Architektur

Setup

Als erstes müssen 3 VM Instanzen auf gcloud erstellt werden.

```
$ gcloud compute instances create s1-node1
            --image-family ubuntu-1604-lts
            --image-project ubuntu-os-cloud
            --machine-type n1-standard-2
            --zone europe-west1-d
$ gcloud compute instances create s1-node2
            --image-family ubuntu-1604-lts
            --image-project ubuntu-os-cloud
            --machine-type n1-standard-2
            --zone europe-west1-d
$ gcloud compute instances create s1-node2
            --image-family ubuntu-1604-lts
            --image-project ubuntu-os-cloud
            --machine-type n1-standard-2
            --zone europe-west1-d
```

Danach muss auf dem Jumphost ein ssh Key generiert werden. Hierbei muss umbeding darauf geachtet werden, dass das Zertifikat ceph heist.

```
$ ssh-keygen
```

Wenn dies gemacht wurde, muss dieser anschliessend auf die erstellten VMs für den User `ceph` kopiert werden. Zuvor müssen jedoch auf allen Maschinen Updates und der entsprechende User angelegt werden. Man verbindet sich nun mit dem gcloud Befehl nacheinander auf alle drei Notes und installiert die Updates. Ebenfalls der User `ceph` muss angelegt werden.

```
$ gcloud compute ssh s1-node1
```

Danach werden die Updates installiert

```
$ sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get dist-upgrade -y
$ sudo useradd -d /home/cephd -m cephd -s /bin/bash
$ sudo passwd cephd
$ echo "cephd ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/cephd
$ sudo chmod 0440 /etc/sudoers.d/cephd
$ sudo su - cephd
```

Hier wird noch der SSH Key vom Host in die authorized Keys geladen, so dass später per SSH Verbindungen augebaut werden können

```
$ mkdir .ssh
$ vim .ssh/authorized_keys
```

Wenn dies nun auf allen Drei VMs gemacht wurden, werden diese nun in die `.ssh/config` eingetragen

```
Host s1-node1 s1-node2 s1-node3
    User cephd
    IdentityFile ~/.ssh/ceph
```

Wie vorher bereits angemerkt ist es wichtig, dass das Zertifikat als ceph benennt wurde, sonnst muss hier ein anderer Name angegeben werden

