**Velero Integration**

There are 4 kinds of data to backup in a kubernetes cluster:

1. Cluster configuration. These are parameters like node configuration, networking and security constructs for the cluster etc.
2. Common kubernetes configurations. Examples are namespaces, rbac policies, pod security policies, quotas, etc.
3. Application manifests. This is based on the specific application that is getting deployed to the cluster.
4. Stateful configurations. These are persistent volumes that is attached to pods.

For item 1, we can use any infrastructure automation tools like Terraform or in the case of GCP, we can use deployment manager. The focus of this blog is on items 2, 3 and 4.

Following are some of the options possible for 2, 3 and 4:

1. Use a Kubernetes backup tool like Velero. This takes care of backing up both kubernetes resources as well as persistent volumes. This covers items 2, 3 and 4, so its pretty complete from a feature perspective. Velero is covered in [detail](https://sreeninet.wordpress.com/2020/04/19/kubernetes-dr/#velero) in this blog.
2. Use GCP “Config sync” feature. This can cover 2 and 3. This approach is more native with Kubernetes declarative approach and the config sync approach tries to recreate the cluster state from stored manifest files. Config sync approach is covered in [detail](https://sreeninet.wordpress.com/2020/04/19/kubernetes-dr/#configsync) in this blog.
3. Use CI/CD pipeline. This can cover 2 and 3. The CI/CD pipeline typically does whole bunch of other stuff in the pipeline and it is a roundabout approach to do DR. An alternative could be to create a separate DR pipeline in CI/CD.
4. [Kubernetes volume snapshot and restore feature](https://kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-cis-volume-snapshot-beta/) was introduced in beta in 1.17 release. This is targeted towards item 4. This will get integrated into kubernetes distributions soon. This approach will use kubernetes api itself to do the volume snapshot and restore.
5. Manual approach can be taken to backup and restore snapshots as described [here](https://stackoverflow.com/questions/60334589/how-to-clone-or-prepopulate-pvc-on-gke). This is targeted towards item 4. The example described here for GCP talks about using cloud provider tool to take a volume snapshot , create a disk from the volume and then manually create a PV and attach the disk to the PV. The kubernetes deployment can use the new PV.
6. Use backup and restore tool like [Stash](https://stash.run/). This is targeted towards item 4. Stash is a pretty comprehensive tool to backup Kubernetes stateful resources. Stash provides a kubernetes operator on top of [restic](https://restic.net/). Stash provides add-ons to backup common kubernetes stateful databases like postgres, mysql, mongo etc.

**Velero Installation**

Velero consists of a client installed on your local computer and a server that runs in your Kubernetes

For the client component, I installed it in mac using brew.

***brew install velero***

For other OS Type Installations

Download the tar files from <https://github.com/vmware-tanzu/velero/releases> and Install in Local Computer.

For the server component, I followed the steps [here](https://github.com/vmware-tanzu/velero-plugin-for-gcp#setup).

## **Setup**

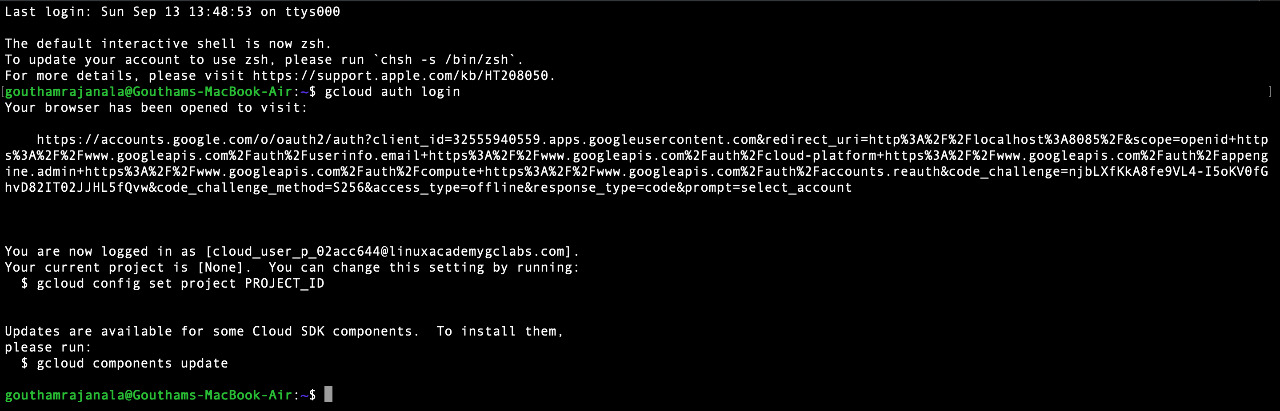
To set up Velero Server on GCP, you:

