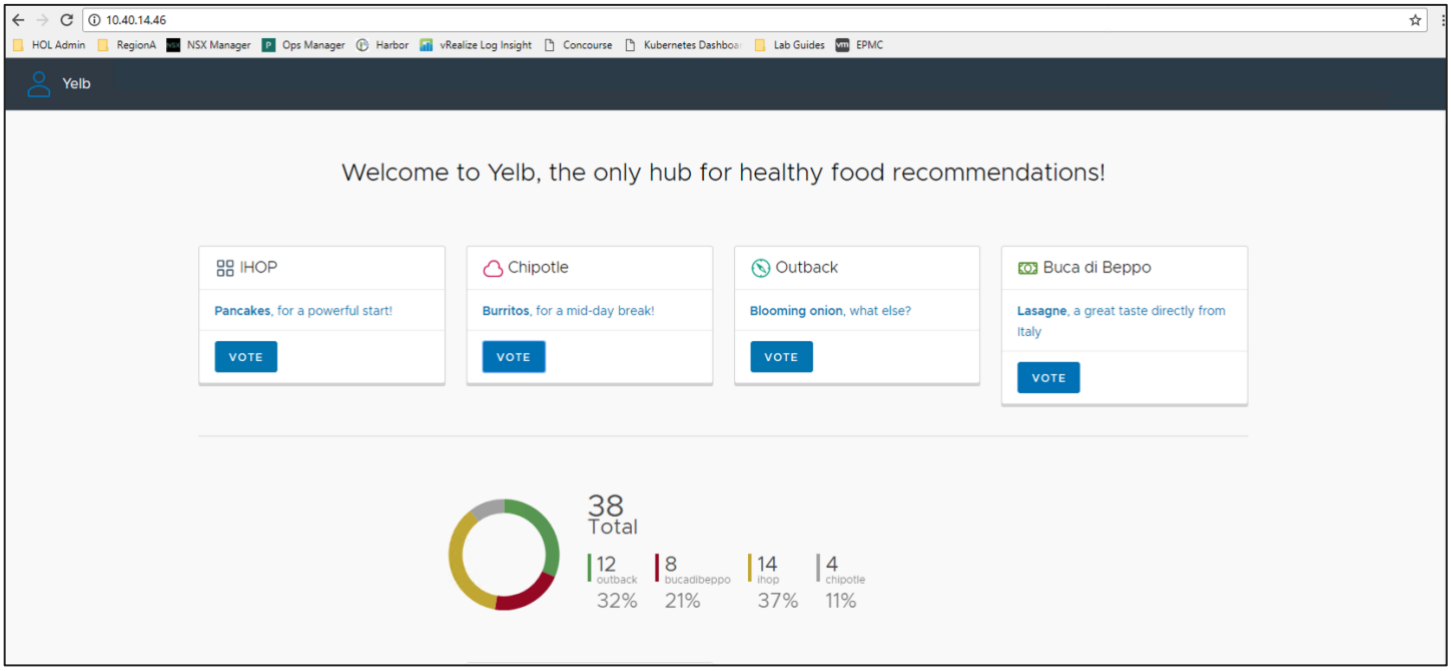



MinIO Browser
Search Buckets...
velero
velero / plugins / vsphere-astrolabe-repo / ivd /
Used: 215.43 KB
Name
Size
Last Modified
data/
md/
peinfo/


MinIO Browser
Search Buckets...
velero
velero / backups /
Used: 215.43 KB
Name
Size
Last Modified
srccluster-snap-backup/


MinIO Browser
Search Buckets...
velero
velero / backups / srccluster-snap-backup /
Used: 215.43 KB
Name
Size
Last Modified
srccluster-snap-backup-volumesnapshots.json.gz
385 bytes
Jul 8, 2020 10:00 AM
srccluster-snap-backup-podvolumebackups.json.gz
29 bytes
Jul 8, 2020 10:00 AM
srccluster-snap-backup-csi-volumesnapshotcontents.json.gz
29 bytes
Jul 8, 2020 10:00 AM
srccluster-snap-backup-csi-volumesnapshots.json.gz
29 bytes
Jul 8, 2020 10:00 AM
srccluster-snap-backup-resource-list.json.gz
4.17 KB
Jul 8, 2020 10:00 AM
srccluster-snap-backup.tar.gz
187.71 KB
Jul 8, 2020 10:00 AM
velero-backup.json
1.27 KB
Jul 8, 2020 10:00 AM
srccluster-snap-backup-logs.gz
21.83 KB
Jul 8, 2020 10:00 AM

Step 6: Login to the Yelb application and more votes to the restaurants listed. The voting data is persisted in the Yelb database, this data will validate the state of the application when restored.



Namespace Backup Using the Restic Plugin

Step 1: Create a backup of the yelb namespace

cd ~/velero/velero-v1.4.0-linux-amd64

./velero backup create <BACKUP NAME> --include-namespaces <NAMESPACE1>

E.g.

./velero backup create yelbbkup --include-namespaces yelb

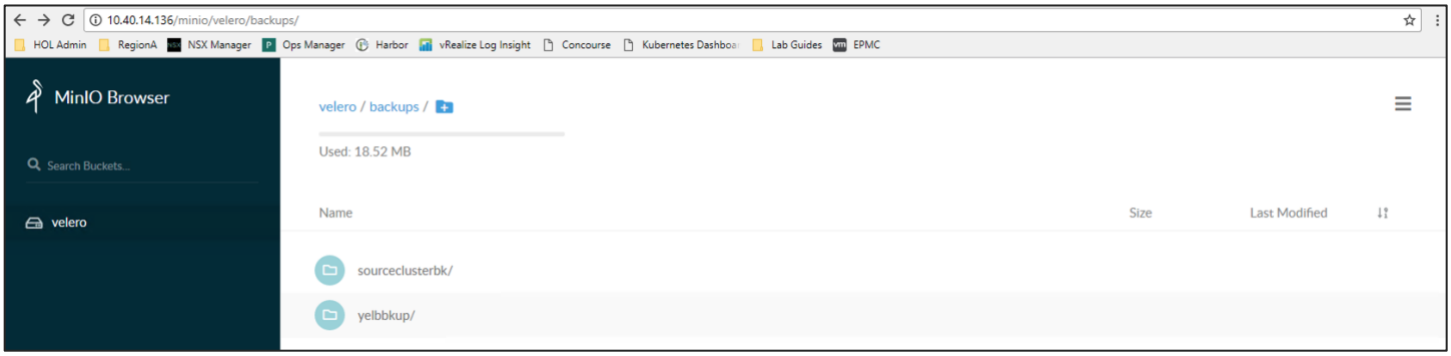
Step 2: Check status of the backup

./velero backup describe yelbbkup

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero backup describe yelbbkup
Name:         yelbbkup
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/source-cluster-k8s-gitversion=v1.15.5
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=15

Phase:  Completed

Namespaces:
  Included:  yelb
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        <none>
  Cluster-scoped:  auto

Label selector:  <none>

Storage Location:  default

Velero-Native Snapshot PVs:  auto

TTL:  720h0m0s

Hooks:  <none>

Backup Format Version:  1

Started:    2020-06-09 04:32:33 +0000 UTC
Completed:  2020-06-09 04:32:39 +0000 UTC

