Backup and Restore Kubernetes Resources with Velero

Document Level Classification

200

VELERO

Introduction

Velero is an open source tool to safely backup and restore, perform disaster recovery, and migrate Kubernetes cluster resources and

persistent volumes. This solution was adopted by Epic since it's stable, free, open-source and officially recommended by AWS.

The primary goal of this tool is to provide a straightforward solution for all teams using Substrate to back up their Kubernetes resources, including persistent volumes. This approach accelerates disaster recovery, minimizes downtime, and enhances overall resilience.

Prerequisites

In order to operate these backups (describe them, restore, etc), any user with access to the Substrate AWS account must have Velero CLI installed. The setup is very simple and the tool is easy to understand. Click the tab below that matches your operating system for installation instructions.

Mosaic macros cannot be exported to this format.

Once this is installed in your OS, make sure to update the default namespace to the one used at Epic:

```
velero client config set namespace=kube-system
```

Current Setup

Velero will be enabled across all Substrate clusters at the end of Nov 2024, so Substrate users **will not need** to install it in their individual clusters.

The following backup capabilities are available:

- K8s resource backups (including all CRDs). This is enabled in all clusters by default as described in the respective section below.
- Volume Snapshots of existing Persistent Volumes via EBS native snapshots capability using Kubernetes CSI Snapshot APIs. This is disabled by default and can be enabled per volume as described in the respective section below.

Velero also offers FSB (File System Backup - not to be mistaken with the aforementioned Volume Snapshots) as explained here, but it's not recommended to be used in production as maintainers themselves describe this as beta. Make sure to read this page in order to understand the current status and limitations.

K8s Resource Backups

Currently, Velero takes **daily backups** of all the resources (including CRDs) in all namespaces inside each K8s cluster. This is the default schedule (literally called `default`) which should fit most teams needs, but this or additional schedules can be configured based on the specific team needs.

Customizing the schedule

If the schedule does not fit, please take a look at the `velero` section here and copy the `velero.values.schedules` block to your cluster config file e.g. if the cluster name is `dead-dev-dontwork`, a block like this (this may get outdated, always copy the block from the previous linked file!) would be added to cluster-configs/dead-dev-dontwork.yaml. This would be an example block overriding the `default` schedule:

```
velero:
  values:
    schedules:
      default:
        schedule: "0 0 * * *"  # midnight, every day
        useOwnerReferencesInBackup: false
        template:
          ttl: 720h  # 30 days, matches default S3 lifecycle rule
          storageLocation: substrate
          excludedNamespaceScopedResources:
            - persistentVolumeClaims
          excludedClusterScopedResources:
            - persistentVolumes
```

A PR is then opened and posted to #cloud-ops-support-ext to seek approval and sync with ArgoCD.

To find out all the possible properties, make sure to check values.yaml file from Velero (search for the `schedules` key), with further explanations in the Schedule API Type docs.

Important note about customizing the TTL: *While the TTL can be customized, in reality the backup will be kept for* ***30 days*** *in the S3 bucket in any case because 1) S3 versioning is enabled (so deleted backups are only really gone after the lifecycle rule is effective) and 2) the S3 lifecycle rule keeps all objects for 30 days. We believe that 30 days is the sweet spot for most of the teams for these kind of backups i.e. not PV/PVC objects. If your team requires a different, longer schedule, please reach out to #cloud-ops-support-ext.*

All AWS accounts have an AWS S3 bucket configured per cluster, which contains the backups of this kind. The cost for having this backup is minimal as it only backs up the resource definitions (which are very few MBs *maximum*) and shouldn't cause any concern to stewards.

It's also possible to create an **on-demand backup** outside of this schedule, please check the "Basic operations" section of this document.

Volume snapshots

Velero is able to create regular EBS Snapshots of any Persistent Volume inside a Substrate K8s cluster. By default, this is disabled since the costs would be too high to be enabled by default for all volumes, but it's very simple to enable it for any volume that requires backups.

In order to achieve this, Substrate's Velero has different schedules: `weekly`, `eod` (every other day), `daily` and `hourly` (please be mindful about this one, it's most likely not necessary). To assign a schedule to a Persistent Volume, simply add the label "`epic-velero.io/schedule: <chosen schedule>`" e.g. "`epic.velero.io/schedule: weekly`" to your Persistent Volume resources.

Each schedule have a different TTL. These can be customized upon request via #cloud-ops-support-ext (a guide to do this by teams themselves will be published later on). This table shows the current TTLs:

Schedule name	TTL
`weekly`	28 days
`eod`	21 days
`daily`	14 days
`hourly`	7 days

Basic operations

All commands described here assume that the user is logged in the Substrate AWS account and cluster with AOP.

Note that this section is a basic overview of Velero commands. While there are more to discover, in reality most of the other available options are already configured for each cluster and therefore Substrate users relevant actions can be summarized as backup get/describe/create/restore.

Official docs are here.

Always use Velero CLI

Velero has a lot of CRDs: `Backups`, `Restores`, `Schedules`, etc. While it is *ok* to query them directly, users should never interact with them directly. For example, if a `Backup` resource is deleted directly instead of using the Velero command, the CRD will disappear but the underlying resources will remain.

Retrieving backups/restores

To obtain a list of existing backups or restores, run:

```
# List backups
velero backup get
# List restores
velero restore get
```

Example (clipped) output of a `velero backup get`:

```
NAME                                                    STATUS      ERROR
velero-default-20241104000022                           Completed   0
velero-default-20241103000021                           Completed   0
velero-daily-pv-20241104000022                          Completed   0
velero-daily-pv-20241103000021                          Completed   0
velero-eod-pv-20241103000021                            Completed   0
velero-eod-pv-20241101000019                            Completed   0
velero-hourly-pv-20241104110022                         Completed   0
velero-hourly-pv-20241104100022                         Completed   0
```

Describing backups/restores

To describe a backup or restore, pass the reference obtained in the previous section to this command:

```
# Describe a backup
velero backup describe <backup name>
# Describe a restore
velero restore describe <restore name>
```

An example backup name is `velero-default-20241104000022`. This command is specially useful to see a backup/restore currently in process. Example output:

```
Name:         velero-default-20241104000022
Namespace:    kube-system
Labels:       app.kubernetes.io/instance=velero
              app.kubernetes.io/managed-by=Helm
```

```
              app.kubernetes.io/name=velero
              argocd.argoproj.io/appname=dead-dev-substrate-velero
              helm.sh/chart=velero-7.2.1
              velero.io/schedule-name=velero-default
              velero.io/storage-location=substrate
Annotations:  kubectl.kubernetes.io/last-applied-configuration={"apiVer

  velero.io/resource-timeout=10m0s
  velero.io/source-cluster-k8s-gitversion=v1.28.13-eks-a737599
  velero.io/source-cluster-k8s-major-version=1
  velero.io/source-cluster-k8s-minor-version=28+

Phase:  Completed


Warnings:
  ...
```

The flag `--details` will provide a very detailed description of all resources that are included.

Creating an on-demand backup

It's also possible to create an **on-demand backup** outside of the predefined schedule. To achieve this, an example command is:

```
velero backup create <backup name> --from-schedule velero-default
```

This would create a backup very shortly afterwards based on the `velero-default` schedule configuration, but there are many other arguments that can be used. Make sure to check the command help (`$ velero backup create -h`) for more information. Example output of the `velero backup create` command:

```
Creating backup from schedule, all other filters are ignored.
Backup request "ruben-test" submitted successfully.
Run `velero backup describe ruben-test` or `velero backup logs ruben-te
```

Restoring a backup

Same cluster as the backup source

No matter if this is an on-demand or a scheduled backup, it works the same way. The basic command is:

```
velero restore create --from-backup <backup name>
```

This would restore *all* resources contained in the backup, which is probably not what it's required. Arguments such as `--exclude-namespaces`/`--include-namespaces`, `--exclude-resources`/`--include-resources`, `-l` (label selector - that's a lowercase L) can be used to limit the scope of the restore. The restore job is shortly started after the command is run.

An example output after running `velero restore create --from-backup ruben-test --include-namespaces team-online-infra-platform --selector app=velero-test` (which will only restore the resources with the label "app=velero-test" from the namespace "team-online-infra-platform") is:

```
Restore request "ruben-test-20241104152814" submitted successfully.
Run `velero restore describe ruben-test-20241104152814` or `velero rest
```

If the restore pertains Persistent Volumes, you can read more details about it here and below that section.

Different cluster than the backup source

This process isn't as straightforward, but it's not that difficult. Let's assume there are two clusters in the same AWS account and the same

region: **cluster-1** and **cluster-2**. The first cluster has a backup called **special-backup** and the team wants to restore this backup to **cluster-2**.

When restoring resources from another cluster, please bear in mind that the resources are restored as-is, and some things (IAM policies, Service Accounts, aws-auth ConfigMap, etc) might not work right away as certain references have changed since the cluster is different. This guide assumes that the reader is aware of these details.

These would be the steps:

1. Extend the **cluster-2** permissions to be able to read the backup from **cluster-1**. The easiest way would be to locate both Velero IAM roles in the AWS account. The roles follow the nomenclature **velero-role-<cluster name>** e.g. **velero-role-cluster-1**. The easiest way would be taking a look into the **cluster-1** IAM role policies and append the resources used in the policies to **cluster-2**. Save the original **cluster-1** policy somewhere as it will need to be restored later.
2. In **cluster-2**, a new **backupstoragelocation** needs to be created, which would point to the. Both of these would be the same as in **cluster-1**. Therefore, the easiest way would be to copy the resources from **cluster-1** and create them in **cluster-2**. For this purpose, let's copy the definition of the current resource to the local machine:

   ```
   # Run this in cluster-1
   kubectl get backupstoragelocation substrate -o yaml -n kube-system
   ```

3. Then, modify the YAML file (or alternatively use kubectl-neat instead):
   1. Leave only **name** and **namespace** under metadata, updating the **name** to something that's not *substrate* (**substrate-temporal** is recommended and will be used throughout the guide).
   2. Remove **status**.
   3. Change **default** to **false** if applicable.

4. Change **accessMode** to **ReadOnly**.

```
# Resulting file after modifications example
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  name: substrate-temporal
  namespace: kube-system
spec:
  accessMode: ReadOnly
  default: false
  objectStorage:
    bucket: 012345678901-cluster-1-velero
  provider: aws
```

4. Apply the YAML file:

```
# Run this in cluster-2. If you are unable to do this, please ask i
kubectl apply -f backupstoragelocation.yaml
```

5. After a minute or so, the backups from **cluster-1** will be visible in **cluster-2**. Restore happens now the same way as described in the *Same cluster as the backup source* section since the backups are accessible now:

```
# Run this in cluster-2
velero restore create --from-backup <backup name>
```

This would restore *all* resources contained in the backup, which is probably not what it's required. Arguments such as `--exclude-namespaces` / `--include-namespaces`, `--exclude-resources` / `--include-resources`, `-l` (label selector - that's a lowercase L) can be used to limit the scope of the restore. An example command that would be more specific is `velero restore create --from-backup ruben-test --include-namespaces team-online-infra-platform --selector app=velero-test` (which will only restore the resources with

the label "app=velero-test" from the namespace "team-online-infra-platform"). The restore job is shortly started after the command is run.

As warned in the beginning, some resources may require modifications. It's possible to either download the backup (`velero backup download ...`) and modify them locally to later apply manually, or alternatively using restore resource modifiers to provide the modifications needed upon restore.

6. Once the restoration procedures are finished, make sure to delete the previously created **backupstoragelocation** to prevent confusion (unless this is explicitly needed):

```
# Run this in cluster-2
velero delete backup-locations substrate-temporal
```

7. Finally, restore the previous IAM policy that was saved in the first step.

Obtaining logs

Logs for backup and restore jobs can be quickly obtained by running:

```
# Display logs for a backup
velero backup logs <backup name>
# Display logs for a restore
velero restore logs <restore name>
```

An example (clipped) output from the restore job created before:

```
Name:         ruben-test-20241104152814
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Phase:                       Completed
```

```
Total items to be restored:  2
Items restored:              2

Started:    2024-11-04 15:28:15 +0100 CET
Completed:  2024-11-04 15:28:17 +0100 CET

Backup:  ruben-test

Namespaces:
  Included:  team-online-infra-platform
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        nodes, events, events.events.k8s.io, backups.velero.
  Cluster-scoped:  auto
...
```
