A storage solution

ACIT4430 - Biraveen Nedunchelian & Kim Phan

Table of content

Task

Alternative 3 - A storage solution

The media department from the company requires a storage solution via Kubernetes. They need:

- Ceph storage that can scale, orchestrated with Rook Links to an external site.
- Distributed persistent volumes in Kubernetes
- Data should be replicated to all worker nodes in the cluster excluding the master node
- If you're using the storage solution to save monitoring data from Prometheus/Sensu or adjacent systems, then the Observability Stack Links to an external site. is recommended

Kubernetes setup

What is kubespray?

According to Kubecost ``Kubespray is an open-source tool that allows for the automated deployment of Kubernetes clusters across computing resources, such as virtual machines. Built to be configurable, fast, and lightweight, Kubespray meets most needs.” [1]

What is Ansible?

According to Ansible themselves, “Ansible is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. It is free to use, and the project benefits from the experience and intelligence of its thousands of contributors.” [2]

Minimum requirements for kubespray

•Master Nodes: 4 GB RAM, 2 CPU and 40 GB free disk space
• Worker Nodes: 2 GB, 2 CPU, 40 GB free disk space
• Ansible Node: 1 GB, 1 CPU and 40 GB disk space
• Internet connectivity on each node
• Regular with sudo admin rights

To set up our kubernetes cluster, we decided to use the kubespray using the and follow the recommended way to set it up, which was provided in the following pdf. The setup process int swiftly and the only “manual” labor we had to perform was to:

- Install python 3
- Clone the kubespray git repository
- CD into the kubespray folder
- Install the requirements.txt
- Create ssh-keys and give the new vm¨s them so that the ansible nodes had access yo the kubernetes nodes
- Setup the other VM´s and plot their respective ip-addresses into the yaml file
- Then run the ansible playbook

The setup process took about 10-minutes and the ansible playbook used around 30 minutes to finish.

Below is a picture of the vm´s we decided to use. You can see that the “node-5” is different from the rest. We will explain why further down

Displaying 6 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
node-5	Ubuntu-22.04-LTS	10.196.36.89	csh.2c2r	ansiblenode_masternodemac	Active	nova	None	Running	3 days, 1 hour	Create Snapshot
node-4	Ubuntu-22.04-LTS	10.196.36.166	css.2c6r.20g	ansiblenode_masternodemac	Active	nova	None	Running	3 days, 1 hour	Create Snapshot
node-3	Ubuntu-22.04-LTS	10.196.39.84	css.2c6r.20g	ansiblenode_masternodemac	Active	nova	None	Running	3 days, 1 hour	Create Snapshot
node-2	Ubuntu-22.04-LTS	10.196.37.150	css.2c6r.20g	ansiblenode_masternodemac	Active	nova	None	Running	3 days, 1 hour	Create Snapshot
node-1	Ubuntu-22.04-LTS	10.196.36.34	css.2c6r.20g	ansiblenode_masternodemac	Active	nova	None	Running	3 days, 1 hour	Create Snapshot
ansiblenode	Ubuntu-22.04-LTS	10.196.37.18	C4R6_10G	Bira_mac_key	Active	nova	None	Running	5 days	Create Snapshot

Why did we choose 5 nodes?
At first we decided to use 4, but when we deployed our ceph cluster we ended up getting a lot of errors. We didn't realize why at first. After a good amount of research we figured out that it was because of the number of nodes we had. It turns out that Ceph works best with an odd number of nodes. The reason begins if we have an even number of nodes. The nodes can end up in a conflict loop (2 v 2), that can end up in the nodes staying in a stalemate, but with an even number of nodes, it's possible to sway in either direction. We also chose the flavor and specs based on what we had left of our limit.

In short, according to the pdf provided. The way to initiate a kubernetes cluster can simply be defined in a short list of multiple steps:
-Configuring a Kubespray Node (usually your Ansible control node)
- Enabling SSH access to the nodes (which is something that is expected to be already done, especially if Terraform is used to deploy cloud instances to be controlled by Ansible).
- Disabling Firewall and enabling ipv4 forwarding (just in case it is not so)
- Starting the Kubernetes deployment
- Accessing the cluster
- Enabling the Kubernetes GUI dashboard

Backup and Recovery

The easiest and quickest way to create a backup is by creating a snapshot in openstack. It's also not the best way. The best solution is to use a ceph which we deployed using the rook orchestrator.

ceph storage
According to Ubuntu “Ceph is an open source software-defined storage solution designed to address the block, file and object storage needs of modern enterprises. Its highly scalable architecture sees it being adopted as the new norm for high-growth block storage, object stores, and data lakes. Ceph provides reliable and scalable storage while keeping CAPEX and OPEX costs in line with underlying commodity hardware prices.” [3]

Rook storage orchestrator:
Rook storage orchestrator is an orchestrator tool which uses the kubernetes tool operator to manage a ceph storage cluster running inside a kubernetes cluster. ITs there to simplify the deployment for us

How did we set up ceph

To setup Ceph i used the “
Guide - Installing Ceph Storage on Kubernetes Cluster using the Rook Storage Orchestrator.pdf” Provided to us by the school.

Step 1: Adding raw devices/partitions to nodes

To add partitions to the nodes was done differently than the one shown in the pdf provided, instead we went into our openstack ui(https://cloud.cs.oslomet.no/) → volumes and then created the volumes separately and attached it to them. All the raw partitions created for ceph are noted as “/dev/sdb on node-X” under the “Attached To” section.

Displaying 12 items

Name	Description	Size	Status	Group	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
node5	-	20GiB	In-use	-	__DEFAULT__	/dev/sdb on node-5	nova	No	No	Edit Volume
node4	-	20GiB	In-use	-	__DEFAULT__	/dev/sdb on node-4	nova	No	No	Edit Volume
node3	-	20GiB	In-use	-	__DEFAULT__	/dev/sdb on node-3	nova	No	No	Edit Volume
node2	-	20GiB	In-use	-	__DEFAULT__	/dev/sdb on node-2	nova	No	No	Edit Volume
node1	-	20GiB	In-use	-	__DEFAULT__	/dev/sdb on node-1	nova	No	No	Edit Volume
6893d8c2-57ae-4de2-bc3c-0740980d4ca8	-	30GiB	In-use	-	__DEFAULT__	/dev/sda on node-5	nova	Yes	No	Edit Volume
d946a7ea-69cf-415e-b964-97a7ac65760d	-	20GiB	In-use	-	__DEFAULT__	/dev/sda on node-3	nova	Yes	No	Edit Volume
81f6282f-43f2-4daf-a54c-0ba202feca7d	-	20GiB	In-use	-	__DEFAULT__	/dev/sda on node-4	nova	Yes	No	Edit Volume
83ef8cf9-fb42-49d1-926d-7f0f41326e06	-	20GiB	In-use	-	__DEFAULT__	/dev/sda on node-2	nova	Yes	No	Edit Volume
f25489b7-c4ff-4a1e-8e2a-d43dce9ba0a1	-	20GiB	In-use	-	__DEFAULT__	/dev/sda on node-1	nova	Yes	No	Edit Volume
mackub	-	50GiB	In-use	-	__DEFAULT__	/dev/sdb on ansible node	nova	No	No	Edit Volume
2d35d36d-8e4c-433f-822a-69100242f733	-	10GiB	In-use	-	__DEFAULT__	/dev/sda on ansible node	nova	Yes	No	Edit Volume

Displaying 12 items

This is a process which is easy to automate in terraform for our juniors deployment

Step 2: Deploy Rook Storage Orchestrator

To deploy the Rook storage orchestrator we first had to clone the git by typing “Git clone –single-branch –branch release-1.11 https://github.com/rook/rook.git

Then we had to “cd” into the rook/deploy/examples folder and use “kubectl” commands to create these different yaml files:

- crds.yaml
- common.yaml
- operator.yaml

These yaml files created the necessary pods for our setup ceph cluster.