Expiration:  2020-07-09 04:32:33 +0000 UTC

Total items to be backed up:  53
Items backed up:              53

Velero-Native Snapshots: <none included>

Restic Backups (specify --details for more information):
  Completed:  2
```

The backup describes the status , the resources etc. In this case as shown , only the resources associated with namespace yelb is backed-up. The Restic backups describe the number of PV's that have been backed up.

Step 3: Login to minio and check if the backup has been created.



Step 4: Other options for backup

./velero backup create planes --selector app=yelb

Check Velero documentation *https://velero.io/docs/v1.4/* for other options

Namespace Backup Using the vSphere plugin

Step 1: Create a Volume snapshot location. This Volume Snapshot location is referenced when a backup is taken

NOTE: If a snapshot location has already been created this step is optional

cd ~/velero/velero-v1.4.0-linux-amd64

./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere
Snapshot volume location "vsphere-snap-loc" configured successfully.
```

Step 2: Create a namespace backup

./velero backup create yelb-vspheresnap-bkp --include-namespaces=yelb --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero backup create yelb-vspheresnap-bkp --include-namespaces=yelb --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc
Backup request "yelb-vspheresnap-bkp" submitted successfully.
Run `velero backup describe yelb-vspheresnap-bkp` or `velero backup logs yelb-vspheresnap-bkp` for more details.
```

Step 3: Check status of the backup

./velero backup describe yelb-vspheresnap-bkp –details

```
    - yelb/redis-server-0.161fddc04e981821
    - yelb/redis-server-0.161fddc0dea8378d
    - yelb/redis-server-0.161fddc3d9d76c06
    - yelb/redis-server-0.161fddc3ea9defee
    - yelb/redis-server-0.161fddc4019fe300
    - yelb/redis-server.161fddc04ceb30f4
    - yelb/yelb-appserver-794d7c9458-cblfn.161fddc0573dbf31
    - yelb/yelb-appserver-794d7c9458-cblfn.161fddc113f84b07
    - yelb/yelb-appserver-794d7c9458-cblfn.161fddc11ea7ceb0
    - yelb/yelb-appserver-794d7c9458-cblfn.161fddc135a79906
    - yelb/yelb-appserver-794d7c9458.161fddc056207e71
    - yelb/yelb-appserver.161fddc05516668a
    - yelb/yelb-db-0.161fddc0526e1081
    - yelb/yelb-db-0.161fddc23e4a39bf
    - yelb/yelb-db-0.161fddc59e98da85
    - yelb/yelb-db-0.161fddcc1cbb2087
    - yelb/yelb-db-0.161fddce111d5a04
    - yelb/yelb-db-0.161fddce229e1360
    - yelb/yelb-db.161fddc050e7083d
    - yelb/yelb-ui-79c68df689-r72hr.161fddc04968bce6
    - yelb/yelb-ui-79c68df689-r72hr.161fddc0f90e255b
    - yelb/yelb-ui-79c68df689-r72hr.161fddc6c35d8f95
    - yelb/yelb-ui-79c68df689-r72hr.161fddc9687bd145
    - yelb/yelb-ui-79c68df689-r72hr.161fddc98668e3be
    - yelb/yelb-ui-79c68df689.161fddc048232b77
    - yelb/yelb-ui.161fddc0470d1430
  v1/Namespace:
    - yelb
  v1/PersistentVolume:
    - pvc-9c5dc2c9-084d-4485-bc4f-a327ca47bd9a
    - pvc-e6166d4e-a21c-44d6-89f2-c3dec5d92199
  v1/PersistentVolumeClaim:
    - yelb/db-pv-claim
    - yelb/redis-pv-claim
  v1/Pod:
    - yelb/redis-server-0
    - yelb/yelb-appserver-794d7c9458-cblfn
    - yelb/yelb-db-0
    - yelb/yelb-ui-79c68df689-r72hr
  v1/Secret:
    - yelb/default-token-p8jj9
  v1/Service:
    - yelb/redis-server
    - yelb/yelb-appserver
    - yelb/yelb-db
    - yelb/yelb-ui
  v1/ServiceAccount:
    - yelb/default

Velero-Native Snapshots:
  pvc-e6166d4e-a21c-44d6-89f2-c3dec5d92199:
    Snapshot ID:        ivd:2f2dc9a4-aee2-47a4-a909-5db8b4f837b9:26662c39-642a-4179-af29-a110e1d0292c
    Type:               ivd
    Availability Zone:
    IOPS:               100
  pvc-9c5dc2c9-084d-4485-bc4f-a327ca47bd9a:
    Snapshot ID:        ivd:15091bc2-0844-4428-b3d5-69399c814620:23500ea7-e487-4313-91d7-27a6f7297c4c
    Type:               ivd
    Availability Zone:
    IOPS:               100
```

NOTE: Even if the velero backup status shows ‘complete’, the backup would not have been complete. When the vSphere plugin is used there is a lot of operations that happen in the background (vSphere etc.).

Step 4: Monitor the uploads

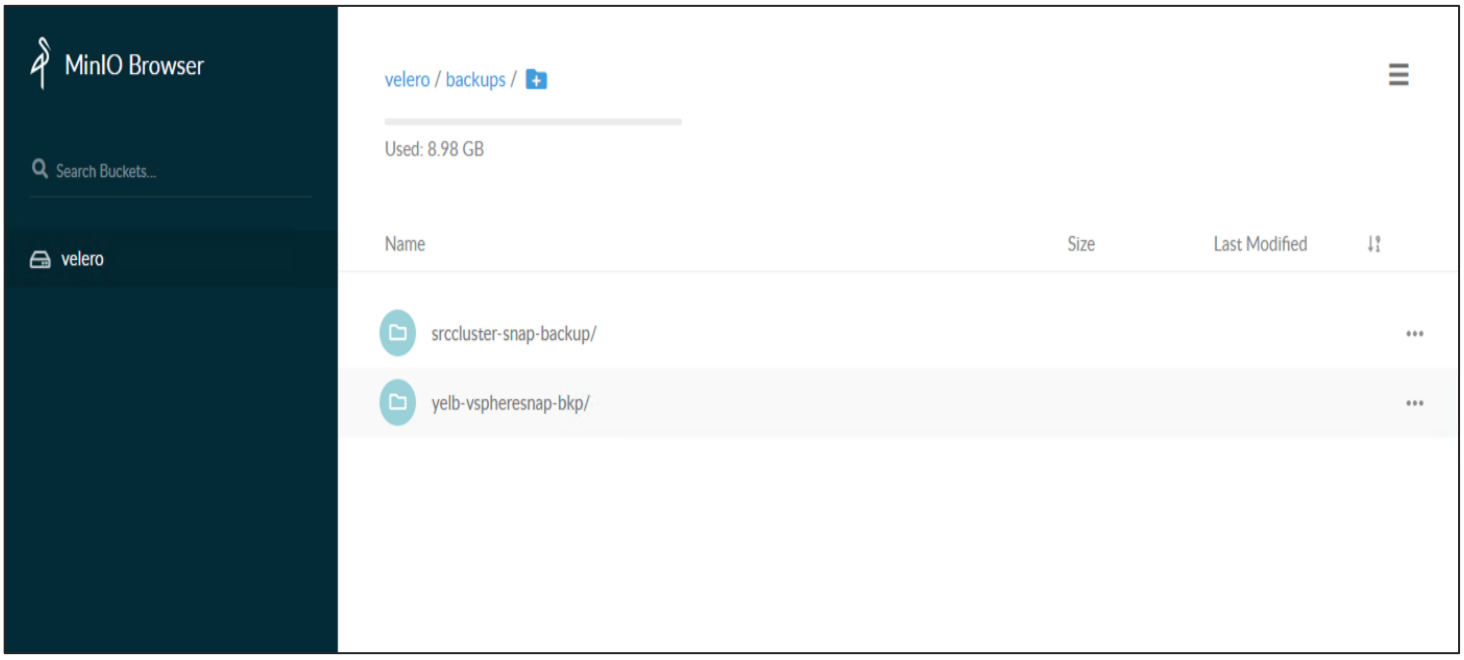
kubectl -n velero get uploads

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl -n velero get uploads
NAME                                          AGE
upload-23500ea7-e487-4313-91d7-27a6f7297c4c   2m34s
upload-26662c39-642a-4179-af29-a110e1d0292c   2m45s
upload-2d9203ed-ac81-4e98-b468-15ca21658766   17m
upload-2dcc8590-d02b-4bf2-9222-f51562c971c5   26m
upload-75423580-9dcc-405d-8e34-7b324a226022   26m
upload-864510ed-73f1-4afd-ba23-5be409eb6b50   26m
upload-8671ebeb-f703-4525-8797-81d3e0e0fd9f   26m
upload-c6d903d2-eda7-41b2-b6de-6f2b339bf009   17m
upload-df77bd51-56db-45e7-9ae6-225e8fb6a3f7   17m
upload-fd1460d6-710a-4c0b-9b1b-422fa5373078   17m
```