Ceph Installation

Zuerst wird auf dem Jumphost der Ordner ceph erstetlt.

```
$ ceph-deploy install s1-node1 s1-node2 s1-node3
$ ceph-deploy new s1-node1 s1-node2 s1-node3
[ceph_deploy.conf][DEBUG ] found configuration file at: /home/student1/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (1.5.32): /usr/bin/ceph-deploy mon create-initial
[ceph_deploy.cli][INFO  ] ceph-deploy options:
[ceph_deploy.cli][INFO  ]  username                      : None
[ceph_deploy.cli][INFO  ]  verbose                       : False
[ceph_deploy.cli][INFO  ]  overwrite_conf                : False
[ceph_deploy.cli][INFO  ]  subcommand                    : create-initial
[ceph_deploy.cli][INFO  ]  quiet                         : False
[ceph_deploy.cli][INFO  ]  cd_conf                       : <ceph_deploy.conf.cephdeploy.Conf
[ceph_deploy.cli][INFO  ]  cluster                       : ceph
[ceph_deploy.cli][INFO  ]  func                          : <function mon at 0x7fe408522488>
[ceph_deploy.cli][INFO  ]  ceph_conf                     : None
[ceph_deploy.cli][INFO  ]  keyrings                      : None
[ceph_deploy.cli][INFO  ]  default_release               : False
[ceph_deploy.mon][DEBUG ] Deploying mon, cluster ceph hosts s1-node1 s1-node2 s1-node3
[ceph_deploy.mon][DEBUG ] detecting platform for host s1-node1 ...
[s1-node1][DEBUG ] connection detected need for sudo
[s1-node1][DEBUG ] connected to host: s1-node1
[s1-node1][DEBUG ] detect platform information from remote host
[s1-node1][DEBUG ] detect machine type
[s1-node1][DEBUG ] find the location of an executable
[ceph_deploy.mon][INFO  ] distro info: Ubuntu 16.04 xenial
[s1-node1][DEBUG ] determining if provided host has same hostname in remote
[s1-node1][DEBUG ] get remote short hostname
[s1-node1][DEBUG ] deploying mon to s1-node1
[s1-node1][DEBUG ] get remote short hostname
[s1-node1][DEBUG ] remote hostname: s1-node1
[s1-node1][DEBUG ] write cluster configuration to /etc/ceph/{cluster}.conf
[s1-node1][DEBUG ] create the mon path if it does not exist
[s1-node1][DEBUG ] checking for done path: /var/lib/ceph/mon/ceph-s1-node1/done
```

```
[s1-node1][DEBUG ] done path does not exist: /var/lib/ceph/mon/ceph-s1-node1/done
[s1-node1][INFO  ] creating keyring file: /var/lib/ceph/tmp/ceph-s1-node1.mon.keyring
[s1-node1][DEBUG ] create the monitor keyring file
[s1-node1][INFO  ] Running command: sudo ceph-mon --cluster ceph --mkfs -i s1-node1 --keyrin
[s1-node1][DEBUG ] ceph-mon: mon.noname-a 10.132.0.17:6789/0 is local, renaming to mon.s1-no
[s1-node1][DEBUG ] ceph-mon: set fsid to 27f1d85f-9b95-4f1e-bd3b-3e8a2cb70e95
[s1-node1][DEBUG ] ceph-mon: created monfs at /var/lib/ceph/mon/ceph-s1-node1 for mon.s1-noo
[s1-node1][INFO  ] unlinking keyring file /var/lib/ceph/tmp/ceph-s1-node1.mon.keyring
[s1-node1][DEBUG ] create a done file to avoid re-doing the mon deployment
[s1-node1][DEBUG ] create the init path if it does not exist
[s1-node1][INFO  ] Running command: sudo systemctl enable ceph.target
[s1-node1][INFO  ] Running command: sudo systemctl enable ceph-mon@s1-node1
[s1-node1][WARNIN] Created symlink from /etc/systemd/system/ceph-mon.target.wants/ceph-mon@s
[s1-node1][INFO  ] Running command: sudo systemctl start ceph-mon@s1-node1
[s1-node1][INFO  ] Running command: sudo ceph --cluster=ceph --admin-daemon /var/run/ceph/ce
[s1-node1][DEBUG ] ********************************************************************************
[s1-node1][DEBUG ] status for monitor: mon.s1-node1
[s1-node1][DEBUG ] {
[s1-node1][DEBUG ]   "election_epoch": 0,
[s1-node1][DEBUG ]   "extra_probe_peers": [
[s1-node1][DEBUG ]     "10.132.0.15:6789/0",
[s1-node1][DEBUG ]     "10.132.0.16:6789/0"
[s1-node1][DEBUG ]   ],
[s1-node1][DEBUG ]   "monmap": {
[s1-node1][DEBUG ]     "created": "2018-02-17 09:30:17.568540",
[s1-node1][DEBUG ]     "epoch": 0,
[s1-node1][DEBUG ]     "fsid": "27f1d85f-9b95-4f1e-bd3b-3e8a2cb70e95",
[s1-node1][DEBUG ]     "modified": "2018-02-17 09:30:17.568540",
[s1-node1][DEBUG ]     "mons": [
[s1-node1][DEBUG ]       {
[s1-node1][DEBUG ]         "addr": "10.132.0.17:6789/0",
[s1-node1][DEBUG ]         "name": "s1-node1",
[s1-node1][DEBUG ]         "rank": 0
[s1-node1][DEBUG ]       },
[s1-node1][DEBUG ]       {
[s1-node1][DEBUG ]         "addr": "0.0.0.0:0/1",
[s1-node1][DEBUG ]         "name": "s1-node2",
[s1-node1][DEBUG ]         "rank": 1
[s1-node1][DEBUG ]       },
[s1-node1][DEBUG ]       {
[s1-node1][DEBUG ]         "addr": "0.0.0.0:0/2",
[s1-node1][DEBUG ]         "name": "s1-node3",
[s1-node1][DEBUG ]         "rank": 2
[s1-node1][DEBUG ]       }
[s1-node1][DEBUG ]     ]
[s1-node1][DEBUG ]   },
```

```
[s1-node1][DEBUG ]   "name": "s1-node1",
[s1-node1][DEBUG ]   "outside_quorum": [
[s1-node1][DEBUG ]     "s1-node1"
[s1-node1][DEBUG ]   ],
[s1-node1][DEBUG ]   "quorum": [],
[s1-node1][DEBUG ]   "rank": 0,
[s1-node1][DEBUG ]   "state": "probing",
[s1-node1][DEBUG ]   "sync_provider": []
[s1-node1][DEBUG ] }
[s1-node1][DEBUG ] ********************************************************************************
[s1-node1][INFO  ] monitor: mon.s1-node1 is running
[s1-node1][INFO  ] Running command: sudo ceph --cluster=ceph --admin-daemon /var/run/ceph/ce
[ceph_deploy.mon][DEBUG ] detecting platform for host s1-node2 ...
...
```

Nun wird das osd Verzeichnis initialisiert.

```
$ ceph-deploy mon create-initial
```

Create a osd directory on every node!

```
$ ssh s1-node1 sudo mkdir /var/local/osd0
$ ssh s1-node2 sudo mkdir /var/local/osd1
$ ssh s1-node3 sudo mkdir /var/local/osd2
```

Prepare the osd for use with the cluster:

```
$ ceph-deploy osd prepare
            s1-node1:/var/local/osd0
            s1-node2:/var/local/osd1
            s1-node3:/var/local/osd2
```

This is an ugly fix for a permission issue that we have with the osd disks. Never do this in production:

```
$ ssh s1-node1 sudo chmod -R 777 /var/local/osd0
$ ssh s1-node2 sudo chmod -R 777 /var/local/osd1
$ ssh s1-node3 sudo chmod -R 777 /var/local/osd2
```

Activate the osd disks

```
$ ceph-deploy osd activate
            s1-node1:/var/local/osd0
            s1-node2:/var/local/osd1
            s1-node3:/var/local/osd2
```

Deploy the cluster keys to the nodes so we can use them to work with the cluster.

```
$ ceph-deploy admin s1-node1 s1-node2 s1-node3
```

Ensure that ceph osds get automatically started on system boot.

```
$ ssh s1-node1 sudo systemctl enable ceph-osd@0
$ ssh s1-node2 sudo systemctl enable ceph-osd@1
$ ssh s1-node3 sudo systemctl enable ceph-osd@2
```

Ceph funktioniert nun, wie man das am Healthcheck sehen kann

```
$ ssh s1-node3 sudo ceph -w
cluster 27f1d85f-9b95-4f1e-bd3b-3e8a2cb70e95
    health HEALTH_OK
         monmap e1: 3 mons at {s1-node1=10.132.0.17:6789/0,s1-node2=10.132.0.16:6789/0,s1-no
            election epoch 6, quorum 0,1,2 s1-node3,s1-node2,s1-node1
         osdmap e14: 3 osds: 3 up, 3 in
            flags sortbitwise,require_jewel_osds
          pgmap v113: 64 pgs, 1 pools, 0 bytes data, 0 objects
            19915 MB used, 9621 MB / 29585 MB avail
                  64 active+clean

2018-02-17 09:42:30.742833 mon.0 [INF] pgmap v112: 64 pgs: 64 active+clean; 0 bytes data, 19
2018-02-17 09:42:32.766504 mon.0 [INF] pgmap v113: 64 pgs: 64 active+clean; 0 bytes data, 19
```

Kubi Installation

Nun wird Kubernetes installiert

Als erstes wird der Ordner k8s auf dem Jumphost erstellt.

Folgende Commands sind wurden auf allen Nodes ausgeführt

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
$ cat <<EOF > /etc/apt/sources.list.d/kubernetes.list
        deb http://apt.kubernetes.io/ kubernetes-xenial main
        EOF

$ apt-get update
$ apt-get install -y docker.io
$ apt-get install -y kubelet kubeadm kubectl kubernetes-cni
```

Anschliessend wird auf dem Node 1 Kubernetes initialisiert.

```
$ kubeadm init --pod-network-cidr 10.244.0.0/16

kubeadm join --token e77f3a.932b35eb750281f7 10.132.0.17:6443 --discovery-token-ca-cert-hash
```

Hierbei muss bei der Ausgabe der KubeADM Befehl kopiert werden. Dieser wird nachher noch benötigt um auf allen Nodes auszuführen.

```
$ kubeadm join --token e77f3a.932b35eb750281f7 10.132.0.17:6443 --discovery-token-ca-cert-ha
```

Danach wird die Kubernetes Konfig mit dem User cephd in dessen .kube Ordner kopiert. Diese kann danach vom Jumphost per scp ab s1-node1 kopiert werden.

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Nun wird das Kubernetes Netzwerk installiert.