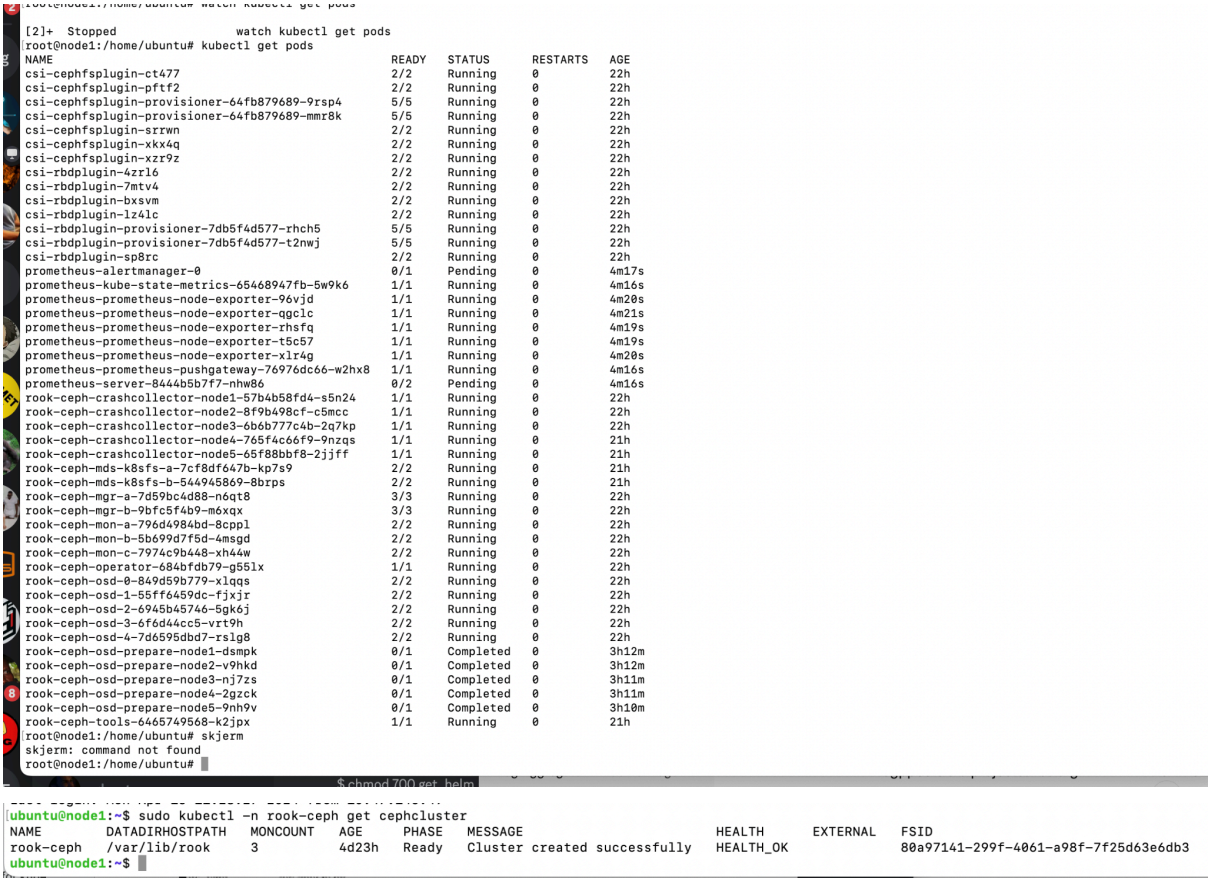
Step 3: Create Ceph Cluster on Kubernetes

After creating those pods, we changed the default namespace to “rook-ceph” by entering this command:

```
# kubectl config set-context --current --namespace rook-ceph
Context "kubernetes-admin@kubernetes" modified.
```

Since the ceph cluster can discover raw partitions by itself, it is okay to use the default cluster deployment manifest file without any modifications. So we now run the “Kubectl create -f cluster.yaml file”-command and wait for our cluster to deploy.

PS- Since we had some errors(because we used 4 nodes instead of an odd number of nodes) along the way before this we decided to let this setup overnight. On our last try we finally managed to set up our cluster. Below is a picture of all out pods and the health of our ceph cluster showing its active



Step 4: Deploy Rook Ceph toolbox in kubernetes

The rook-ceph toolbox is a container which is commonly used for rook debugging and testing. The toolbox is based on CentOS and a added benefit is that any additional tools can be installed via yum commands

To set up the toolbox, we just navigate inside the “rook/deploy/examples' '-folder and write in the command “kubectl apply -f toolbox.yaml' ', then we wait for about 1 minute.

To connect to the pod we use the exec option

```
[root@k8s-bastion ~]# kubectl -n rook-ceph exec -it deploy/rook-ceph-tools --
bash
[root@rook-ceph-tools-96c99fbf-qb9cj /]#
```

From there i can use bash commands such as:

- ceph status → to check cluster health
- ceph osd status → to check the OSD´s current statue
- ceph df → to check raw storage and pools
    - fun fact: we never had the k8fs-data0 datapool, so we ended up using the k8sfs-replicated pool instead

```
ubuntu@node1:~/rook/deploy/examples$ sudo su
root@node1:/home/ubuntu/rook/deploy/examples# kubectl -n rook-ceph exec -it deploy/rook-ceph-tools
-- bash
error: you must specify at least one command for the container
--: command not found
root@node1:/home/ubuntu/rook/deploy/examples# kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash
bash-4.4$ ceph osd lspools
1 .mgr
2 k8sfs-metadata
3 k8sfs-replicated
4 replicapool
bash-4.4$
```

Step 5: Working with Ceph Storage Modes

By rook we already have three types of storage exposed by Rook:

- Shared File System: Create a filesystem to be shared across multiple pods (RWX)
- Block: Create block storage to be consumed by a pod (RWO)
- Object: Create an object store that is accessible inside or outside the Kubernetes cluster

Cephas

We still have to be in the rook/deploy/examples folder. While in here we have to edit the filesystem.yaml file, we had to enter and set the pool name and replication size. and then do a “kubectl create -f filesystem.yaml”

all the datapools should then have been created, but for some reason we didn't get the last datapool named “k8fs-data " which was needed for this setup. We coulndt find the reason as to why it didn't get created, so instead we opted to use the k8fs-replicated datapool as an alternative.

We then “nano-ed” into csi/cephfs/storageclass.yaml and changed the configurations. We didn't use k8fs-data0 as mentioned in the setup. instead used k8fs-replicated. Below is what the pdf says we should do.