NOTE: Until the uploads are complete do not perform a restore operation

Step 5: Check vCenter to make sure that none of the operations related to the backup are running

Step 6: Login to Minio and check if the backup exists



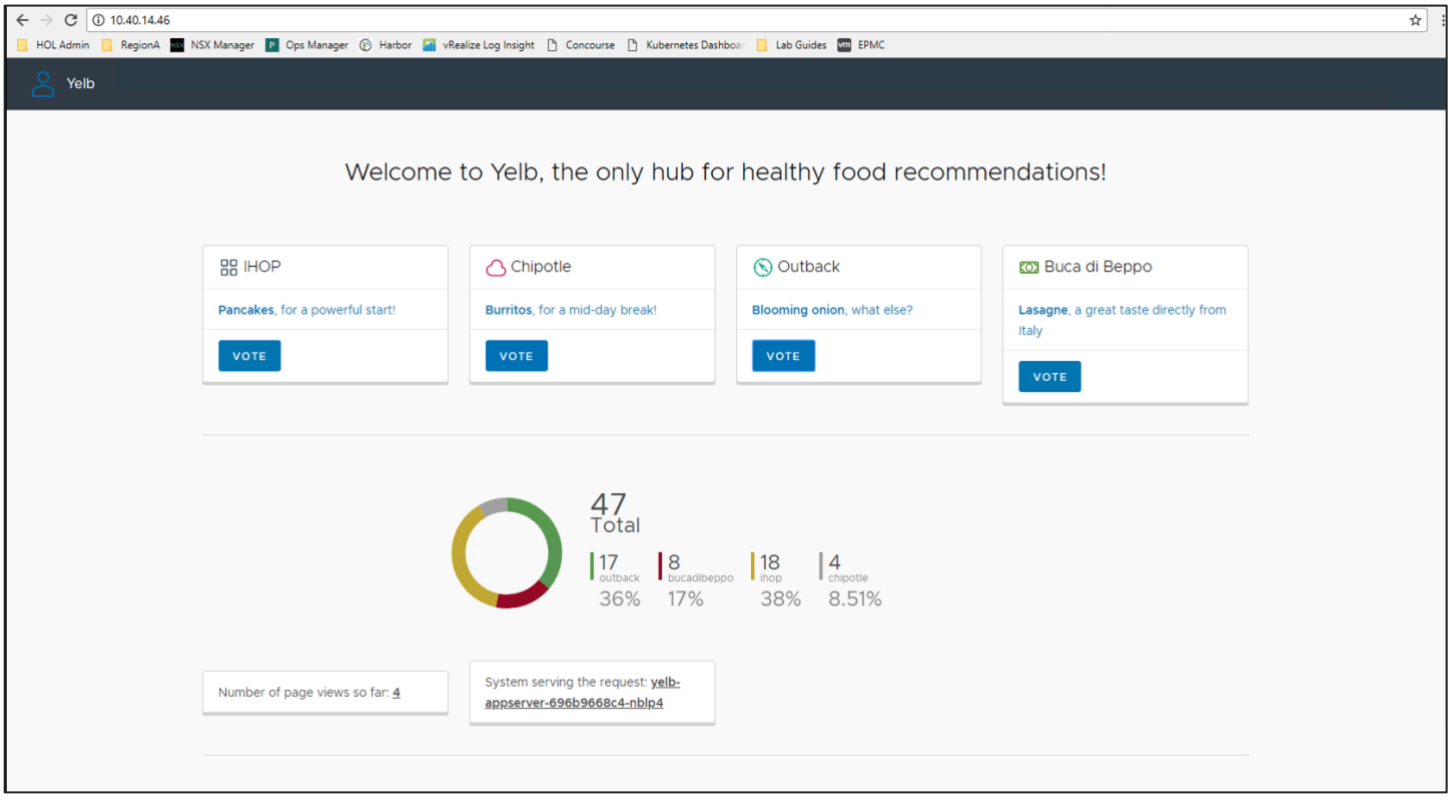
Scheduled Backups

Schedule Backup Using the Restic Plugin

The schedule operation allows you to back up your data at recurring intervals. The first backup is performed when the schedule is first created, and subsequent backups happen at the schedule's specified interval. These intervals are specified by a Cron expression.

Scheduled backups are point in time backups and can be used for disaster recovery use cases.

Step 1: Login to the Yelb application and more votes to the restaurants listed. The voting data is persisted in the Yelb database it will validate the state of the application when restored.



Step 2: Create a backup scheduler

./velero schedule create <scheduler-name> --schedule <cron-expression>

E.g.

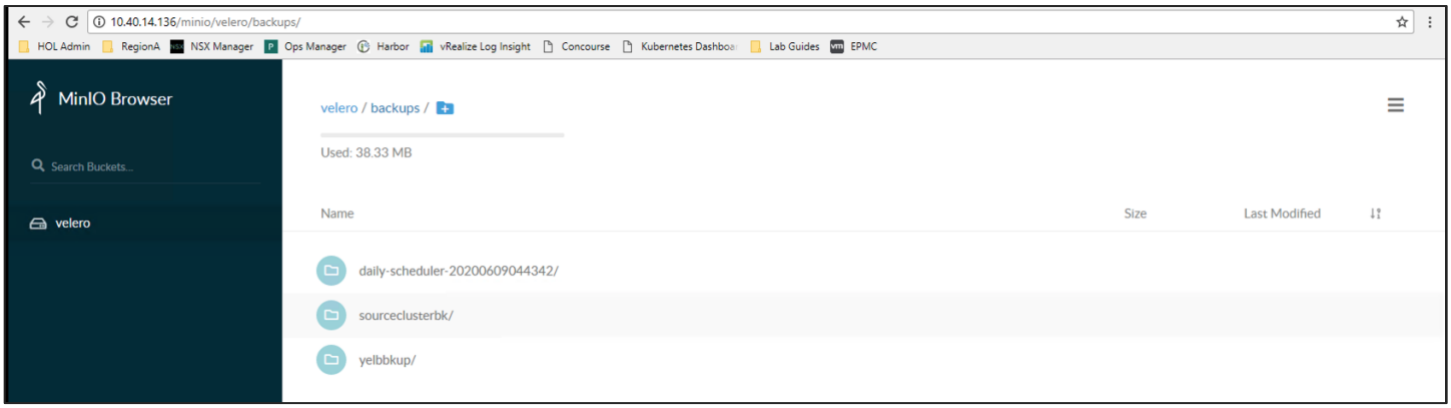
To create a daily scheduler (use *https://crontab.guru/every-15-minutes* for cron expressions if needed)

./velero schedule create daily-scheduler --schedule="@every 24h" --ttl 24h0m0s

Note: The TTL flag allows the user to specify the backup retention period with the value specified in hours, minutes, and seconds in the form --ttl 24h0m0s.

