INSTALLATION (KUBERNETES + HELM)

The ceph-helm project enables you to deploy Ceph in a Kubernetes environement. This documentation assumes a Kubernetes environement is available.

CURRENT LIMITATIONS

- The public and cluster networks must be the same
- If the storage class user id is not admin, you will have to manually create the user in your Ceph cluster and create its secret in Kubernetes
- ceph-mgr can only run with 1 replica

INSTALL AND START HELM

Helm can be installed by following these instructions.

Helm finds the Kubernetes cluster by reading from the local Kubernetes config file; make sure this is downloaded and accessible to the `helm` client.

A Tiller server must be configured and running for your Kubernetes cluster, and the local Helm client must be connected to it. It may be helpful to look at the Helm documentation for init. To run Tiller locally and connect Helm to it, run:

```
$ helm init
```

The ceph-helm project uses a local Helm repo by default to store charts. To start a local Helm repo server, run:

```
$ helm serve &
$ helm repo add local http://localhost:8879/charts
```

ADD CEPH-HELM TO HELM LOCAL REPOS

```
$ git clone https://github.com/ceph/ceph-helm
$ cd ceph-helm/ceph
$ make
```

CONFIGURE YOUR CEPH CLUSTER

Create a `ceph-overrides.yaml` that will contain your Ceph configuration. This file may exist anywhere, but for this docuiment will be assumed to reside in the user's home directory.:

```
$ cat ~/ceph-overrides.yaml
network:
  public:   172.21.0.0/20
  cluster:   172.21.0.0/20

osd_devices:
  - name: dev-sdd
    device: /dev/sdd
    zap: "1"
  - name: dev-sde
    device: /dev/sde
    zap: "1"

storageclass:
  name: ceph-rbd
  pool: rbd
  user_id: k8s
```

Note: If journal is not set it will be colocated with device

Note: The `ceph-helm/ceph/ceph/values.yaml` file contains the full list of option that can be set

CREATE THE CEPH CLUSTER NAMESPACE

By default, ceph-helm components assume they are to be run in the ceph Kubernetes namespace. To create the namespace, run:

```
$ kubectl create namespace ceph
```

CONFIGURE RBAC PERMISSIONS

Kubernetes >=v1.6 makes RBAC the default admission controller. ceph-helm provides RBAC roles and permissions for each component:

```
$ kubectl create -f ~/ceph-helm/ceph/rbac.yaml
```

The `rbac.yaml` file assumes that the Ceph cluster will be deployed in the ceph namespace.

LABEL KUBELETS

The following labels need to be set to deploy a Ceph cluster:

- ceph-mon=enabled
- ceph-mgr=enabled
- ceph-osd=enabled
- ceph-osd-device-<name>=enabled

The `ceph-osd-device-<name>` label is created based on the osd_devices name value defined in our `ceph-overrides.yaml`. From our example above we will have the two following label: `ceph-osd-device-dev-sdb` and `ceph-osd-device-dev-sdc`.

For each Ceph Monitor:

```
$ kubectl label node <nodename> ceph-mon=enabled ceph-mgr=enabled
```

For each OSD node:

```
$ kubectl label node <nodename> ceph-osd=enabled ceph-osd-device-dev-sdb=enabled ceph-osd-dev
```

CEPH DEPLOYMENT

Run the helm install command to deploy Ceph:

```
$ helm install --name=ceph local/ceph --namespace=ceph -f ~/ceph-overrides.yaml
NAME:   ceph
LAST DEPLOYED: Wed Oct 18 22:25:06 2017
NAMESPACE: ceph
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME                    TYPE    DATA  AGE
ceph-keystone-user-rgw  Opaque  7     1s

==> v1/ConfigMap
NAME              DATA  AGE
```

```
ceph-bin-clients  2     1s
ceph-bin          24    1s
ceph-etc          1     1s
ceph-templates    5     1s

==> v1/Service
NAME      CLUSTER-IP      EXTERNAL-IP  PORT(S)   AGE
ceph-mon  None            <none>       6789/TCP  1s
ceph-rgw  10.101.219.239  <none>       8088/TCP  1s

==> v1beta1/DaemonSet
NAME              DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE-SELECTOR
ceph-mon          3        3        0      3           0          ceph-mon=enabled
ceph-osd-dev-sde  3        3        0      3           0          ceph-osd-device-dev-sde=ena
ceph-osd-dev-sdd  3        3        0      3           0          ceph-osd-device-dev-sdd=ena

==> v1beta1/Deployment
NAME                  DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
ceph-mds              1        1        1           0          1s
ceph-mgr              1        1        1           0          1s
ceph-mon-check        1        1        1           0          1s
ceph-rbd-provisioner  2        2        2           0          1s
ceph-rgw              1        1        1           0          1s

==> v1/Job
NAME                                 DESIRED  SUCCESSFUL  AGE
ceph-mgr-keyring-generator           1        0           1s
ceph-mds-keyring-generator           1        0           1s
ceph-osd-keyring-generator           1        0           1s
ceph-rgw-keyring-generator           1        0           1s
ceph-mon-keyring-generator           1        0           1s
ceph-namespace-client-key-generator  1        0           1s
ceph-storage-keys-generator          1        0           1s

==> v1/StorageClass
NAME     TYPE
ceph-rbd  ceph.com/rbd
```