Update the *fsName* and pool name in Cephfs *Storageclass* configuration file:

```
$ vim csi/cephfs/storageclass.yaml
parameters:
  clusterID: rook-ceph # namespace:cluster
```

```
  fsName: k8sfs
  pool: k8fs-data0
```

and below is what we actually did(look at “pool:”)

biraveennedunchelian — root@node1: /home/ubuntu/rook/deploy/examples — ssh ubuntu@10.196.36.34 — 154×27

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: rook-cephfs
# Change "rook-ceph" provisioner prefix to match the operator namespace if needed
provisioner: rook-ceph.cephfs.csi.ceph.com # driver:namespace:operator
parameters:
  # clusterID is the namespace where the rook cluster is running
  # If you change this namespace, also change the namespace below where the secret namespaces are defined
  clusterID: rook-ceph # namespace:cluster

  # CephFS filesystem name into which the volume shall be created
  fsName: k8sfs

  # Ceph pool into which the volume shall be created
  # Required for provisionVolume: "true"
  pool: k8sfs-replicated

  # The secrets contain Ceph admin credentials. These are generated automatically by the operator
  # in the same namespace as the cluster.
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: rook-ceph # namespace:cluster
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: rook-ceph # namespace:cluster
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-cephfs-node
  csi.storage.k8s.io/node-stage-secret-namespace: rook-ceph # namespace:cluster
"csi/cephfs/storageclass.yaml" 36L, 1691B                                    1,1           Top
```

I then created the PVC and pod to test the usage of persistent volume by writing the command “kubectl create -f csi/cephfs/pvc.yaml”. below is a picture of the PC creation manifest file content: we will come back to this later

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cephfs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: rook-cephfs
```

RBD

according to the pdf provided "Block storage allows a single pod to mount storage (RWO mode). Before Rook can provision storage, a **StorageClass** and **CephBlockPool** need to be created”. To do so, we just have to write these commands:

```
[root@k8s-bastion ~]# cd
[root@k8s-bastion ~]# cd rook/deploy/examples
[root@k8s-bastion csi]# kubectl  create  -f csi/rbd/storageclass.yaml
cephblockpool.ceph.rook.io/replicapool created
storageclass.storage.k8s.io/rook-ceph-block created

[root@k8s-bastion csi]# kubectl  create  -f csi/rbd/pvc.yaml
persistentvolumeclaim/rbd-pvc created
```

To list them i just have to write this command:

```
[root@node1:/home/ubuntu/rook/deploy/examples# kubectl get sc
NAME              PROVISIONER                  RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
rook-ceph-block   rook-ceph.rbd.csi.ceph.com   Delete          Immediate           true                   4d22h
rook-cephfs       rook-ceph.cephfs.csi.ceph.com   Delete          Immediate           true                   4d22h
root@node1:/home/ubuntu/rook/deploy/examples#
```

Step 6: Accessing Ceph Dashboard

The dashboards felt more of a novelty setup on this project, but we still tried to set it up. according to the pdf. “The Ceph dashboard gives you an overview of the status of your Ceph cluster:

- The overall health
- The status of the mon quorum

• The status of the mgr, and osds
• Status of other Ceph daemons
• View pools and PG status
• Logs for the daemons, and much more”

For this part you are supposed to port forward using this command:

```
$ kubectl port-forward service/rook-ceph-mgr-dashboard 8443:8443 -n rook-ceph
```

and then run this command to get the credentials:

Login username is **admin** and password can be extracted using the following command:

```
kubectl -n rook-ceph get secret rook-ceph-dashboard-password -o
jsonpath="{['data']['password']}" | base64 --decode && echo
```

then:

To create a service with the NodePort, save this yaml as ***dashboard-external-https.yaml***.

```
# cd
# vim dashboard-external-https.yaml
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-mgr-dashboard-external-https
  namespace: rook-ceph
  labels:
    app: rook-ceph-mgr
    rook_cluster: rook-ceph
spec:
  ports:
  - name: dashboard
    port: 8443
    protocol: TCP
    targetPort: 8443
  selector:
    app: rook-ceph-mgr
    rook_cluster: rook-ceph
  sessionAffinity: None
  type: NodePort
```

Create a service that listens on Node Port:

```
[root@k8s-bastion ~]# kubectl create -f dashboard-external-https.yaml
service/rook-ceph-mgr-dashboard-external-https created
```

This was all we needed to set up the dashboard, but for some unknown reason we could not access the page through normal means. The only way we could check if the page was up and running was by curling it inside the vm as shown below:


node5 Ready <none> 22h v1.29.3 10.196.36.89 <none> Ubuntu 22.04.3 LTS 5.15.0-82-generic containerd://1.7.13
root@node1:/home/ubuntu/.ssh# curl -kL https://10.233.48.91:8443
<!DOCTYPE html><html lang="en-US" dir="ltr"><head>
<meta charset="utf-8">
<title>Ceph</title>
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link rel="icon" type="image/x-icon" id="cdFavicon" href="favicon.ico">
<style>@charset "UTF-8";:root{--white:#fff;--gray-100:#f8f9fa;--gray-200:#e9ecef;--gray-300:#dee2e6;--gray-400:#ced4da;--gray-500:#adb5bd;--gray-600:#6c757d;--gray-700:#495057;--gray-800:#343a40;--gray-900:#212529;--black:#000;--blue:#007bff;--indigo:#6610f2;--purple:#6f42c1;--pink:#a94442;--red:#dc3545;--orange:#fd7e14;--yellow:#d48200;--green:#008a00;--teal:#20c997;--cyan:#17a2b8;--barley-white:#fcecba;--primary:#25828e;--primary-500:#2b99a8;--secondary:#374249;--success:#008a00;--info:#25828e;--warning:#d48200;--danger:#dc3545;--light:#f8f9fa;--dark:#343a40;--green-300:#6ec664;--cyan-300:#009596;--purple-300:#a18fff;--light-blue-300:#35caed;--gold-300:#f4c145;--light-green-300:#ace12e;--accent:#25828e;--warning-dark:#fd7e14;--fg-color-over-dark-bg:#fff;--fg-hover-color-over-dark-bg:#adb5bd;--body-color-bright:#f8f9fa;--body-bg:#fff;--body-color:#212529;--body-bg-alt:#e9ecef;--health-color-error:#dc3545;--health-color-healthy:#008a00;--health-color-warning:#d48200;--health-color-warning-800:#9d6d10;--chart-color-red:#dc3545;--chart-color-blue:#06c;--chart-color-orange:#ef9234;--chart-color-yellow:#f6d173;--chart-color-green:#008a00;--chart-color-gray:#ededed;--chart-color-cyan:#2b99a8;--chart-color-purple:#3c3d99;--chart-color-center-text:#151515;--chart-color-center-text-description:#72767b;--chart-color-tooltip-background:#000;--chart-danger:#c9190b;--font-family-sans-serif:"Helvetica Neue", Helvetica, Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";--card-cap-bg:#f8f9fa;--grid-gutter-width:30px;--datatable-divider-color:rgba(0, 0, 0, .09);--nav-tabs-margin-bottom:1rem;--tooltip-color:#fff;--tooltip-bg:#212529;--tooltip-opacity:1;--screen-sm-min:576px;--screen-md-min:768px;--screen-lg-min:992px;--screen-xl-min:1200px;--screen-xs-max:575px;--screen-sm-max:767px;--screen-md-max:991px;--screen-lg-max:1199px;--navbar-height:43px}:root{--bs-blue:#007bff;--bs-indigo:#6610f2;--bs-purple:#6f42c1;--bs-pink:#a94442;--bs-red:#dc3545;--bs-orange:#fd7e14;--bs-yellow:#d48200;--bs-green:#008a00;--bs-teal:#20c997;--bs-cyan:#17a2b8;--bs-white:#fff;--bs-gray:#6c757d;--bs-gray-dark:#343a40;--bs-accent:#25828e;--bs-warning-dark:#fd7e14;--bs-primary:#25828e;--bs-secondary:#374249;--bs-success:#008a00;--bs-info:#25828e;--bs-warning:#d48200;--bs-danger:#dc3545;--bs-light:#f8f9fa;--bs-dark:#343a40;--bs-font-sans-serif:"Helvetica Neue", Helvetica, Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";--bs-font-monospace:SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;--bs-gradient:linear-gradient(180deg, rgba(255, 255, 255, .15), rgba(255, 255, 255, 0))}*,*:before,*:after{box-sizing:border-box}@media (prefers-reduced-motion: no-preference){:root{scroll-behavior:smooth}}body{margin:0;font-family:var(--bs-font-sans-serif);font-size:1rem;font-weight:400;line-height:1.5;color:#212529;background-color:#fff;-webkit-text-size-adjust:100%;-webkit-tap-highlight-color:rgba(0,0,0,0)}html{background-color:#fff}html,body{font-size:12px;height:100%;width:100%}</style><link rel="stylesheet" href="styles.8b6796664b673424.css" media="print" onload="this.media='all'"><noscript><link rel="stylesheet" href="styles.8b6796664b673424.css"></noscript></head>
<body>
<noscript>
<div class="noscript container"
ng-if="false">