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero schedule create daily-scheduler --schedule="@every 24h" --ttl 24h0m0s
Schedule "daily-scheduler" created successfully.
```

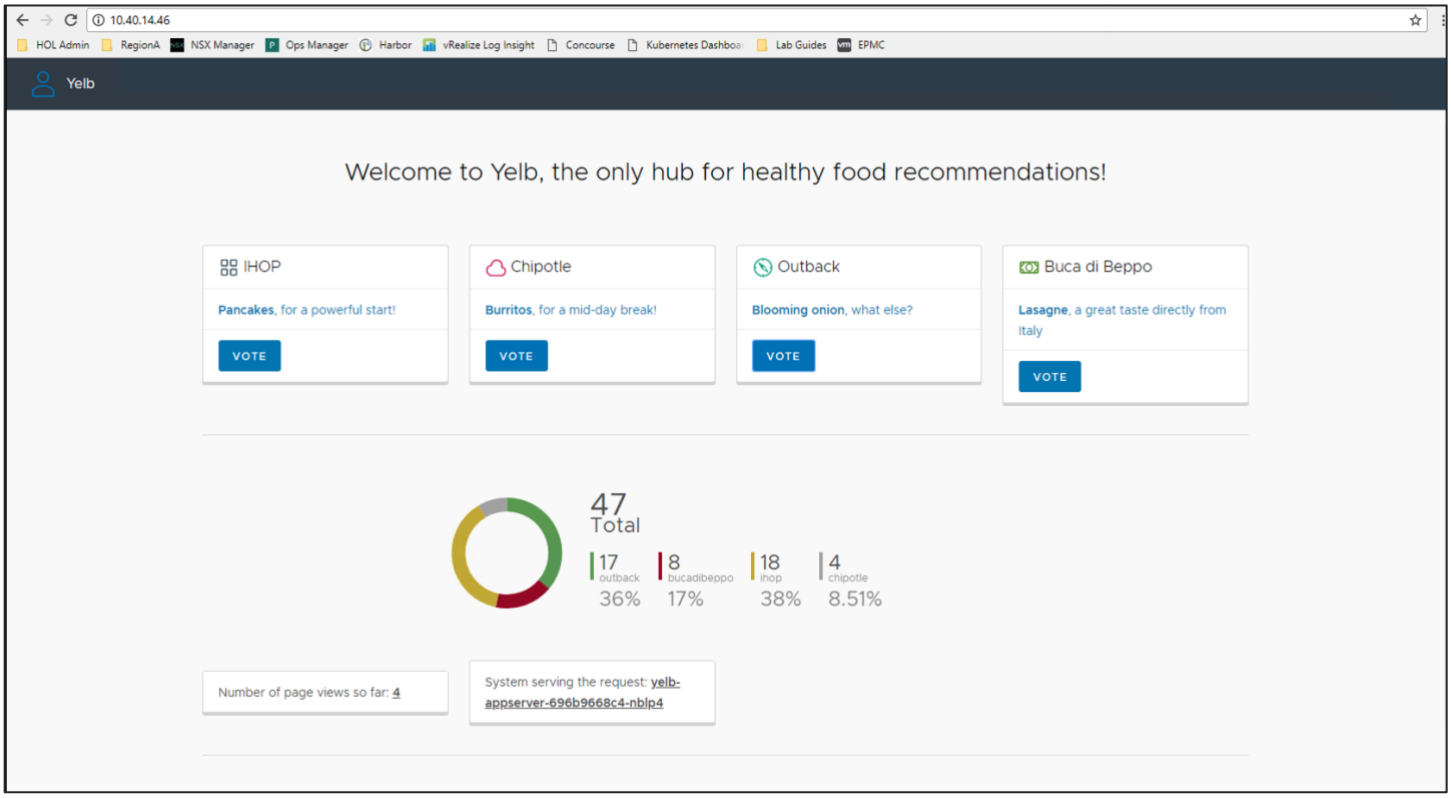
Step 3: Login to Minio and check if the backup has been created. When creating a scheduler, the first backup will be created soon as the schedule is submitted.



Note: The state data of the application before each backup was performed was as below

Schedule Backup Using the vSphere plugin

Step 1: Login to the Yelb application and more votes to the restaurants listed. The voting data is persisted in the Yelb database , this data will validate the state of the application when restored.



Step 2: Create a Volume snapshot location. This Volume Snapshot location is referenced when a backup is taken

NOTE: If a snapshot location has already been created this step is optional

cd ~/velero/velero-v1.4.0-linux-amd64

./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere
Snapshot volume location "vsphere-snap-loc" configured successfully.
```

Step 3: Create a backup scheduler

./velero schedule create <scheduler-name> --schedule <cron-expression> --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc

E.g.

To create a daily scheduler (use *https://crontab.guru/every-15-minutes* for cron eexpressions if needed)

./velero schedule create daily-scheduler --schedule="@every 24h" --ttl 24h0m0s --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc

Note: The TTL flag allows the user to specify the backup retention period with the value specified in hours, minutes, and seconds in the form --ttl 24h0m0s.

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero schedule create daily-scheduler --schedule="@every 24h" --ttl 24h0m0s --snapshot-volumes --volume-snapshot-locations vsphere-snap-loc
Schedule "daily-scheduler" created successfully.
```

Step 4: Monitor the uploads

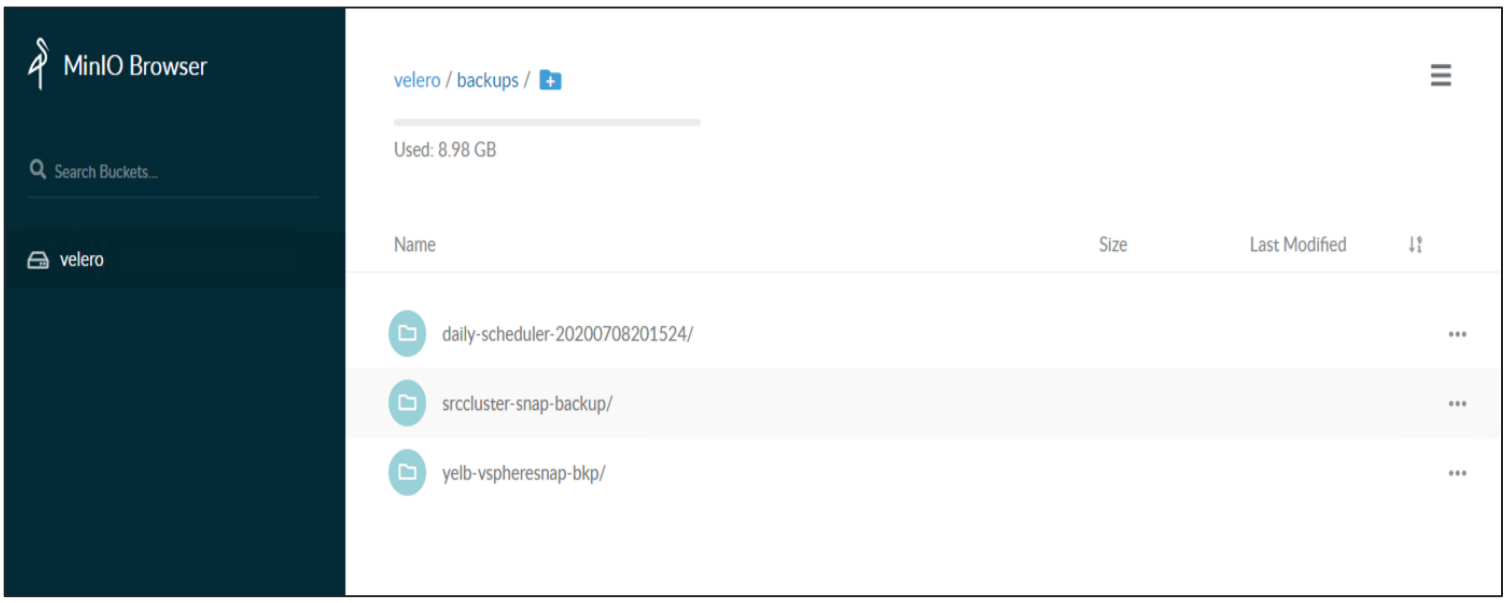
kubectl -n velero get uploads

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl -n velero get uploads
NAME                                          AGE
upload-23500ea7-e487-4313-91d7-27a6f7297c4c   20m
upload-26662c39-642a-4179-af29-a110e1d0292c   20m
upload-2d9203ed-ac81-4e98-b468-15ca21658766   35m
upload-2dcc8590-d02b-4bf2-9222-f51562c971c5   44m
upload-75423580-9dcc-405d-8e34-7b324a226022   44m
upload-864510ed-73f1-4afd-ba23-5be409eb6b50   44m
upload-8671ebeb-f703-4525-8797-81d3e0e0fd9f   44m
upload-92dd880c-dd17-49fc-92d8-9ff5c57d5503   85s
upload-98905351-7d62-4803-bf94-1c14dea33757   89s
upload-994551e3-7627-47d3-aba8-d8a93028ef97   80s
upload-c6d903d2-eda7-41b2-b6de-6f2b339bf009   35m
upload-df77bd51-56db-45e7-9ae6-225e8fb6a3f7   35m
upload-ee4c66b3-ab98-4ea0-90e1-b6ecc6ca67d5   96s
upload-fd1460d6-710a-4c0b-9b1b-422fa5373078   35m
```

