VELERO KUBERNETES BACKUP/RESTORE

Appunti di Pietro Cornelio

Rel. doc. 0.1.2 – 25.Marzo.2024
Rel. doc. 0.1.3 – 26.Marzo.2024
Rel. doc. 0.1.4 – 27.Marzo.2024
Rel. doc. 0.1.5 – 28.Marzo.2024

Prefazione

Alcune delle funzionalità che rendono Kubernetes un potente e versatile ambiente di distribuzione delle applicazioni containerizzate e di concertazione dei containers possono creare nuove sfide quando si tratta di gestire e proteggere i dati. L'implementazione del backup per le applicazioni in esecuzione su Kubernetes è fondamentale per proteggere i dati in caso di incidente, guasto del sistema o persino un attacco deliberato e rappresenta una problematica non banale.

Inoltre, gli ambienti Kubernetes sono spesso dinamici, con applicazioni e dati in continua evoluzione, ciò significa che i backup devono essere aggiornati per ripristinare i dati, in caso di disaster recovery. Kubernetes non ha alcuna soluzione di backup e ripristino integrata.

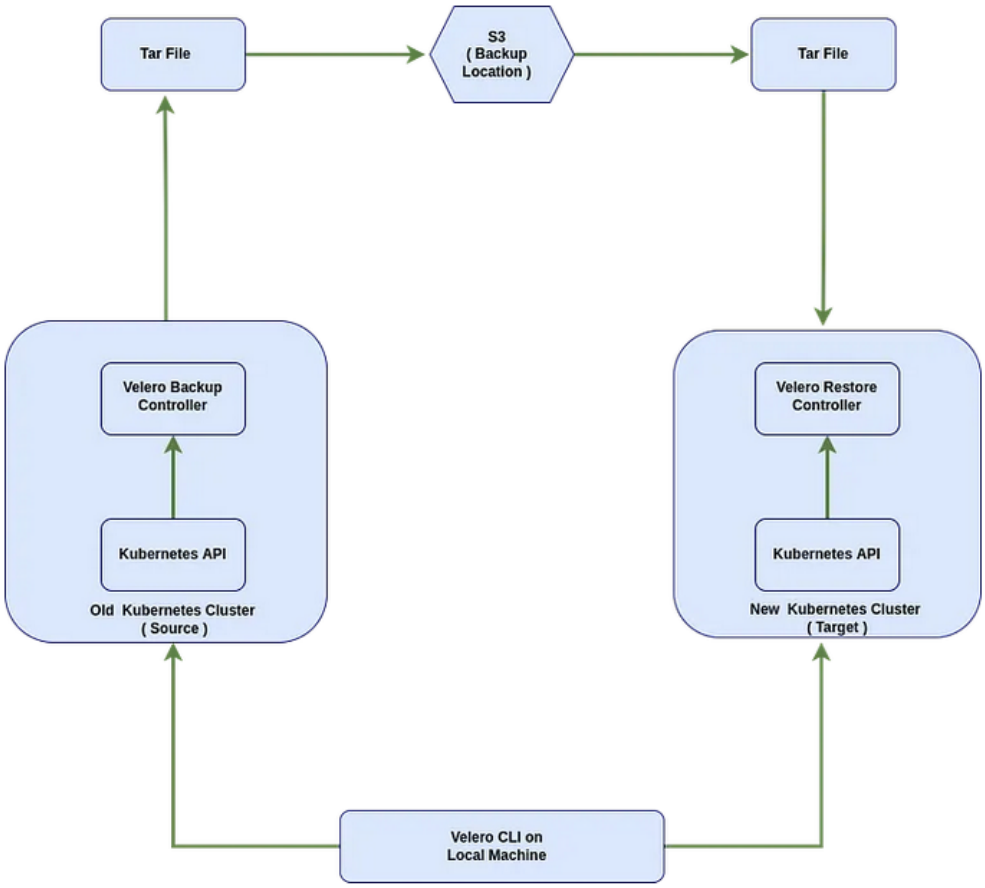

Figura 1- Architettura Velero

Introduzione

Velero è un software open source per il backup e il ripristino delle risorse in esecuzione su un cluster Kubernetes. Velero utilizza le funzionalità API di Kubernetes per raccogliere i dati da eseguire necessari al backup e restore degli oggetti. Velero può eseguire backup manuali e backup automatizzati, ripristino, disaster recovery, migrazione di risorse e PV da un cluster Kubernetes a un altro cluster Kubernetes.

In questo tutorial, vedremo come fare un backup delle risorse (oggetti) in esecuzione su un cluster Kubernetes di origine e come migrare quei backup al cluster di destinazione.

Nel diagramma di architettura di cui sopra, ci sono 2 cluster Kubernetes Sorgente e Destinazione. Il controller (agent) velero è distribuito su entrambi i cluster.

Cos'è il software Velero

Velero può affrontare e risolvere elegantemente la sfida del backup e ripristino delle risorse di Kubernetes. Può eseguire il backup e il ripristino in modo sicuro, eseguire il disaster recovery e migrare le risorse di un cluster di Kubernetes e anche i volumi persistenti.

