# Maintenance

- Convert PV from ReadWriteOnce to ReadWriteMany
- Restore Longhorn PV from backup

# Convert PV from ReadWriteOnce to ReadWriteMany

# Change reclaim policy of the persistent volume to `Retain`

**This is required**!!! Before you delete the persistent volume claim to avoid a surprise that your data got wiped. The default reclaim policy is `Delete` and we do not want that to happen.

```
kubectl patch pv grafana-pv -n grafana -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
```

# Scale down any workloads (deployments, stateful sets etc etc) using the volume

In this step you are removing binds to the persistent volume. Check what is using it and scale it down to 0. No need to delete deployments, just scale them down to no pods running.

```
kubectl scale --replicas=0 -n grafana deployment grafana
```

When all workloads using the persistent volume have been scaled down to zero you should see the status of it change from `Bound` to `Released`. We are not done yet though.

# Delete existing persistent volume claim to be able to free persistent volume

NOTE: **Did you back up its manifest just in case?**

```
kubectl delete pvc grafana-pvc -n grafana
```

# Free persistent volume status to become `Available`

Here we just remove the reference to the deleted persistent volume claim from the persistent volume so that a new persistent volume claim can be set.

```
kubectl patch pv grafana-pv -n grafana -p '{"spec":{"claimRef":{"uid":""}}}'
```

You should see that status of the persistent volume changed to `Available`

# Change the persistent volume access mode to ReadWriteMany

Just run this command:

```
kubectl patch pv grafana-pv -n grafana -p '{"spec":{"accessModes":["ReadWriteMany"]}}'
```

# Recreate persistent volume claim with access mode ReadWriteMany

Here you either use your existing code to create this kubernetes resource with access mode `ReadWriteMany` or just use the backup we did in first step, edit its spec.accessModes and apply that.

I am just showing what the part of `spec` that is interesting to us should be, if you are editing existing manifest or backup, you will easily find the place to change it.

```
spec:
  accessModes:
    - ReadWriteMany
```

Apply the new persistent volume claim manifest (assuming we saved it to a file called `persistent_volume_claim_read_write_many.yaml`:

```
kubectl -n grafana apply -f pvc.yaml
```

# Scale the workloads back up to start using the volume with new access mode.

Adjust number of replicas to what your workloads manifests actually specify.

```
kubectl scale --replicas=1 -n grafana deployment grafana
```

The status of you persistent volume should become `Bound` now.

# Restore Longhorn PV from backup

Unfortunately, the longhorn UIs restore function does not work for in place backups. This means the PV needs to be deleted for the new volume to be created! As of this writing, there is a Github issue open from 2023 regarding this problem.

## Steps to restore from backup

1. Ensure the backup exists in the UI. Navigate to Volumes > <broken PV> > Backups. I open this in a new window. We need to take note of the PV and PVC names.
2. Scale down the deployment/statefulset using the PV.

```
kubectl scale statefulset <name> --replicas 0
```

3. Delete the PV/PVC from the longhorn UI
4. In the backup windows in the longhorn UI. Select Restore. Check the box "Use Previous Name"
5. Navigate back to volumes and find the restored PV. On the right dropdown click Create PV/PVC. It should ask to use the old information, if not we need to fill in the information from step 1.
6. Verify the PV/PVC exist and we can edit the PV and look for "volumeName" to match step 1.

```
spec:
  storageClassName: longhorn
  volumeMode: Filesystem
  volumeName: pvc-<ID from step 1>
```

7. Scale the deployment back up

```
kubectl scale statefulset <name> --replicas 1
```