<h2 i18n>JavaScript required!</h2>
<p i18n>A browser with JavaScript enabled is required in order to use this service.</p>
<p i18n>When using Internet Explorer, please check your security settings and add this address to your trusted sites.</p>

</noscript>
<cd-root></cd-root>
<script src="runtime.4fd39655e7ea619b.js" type="module"></script><script src="polyfills.4b60b22744014b0b.js" type="module"></script><script src="scripts.cfd741a72b67f696.js" defer></script><script src="main.ddd4de0999172734.js" type="module"></script>
</body></html>root@node1:/home/ubuntu/.ssh# kubectl -n rook-ceph get cephcluster
NAME DATADIRHOSTPATH MONCOUNT AGE PHASE MESSAGE HEALTH EXTERNAL FSID
rook-ceph /var/lib/rook 3 22h Ready Cluster created successfully HEALTH_OK 80a97141-299f-4061-a98f-7f25d63e6db3
root@node1:/home/ubuntu/.ssh#
Last inn på

Monitoring and logging

For our monitoring solutions we set up prometheus and grafana, the way we set it up was by installing and using helm using guide: https://medium.com/@gayatripawar401/deploy-prometheus-and-grafana-on-kubernetes-using-helm-5aa9d4fbae66

Prometheus

The guide was easy to follow, below you can see a picture of our prometheus pods running

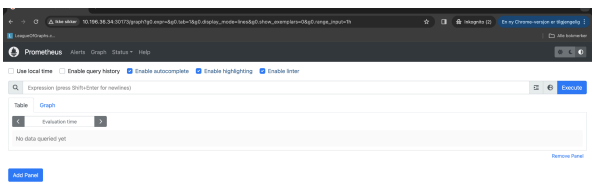
```
root@node1:/home/ubuntu# kubectl get service
NAME                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
prometheus-alertmanager                 ClusterIP   10.233.60.94    <none>        9093/TCP            23m
prometheus-alertmanager-headless        ClusterIP   None            <none>        9093/TCP            23m
prometheus-kube-state-metrics           ClusterIP   10.233.39.254   <none>        8080/TCP            23m
prometheus-prometheus-node-exporter     ClusterIP   10.233.44.6     <none>        9100/TCP            23m
prometheus-prometheus-pushgateway       ClusterIP   10.233.54.175   <none>        9091/TCP            23m
prometheus-server                       ClusterIP   10.233.27.61    <none>        80/TCP              23m
rook-ceph-mgr                           ClusterIP   10.233.39.134   <none>        9283/TCP            22h
rook-ceph-mgr-dashboard                 ClusterIP   10.233.48.91    <none>        8443/TCP            22h
rook-ceph-mon-a                         ClusterIP   10.233.63.154   <none>        6789/TCP,3300/TCP   22h
rook-ceph-mon-b                         ClusterIP   10.233.43.244   <none>        6789/TCP,3300/TCP   22h
rook-ceph-mon-c                         ClusterIP   10.233.20.111   <none>        6789/TCP,3300/TCP   22h
```

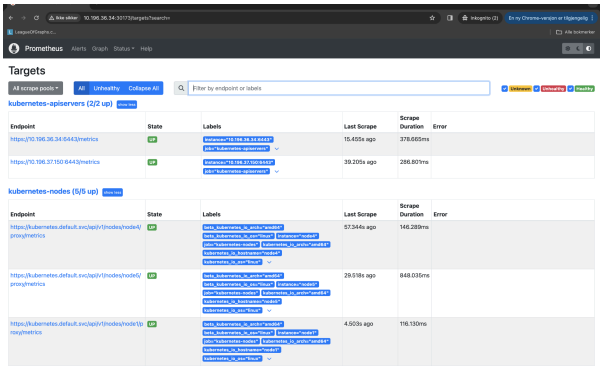
To get prometheus up and running we had to expand our storageclass form 1Gi to 10Gi and also add the metadata prometheus-server to the pvc.yaml file. The reasons is that we prometheus don't have enough space within 1Gi, to change this all we had to do was to change the values as mentioned, then do a "kubectl delete -f csi/cephfs/pvc.yaml" and then "kubectl create -f csi/cephfs/pvc.yaml" again

If i wanted to enter prometheus i could enter inn 1 of 5 links:

- For node1, access Prometheus at http://10.196.36.34:30173/
- For node2, access Prometheus at http://10.196.37.150:30173/
- For note 3, access Prometheus at http://10.196.39.84:30173/
- For node4, access Prometheus at http://10.196.36.166:30173/
- For node5, access Prometheus at http://10.196.36.89:30173/

Proof of our prometheus is up and running:





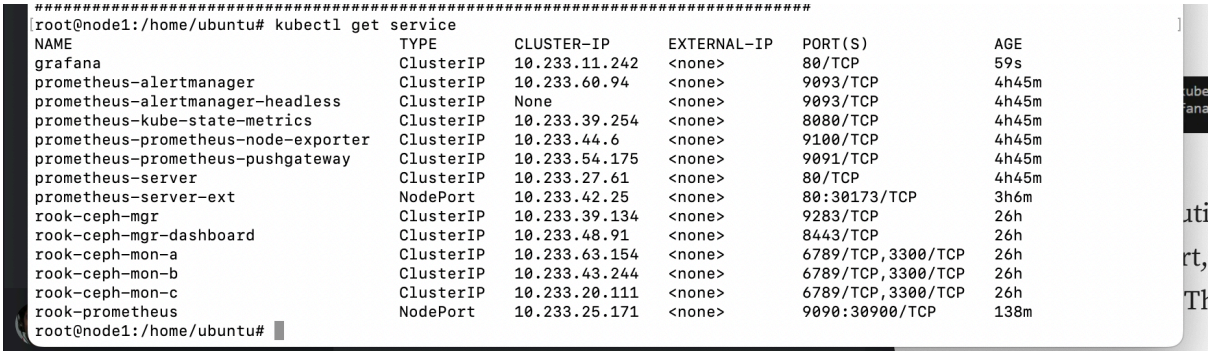
Pictures of all our pods running. Take note of "prometheus-alertmanager-0" not beginning up. This is because I've chosen not to set it up as it's not a part of our task. To get this up and running i had to put the metadata name in the pvc.yaml file and do a “delete” and then a