Velero consiste in un processo server (agent) in esecuzione come distribuzione sul cluster Kubernetes e in un'interfaccia a riga di comando (CLI) in locale o remoto (multipiattaforma) con la quale i team possono configurare backup pianificati, attivare backup ad-hoc, eseguire ripristini e altro ancora. Velero CLI quindi è un piccolo software client standalone installato sulla macchina locale (MS Windows, Mac OS, GNU/Linux) o su un server remoto, che ha accesso (selezionando il **kube config** specifico) al cluster Kubernetes sorgente o al cluster Kubernetes destinazione.

Velero utilizza una posizione di backup per eseguire il backup e ripristinare il cluster. La posizione di backup può essere uno storage AWS S3, Azure Blob o qualsiasi storage compatibile con S3, i backup su S3 sono in formato tar.

Per questo tutorial, utilizzeremo AWS S3 per eseguire backup e ripristinare (restore) i dati del cluster.

https://velero.io/

A differenza di altri strumenti che accedono direttamente al database Kubernetes etcd per eseguire backup e ripristini, Velero utilizza l'API Kubernetes per acquisire lo stato delle risorse del cluster e ripristinarle quando necessario. Questo approccio basato su API ha una serie di vantaggi :

1. I backup possono acquisire sottoinsiemi delle risorse del cluster, filtrando per spazio dei nomi, tipo di risorsa e/o selettore di etichette, fornendo un elevato grado di flessibilità su ciò che viene di backup e ripristinato;
2. Gli utenti settati su Kubernetes spesso non hanno i privilegi di accesso al database sottostante etcd, quindi non sono possibili backup/ripristino diretti;
3. Le risorse esposte attraverso i server API aggregati possono essere facilmente ripristinate e anche se sono archiviate in un database etcd separato.

Inoltre, Velero consente di eseguire il backup e il ripristino dei dati persistenti delle applicazioni insieme alle loro configurazioni, utilizzando la funzionalità snapshot nativa della piattaforma di archiviazione o uno strumento di backup integrato a livello di file chiamato **restic**.

Velero può essere configurato per eseguire il backup dei dati persistenti senza necessariamente utilizzare Restic o gli snapshot. Puoi configurare Velero per eseguire il backup dei dati persistenti utilizzando altri metodi, come la copia diretta dei dati o la creazione di volumi di backup.

Come funziona il Velero?

Ogni operazione di Velero come il backup on-demand, il backup programmato, il ripristino ecc. è una risorsa personalizzata definita come risorsa personalizzata di Kubernetes (CRD) e archiviata. Velero include anche i controllori che elaborano le risorse personalizzate per eseguire backup, ripristini e tutte le operazioni correlate.

IMPORTANTE:

Per impostazione predefinita, Velero esegue un ripristino non distruttivo, il che significa che non eliminerà alcun dato sul cluster di destinazione. Se una risorsa nel backup esiste già nel cluster di destinazione, Velero salterà tale risorsa.

Installazione di Velero client (CLI)

Prima di installare Velero, bisogna avere il client Kubernetes kubectl perfettamente configurato su Windows o Linux o Mac OS. Accertiamoci che kubectl punti correttamente al cluster sorgente:

```
C:\Users\P.Cornelio>kubectl cluster-info

Kubernetes control plane is running at https://rancher-sac-
cert.assets.giottolabs.com/k8s/clusters/c-mwzxd
CoreDNS is running at https://rancher-sac-cert.assets.giottolabs.com/k8s/clusters/c-
mwzxd/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
C:\Users\P.Cornelio>kubectl get nodes

NAME                                          STATUS   ROLES    AGE     VERSION
ip-10-15-76-123.eu-west-1.compute.internal    Ready    <none>   3d17h   v1.27.9-eks-5e0fdde
ip-10-15-76-235.eu-west-1.compute.internal    Ready    <none>   3d17h   v1.27.9-eks-5e0fdde
ip-10-15-77-120.eu-west-1.compute.internal    Ready    <none>   3d17h   v1.27.9-eks-5e0fdde
ip-10-15-77-138.eu-west-1.compute.internal    Ready    <none>   43h     v1.27.9-eks-5e0fdde
ip-10-15-77-170.eu-west-1.compute.internal    Ready    <none>   3d17h   v1.27.9-eks-5e0fdde
```

Scaricare il client di Velero

Scaricare il client da: https://github.com/vmware-tanzu/velero/releases/tag/v1.13.1
Scompattare il client in una dir di comodo e settare il PATH di ricerca dei comandi nel proprio sistema operativo.
Verifichiamo la versione di Velero:

```
C:\Users\P.Cornelio>velero version --client-only
Client:
        Version: v1.13.1
        Git commit: ea5a89f83b89b2cb7a27f54148683c1ee8d57a37
```

Configurazione di Velero

Creare un file denominato velero-credentials nella directory di Velero e scrivere dentro al file i seguenti parametri per accedere al Bucker S3 che si vuole utilizzare per il backup

```
[default]
aws_access_key_id = AKXXXXXXXXXXXXXXXT
aws_secret_access_key = bXXXXXXXXXXXXXXXXXXXXz
```