* [Create an GCS bucket](https://github.com/vmware-tanzu/velero-plugin-for-gcp#Create-an-GCS-bucket)
* [Set permissions for Velero](https://github.com/vmware-tanzu/velero-plugin-for-gcp#Set-permissions-for-Velero)
* [Install and start Velero](https://github.com/vmware-tanzu/velero-plugin-for-gcp#Install-and-start-Velero)

If you do not have the gcloud and gsutil CLIs locally installed, follow the [user guide](https://cloud.google.com/sdk/docs/) to set them up.

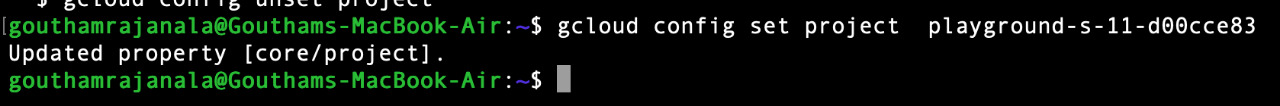
**Login to gcloud From SDK/Terminal**

***gcloud auth login***



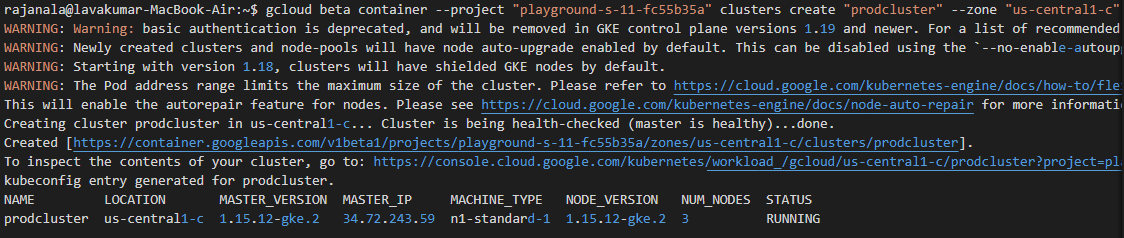
Set your Kubernates to GCP Project

***gcloud config set project PROJECT\_ID***

******

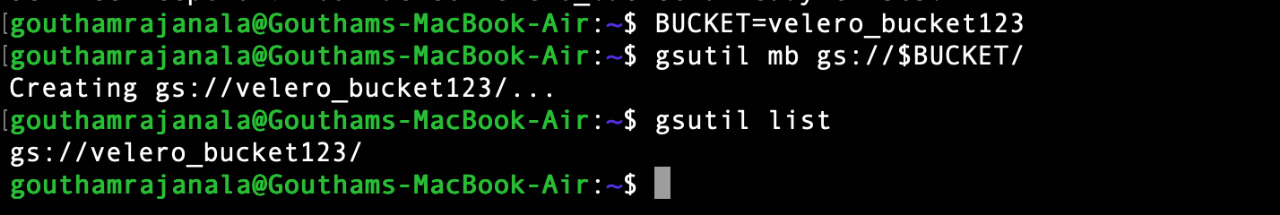
**Create GKE cluster**

***gcloud beta container --project "lavaproject" clusters create "velerocluster" --zone "us-central1-c" --enable-ip-alias***



**Create storage bucket**

***BUCKET=lavakumar-test-velero-backup  
gsutil mb gs://$BUCKET/***

******

**Create service account with right permissions**

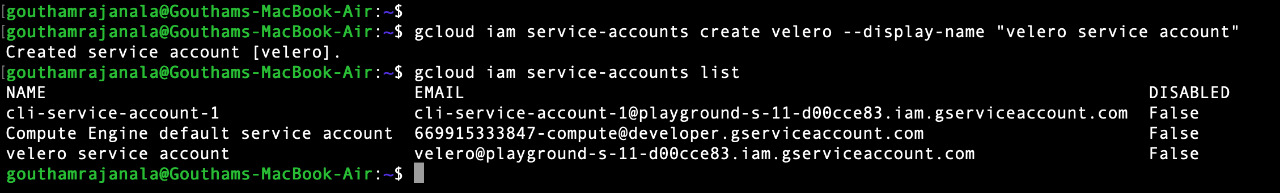
***gcloud iam service-accounts create velero \***