“create”

```
root@node1:/home/ubuntu/rook/deploy/examples# kubectl get pods -n rook-ceph | grep prometheus
prometheus-alertmanager-0                          0/1     Pending     0          157m
prometheus-kube-state-metrics-65468947fb-5w9k6     1/1     Running     0          157m
prometheus-prometheus-node-exporter-96vjd          1/1     Running     0          157m
prometheus-prometheus-node-exporter-qgclc          1/1     Running     0          157m
prometheus-prometheus-node-exporter-rhsfq          1/1     Running     0          157m
prometheus-prometheus-node-exporter-t5c57          1/1     Running     0          157m
prometheus-prometheus-node-exporter-xlr4g          1/1     Running     0          157m
prometheus-prometheus-pushgateway-76976dc66-w2hx8  1/1     Running     0          157m
prometheus-server-8444b5b7f7-nhw86                 2/2     Running     0          157m
root@node1:/home/ubuntu/rook/deploy/examples# kubectl get nodes -o wide
```

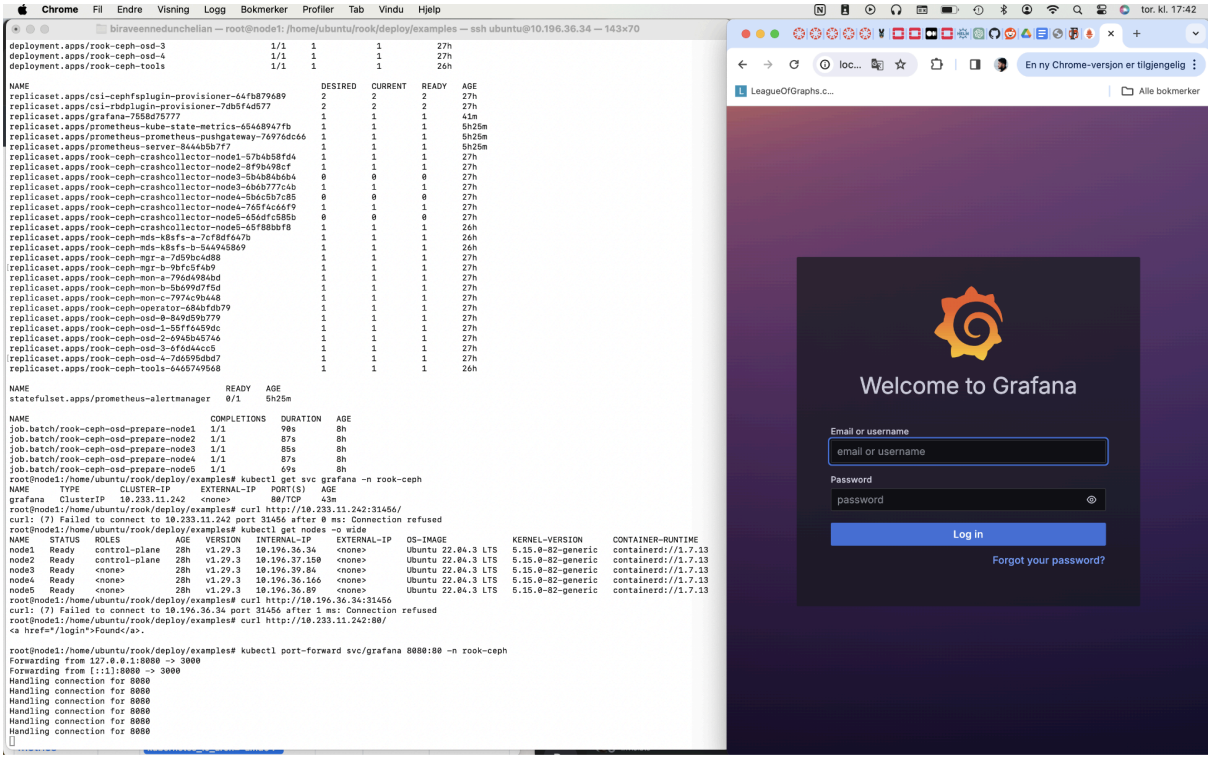
Grafana

I used Helm chart on kubernetes to install grafana on our cluster as you can see below: here i can see on top that grafana is up and running



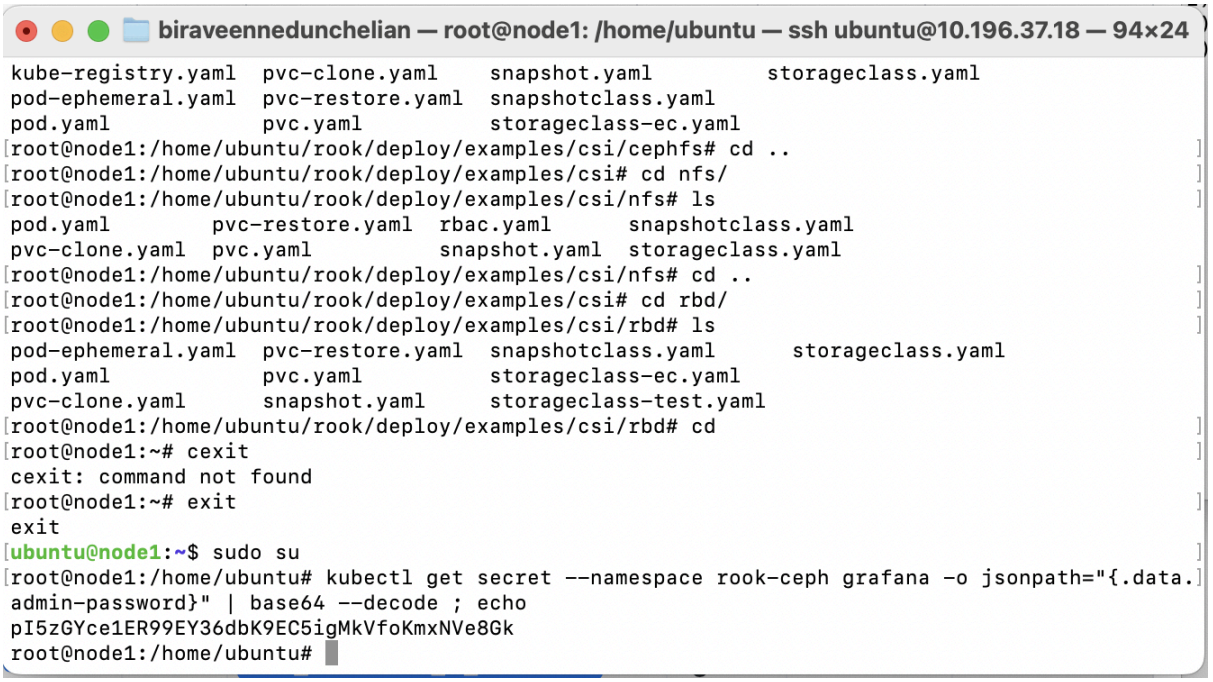
```
################################################################################
root@node1:/home/ubuntu# kubectl get service
NAME                                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
grafana                               ClusterIP   10.233.11.242   <none>        80/TCP              59s
prometheus-alertmanager               ClusterIP   10.233.60.94    <none>        9093/TCP            4h45m
prometheus-alertmanager-headless      ClusterIP   None            <none>        9093/TCP            4h45m
prometheus-kube-state-metrics         ClusterIP   10.233.39.254   <none>        8080/TCP            4h45m
prometheus-prometheus-node-exporter   ClusterIP   10.233.44.6     <none>        9100/TCP            4h45m
prometheus-prometheus-pushgateway     ClusterIP   10.233.54.175   <none>        9091/TCP            4h45m
prometheus-server                     ClusterIP   10.233.27.61    <none>        80/TCP              4h45m
prometheus-server-ext                 NodePort    10.233.42.25    <none>        80:30173/TCP        3h6m
rook-ceph-mgr                         ClusterIP   10.233.39.134   <none>        9283/TCP            26h
rook-ceph-mgr-dashboard               ClusterIP   10.233.48.91    <none>        8443/TCP            26h
rook-ceph-mon-a                       ClusterIP   10.233.63.154   <none>        6789/TCP,3300/TCP   26h
rook-ceph-mon-b                       ClusterIP   10.233.43.244   <none>        6789/TCP,3300/TCP   26h
rook-ceph-mon-c                       ClusterIP   10.233.20.111   <none>        6789/TCP,3300/TCP   26h
rook-prometheus                       NodePort    10.233.25.171   <none>        9090:30900/TCP      138m
root@node1:/home/ubuntu#
```