Installazione dell'Agent Velero sul Cluster Kubernetes

Dal client Velero è possibile installare l'agent sfruttando kubectl fornendo una serie di opzioni:

In questo caso il bucket S3 (non su AWS ma su filebase) è il seguente:
`Nome bucket S3 = cornelius-bucket`
Indirizzo Server Storage S3 = https://s3.filebase.com
File delle credenziali Bucket S3 = `velero-credentials`

```
C:\Users\P.Cornelio>velero install --use-node-agent --provider aws --bucket cornelius-bucket --secret-file velero-credentials --use-volume-snapshots=false --plugins velero/velero-plugin-for-aws:v1.4.0 --backup-location-config region=us-east-1,s3ForcePathStyle="true",s3Url=https://s3.filebase.com
```

Segue output a video dell'installazione :

```
CustomResourceDefinition/backuprepositories.velero.io: attempting to create resource
CustomResourceDefinition/backuprepositories.velero.io: attempting to create resource client
CustomResourceDefinition/backuprepositories.velero.io: already exists, proceeding
CustomResourceDefinition/backuprepositories.velero.io: created
CustomResourceDefinition/backups.velero.io: attempting to create resource
CustomResourceDefinition/backups.velero.io: attempting to create resource client
CustomResourceDefinition/backups.velero.io: already exists, proceeding
CustomResourceDefinition/backups.velero.io: created
CustomResourceDefinition/backupstoragelocations.velero.io: attempting to create resource
CustomResourceDefinition/backupstoragelocations.velero.io: attempting to create resource client
CustomResourceDefinition/backupstoragelocations.velero.io: already exists, proceeding
CustomResourceDefinition/backupstoragelocations.velero.io: created
CustomResourceDefinition/deletebackuprequests.velero.io: attempting to create resource
CustomResourceDefinition/deletebackuprequests.velero.io: attempting to create resource client
CustomResourceDefinition/deletebackuprequests.velero.io: already exists, proceeding
CustomResourceDefinition/deletebackuprequests.velero.io: created
CustomResourceDefinition/downloadrequests.velero.io: attempting to create resource
CustomResourceDefinition/downloadrequests.velero.io: attempting to create resource client
CustomResourceDefinition/downloadrequests.velero.io: already exists, proceeding
CustomResourceDefinition/downloadrequests.velero.io: created
CustomResourceDefinition/podvolumebackups.velero.io: attempting to create resource
CustomResourceDefinition/podvolumebackups.velero.io: attempting to create resource client
CustomResourceDefinition/podvolumebackups.velero.io: already exists, proceeding
CustomResourceDefinition/podvolumebackups.velero.io: created
CustomResourceDefinition/podvolumerestores.velero.io: attempting to create resource
CustomResourceDefinition/podvolumerestores.velero.io: attempting to create resource client
CustomResourceDefinition/podvolumerestores.velero.io: already exists, proceeding
CustomResourceDefinition/podvolumerestores.velero.io: created
CustomResourceDefinition/restores.velero.io: attempting to create resource
CustomResourceDefinition/restores.velero.io: attempting to create resource client
CustomResourceDefinition/restores.velero.io: already exists, proceeding
CustomResourceDefinition/restores.velero.io: created
CustomResourceDefinition/schedules.velero.io: attempting to create resource
CustomResourceDefinition/schedules.velero.io: attempting to create resource client
CustomResourceDefinition/schedules.velero.io: already exists, proceeding
CustomResourceDefinition/schedules.velero.io: created
CustomResourceDefinition/serverstatusrequests.velero.io: attempting to create resource
CustomResourceDefinition/serverstatusrequests.velero.io: attempting to create resource client
CustomResourceDefinition/serverstatusrequests.velero.io: already exists, proceeding
CustomResourceDefinition/serverstatusrequests.velero.io: created
CustomResourceDefinition/volumesnapshotlocations.velero.io: attempting to create resource
CustomResourceDefinition/volumesnapshotlocations.velero.io: attempting to create resource client
CustomResourceDefinition/volumesnapshotlocations.velero.io: already exists, proceeding
CustomResourceDefinition/volumesnapshotlocations.velero.io: created
CustomResourceDefinition/datadownloads.velero.io: attempting to create resource
CustomResourceDefinition/datadownloads.velero.io: attempting to create resource client
CustomResourceDefinition/datadownloads.velero.io: already exists, proceeding
CustomResourceDefinition/datadownloads.velero.io: created
CustomResourceDefinition/datauploads.velero.io: attempting to create resource
```

```
CustomResourceDefinition/datauploads.velero.io: attempting to create resource client
CustomResourceDefinition/datauploads.velero.io: already exists, proceeding
CustomResourceDefinition/datauploads.velero.io: created
Waiting for resources to be ready in cluster...
Namespace/velero: attempting to create resource
Namespace/velero: attempting to create resource client
Namespace/velero: created
ClusterRoleBinding/velero: attempting to create resource
ClusterRoleBinding/velero: attempting to create resource client
ClusterRoleBinding/velero: created
ServiceAccount/velero: attempting to create resource
ServiceAccount/velero: attempting to create resource client
ServiceAccount/velero: created
Secret/cloud-credentials: attempting to create resource
Secret/cloud-credentials: attempting to create resource client
Secret/cloud-credentials: created
BackupStorageLocation/default: attempting to create resource
BackupStorageLocation/default: attempting to create resource client
BackupStorageLocation/default: created
Deployment/velero: attempting to create resource
Deployment/velero: attempting to create resource client
Deployment/velero: created
DaemonSet/node-agent: attempting to create resource
DaemonSet/node-agent: attempting to create resource client
DaemonSet/node-agent: created
Velero is installed! ⛵ Use 'kubectl logs deployment/velero -n velero' to view the status.
```