***--display-name "Velero service account"***

SERVICE\_ACCOUNT\_EMAIL=$(gcloud iam service-accounts list \

--filter="displayName:Velero service account" \

--format 'value(email)')



ROLE\_PERMISSIONS=(

compute.disks.get

compute.disks.create

compute.disks.createSnapshot

compute.snapshots.get

compute.snapshots.create

compute.snapshots.useReadOnly

compute.snapshots.delete

compute.zones.get

)

***gcloud iam roles create velero.server \***

***--project $PROJECT\_ID \***

***--title "Velero Server" \***

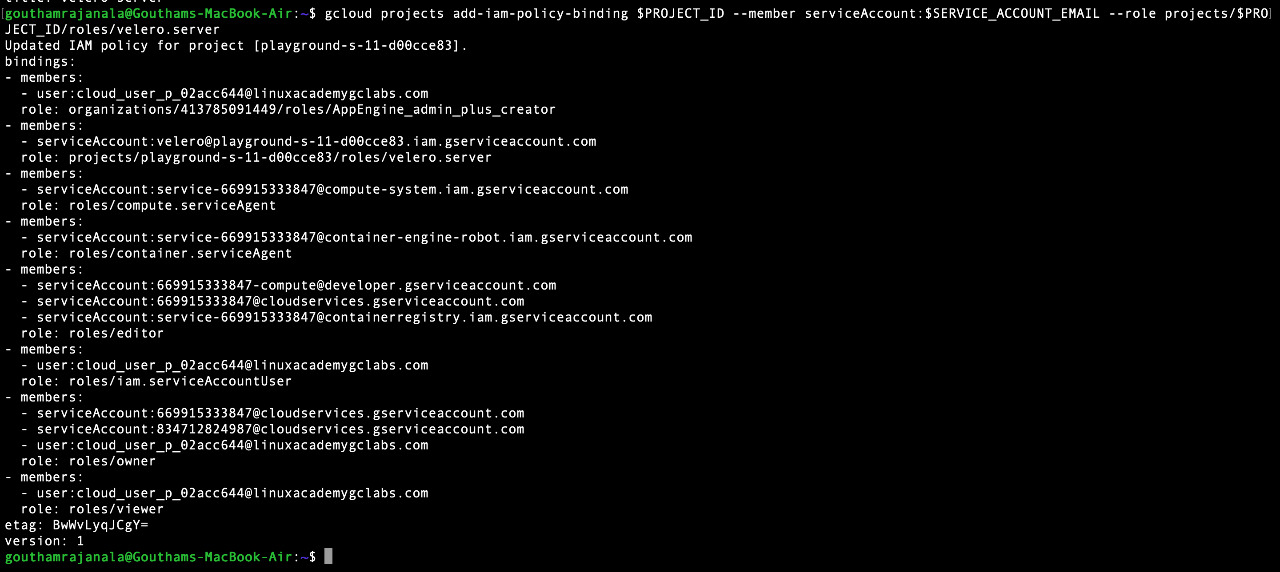
***--permissions "$(IFS=","; echo "${ROLE\_PERMISSIONS[\*]}")"***

******

***gcloud projects add-iam-policy-binding $PROJECT\_ID \***

***--member serviceAccount:$SERVICE\_ACCOUNT\_EMAIL \***

***--role projects/$PROJECT\_ID/roles/velero.server***

****

**Download service account locally**

This service account is needed for the installation of Velero server.

***gcloud iam service-accounts keys create credentials-velero \***

***--iam-account $SERVICE\_ACCOUNT\_EMAIL***

**C:\Users\rajanalal\Desktop\Velero_GCP\WhatsApp Image 2020-09-13 at 5.21.41 PM (12).jpeg**

**Set appropriate bucket permission**

***gsutil iam ch serviceAccount:$SERVICE\_ACCOUNT\_EMAIL:objectAdmin gs://${BUCKET}***

**Install Velero server**

This has to be done after setting the right Kubernetes context.