Then we created a ssh tunnel to connect to the cluster. we've exposed the port 8080 for accessing it through the localhost:8080 on our computer. As you can see in this picture i've managed to establish a blissful connection

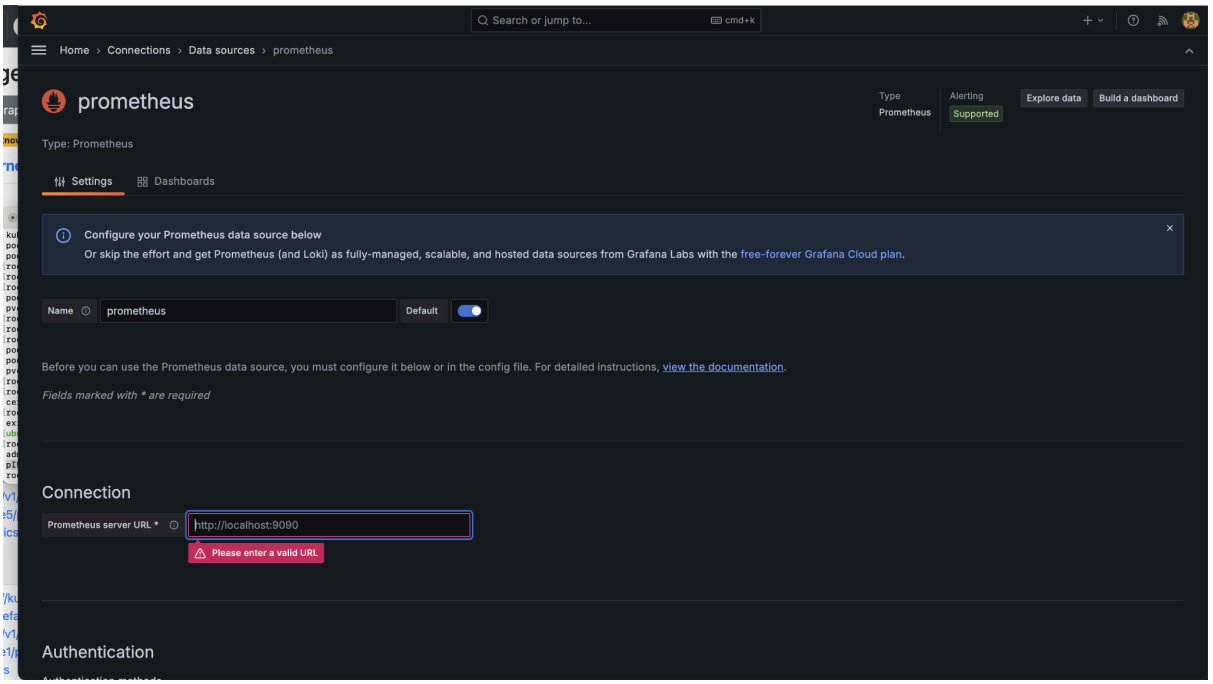


to get our admin password for grafana we had to use the command: "kubectl get secret --namespace rook-ceph grafana -o jsonpath="{.data.admin-password}" | base64 --decode ;

echo" to get it to expose it for me. In the picture you can see what the password it

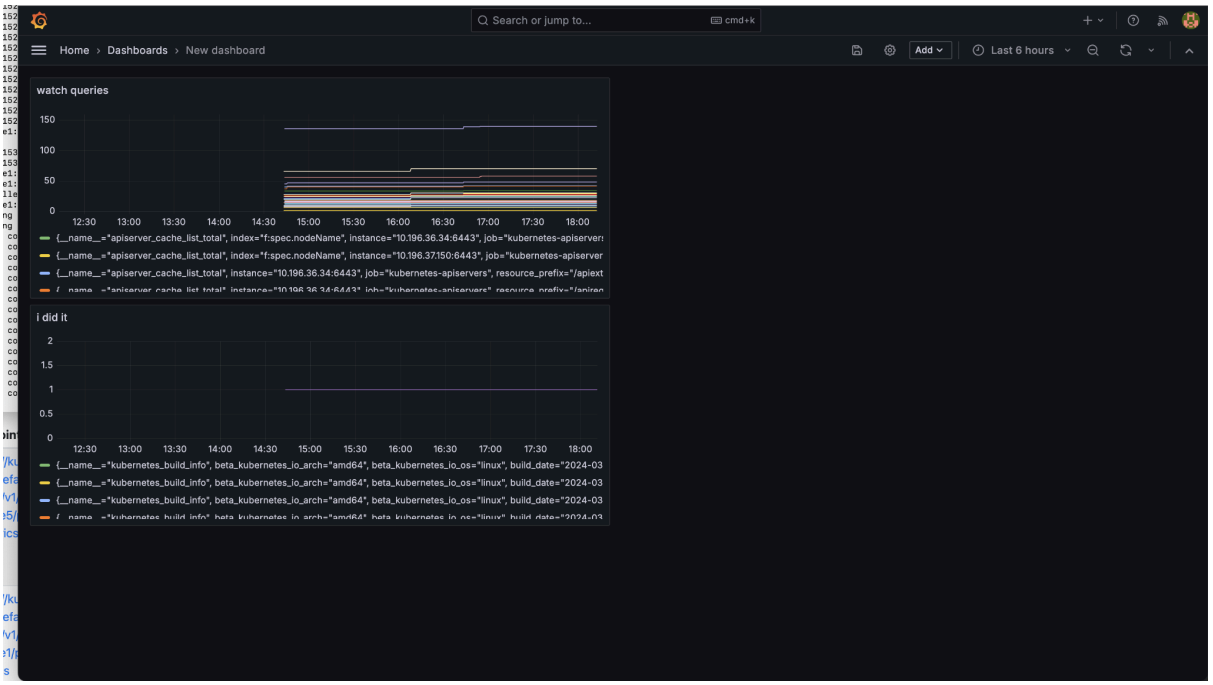


For the grafana chart we will add the url to prometheus. Which is the one i gave above:(1 of the 5: http://10.196.36.34:30173/



After connecting prometheus with grafana I've made a few visualizations. here you can see

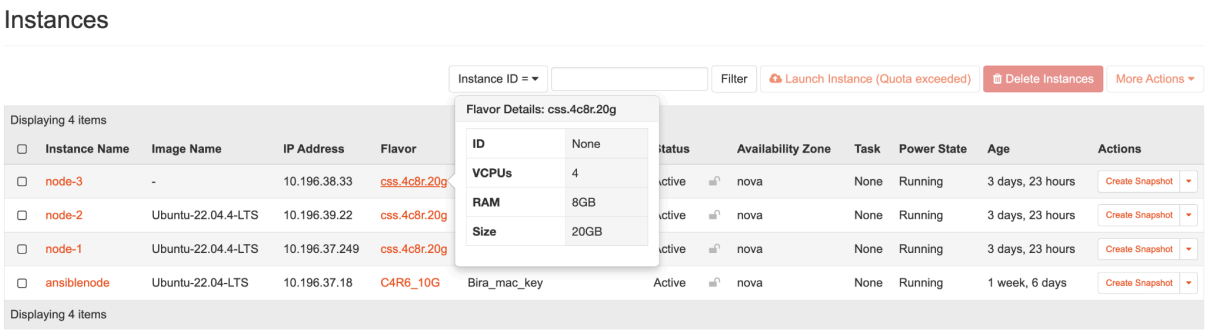
two of the graphs we made:



ELK-stack

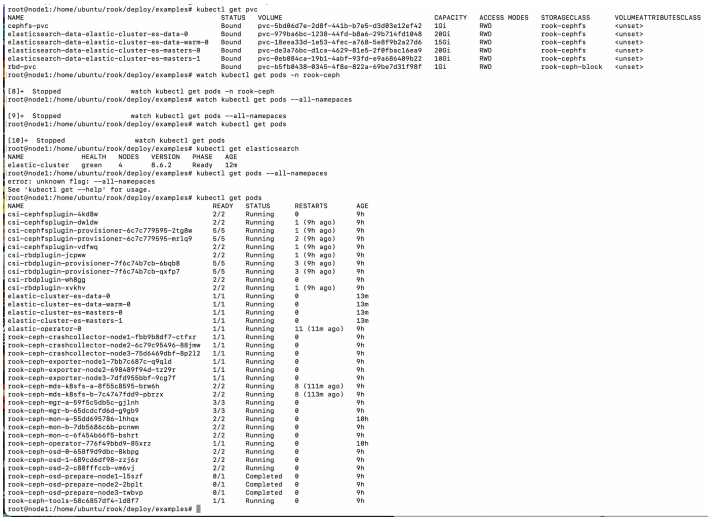
I forgot to add the pictures, but we had resource constraints and since we thought it was a good idea to restart our project we lost all our progress. What happened next is that we wasn't able to get ceph running since our class reached the limit for image pulls and we couldn't properly login on our docker account. So we was stuck :). But before this we actually got everything in elastic up and running, the only issue was that our node 5 was too weak and that we didn't have enough memory to run more things in our cluster. We tried adding more storage, but we mistook it and thought they meant volume, not ram. To install the elk stack we used a helm, the installation went swiftly. We also have to mention that we tried to make the setting to use as few resources as possible, but to no avail.(modifying the original yaml.file)

What we did instead was to then delete all our vm-^s and then create new one as you can see below:



These vms will serve as more powerful nodes to get our cluster up and running and make me avoid running into problems.

ELASTIC:
As seen below, we got our elastic cluster up and running with no issues. The only thing that we had to do was to bind the PVC to our storage-class by modifying the pvc.yaml file. The cluster used 20 minutes to get up.



The yaml file made to create the clusters can be seen in appendix 1 and appendix2 as the yaml file is way too big to be put here. The setup of the cluster is after figuring out how to bind the pods to our pvc.

Kibana and logstash:
Setting up kibana and logstash was not as easy as setting up our elastic cluster. What happened here was that our entire vms crashed and OOM-killer turned off our vms, resulting in me not gaining access to our clusters anymore. What I think happened is that we kind of nested 2 kibana yml(accidentally created 2) files into one and then everything went down. We cannot say for sure, and with the time-constraint we don't have the time to redo the project as we used a lot of time to get the ceph cluster up.

```
[ubuntu@node1:~$ sudo su
[root@node1:/home/ubuntu# kubectl get nodes
E0422 12:16:09.270624   43872 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:09.273755   43872 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:09.276880   43872 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:09.280771   43872 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:09.284174   43872 memcache.go:265] couldn't get current server API group list: unknown
Error from server (Forbidden): unknown
[root@node1:/home/ubuntu# kubectl get nodes
E0422 12:16:12.922128   43989 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:12.925645   43989 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:12.928746   43989 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:12.932287   43989 memcache.go:265] couldn't get current server API group list: unknown
E0422 12:16:12.935524   43989 memcache.go:265] couldn't get current server API group list: unknown
Error from server (Forbidden): unknown
```

Update: We got kibana and logstash running after restarting all our vms and running the kubespray again on our master node(and waiting 3 days). We did so by installing it via helm. We did not have to do anything more than that.

```
elastic-cluster-es-masters-1                      1/1     Running     1 (3h47m ago)    3d7h
elastic-operator-0                                1/1     Running     15 (15m ago)     3d16h
kibana-kb-689d7587d6-m2ztk                        1/1     Running     1 (3h47m ago)    3d7h
rook-ceph-crashcollector-node1-fbb9b8df7-ctfxr    1/1     Running     1 (3h47m ago)    3d17h
rook-ceph-crashcollector-node2-6c79c95496-88jmw   1/1     Running     1 (3h47m ago)    3d17h
rook-ceph-crashcollector-node3-75d6469dbf-8p2l2   1/1     Running     1 (3h47m ago)    3d17h
rook-ceph-exporter-node1-7bb7c687c-q9qld          1/1     Running     2 (43m ago)      3d17h
rook-ceph-exporter-node2-698489f94d-tz29r         1/1     Running     2 (41m ago)      3d17h
rook-ceph-exporter-node3-7dfd955bbf-9cg7f         1/1     Running     2 (42m ago)      3d17h
rook-ceph-mds-k8sfs-a-8f55c8595-brw6h             2/2     Running     12 (40m ago)     3d17h
rook-ceph-mds-k8sfs-b-7c4747fdd9-pbrzx            2/2     Running     11 (42m ago)     3d17h
rook-ceph-mgr-a-59f5c5db5c-gjlnh                  3/3     Running     11 (43m ago)     3d17h
rook-ceph-mgr-b-65dcdcfd6d-g9gb9                  3/3     Running     13 (42m ago)     3d17h
rook-ceph-mon-a-55dd695786-lhhqx                  2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-mon-b-7db5686c6b-pcnwm                  2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-mon-c-6f454b66f5-bshrt                  2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-operator-776f49bbd9-85xrz               1/1     Running     4 (41m ago)      3d17h
rook-ceph-osd-0-658f9d9dbc-8kbpg                  2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-osd-1-689cd6df98-zzj6r                  2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-osd-2-c88fffccb-vm6vj                   2/2     Running     2 (3h47m ago)    3d17h
rook-ceph-osd-prepare-node1-9zc86                 0/1     Completed   0                26m
rook-ceph-osd-prepare-node2-vrsgk                 0/1     Completed   0                26m
rook-ceph-osd-prepare-node3-v2q4z                 0/1     Completed   0                25m
rook-ceph-tools-58c6857df4-ld8f7                  1/1     Running     1 (3h47m ago)    3d17h
root@node1:/home/ubuntu/rook/deploy/examples# kubectl get beat
NAME         HEALTH   AVAILABLE   EXPECTED   TYPE       VERSION   AGE
quickstart                                   filebeat             3d7h
rook-ceph                                    filebeat             3d7h
root@node1:/home/ubuntu/rook/deploy/examples# kubectl describe beat quickstart -n rook-ceph
```

Discussion:

Our initial idea was to create the project and set it up separately and then create the automations and necessary files to make it easy for junior developers to set up everything by creating a few scripting files and then create another “master” scripting file to run the other scripting files in the correct order. This will make the deployment easy for our junior developers. Unfortunately due to time constraints and a few hiccups we was only able to partially finish the project.

What did work:

Setting up ceph kubernetes cluster:

By following the PDF provided by the course, we managed to set up the kubernetes using kubespray.