The output from helm install shows us the different types of ressources that will be deployed.

A StorageClass named `ceph-rbd` of type `ceph.com/rbd` will be created with `ceph-rbd-provisioner` Pods. These will allow a RBD to be automatically provisioned upon creation of a PVC. RBDs will also be formatted when mapped for the first time. All RBDs will use the ext4 filesystem. `ceph.com/rbd` does not support the `fsType` option. By default, RBDs will use image format 2 and layering. You can overwrite the following storageclass' defaults in your values file:

```
storageclass:
  name: ceph-rbd
  pool: rbd
  user_id: k8s
  user_secret_name: pvc-ceph-client-key
  image_format: "2"
  image_features: layering
```

Check that all Pods are running with the command below. This might take a few minutes:

```
$ kubectl -n ceph get pods
NAME                                        READY     STATUS    RESTARTS   AGE
ceph-mds-3804776627-976z9                   0/1       Pending   0          1m
ceph-mgr-3367933990-b368c                   1/1       Running   0          1m
ceph-mon-check-1818208419-0vkb7             1/1       Running   0          1m
ceph-mon-cppdk                              3/3       Running   0          1m
ceph-mon-t4stn                              3/3       Running   0          1m
ceph-mon-vqzl0                              3/3       Running   0          1m
ceph-osd-dev-sdd-6dphp                      1/1       Running   0          1m
ceph-osd-dev-sdd-6w7ng                      1/1       Running   0          1m
ceph-osd-dev-sdd-l80vv                      1/1       Running   0          1m
ceph-osd-dev-sde-6dq6w                      1/1       Running   0          1m
ceph-osd-dev-sde-kqt0r                      1/1       Running   0          1m
ceph-osd-dev-sde-lp2pf                      1/1       Running   0          1m
ceph-rbd-provisioner-2099367036-4prvt       1/1       Running   0          1m
```

```
ceph-rbd-provisioner-2099367036-h9kw7   1/1       Running   0          1m
ceph-rgw-3375847861-4wr74               0/1       Pending   0          1m
```

Note: The MDS and RGW Pods are pending since we did not label any nodes with ceph-rgw=enabled or ceph-mds=enabled

Once all Pods are running, check the status of the Ceph cluster from one Mon:

```
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- ceph -s
cluster:
  id:     e8f9da03-c2d2-4ad3-b807-2a13d0775504
  health: HEALTH_OK

services:
  mon: 3 daemons, quorum mira115,mira110,mira109
  mgr: mira109(active)
  osd: 6 osds: 6 up, 6 in

data:
  pools:   0 pools, 0 pgs
  objects: 0 objects, 0 bytes
  usage:   644 MB used, 5555 GB / 5556 GB avail
  pgs:
```

CONFIGURE A POD TO USE A PERSISTENTVOLUME FROM CEPH

Create a keyring for the k8s user defined in the `~/ceph-overwrite.yaml` and convert it to base64:

```
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- bash
# ceph auth get-or-create-key client.k8s mon 'allow r' osd 'allow rwx pool=rbd'  | base64
QVFCLzdPaFoxeUxCRVJBQUVEVGdHcE9YU3BYMVBSdURHUEU0T0E9PQo=
# exit
```

Edit the user secret present in the ceph namespace:

```
$ kubectl -n ceph edit secrets/pvc-ceph-client-key
```