Verifichiamo lo status di Velero lato client e lato server:

```
C:\Users\P.Cornelio>velero version
Client:
        Version: v1.13.1
        Git commit: ea5a89f83b89b2cb7a27f54148683c1ee8d57a37
Server:
        Version: v1.13.1
```

Verifichiamo che la location del backup sia corretta, col comando:

```
C:\Users\P.Cornelio>velero backup-location get
NAME      PROVIDER   BUCKET/PREFIX      PHASE       LAST VALIDATED                  ACCESS MODE   DEFAULT
default   aws        cornelius-bucket   Available   2024-03-25 09:15:08 +0100 CET   ReadWrite     true
```

Verifichiamo che tutti i pod di Velero sono in esecuzione corretta

```
C:\Users\P.Cornelio>kubectl get all -n velero
```

Esempi di Backup

Quando eseguiamo il comando velero backup, verrà effettuata una chiamata al server API Kubernetes e verrà creato un backup di tutti gli oggetti come namespaces, storageclass, pv, pvc ecc. Il controller di backup interrogherà il server API per gli oggetti di cui eseguire il backup e quindi effettua una chiamata al bucket S3 dove il backup verrà salvato in formato tar.

Facciamo il backup dell'applicazione IOT su Kubernetes

```
C:\Users\P.Cornelio>velero backup create sac-app-iot --selector app=iot --wait

Backup request "sac-app-iot" submitted successfully.
Waiting for backup to complete. You may safely press ctrl-c to stop waiting - your
backup will continue in the background.
.........
Backup completed with status: Completed. You may check for more information using the
commands `velero backup describe sac-app-iot` and `velero backup logs sac-app-iot`.
```

L'opzione --wait dice a Velero di non uscire dall'esecuzione (che rimarrebbe comunque in backgroud sul cluster K8S) ma di attendere finchè l'attività non è conclusa.

Verifichiamo il backup

Verifichiamo il backup appena fatto

`C:\Users\P.Cornelio>velero backup get` (elenca tutti i backup presenti sullo storage)

```
C:\Users\P.Cornelio\.kube>velero backup get
NAME                        STATUS      ERRORS   WARNINGS   CREATED                         EXPIRES   STORAGE LOCATION   SELECTOR
bck-namespace-iot           Completed   0        0          2024-03-25 09:33:55 +0100 CET   27d       default            <none>
bck-namespace-rabbitmq      Completed   0        0          2024-03-27 16:58:20 +0100 CET   29d       default            <none>
sac-app-iot                 Completed   0        0          2024-03-25 09:22:25 +0100 CET   27d       default            app=iot
sac-backup-tutti-namespace  Completed   0        0          2024-03-25 10:39:34 +0100 CET   27d       default            <none>
sac-backup-tutto            Completed   0        0          2024-03-25 11:52:06 +0100 CET   27d       default            <none>
```

Oppure con il describe del backup specifico

`C:\Users\P.Cornelio>velero backup describe sac-app-iot`

```
Name:         sac-app-iot
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/resource-timeout=10m0s
              velero.io/source-cluster-k8s-gitversion=v1.27.10-eks-508b6b3
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=27+
Phase:  Completed
Namespaces:
  Included:  *
  Excluded:  <none>
Resources:
  Included:        *
  Excluded:        <none>
  Cluster-scoped:  auto
Label selector:  app=iot
Or label selector:  <none>
Storage Location:  default
Velero-Native Snapshot PVs:  auto
Snapshot Move Data:          false
Data Mover:                  velero
TTL:  720h0m0s
CSISnapshotTimeout:    10m0s
ItemOperationTimeout:  4h0m0s
Hooks:  <none>
Backup Format Version:  1.1.0
Started:    2024-03-25 09:22:25 +0100 CET
Completed:  2024-03-25 09:22:34 +0100 CET
Expiration:  2024-04-24 10:22:25 +0200 CEST
Total items to be backed up:  23
Items backed up:              23
Backup Volumes:
  Velero-Native Snapshots: <none included>
  CSI Snapshots: <none included>
  Pod Volume Backups: <none included>

HooksAttempted:  0
HooksFailed:     0
```

Notare che esiste una data di scadenza del backup fissata di default nel **TTL = 720h0m0s**

In Velero, la data di scadenza (expiration) indica il momento in cui un backup o un snapshot non saranno più conservati sullo storage.