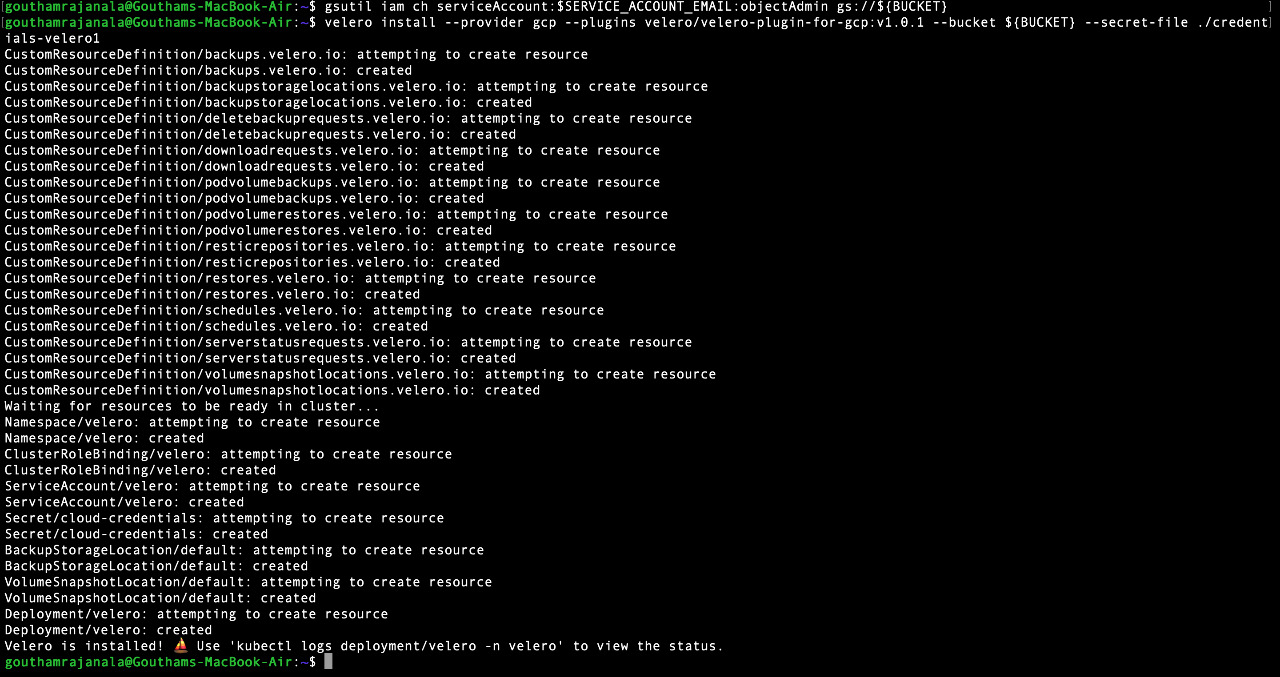
***velero install \***

***--provider gcp \***

***--plugins velero/velero-plugin-for-gcp:v1.0.***1 \

***--bucket $BUCKET \***

***--secret-file ./credentials-velero***



After this, we can check that Velero is successfully installed:

***$ velero version***

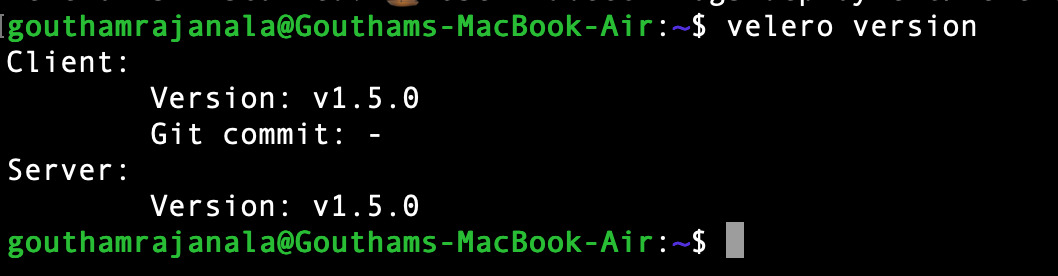
***Client:***

***Version: v1.3.1***

***Git commit: -***

***Server:***

***Version: v1.3.1***



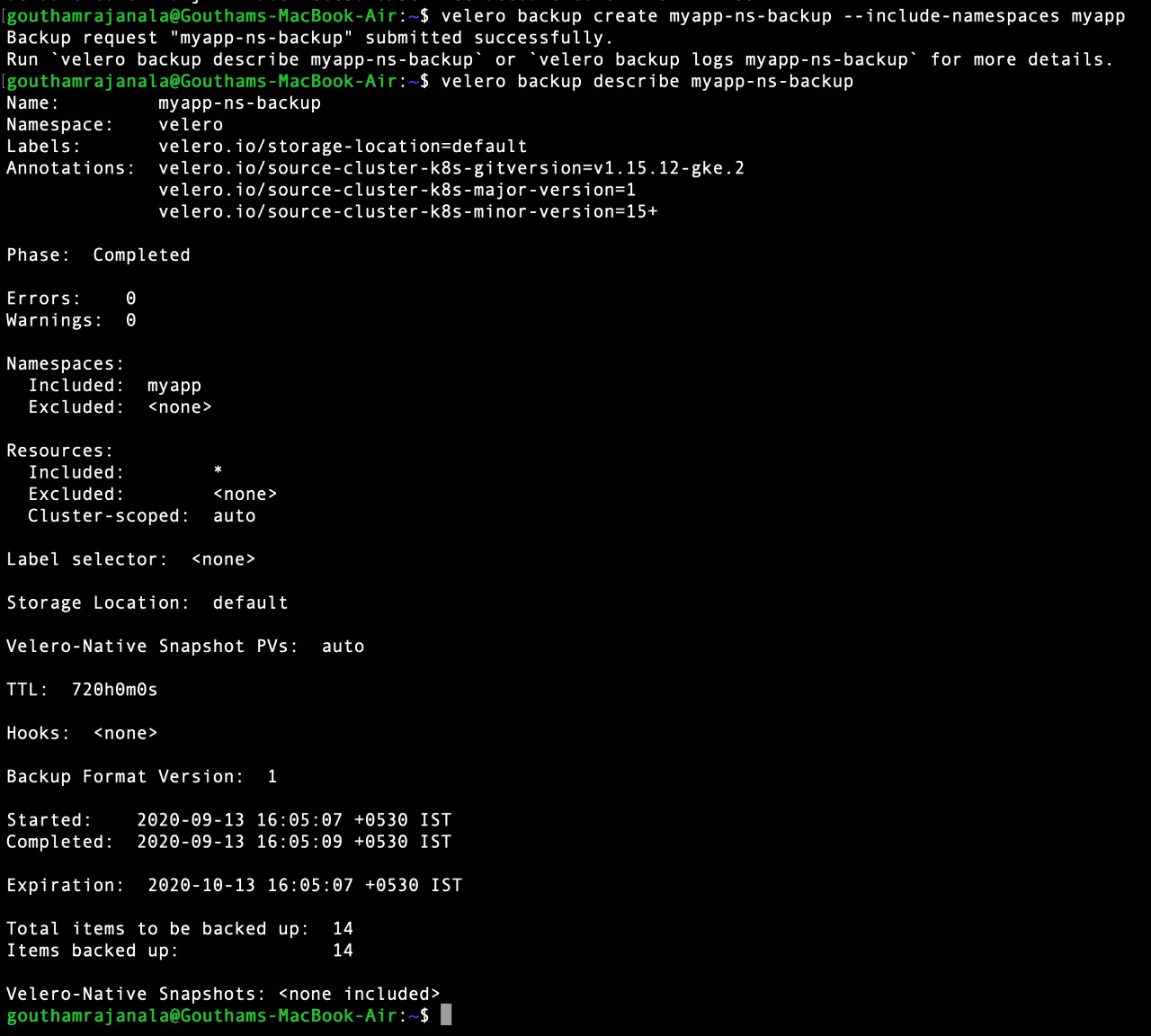
### Install application

Install Your Kubernates Application with Pods, Deployments, Nodes, Services, Secrets, PV, PVC, and Namespaces…

### Backup kubernetes cluster

The backup can be done at the complete cluster level or for individual namespaces. I will create a namespace backup now.

***$ velero backup create myapp-ns-backup --include-namespaces myapp***

******

Backup request "myapp-ns-backup" submitted successfully.

Run `velero backup describe myapp-ns-backup` or `velero backup logs myapp-ns-backup` for more details.