Add the base64 value to the key value with your own and save:

```
apiVersion: v1
data:
  key: QVFCLzdPaFoxeUxCRVJBQUVEVGdHcE9YU3BYMVBSdURHUEU0T0E9PQo=
kind: Secret
metadata:
  creationTimestamp: 2017-10-19T17:34:04Z
  name: pvc-ceph-client-key
  namespace: ceph
  resourceVersion: "8665522"
  selfLink: /api/v1/namespaces/ceph/secrets/pvc-ceph-client-key
  uid: b4085944-b4f3-11e7-add7-002590347682
type: kubernetes.io/rbd
```

We are going to create a Pod that consumes a RBD in the default namespace. Copy the user secret from the ceph namespace to `default`:

```
$ kubectl -n ceph get secrets/pvc-ceph-client-key -o json | jq '.metadata.namespace = "defaul
secret "pvc-ceph-client-key" created
$ kubectl get secrets
NAME                  TYPE                                  DATA      AGE
default-token-r43wl   kubernetes.io/service-account-token   3         61d
pvc-ceph-client-key   kubernetes.io/rbd                     1         20s
```

Create and initialize the RBD pool:

```
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- ceph osd pool create rbd 256
pool 'rbd' created
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- rbd pool init rbd
```

Important: Kubernetes uses the RBD kernel module to map RBDs to hosts. Luminous requires CRUSH_TUNABLES 5 (Jewel). The minimal kernel version for these tunables is 4.5. If your kernel does not support these tunables, run `ceph osd crush tunables hammer`

Important: Since RBDs are mapped on the host system. Hosts need to be able to resolve the ceph-mon.ceph.svc.cluster.local name managed by the kube-dns service. To get the IP address of the kube-dns service, run `kubectl -n kube-system get svc/kube-dns`

Create a PVC:

```
$ cat pvc-rbd.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ceph-pvc
spec:
  accessModes:
   - ReadWriteOnce
  resources:
    requests:
       storage: 20Gi
  storageClassName: ceph-rbd

$ kubectl create -f pvc-rbd.yaml
persistentvolumeclaim "ceph-pvc" created
$ kubectl get pvc
NAME       STATUS    VOLUME                                     CAPACITY   ACCESSMODES   STOR
ceph-pvc   Bound     pvc-1c2ada50-b456-11e7-add7-002590347682   20Gi       RWO           ceph
```

You can check that the RBD has been created on your cluster:

```
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- rbd ls
kubernetes-dynamic-pvc-1c2e9442-b456-11e7-9bd2-2a4159ce3915
$ kubectl -n ceph exec -ti ceph-mon-cppdk -c ceph-mon -- rbd info kubernetes-dynamic-pvc-1c2e
rbd image 'kubernetes-dynamic-pvc-1c2e9442-b456-11e7-9bd2-2a4159ce3915':
    size 20480 MB in 5120 objects
    order 22 (4096 kB objects)
    block_name_prefix: rbd_data.10762ae8944a
    format: 2
    features: layering
    flags:
    create_timestamp: Wed Oct 18 22:45:59 2017
```

Create a Pod that will use the PVC:

```
$ cat pod-with-rbd.yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: busybox
      image: busybox
      command:
        - sleep
        - "3600"
      volumeMounts:
      - mountPath: "/mnt/rbd"
        name: vol1
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: ceph-pvc

$ kubectl create -f pod-with-rbd.yaml
pod "mypod" created
```

Check the Pod:

```
$ kubectl get pods
NAME      READY     STATUS    RESTARTS   AGE
mypod     1/1       Running   0          17s
$ kubectl exec mypod -- mount | grep rbd
/dev/rbd0 on /mnt/rbd type ext4 (rw,relatime,stripe=1024,data=ordered)
```

LOGGING

OSDs and Monitor logs can be accessed via the `kubectl logs [-f]` command. Monitors have multiple stream of logging, each stream is accessible from a container running in the ceph-mon Pod.

There are 3 containers running in the ceph-mon Pod:

- ceph-mon, equivalent of ceph-mon.hostname.log on baremetal
- cluster-audit-log-tailer, equivalent of ceph.audit.log on baremetal
- cluster-log-tailer, equivalent of ceph.log on baremetal or `ceph -w`

Each container is accessible via the `--container` or `-c` option. For instance, to access the cluster-tail-log, one can run:

```
$ kubectl -n ceph logs ceph-mon-cppdk -c cluster-log-tailer
```