Step 5: Check vCenter to make sure that none of the operations related to the backup are running

NOTE: Event if the velero backup status shows complete the backup would not have been complete. When the vsphere plugin is used there is a lot of operations that happen in the background. During the backup there are a number of operations that occur in Vsphere.

Step 6: Login to Minio and check if the scheduled backup has been created



Backup Type	Votes	Outback	Buca	IHop	Chipotle
Full Cluster	15	4 (27%)	3(20%)	5(33%)	3(20%)
Namespace	38	12(8%)	8(21%)	14(37%)	4(11%)
Scheduled backup	47	17(36%)	8(17%)	18(38%)	4(8.51%)

Check Velero documentation *https://velero.io/docs/v1.4/* for other options

Restore Backups

This section describes steps to restore a Velero backup to a target cluster. The steps give an overview of restoring a backup of a namespace and an entire clusterbackup. The restore procedure is the same if using a backup that used the restic plugin or the vsphere plugin. If the status of the restore indicates complete it does mean the restore is complete unlike the backup process.

Step 1: Get kube config for the source cluster

pks login -a <pks api> -u <user> -p <password> -k
pks get-credentials <cluster>

Alternatively
pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k
E.g.
pks login -a pks.corp.local -u riaz -p VMware1! -k
pks get-credentials my-cluster
pks get-kubeconfig my-cluster -a pks.corp.local -u riaz -p VMware1! -k

Step 2: Set kubectl context to the target cluster

kubectl config use-context <target-cluster>
E.g.
kubectl config use-context my-cluster

Step 3: Check all resources running on the target cluster

kubectl get ns

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS        AGE
default           Active        33d
kube-node-lease   Active        33d
kube-public       Active        33d
kube-system       Active        33d
pks-system        Active        33d
spc               Terminating   32d
velero            Active        10d
```

NOTE: yelb, x1, y1 and z1 namespaces do not exist

Step 4: Before backing up to a cluster, make sure you have defined a storage class to be used by the stateful applications that is being restored. Create a storage-class on the target cluster with the following storage class definition.

Copy the contents of the file below to a file storage-class.yaml and create the storage class.

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  storageclass.kubernetes.io/is-default-class: "true"
  name: thin-disk
provisioner: kubernetes.io/vsphere-volume
parameters:
   diskformat: thin
kubectl apply -f storage-class.yaml
```

kubectl get sc

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get sc
NAME                 PROVISIONER                    AGE
thin-disk (default)  kubernetes.io/vsphere-volume   33d
```

NOTE: If setting up the storage class using the csi driver , follow the steps provided in the VMware Tanzu documentation to set up the CSI driver on the cluster you before creating the storage class. https://docs.pivotal.io/pks/1-7/vsphere-cns.html

Storage class definition when using a CSI driver

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sc
  annotations:
     storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/5cef81a9-a9328547-8d05-00505601dfda/"
```

kubectl get sc

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get sc
NAME              PROVISIONER              AGE
csi-sc (default)  csi.vsphere.vmware.com   20h
```

Step 5: Create a Volume snapshot location. (If using the velero vsphere plugin)

NOTE: If a snapshot location has already been created this step is optional

cd ~/velero/velero-v1.4.0-linux-amd64

./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero snapshot-location create vsphere-snap-loc --provider velero.io/vsphere
Snapshot volume location "vsphere-snap-loc" configured successfully.
```

Restore Namespace Backup

Step 6: Restore the yelb namespace from the yelbbkup created in the previous step.

cd ~/velero/velero-v1.4.0-linux-amd64

./velero restore create --from-backup yelbbkup

In the above example for vsphere plugin the namespace backup was named yelb-vspheresnap-bkp

./velero restore create --from-backup yelb-vspheresnap-bkp

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero restore create --from-backup yelbbkup
Restore request "yelbbkup-20200609031343" submitted successfully.
Run `velero restore describe yelbbkup-20200609031343` or `velero restore logs yelbbkup-20200609031343` for more details.
```

Step 7: Check the status of the restore in the cluster. The yelb namespace should be created and the pods should be up and running. Make sure that the pv is also created and bound:

kubectl get ns

kubectl get po -n yelb

kubectl get pvc -n yelb

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS        AGE
default           Active        33d
kube-node-lease   Active        33d
kube-public       Active        33d
kube-system       Active        33d
pks-system        Active        33d
spc               Terminating   32d
velero            Active        10d
yelb              Active        4m1s
```