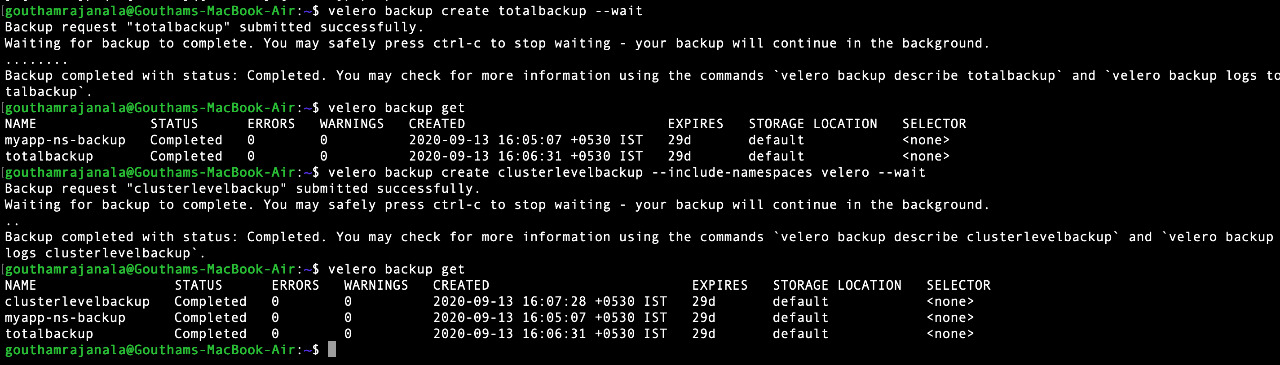
We can look at different commands like “velero backup describe”, “velero backup logs”, “velero get backup” to get the status of the backup. The following output shows that the backup is completed.

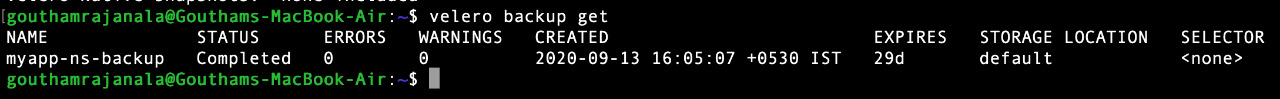
***$ velero get backup***

NAME STATUS CREATED EXPIRES STORAGE LOCATION SELECTOR

myapp-ns-backup Completed 2020-04-19 14:35:51 +0530 IST 29d default <none>

Let’s look at the snapshots created in GCP.





***$ gcloud compute snapshots list***

NAME DISK\_SIZE\_GB SRC\_DISK STATUS

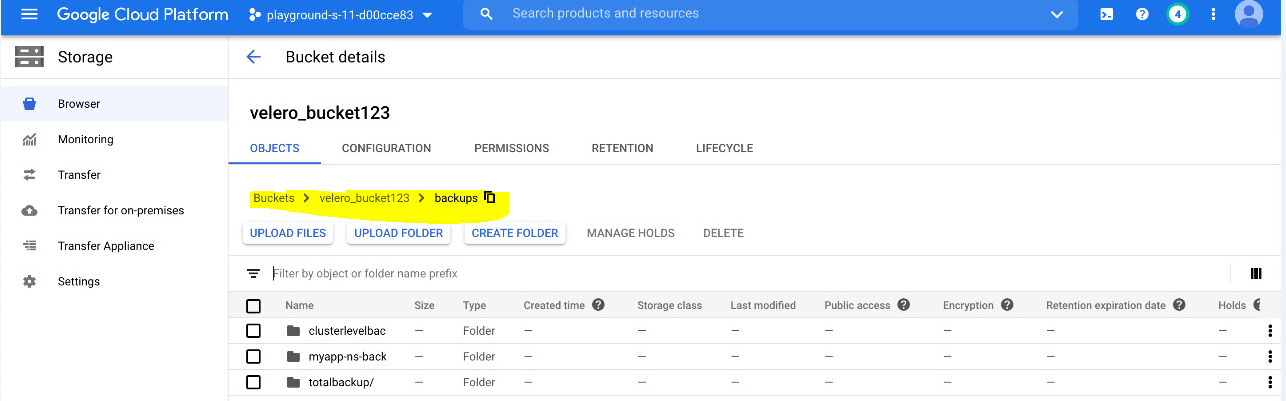
gke-prodcluster-96c83f-pvc-7a72c7dc-74ff-4301-b64d-0551b7d98db3 200 us-central1-c/disks/gke-prodcluster-96c83f-pvc-3ebf86a0-8162-11ea-9370-42010a800047 READY

gke-prodcluster-96c83f-pvc-c9c93573-666b-44d8-98d9-129ecc9ace50 200 us-central1-c/disks/gke-prodcluster-96c83f-pvc-4017a2ab-8162-11ea-9370-42010a800047 READY