```
$ kubectl taint node --all node-role.kubernetes.io/master:NoSchedule-
$ wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.ym
$ kubectl apply -f kube-flannel.yml --namespace=kube-system
```

Nachdem die anderen beiden Nodes nun in Kubernetes gejoint wurden gibt folgender Commend nun diesen Output

```
$ kubectl get nodes
NAME       STATUS    ROLES     AGE       VERSION
s1-node1   Ready     master    10m       v1.9.3
s1-node2   Ready     <none>    1m        v1.9.3
s1-node3   Ready     <none>    37s       v1.9.3
```

Zugriff auf ceph

Als erstes wird sichergestellt, dass auf Node1 ceph richtig gestartet ist

```
$ sudo systemctl start ceph-mon
$ sudo systemctl start ceph-osd@0
```

Danach werden die beiden Yaml Files provisioner.yaml und storage.yaml erstellt.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: rbd-provisioner
   namespace: kube-system
spec:
   replicas: 1
   template:
      metadata:
          labels:
              app: rbd-provisioner
spec:
   containers:
      name: rbd-provisioner
      image: "quay.io/external_storage/rbd-provisioner:v0.1.1"
      serviceAccountName: persistent-volume-binder
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   mame: ceph
```

```
   provisioner: ceph.com/rbd
      parameters:
         monitors: s1-node1,s1-node2,s1-node3
            pool: rbd
            adminId: admin
            adminSecretNamespace: kube-system
            adminSecretName: ceph-secret
            userId: admin
            userSecretName: ceph-secret

apiVersion: v1
kind: Secret
metadata:
name: ceph-secret
type: "kubernetes.io/rbd"
   data:
      key: test
```

Da wir im storage.yaml ceph-secret definiert haben, muss auf dem Node1 das Passwort in Base 64 encoded werden.

```
$ grep key /etc/ceph/ceph.client.admin.keyring |awk '{printf "%s", $NF}'|base64
```

Anschliessend werden die einznen Konfigs in Kubernetes eingespielt

```
$ kc create -f provisioner.yaml
$ kc create -f secret.yaml
$ kc create -f secret.yaml --namespace=kube-system
$ kc create -f storage.yaml storageclass "ceph" created
```

Jetzt da alles eingerichtet ist, wir PVC

```
$ kc get pvc
NAME            STATUS    VOLUME                                     CAPACITY   ACCESS MODES
pvc-guestbook   Bound     pvc-51714cbe-13d7-11e8-a958-42010a840011   1Gi        RWO
```

Guestboot installieren

Source auf Jumphost laden

```
$ scp -r guestbook-go student1@test:/home/student1/
```

Nun können wir die Applikation erstellen:

```
$ kc create -f redis-master-controller.json
$ kc create -f redis-master-service.json
$ kc create -f redis-slave-controller.json
$ kc create -f redis-slave-service.json
$ kc create -f guestbook-controller.json
$ kc create -f guestbook-service.json
```

```
$ gcloud compute instances list
NAME      ZONE            MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP     STATUS
jumphost  europe-west1-d  f1-micro                    10.132.0.2   35.205.224.103  RUNNING
s1-node1  europe-west1-d  n1-standard-2               10.132.0.17  35.187.68.150   RUNNING
s1-node2  europe-west1-d  n1-standard-2               10.132.0.16  35.195.163.210  RUNNING
s1-node3  europe-west1-d  n1-standard-2               10.132.0.15  35.195.143.143  RUNNING
```

Testing

Tests konnten nur bedingt ausgeführt werden. Das Guestbook konnte per external IP aufgerufen werden. Jedoch hatte die Applikation probleme auf die Datenbank zuzugreiffen. Bei einem erneuten start des Kubernetes Clusters rauchte jedoch die ganze Umgebung ab.

Bei Lars konnte ich jedoch nachfollziehen, dass wenn das Volume abgeschalten ist, die Einträge im Guestbook gelöscht werden.

Bei einem Restart der Pods sind die Einträge jedoch noch vorhanden.

Schwierigkeiten

Die meisten Schwierigkeiten hatte ich mit dem Ceph Setup. Ich musste hierbei alle VMs von neuem bauen, da mein Setup die Nodes nicht hinzfügen konnte.

Vorteile

Die Vorteile von einem System wie Kubernetes sind, dass sich um Konfiguration auf einzelnen Servern sowie deren Wartung keine Gedanken mehr gemacht werden müssen. Es ist lediglich noch der Kubernetes Kluster vorhanden welcher gewartet werden muss. Dies erleichtert die Arbeit für den Betrieb vehement.

Lessons learend

Was ich vor allem gelernt habe, ist dass hinter dem Setup von Kubernetes ziemlich viel Magic passiert. Ohne Dokumentation würde ich wahrscheinlich nur mit emensem Zeitaufwand ein solches Setup erarbeiten können.

Trotz allem war es sehr interresant mit den einzelnen Komponenten in berührung zu kommen.