Setting up ceph cluster:

By setting up the ceph cluster int ill by following the pdf provided by our course

Setting up prometheus and grafana

Using our **Medium[5]** source provided to me by our class TA, we was able to use helm and setup prometheus and grafana

Setting up ELK:

By following the **Medium[6]** source we was able to set up our ELK-stack.

What didn't work:

Automation

We didn't get the automation up because we didn't finish the initial project, but if I did we would have used a combination of terraform, python and ansible to simplify the project. The way we would use them is:

- Terraform: setting up infrastructure and dividing the master-ssh-key between the other vms
- Python To automate the setup of ceph cluster, prometheus, grafana and elk-stack
- Use ansible to setup our kubernetes cluster using kubespray

Backup:

We never fully set up a backup. The only thing we did was to take a snapshot of our masternode. This in itself might be “okay” for some tasks, but for this project we wanted to

set up a backup using our ceph storage cluster. The way to that is by using Ceph RBD backup and having the backup save in another fully separate vm.

As to quote **Storeware** “When Ceph RBD is used as a backend in virtual environment (such as OpenStack) – disk attachment method means that you use a Proxy VM, which requests from the virtualization platform several operations: snapshot, volume creation and attachment of each volume to this proxy. In this scenario you can read data directly from the volume without any direct interaction with Ceph itself.”

Proposed Workflow:

Since I unfortunately didn't finish the project, I wasn't able to properly create a workflow, so instead I wanted to talk about a fictional workflow instead.
The way I wanted to set up our workflow was that I used a combination of Terraform, python and ansible. Each of them have their own respective purpose:

Terraform -
The way I wanted to use terraform is to deploy our infrastructure, by setting up our vm¨s, volumes and install the necessary packages. This should have been a terraform script that the juniors could use to just deploy and wait.

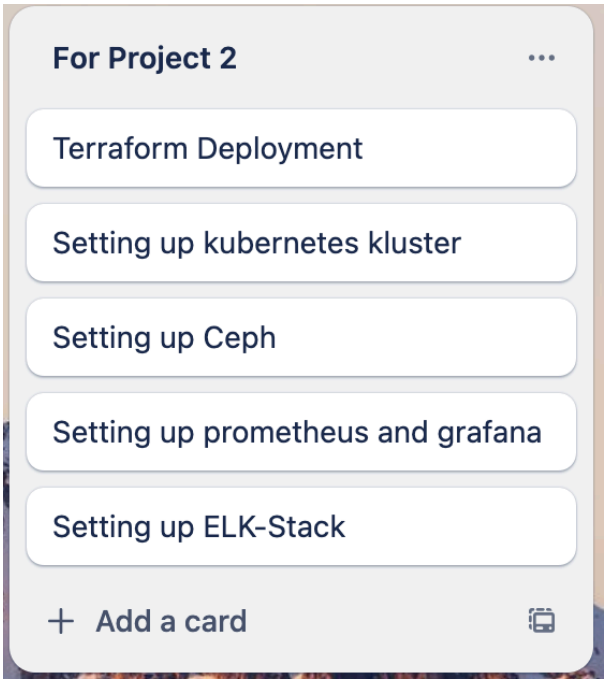
Ansible:
After the deployment of our infrastructure i wanted them to use ansible to deploy the kubernetes using kubespray with ansible. The guide they should follow is the one in the pdf provided.

Python:
At last I would use a python script to install helm and the other pods, as well as write scripts to write in the correct pvc¨s into the yaml files and also get them to apply without them needing to do much.

The solution should have been seamless and easy to set up, for our juniors. Their job should have essentially just run the files.

Kanban/trello

In our trello board, the workflow would have looked like the picture below. The terraform script will be inside our ansible node and at the end of the deployment it will give me the ip-addresses of each of the new VM¨s created. After the Terraform deployment, they will have to set up the kubernetes cluster and since a lot of the groundwork has been laid beforehand, the only thing they would have to do is to add the new vm-ip addresses. Then run the ansible playbook. To set up Ceph will be done with the rook orchestrator with the instructions provided on canvas. Prometheus, grafana and ELK will be deployed through a python script. I will make 3 separate python scripts and then make one more to run them all. The reason for this choice is because I want to separate them in case we are switching out some of them in the future.



Conclusion

To conclude our project, w will have to say that the project was somewhat successful. we didn't implement a proper backup system; nor did we have time to set up a terraform deployment script file, but overall I'm happy with what we achieved. The project was difficult, but we learned alot and the experience gained teached me to plan better and also gave me more resources to work with for our future works.

References:

[1] https://www.kubecost.com/kubernetes-devops-tools/kubespray/
[2] https://www.ansible.com/
[3] https://ubuntu.com/ceph/what-is-ceph
[4] https://storware.eu/blog/backup-strategies-for-ceph/
[5] https://medium.com/@KushanJanith/run-elastic-stack-on-kubernetes-29e295cd6531
[6]https://medium.com/@gayatripawar401/deploy-prometheus-and-grafana-on-kubernetes-using-helm-5aa9d4fbae66

APPENDIX 1:

```yaml
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elastic-cluster
spec:
  version: 8.6.2
  volumeClaimDeletePolicy: DeleteOnScaledownOnly
  nodeSets:
  - name: masters
    count: 2
    config:
      node.roles: ["master"]
    podTemplate:
      spec:
        priorityClassName: elastic-cluster-high-priority
        containers:
        - name: elasticsearch
          resources:
            limits:
              cpu: 2
              memory: 2.5Gi
            requests:
              cpu: 1
              memory: 2Gi
        initContainers:
        - name: sysctl
          securityContext:
            privileged: true
            runAsUser: 0
          command: ['sh', '-c', 'sysctl -w vm.max_map_count=262144']
    volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
  - name: data
    count: 1
    config:
      node.roles: ["data", "ingest"]
    podTemplate:
      spec:
        priorityClassName: elastic-cluster-high-priority
        containers:
        - name: elasticsearch
          resources:
            limits:
              cpu: 2
              memory: 4Gi
            requests:
              cpu: 1
              memory: 2Gi
        initContainers:
        - name: sysctl
          securityContext:
            privileged: true
            runAsUser: 0
          command: ['sh', '-c', 'sysctl -w vm.max_map_count=262144']
    volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 100Gi
```

APPENDIX 2:

```yaml
  - name: data-warm
    count: 1
    config:
      node.roles: ["data", "data_warm"]
    podTemplate:
      spec:
        priorityClassName: elastic-cluster-high-priority
        containers:
        - name: elasticsearch
          resources:
            limits:
              cpu: 1.5
              memory: 3Gi
            requests:
              cpu: 1
              memory: 2Gi
        initContainers:
        - name: sysctl
          securityContext:
            privileged: true
            runAsUser: 0
          command: ['sh', '-c', 'sysctl -w vm.max_map_count=262144']
    volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 150Gi
```

elasticsearch-cluster.yaml hosted with ❤ by GitHub view raw

APPENDIX 3:

our PVC file

```
[root@node1:/home/ubuntu/rook/deploy/examples# cat csi/cephfs/pvc.yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cephfs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: rook-cephfs
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: elasticsearch-data-elastic-cluster-es-data-0
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: rook-cephfs

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: elasticsearch-data-elastic-cluster-es-data-warm-0
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 15Gi
  storageClassName: rook-cephfs

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: elasticsearch-data-elastic-cluster-es-masters-0
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: rook-cephfs
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: elasticsearch-data-elastic-cluster-es-masters-1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: rook-cephfs
```