Let’s look at the contents of velero storage bucket:

***gsutil ls gs://lavakumar-test-velero-backup/backups/***

***gs://lavakumar-test-velero-backup/backups/myapp-ns-backup/***



When creating snapshots, it is necessary that the snapshots are created in a consistent state when the writes are in the fly. The way Velero achieves this is by using [backup hooks](https://velero.io/docs/master/hooks/) and sidecar container. The backup hook freezes the filesystem when backup is running and then unfreezes the filesystem after backup is completed.

### Creating Scheduling Backup

### *velero schedule create velero\_schedule –schedule “0/4 \* \* \* \*” –include-namespace myapp*

### C:\Users\rajanalal\Desktop\Velero_GCP\WhatsApp Image 2020-09-13 at 5.21.41 PM (25).jpeg

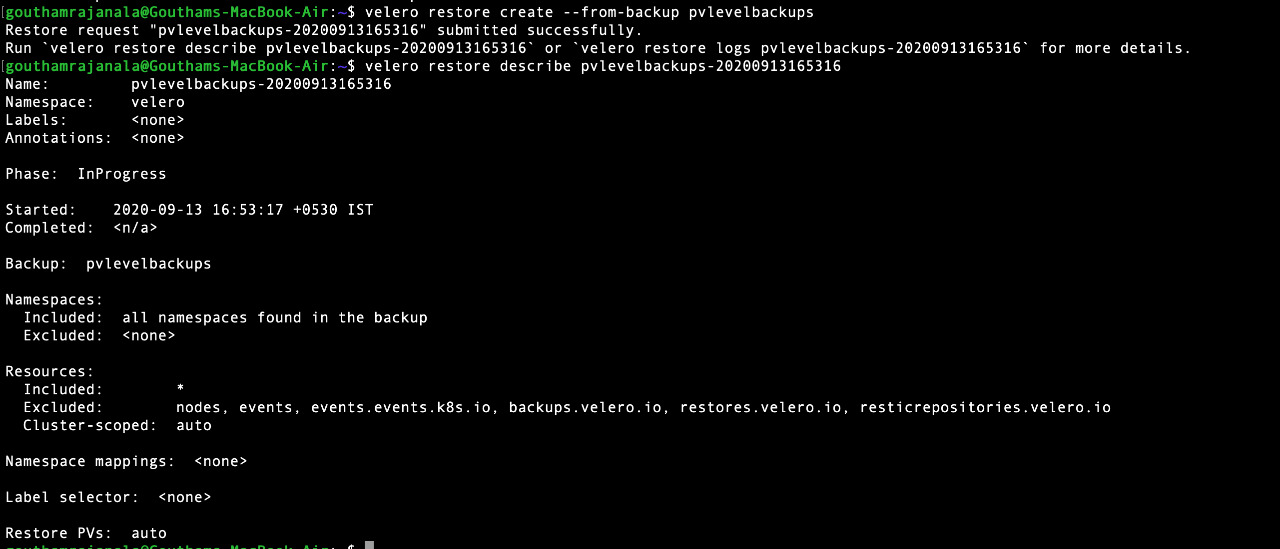
### Restore Kubernetes cluster

For this example, we will create a new cluster and restore the contents of namespace “myapp” to this cluster. We expect that both the kubernetes manifests as well as persistent volumes are restored.

***$ velero restore create --from-backup myapp-ns-backup***

Restore request "myapp-ns-backup-20200419151242" submitted successfully.

Run `velero restore describe myapp-ns-backup-20200419151242` or `velero restore logs myapp-ns-backup-20200419151242` for more details.



Let’s make sure that the restore is completed successfully:

***$ velero restore get***

NAME BACKUP STATUS WARNINGS ERRORS CREATED SELECTOR

myapp-ns-backup-20200419151242 myapp-ns-backup Completed 1 0 2020-04-19 15:12:44 +0530 IST <none>

The restore command above would create all the manifests including namespaces, deployments and services. It will also create PVs and attach to the appropriate pods.

Let’s look at the some of the resources created: