Steps to deploy!

- Navigate to the Backup & Restore directory:
- Create the Service Principal, representing Velero, to perform backups & restores:

```
az ad sp create-for-rbac --name sp-velero-aks1 --role Reader --scopes
/subscriptions/{subscriptionId}
```

- Deploy the Terraform sample code:

```
terraform init
terraform plan
terraform apply
```

- Connect to the Primary AKS Cluster (following the sample code as is):

```
   az aks get-credentials --name primary-aks1 --overwrite-existing --resource-
group primary-aks1
```

- Check that velero is installed and running correctly:

```
```bash
kubectl get pods -n velero
```
```

- Deploy sample stateful applications in the primary cluster:

```
   kubectl apply -f ../applications_samples/
```

  - Wait for the applications to be running

```
kubectl get pods --all-namespaces -w
```

  - Create some data files (to test backups and restores):

```
kubectl exec -it nginx-csi-disk-zrs -n csi-disk-zrs -- touch
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-csi-disk-lrs -n csi-disk-lrs -- touch
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-csi-file-zrs -n csi-file-zrs -- touch
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-file-lrs -n file-lrs -- touch /mnt/azuredisk/some-
data-file.txt
```

```
kubectl exec -it nginxstatefulset-0 -n diskstatefulset -- touch
/mnt/azuredisk/some-data-file.txt
```

- Check that data is created :

```
kubectl exec -it nginx-csi-disk-zrs -n csi-disk-zrs -- ls
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-csi-disk-lrs -n csi-disk-lrs -- ls
/mnt/azuredisk/some-data-file.txt
```

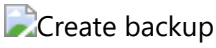
```
kubectl exec -it nginx-csi-file-zrs -n csi-file-zrs -- ls
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-file-lrs -n file-lrs -- ls /mnt/azuredisk/some-data-
file.txt
```

```
kubectl exec -it nginxstatefulset-0 -n diskstatefulset -- ls
/mnt/azuredisk/some-data-file.txt
```

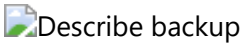
- Create a backup for primary AKS cluster: (You can filter resources to backup)

```
velero backup create manual-backup1  -w
```



- Describe created backup:

```
velero backup describe manual-backup1 --details
```



- Restore to secondary AKS cluster:
  - Connect to the Secondary / Backup AKS Cluster (following the sample code as is):

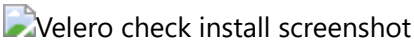
```
az aks get-credentials --name aks-dr --overwrite-existing --resource-
group aks-dr
```

  - Check running pods :

```
kubectl get pods --all-namespaces
```

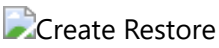
  - As Velero is configured, in the secondary backup cluster, to reference the same backup location (storage account container), You should see the same backups available :

```
velero backup get
```



  - Restore from backup : (you may get a partially failed status when trying to restore existing objects, such as kube-system resources).

```
velero restore create restore1 --from-backup manual-backup1 -w
```



- Check that Restore is successful:
  - Check restored applications / pods

```
kubectl get pods --all-namespaces
```

  - check restore details

```
velero restore get restore1
```

```
velero restore describe restore1 --details
```

- check restore logs

```
velero restore logs restore1
```

- Check that data is restored (verify existence of data files):

```
kubectl exec -it nginx-csi-disk-zrs -n csi-disk-zrs -- ls
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-csi-disk-lrs -n csi-disk-lrs -- ls
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-csi-file-zrs -n csi-file-zrs -- ls
/mnt/azuredisk/some-data-file.txt
```

```
kubectl exec -it nginx-file-lrs -n file-lrs -- ls /mnt/azuredisk/some-data-
file.txt
```

```
kubectl exec -it nginxstatefulset-0 -n diskstatefulset -- ls
/mnt/azuredisk/some-data-file.txt
```