Scadenza dei backup

Esistono due tipi di date di scadenza:

1. Data di scadenza del backup (Backup Expiration):
- Si applica ai backup completi e incrementali.
- Determina quando il backup stesso verrà eliminato.
- È impostata per impostazione predefinita su 30 giorni, ma può essere modificata.

2. Data di scadenza dello snapshot (Snapshot Expiration):
- Si applica solo agli snapshot.
- Determina quando i dati contenuti nello snapshot verranno eliminati.
- È indipendente dalla data di scadenza del backup da cui è stato creato lo snapshot.
- Può essere impostata manualmente o automaticamente in base a una policy.

3. Cosa succede quando un backup o uno snapshot scade?
- I backup scaduti vengono automaticamente eliminati da Velero.
- Per gli snapshot scaduti, i dati contenuti nello snapshot vengono eliminati, ma lo snapshot stesso rimane presente.

4. Perché impostare una data di scadenza?
- Consente di risparmiare spazio di archiviazione eliminando i backup e gli snapshot non più necessari.
- Può aiutare a migliorare le prestazioni del backup riducendo il numero di backup da gestire.

5. Come impostare la data di scadenza?
- La data di scadenza del backup può essere impostata durante la creazione del backup o modificata in seguito.
- La data di scadenza dello snapshot può essere impostata manualmente o automaticamente in base a una policy.

Inclusione dei PVC nei backup

Allora, i dati dei **Persistent Volume Claim (PVC)** vengono inclusi nei backup di Velero per impostazione predefinita. Velero utilizza un'opzione chiamata --include-pvcs per includere i PVC nei backup. Questa opzione è attivata per impostazione predefinita, il che significa che tutti i PVC presenti nel cluster Kubernetes al momento del backup saranno inclusi.

Esistono però alcune eccezioni:

- PVC con la specifica annotations.velero.io/exclude-backup: "true": I PVC con questa annotazione non saranno inclusi nei backup.
- PVC con la specifica volumeMode: Ephemeral: I PVC con questa specifica sono volatili e non vengono conservati sul disco, **quindi non è necessario includerli nei backup**.

È importante notare che includere i PVC nei backup può aumentare significativamente le dimensioni del backup. Se si desidera ridurre le dimensioni del backup, è possibile escludere i PVC non necessari utilizzando l'opzione --exclude-pvcs.

Ecco alcuni esempi di come utilizzare l'opzione --exclude-pvcs:
Per escludere tutti i PVC con un nome che inizia con "data-":

```
velero backup create my-backup --exclude-pvcs "data-*"
```

Per escludere un singolo PVC con nome "my-pvc":

```
velero backup create my-backup --exclude-pvcs "my-pvc"
```

Come cancellare un backup sullo storage

```
C:\Users\P.Cornelio\.kube>velero backup delete bck-namespace-rabbitmq
Are you sure you want to continue (Y/N)? y
Request to delete backup "bck-namespace-rabbitmq" submitted successfully.
The backup will be fully deleted after all associated data (disk snapshots, backup files,
restores) are removed.
```

Backup di un intero namespace

Proviamo ora a fare un backup di un intero namespace compresi tutti i suoi oggetti e applicazioni.

Notare, questa volta lanciamo il backup senza l'opzione --wait

```
C:\Users\P.Cornelio>velero backup create bck-namespace-iot --include-namespaces iot

Backup request "bck-namespace-iot" submitted successfully.
```

Possiamo verificare il backup in corso con il seguente comando:

```
C:\Users\P.Cornelio>velero backup logs bck-namespace-iot

time="2024-03-25T08:33:55Z" level=info msg="Setting up backup temp file" backup=velero/bck-
namespace-iot logSource="pkg/controller/backup_controller.go:620"
time="2024-03-25T08:33:55Z" level=info msg="Setting up plugin manager" backup=velero/bck-
namespace-iot logSource="pkg/controller/backup_controller.go:627"
time="2024-03-25T08:33:55Z" level=info msg="Getting backup item actions" backup=velero/bck-
namespace-iot logSource="pkg/controller/backup_controller.go:631"
time="2024-03-25T08:33:55Z" level=info msg="Setting up backup store to check for backup
existence" backup=velero/bck-namespace-iot logSource="pkg/controller/backup_controller.go:636"
time="2024-03-25T08:33:55Z" level=info msg="Writing backup version file" backup=velero/bck-
namespace-iot logSource="pkg/backup/backup.go:197"
time="2024-03-25T08:33:55Z" level=info msg="Including namespaces: iot" backup=velero/bck-
namespace-iot logSource="pkg/backup/backup.go:203"
time="2024-03-25T08:33:55Z" level=info msg="Excluding namespaces: <none>" backup=velero/bck-
namespace-iot logSource="pkg/backup/backup.go:204"
time="2024-03-25T08:33:55Z" level=info msg="Including resources: *" backup=velero/bck-namespace-
iot logSource="pkg/util/collections/includes_excludes.go:506"
time="2024-03-25T08:33:55Z" level=info msg="Excluding resources: <none>" backup=velero/bck-
namespace-iot logSource="pkg/util/collections/includes_excludes.go:507"
time="2024-03-25T08:33:55Z" level=info msg="Backing up all volumes using pod volume backup:
false" backup=velero/bck-namespace-iot logSource="pkg/backup/backup.go:222"
time="2024-03-25T08:33:55Z" level=info msg="Getting items for group" backup=velero/bck-namespace-
iot group=v1 logSource="pkg/backup/item_collector.go:105"
time="2024-03-25T08:33:55Z" level=info msg="Getting items for resource" backup=velero/bck-
namespace-iot group=v1 logSource="pkg/backup/item_collector.go:196" resource=pods
time="2024-03-25T08:33:55Z" level=info msg="Listing items" backup=velero/bck-namespace-iot
group=v1 logSource="pkg/backup/item_collector.go:323" namespace=iot resource=pods
time="2024-03-25T08:33:55Z" level=info msg="list for groupResource pods was not paginated"
backup=velero/bck-namespace-iot logSource="pkg/backup/item_collector.go:496"
time="2024-03-25T08:33:55Z" level=info msg="Retrieved 28 items" backup=velero/bck-namespace-iot
group=v1 logSource="pkg/backup/item_collector.go:354" namespace=iot resource=pods
time="2024-03-25T08:33:55Z" level=info msg="Getting items for resource" backup=velero/bck-
namespace-iot group=v1 logSource="pkg/backup/item_collector.go:196"
resource=persistentvolumeclaims ...

etc.
```

Backup di tutto il cluster k8s

C:\Users\P.Cornelio>velero backup create all-sac-k8s –wait

Backup request "all-sac-k8s" submitted successfully.
Waiting for backup to complete. You may safely press ctrl-c to stop waiting - your backup will continue in the background.
.....
Backup completed with status: Completed. You may check for more information using the commands `velero backup describe all-sac-k8s` and `velero backup logs all-sac-k8s`.

Verifica backup

```
C:\Users\P.Cornelio>velero backup describe all-sac-k8s

Name:         all-sac-k8s
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/resource-timeout=10m0s
              velero.io/source-cluster-k8s-gitversion=v1.27.10-eks-508b6b3
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=27+
Phase:  Completed

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

Label selector:  <none>
Or label selector:  <none>
Storage Location:  default

Velero-Native Snapshot PVs:  auto
Snapshot Move Data:          false
Data Mover:                  velero

TTL:  720h0m0s

CSISnapshotTimeout:    10m0s
ItemOperationTimeout:  4h0m0s
Hooks:  <none>
Backup Format Version:  1.1.0

Started:    2024-03-25 10:30:02 +0100 CET
Completed:  2024-03-25 10:30:34 +0100 CET

Expiration:  2024-04-24 11:30:02 +0200 CEST
Total items to be backed up:  2103
Items backed up:              2103

Backup Volumes:
  Velero-Native Snapshots: <none included>
  CSI Snapshots: <none included>
  Pod Volume Backups: <none included>

HooksAttempted:  0
HooksFailed:     0
```

Notare le opzioni Namespaces e Resources sopra (evidenziate in giallo) con l'asterisco, il che significa includere tutto.

Restore

Vediamo come effettuare un restore di un namespace, di un'app su un cluster Kubernetes di destinazione

Importante riguardo le storage class:

Quando esegui un backup con Velero su un cluster Kubernetes, le storage class non vengono automaticamente create sul cluster di destinazione. Velero si occupa principalmente di fare il backup e il ripristino delle risorse Kubernetes, come i pod, i deployment, i service, i PersistentVolumeClaim (PVC) e i PersistentVolume (PV), ma non si estende alla creazione di risorse di infrastruttura come le storage class.

Velero ripristina i dati e le risorse dal backup, inclusi i **PersistentVolumeClaim (PVC)** e i **PersistentVolume (PV)** associati. Se i PVC e i PV nel backup sono associati a una storage class che non esiste ancora nel cluster di destinazione, il ripristino dei PVC e dei PV non avrà successo fino a quando non viene creata la storage class corrispondente manualmente. Pertanto, prima di eseguire un ripristino (restore) con Velero, bisogna assicurarsi che le storage class necessarie siano già presenti nel cluster di destinazione. Se le storage class nel backup non corrispondono a quelle disponibili nel cluster di destinazione, potrebbe essere necessario aggiornare i PVC e i PV nel backup per utilizzare le storage class disponibili prima di eseguire il ripristino.

Collegarsi al cluster K8S di destinazione

Abbiamo 2 possibilità per passare da un kubeconfig a un altro, ma noi useremo la più sicura cioè quella di sovrascrivere il file config; l'altra opzione è quella di usare l'opzione di Velero --kubeconfig=/path/to/my-kubeconfig.yaml ma è facile sbagliare puntando al cluster K8S errato.

Sovrascrivere il file **.kube\config** con il file yaml del cluster di destinazione ad esempio :

```
C:\Users\P.Cornelio\.kube>dir

Il volume nell'unità C non ha etichetta.
Numero di serie del volume: FECA-3DF1
Directory di C:\Users\P.Cornelio\.kube

25/03/2024  13:58    <DIR>          .
25/03/2024  09:14    <DIR>          ..
14/02/2024  15:17             1.436 ambienti-esteri.yaml
22/03/2024  10:55    <DIR>          cache
25/03/2024  13:57             4.974 cluster-be.yaml
22/03/2024  10:56        12.164.955 cluster-k8s-sac.txt
08/03/2024  10:02               396 config
26/02/2024  15:39             4.592 local.yaml
08/03/2024  10:02               396 sac-cert.yaml
               6 File     12.176.749 byte
               3 Directory  265.191.727.104 byte disponibili

C:\Users\P.Cornelio\.kube>
C:\Users\P.Cornelio\.kube>cp cluster-be.yaml config
```

Nota: su sistemi GNU/Linux la procedura è identica, cambiano solo i comandi e la sintassi di path.

Mentre prima da kubectl puntavamo a **sac-cert.yaml** ora puntiamo **a cluster-be.yaml**

Verifichiamo col seguente comando le info del cluster K8S di destinazione:

```
C:\Users\P.Cornelio>kubectl cluster-info
Kubernetes control plane is running at https://rancher.almaviva.lab/k8s/clusters/c-m-97stwr7n
CoreDNS is running at https://rancher.almaviva.lab/k8s/clusters/c-m-97stwr7n/api/v1/namespaces/kube-system/services/rke2-coredns-rke2-coredns:udp-53/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Verifichiamo anche i nodi, meglio assicurarci vediamo i nodi corretti 😉

```
C:\Users\P.Cornelio>kubectl get nodes

NAME                STATUS   ROLES                       AGE    VERSION
ingressbe01-rke2    Ready    worker                      403d   v1.24.9+rke2r2
ingressbe02-rke2    Ready    worker                      403d   v1.24.9+rke2r2
ingressbe03-rke2    Ready    worker                      139d   v1.24.9+rke2r2
managerbe01-rke2    Ready    control-plane,etcd,master   403d   v1.24.9+rke2r2
managerbe02-rke2    Ready    control-plane,etcd,master   403d   v1.24.9+rke2r2
managerbe03-rke2    Ready    control-plane,etcd,master   193d   v1.24.9+rke2r2
workerbe01-rke2     Ready    worker                      403d   v1.24.9+rke2r2
workerbe02-rke2     Ready    worker                      403d   v1.24.9+rke2r2
workerbe03-rke2     Ready    worker                      139d   v1.24.9+rke2r2
```

Installazione agent Velero sul cluster K8S di destinazione

Bene, ora passiamo a installare l'agent sul cluster di destinazione che ci permetterà di effettuare il restore dei backup fatti in precedenza. L'installazione è identica al cluster K8S sorgente:

```
C:\Users\P.Cornelio>velero install --use-node-agent --provider aws --bucket cornelius-bucket --secret-file velero-credentials --use-volume-snapshots=false --plugins velero/velero-plugin-for-aws:v1.4.0 --backup-location-config region=us-east-1,s3ForcePathStyle="true",s3Url=https://s3.filebase.com
```

Output del comando:

```
CustomResourceDefinition/backuprepositories.velero.io: attempting to create resource
CustomResourceDefinition/backuprepositories.velero.io: attempting to create resource client
CustomResourceDefinition/backuprepositories.velero.io: created
CustomResourceDefinition/backups.velero.io: attempting to create resource
CustomResourceDefinition/backups.velero.io: attempting to create resource client
CustomResourceDefinition/backups.velero.io: created
CustomResourceDefinition/backupstoragelocations.velero.io: attempting to create resource
CustomResourceDefinition/backupstoragelocations.velero.io: attempting to create resource client
CustomResourceDefinition/backupstoragelocations.velero.io: created
CustomResourceDefinition/deletebackuprequests.velero.io: attempting to create resource
CustomResourceDefinition/deletebackuprequests.velero.io: attempting to create resource client
CustomResourceDefinition/deletebackuprequests.velero.io: created
CustomResourceDefinition/downloadrequests.velero.io: attempting to create resource
CustomResourceDefinition/downloadrequests.velero.io: attempting to create resource client
CustomResourceDefinition/downloadrequests.velero.io: created
CustomResourceDefinition/podvolumebackups.velero.io: attempting to create resource
CustomResourceDefinition/podvolumebackups.velero.io: attempting to create resource client
CustomResourceDefinition/podvolumebackups.velero.io: created
CustomResourceDefinition/podvolumerestores.velero.io: attempting to create resource
CustomResourceDefinition/podvolumerestores.velero.io: attempting to create resource client
CustomResourceDefinition/podvolumerestores.velero.io: created
CustomResourceDefinition/restores.velero.io: attempting to create resource
CustomResourceDefinition/restores.velero.io: attempting to create resource client
CustomResourceDefinition/restores.velero.io: created
CustomResourceDefinition/schedules.velero.io: attempting to create resource
CustomResourceDefinition/schedules.velero.io: attempting to create resource client
CustomResourceDefinition/schedules.velero.io: created
CustomResourceDefinition/serverstatusrequests.velero.io: attempting to create resource
CustomResourceDefinition/serverstatusrequests.velero.io: attempting to create resource client
CustomResourceDefinition/serverstatusrequests.velero.io: created
CustomResourceDefinition/volumesnapshotlocations.velero.io: attempting to create resource
CustomResourceDefinition/volumesnapshotlocations.velero.io: attempting to create resource client
CustomResourceDefinition/volumesnapshotlocations.velero.io: created
CustomResourceDefinition/datadownloads.velero.io: attempting to create resource
CustomResourceDefinition/datadownloads.velero.io: attempting to create resource client
CustomResourceDefinition/datadownloads.velero.io: created
CustomResourceDefinition/datauploads.velero.io: attempting to create resource
CustomResourceDefinition/datauploads.velero.io: attempting to create resource client
CustomResourceDefinition/datauploads.velero.io: created
```

```
Waiting for resources to be ready in cluster...
Namespace/velero: attempting to create resource
Namespace/velero: attempting to create resource client
Namespace/velero: created
ClusterRoleBinding/velero: attempting to create resource
ClusterRoleBinding/velero: attempting to create resource client
ClusterRoleBinding/velero: created
ServiceAccount/velero: attempting to create resource
ServiceAccount/velero: attempting to create resource client
ServiceAccount/velero: created
Secret/cloud-credentials: attempting to create resource
Secret/cloud-credentials: attempting to create resource client
Secret/cloud-credentials: created
BackupStorageLocation/default: attempting to create resource
BackupStorageLocation/default: attempting to create resource client
BackupStorageLocation/default: created
Deployment/velero: attempting to create resource
Deployment/velero: attempting to create resource client
Deployment/velero: created
DaemonSet/node-agent: attempting to create resource
DaemonSet/node-agent: attempting to create resource client
DaemonSet/node-agent: created
Velero is installed! ⛵ Use 'kubectl logs deployment/velero -n velero' to view the status.
```

Fatto, ora possiamo fare la verifica che l'agent risponde e ci indichi a quale storage sta puntando

```
C:\Users\P.Cornelio>velero backup-location get

NAME      PROVIDER   BUCKET/PREFIX      PHASE       LAST VALIDATED                  ACCESS MODE   DEFAULT
default   aws        cornelius-bucket   Available   2024-03-25 16:25:16 +0100 CET   ReadWrite     true
```

Restoriamo il namespace IOT di cui backup bck-namespace-iot

Proviamo ad effettuare il restore di un intero namespace precedentemente backuppato sul cluster di origine

```
C:\Users\P.Cornelio>velero restore create --from-backup bck-namespace-iot

Restore request "bck-namespace-iot-20240325164741" submitted successfully.
Run `velero restore describe bck-namespace-iot-20240325164741` or `velero restore logs
bck-namespace-iot-20240325164741` for more details.
```

Verifichiamo l'esito del restore con i seguenti comandi:

```
C:\Users\P.Cornelio>velero restore get
C:\Users\P.Cornelio>velero restore describe bck-namespace-iot
C:\Users\P.Cornelio>velero restore logs bck-namespace-iot
```

Verifichiamo il deployment appena restorato su K8S destinazione

```
C:\Users\P.Cornelio>kubectl get deployments --namespace=iot
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
iot-jsexecutor   0/15    15           0           <invalid>
iot-web-ui       0/2     2            0           <invalid>
```

ripetendo il comando si vedono i deployment work in progress:

```
C:\Users\P.Cornelio>kubectl get deployments --namespace=iot
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
iot-jsexecutor   0/15    15           0           41s
iot-web-ui       2/2     2            2           40s
```

In questo particolare esempio di restoring il namespace IOT contiene delle applicazioni che necessitano di kafka-zookeeper, quindi prima va restorato kafka-zookeeper. Ma ai fini dell'esempio va bene anche così.

CONTINUA NELLA RELEASE SUCCESSIVA…

Appendice

```
C:\Users\P.Cornelio>velero create backup --help
Create a backup

Usage:
  velero create backup NAME [flags]

Examples:
  # Create a backup containing all resources.
  velero backup create backup1

  # Create a backup including only the nginx namespace.
  velero backup create nginx-backup --include-namespaces nginx

  # Create a backup excluding the velero and default namespaces.
  velero backup create backup2 --exclude-namespaces velero,default

  # Create a backup based on a schedule named daily-backup.
  velero backup create --from-schedule daily-backup

  # View the YAML for a backup that doesn't snapshot volumes, without sending it to the
server.
  velero backup create backup3 --snapshot-volumes=false -o yaml

  # Wait for a backup to complete before returning from the command.
  velero backup create backup4 --wait
```