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get po -n yelb
NAME                          READY   STATUS    RESTARTS   AGE
redis-server-0                1/1     Running   0          4m10s
yelb-appserver-696b9668c4-l9ds5   1/1     Running   0          4m10s
yelb-db-0                     1/1     Running   0          4m9s
yelb-ui-6665575695-58dfk      1/1     Running   0          4m8s
```

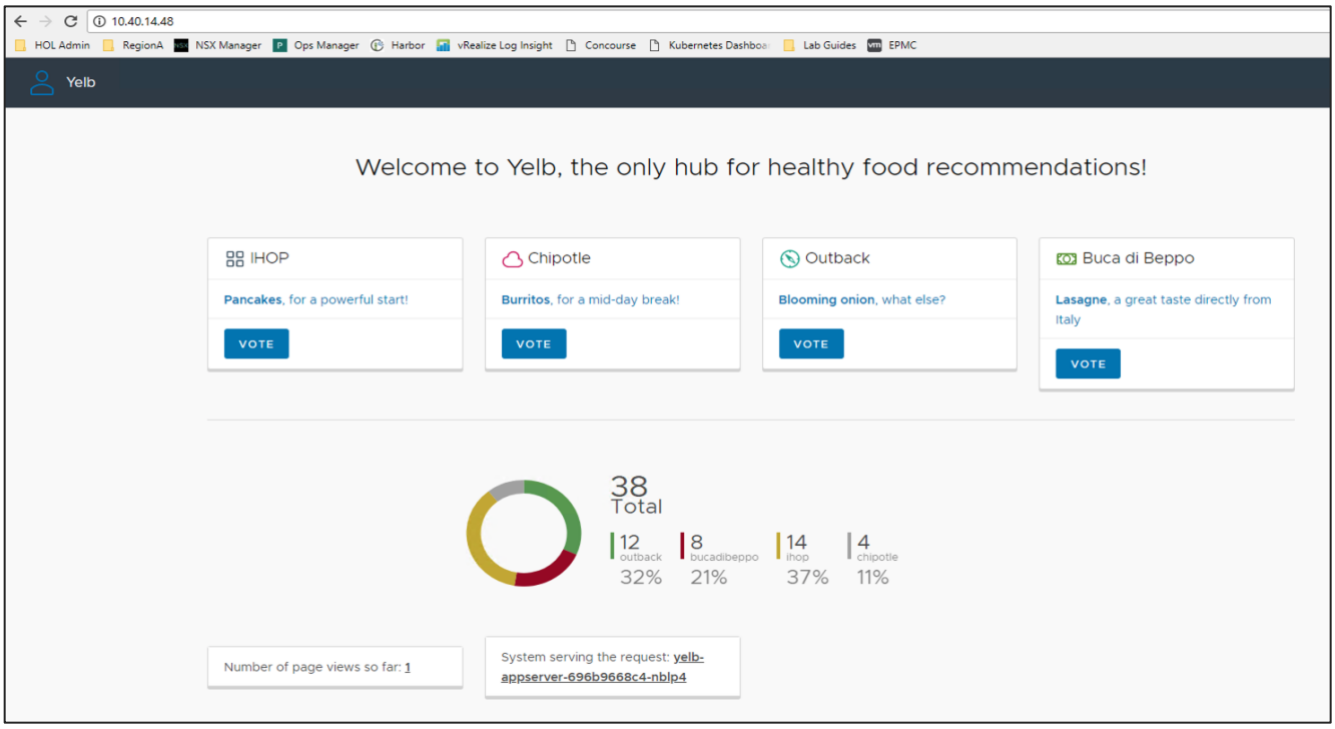
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get pvc -n yelb
NAME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
db-pv-claim      Bound    pvc-309cfeda-c316-4812-ac1f-1c4a300a487a   5Gi        RWO            thin-disk      4m29s
redis-pv-claim   Bound    pvc-e5c5f9fe-1e69-4dd6-a4b4-81a01aacf6f2   2Gi        RWO            thin-disk      4m29s
```

Note: Only the yelb namespace and its resources have been restored

Step 8: Get the external ip of yelb-ui, point the browser to it and make sure all the data is visible in the application, and the application is reachable. Compare the voting data recorded in the table before the namespace backup was taken.

kubectl get svc -n yelb

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n yelb
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
redis-server     ClusterIP      10.100.200.135   <none>        6379/TCP       47s
yelb-appserver   ClusterIP      10.100.200.52    <none>        4567/TCP       47s
yelb-db          ClusterIP      10.100.200.247   <none>        5432/TCP       47s
yelb-ui          LoadBalancer   10.100.200.146   10.40.14.48   80:32015/TCP   47s
```


10.40.14.48
HOL Admin
RegionA
NSX Manager
Ops Manager
Harbor
vRealize Log Insight
Concourse
Kubernetes Dashboa
Lab Guides
EPMC
Yelb
Welcome to Yelb, the only hub for healthy food recommendations!
IHOP
Pancakes, for a powerful start!
VOTE
Chipotle
Burritos, for a mid-day break!
VOTE
Outback
Blooming onion, what else?
VOTE
Buca di Beppo
Lasagne, a great taste directly from Italy
VOTE
38
Total
12
outback
32%
8
bucadibeppo
21%
14
ihop
37%
4
chipotle
11%
Number of page views so far: 1
System serving the request: yelb-appserver-696b9668c4-nblp4

Step 9: Delete the yelb namespace which will delete the application and the PV:

kubectl delete ns yelb

kubectl get ns

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl delete ns yelb
namespace "yelb" deleted
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS        AGE
default           Active        33d
kube-node-lease   Active        33d
kube-public       Active        33d
kube-system       Active        33d
pks-system        Active        33d
spc               Terminating   32d
velero            Active        10d
```

Restore Cluster Backup

Step 10: Restore the back of all resources from the source cluster to the target cluster:

cd ~/velero/velero-v1.4.0-linux-amd64

./velero restore create --from-backup sourceclusterbk

In the above example for vsphere plugin the namespace backup was named yelb-vspheresnap-bkp

./velero restore create --from-backup srccluster-snap-backup

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero restore create --from-backup sourceclusterbk
Restore request "sourceclusterbk-20200601213029" submitted successfully.
Run `velero restore describe sourceclusterbk-20200601213029` or `velero restore logs sourceclusterbk-20200601213029` for more details.
```

Step 11: Monitor the resources created in the target cluster. The yelb , x1, y1 and z1 namespaces should be created. Pods,pv's, deployments and services should also be created.

kubectl get ns

kubectl get po --all-namespaces

kubectl get pvc --all-namespaces

kubectl get svc --all-namespaces

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS        AGE
default           Active        33d
kube-node-lease   Active        33d
kube-public       Active        33d
kube-system       Active        33d
pks-system        Active        33d
spc               Terminating   32d
velero            Active        10d
x1                Active        104s
y1                Active        102s
yelb              Active        106s
z1                Active        104s
```

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get pvc --all-namespaces
NAMESPACE   NAME              STATUS        VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
spc         database-server   Terminating   pvc-7c5d2222-389d-4dfd-9187-a52d4db37a06   8Gi        RWO            thin-disk      32d
yelb        db-pv-claim       Bound         pvc-74a65359-4a58-4087-a79e-01d67598197e   5Gi        RWO            thin-disk      4m35s
yelb        redis-pv-claim    Bound         pvc-bcfdb7eb-78e1-400d-b58c-3e954b8dd9ad   2Gi        RWO            thin-disk      4m35s
```

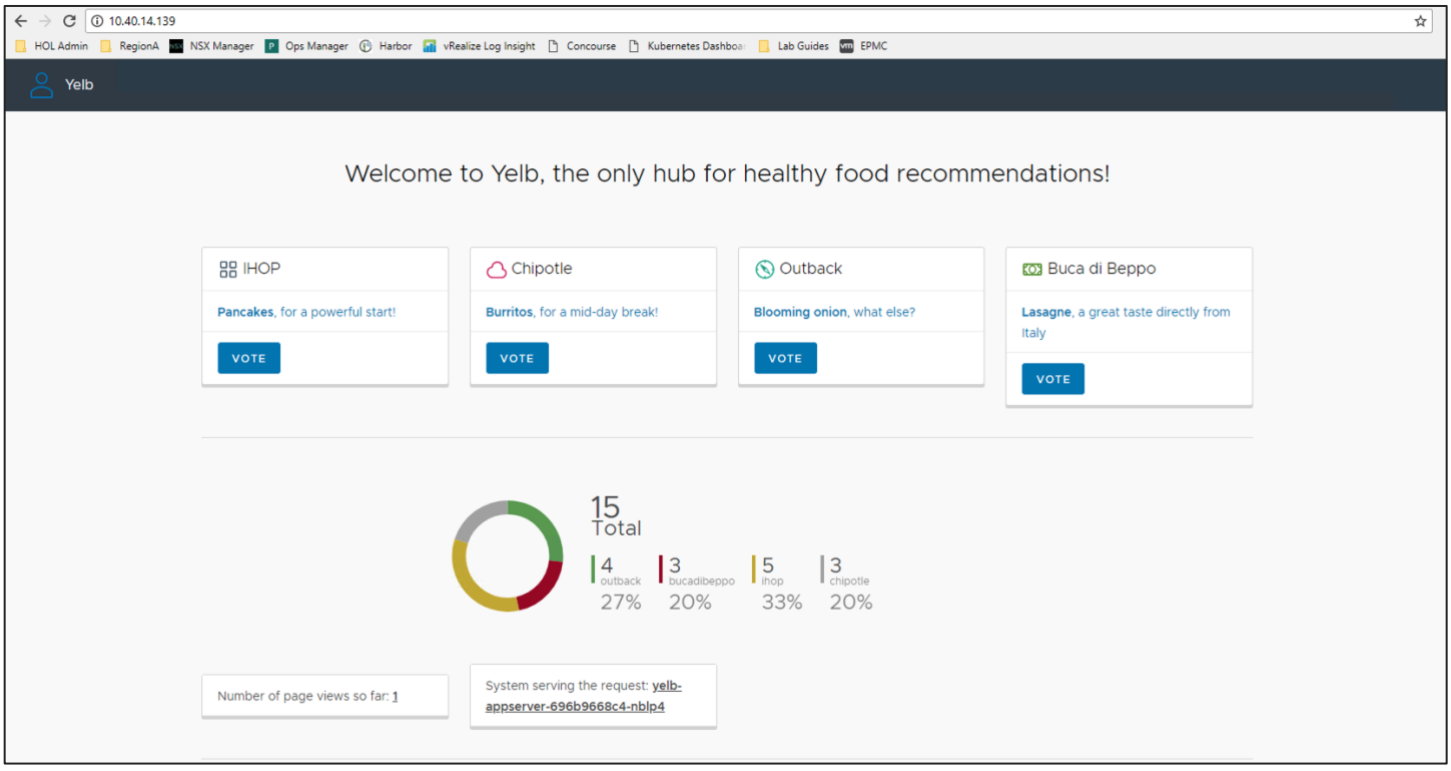
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc --all-namespaces
NAMESPACE     NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)          AGE
default       kubernetes             ClusterIP      10.100.200.1     <none>          443/TCP          33d
kube-system   kube-dns               ClusterIP      10.100.200.2     <none>          53/UDP,53/TCP    33d
kube-system   kubernetes-dashboard   NodePort       10.100.200.95    <none>          443:31379/TCP    33d
kube-system   metrics-server         ClusterIP      10.100.200.6     <none>          443/TCP          33d
pks-system    fluent-bit             ClusterIP      10.100.200.80    <none>          24224/TCP        33d
pks-system    validator              ClusterIP      10.100.200.207   <none>          443/TCP          33d
x1            service-a-lb           LoadBalancer   10.100.200.147   10.40.14.138    80:32641/TCP     9m11s
x1            svc-service-a          ClusterIP      10.100.200.22    <none>          80/TCP           9m11s
y1            svc-service-b          ClusterIP      10.100.200.138   <none>          80/TCP           9m11s
yelb          redis-server           ClusterIP      10.100.200.4     <none>          6379/TCP         9m11s
yelb          yelb-appserver         ClusterIP      10.100.200.5     <none>          4567/TCP         9m11s
yelb          yelb-db                ClusterIP      10.100.200.133   <none>          5432/TCP         9m11s
yelb          yelb-ui                LoadBalancer   10.100.200.33    10.40.14.139    80:30859/TCP     9m11s
z1            svc-service-c          ClusterIP      10.100.200.109   <none>          80/TCP           9m13s
z1            svc-service-d          ClusterIP      10.100.200.112   <none>          80/TCP           9m13s
```

Step 12: Get the external ip for the yelb-ui app

kubectl get svc -n yelb

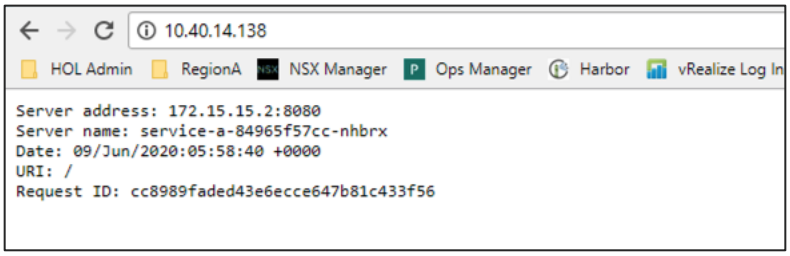
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n yelb
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
redis-server     ClusterIP      10.100.200.4     <none>         6379/TCP       9m46s
yelb-appserver   ClusterIP      10.100.200.5     <none>         4567/TCP       9m46s
yelb-db          ClusterIP      10.100.200.133   <none>         5432/TCP       9m46s
yelb-ui          LoadBalancer   10.100.200.33    10.40.14.139   80:30859/TCP   9m46s
```

Step 13: Point the browser to it and make sure all the data is visible in the application and the application is reachable. Compare the voting data recorded in the table before the cluster backup was taken:



Step 14: The x1, y1 and z1 namespaces run nginx pods with busybox. The nginx pod running in namespace x1 is exposed as a load balancer. Get the external ip to this pod and point the browser to access the service:

```
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
service-a-lb     LoadBalancer   10.100.200.179   10.40.14.138    80:32017/TCP   4m17s
svc-service-a    ClusterIP      10.100.200.97    <none>          80/TCP         4m16s
```



Note: With the cluster restore all resources , including namespaces, pv's etc are restored

From a cluster backup you could selectively restore the resources that are required to migrate as well.

Restore a Point in Time Backup (Scheduled Backup)

Step 14: Delete the yelb, x1 , y1 and z1 namespaces which will delete the application and the associated PV's and other resources:

kubectl delete ns yelb

kubectl delete ns x1

kubectl delete ns y1

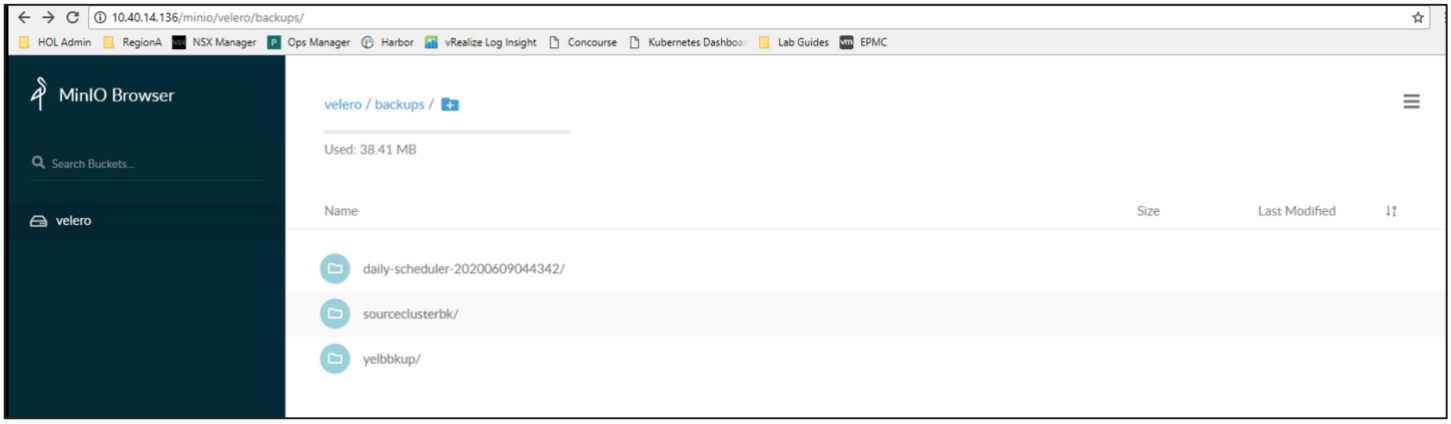
kubectl delete ns z1

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl delete ns yelb
namespace "yelb" deleted
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl delete ns x1
namespace "x1" deleted
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl delete ns y1
namespace "y1" deleted
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl delete ns z1
namespace "z1" deleted
```

kubectl get ns

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get ns
NAME              STATUS        AGE
default           Active        34d
kube-node-lease   Active        34d
kube-public       Active        34d
kube-system       Active        34d
pks-system        Active        34d
spc               Terminating   32d
velero            Active        30m
```

Step 15: Login to Minio, and copy the name of the backup that was created when a backup scheduler was created. Eg. daily-scheduler-20200609044342:



Step 15: Restore the point in time backup to the cluster:

cd ~/velero/velero-v1.4.0-linux-amd64

./velero restore create --from-backup daily-scheduler-20200609044342

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ ./velero restore create --from-backup daily-scheduler-20200609044342
Restore request "daily-scheduler-20200609044342-20200609061417" submitted successfully.
Run `velero restore describe daily-scheduler-20200609044342-20200609061417` or `velero restore logs daily-scheduler-20200609044342-20200609061417` for more details.
```

Step 16: Check the status of the restore:

./velero restore describe daily-scheduler-20200609044342-20200609061417

Step 17: Monitor the resources created in the target cluster. The yelb , x1, y1 and z1 namespaces should be created. Pods,pv's, deployments and services should also be created.

kubectl get ns

kubectl get po --all-namespaces

kubectl get pvc --all-namespaces

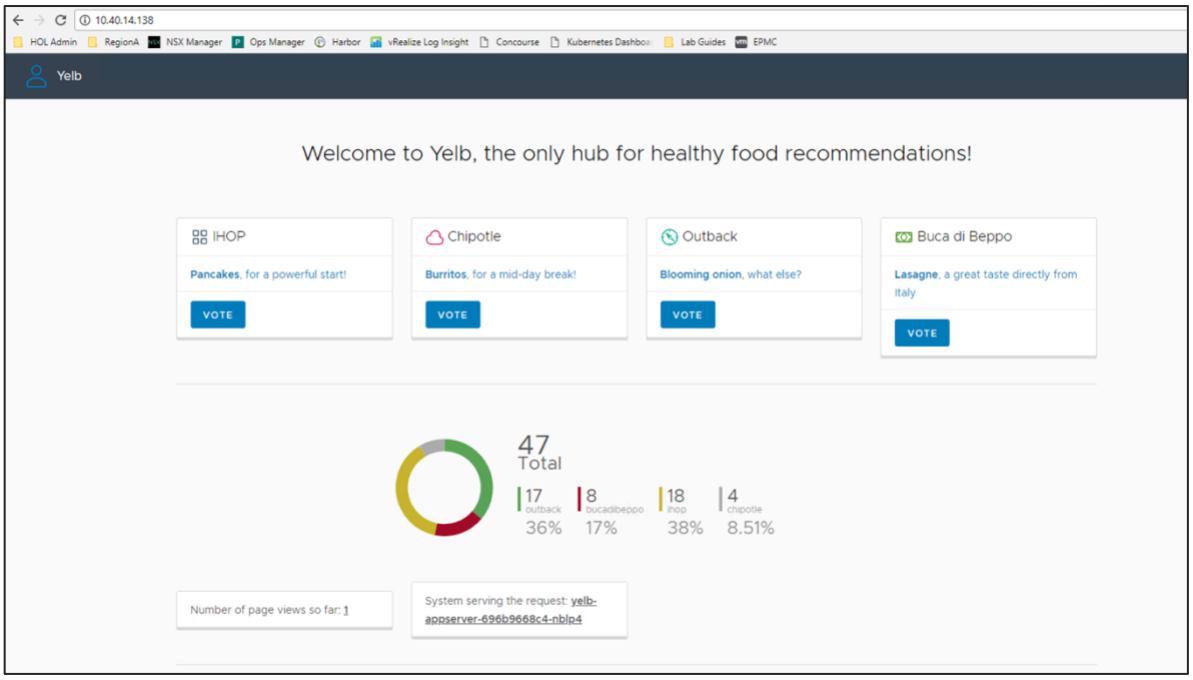
kubectl get svc --all-namespaces

Step 18: Get the external ip for the yelb-ui app

kubectl get svc -n yelb

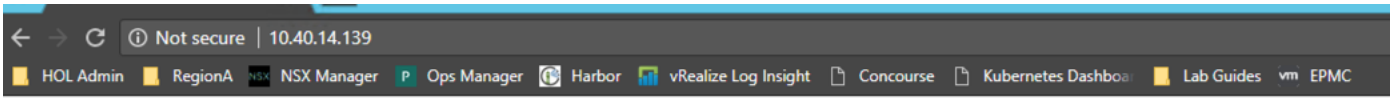
```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n yelb
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
redis-server     ClusterIP      10.100.200.110   <none>          6379/TCP       3m8s
yelb-appserver   ClusterIP      10.100.200.206   <none>          4567/TCP       3m8s
yelb-db          ClusterIP      10.100.200.157   <none>          5432/TCP       3m7s
yelb-ui          LoadBalancer   10.100.200.75    10.40.14.138    80:31757/TCP   3m7s
```

Step 19: Point the browser to it,make sure all the data is visible in the application, and the application is reachable. Compare the voting data recorded in the table before the cluster backup was taken.



Step 20: The x1, y1 and z1 namespaces run nginx pods with busybox. The nginx pod running in namespace x1 is exposed as a load balancer. Get the external ip to this pod and point the browser to access the service.

```
ubuntu@cli-vm:~/velero/velero-v1.4.0-linux-amd64$ kubectl get svc -n x1
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
service-a-lb   LoadBalancer   10.100.200.216   10.40.14.139   80:31705/TCP   5m28s
svc-service-a  ClusterIP      10.100.200.126   <none>         80/TCP         5m28s
```




```
Server address: 172.15.8.3:8080
Server name: service-a-84965f57cc-nhbrx
Date: 09/Jun/2020:06:21:47 +0000
URI: /
Request ID: 98f40e22b022a1ddbb7ae3ffe5024947
```

Note: With the cluster restore all resources , including namespaces, pv's etc are restored from a backup in time.

Cleanup

Step 1: Get kube config for the source cluster

pks login -a <pks api> -u <user> -p <password> -k

pks get-credentials <cluster>

Alternatively

pks get-kubeconfig <cluster> -a <pks api> -u <user> -p <password> -k

E.g.

pks login -a pks.corp.local -u riaz -p VMware1! -k

pks get-credentials ci-cluster

pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

```
ubuntu@cli-vm:~/velero$ pks get-kubeconfig ci-cluster -a pks.corp.local -u riaz -p VMware1! -k

Fetching kubeconfig for cluster ci-cluster and user riaz.
You can now use the kubeconfig for user riaz:
$kubectl config use-context ci-cluster
```

Step 2: Set kubectl context to the source cluster

kubectl config use-context <source-cluster>

E.g.

kubectl config use-context ci-cluster

Step 3: Get all backup's taken from of a specific cluster

```
./velero backup get
```

Step 4: To delete a specific backup

./velero backup delete <backupname>

E.g.
./velero backup delete sourceclusterbk

Step 4: To delete all backups

./velero backup delete --all

Step 5: To delete a backup schedule

./velero schedule delete <schedule-name>

Conclusion

We hope this document was useful. As you try these configuration steps, please provide any feedback or questions in the comments section for this document on code.vmware.com. Also, do let us know if you have any suggestions or if you would like to see guidance on other topics.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.