Deploying a multi cluster distributed telemetry stack of Prometheus, M3DB and Collectd using EMCO

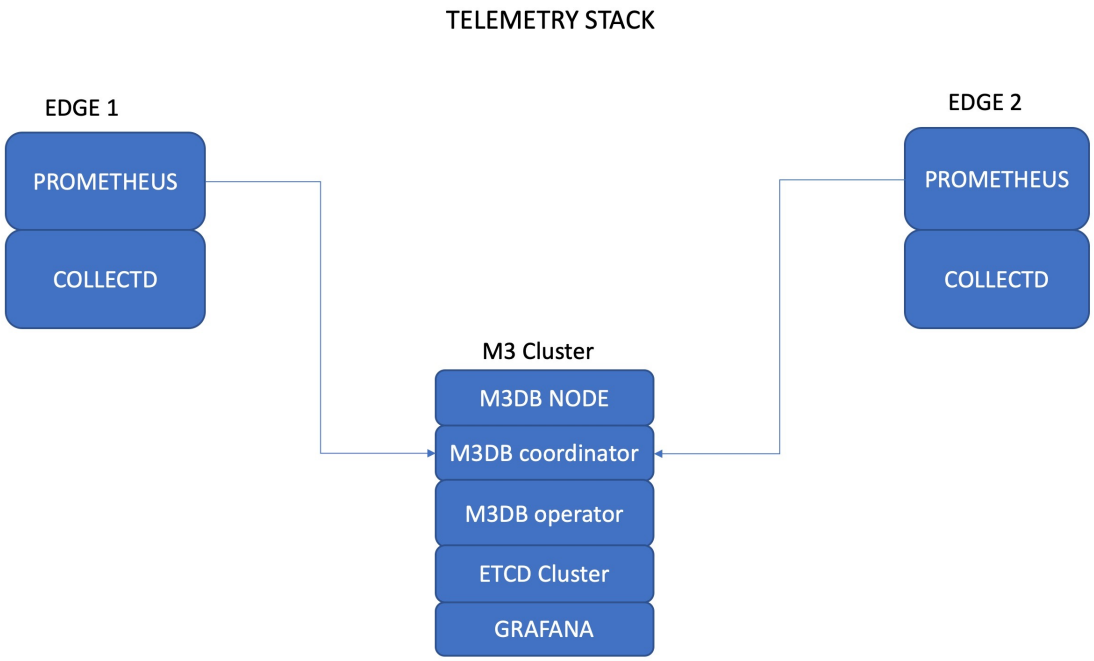
Often we require a multi cluster deployment where we need to replicate a stack across multiple locations.
These multiple clusters should be able to connect and communicate among themselves.

We tried deploying such a stack through EMCO platform.

While most of the deployment is automated, there are few steps which were done manually and each of them have been mentioned here.

So, these steps shall also be automated and the entire stack shall be a 1 click deployment through EMCO.

The components of the stack are below :



Step-by-step guide

For smooth deployment we have created 3 scripts. **One must edit the script to give the correct cluster details to each script.** For eg, kubeconfig file, cluster ips
and also the corrects URLs where EMCO binaries like orchestrator, clm etc are running.

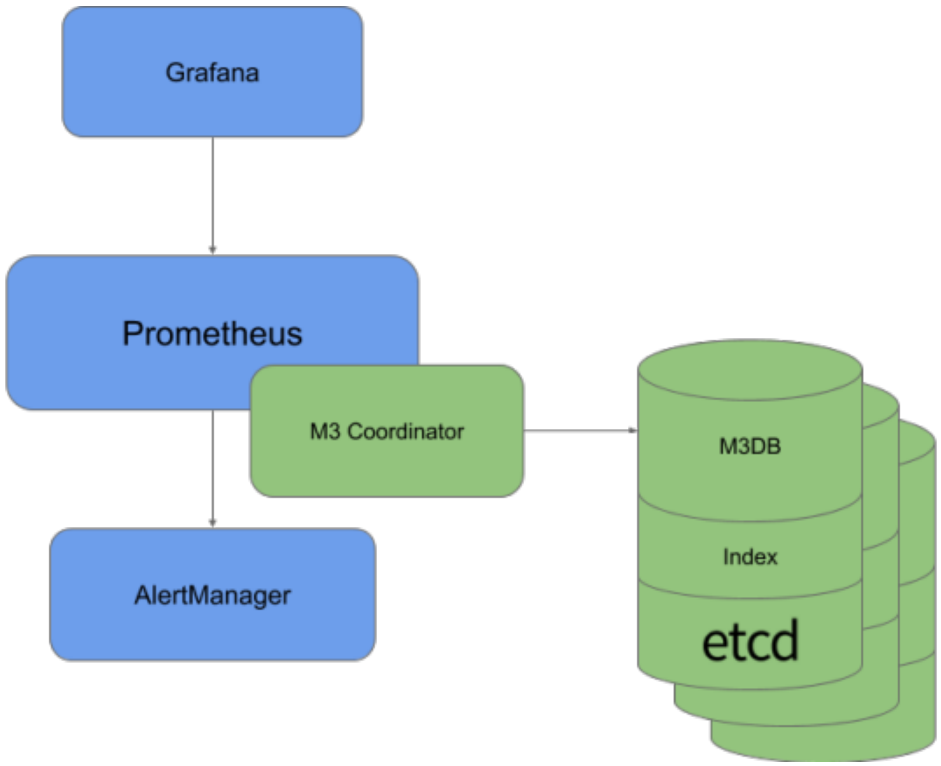
- m3operatorScript -
- This script has to run first as it deploys the m3operator.
- M3operator is a pre-requisite for the m3 cluster deployment.
- Make necessary changes in the script as per the cluster details at your end.
- Helm charts and profile can be found at : operatorHelmChartsAndProfile

After you are done with the m3Operator installation, the following pods shall be running

m3operator

```
vagrant@emco:~/multicloud-k8s/kud/tests$ kcc2 get pods
NAME                          READY   STATUS    RESTARTS   AGE
etcd-0                        1/1     Running   0          30s
etcd-1                        1/1     Running   0          18s
etcd-2                        1/1     Running   0          10s
m3db-operator-0               1/1     Running   0          30s
```

- m3dbInstallerScript -
- m3db shall be deployed only on a 3 node cluster, and the nodes shall be labelled. For labelling of the nodes, refer the manual steps in the same doc. Labelling of nodes is a pre-requisite.
- This script shall install m3db nodes.
- Helm charts and profile : m3dbHelmCharts
- This script shall also bring m3coordinator-m3db-cluster service.
- A sidecar process, M3Coordinator, allows M3DB to act as the long-term storage for Prometheus. Its responsible for integration with prometheus



After the m3installer script is run , the pods and services visible shall be :

M3DB installer

```
vagrant@emco:~/multicloud-k8s/kud/tests$ kcc2 get pods
NAME                          READY   STATUS    RESTARTS   AGE
etcd-0                        1/1     Running   0          5m11s
etcd-1                        1/1     Running   0          4m59s
etcd-2                        1/1     Running   0          4m51s
gr-grafana-5c57fbd899-llmmm   1/1     Running   0          2d5h
m3db-cluster-rep0-0           1/1     Running   0          105s
m3db-cluster-rep1-0           1/1     Running   0          84s
m3db-cluster-rep2-0           1/1     Running   0          60s
m3db-operator-0               1/1     Running   0          5m11s
vagrant@emco:~/multicloud-k8s/kud/tests$ kcc2 get svc
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT
(S)                                                           AGE
etcd                          ClusterIP   None            <none>        2379/TCP,2380
/TCP                                                  5m22s
etcd-cluster                  ClusterIP   10.244.13.90    <none>        2379
/TCP                                                          5m22s
gr-grafana                    NodePort    10.244.12.2     <none>        80:30007
/TCP                                                  2d5h
kubernetes                    ClusterIP   10.244.0.1      <none>        443
/TCP                                                       2d18h
m3coordinator-m3db-cluster    ClusterIP   10.244.36.133   <none>        7201/TCP,7203
/TCP                                                  116s
m3dbnode-m3db-cluster         ClusterIP   None            <none>        9000/TCP,9001/TCP,9002/TCP,9003/TCP,9004
/TCP,7201/TCP,7203/TCP   116s
```

NOTE : Once all the m3db pods and services are up and running, edit the service type to be NodePort as discussed in the manual steps in the same doc.

- CollectD-PrometheusScript -
- This script shall install collectd and prometheus.
- Helm charts for collectd and prometheus : collectd and prometheus
- For connecting prometheus and m3coordinator, in the remote write section of the values.yaml of the prometheus helm charts, add the following:

Connecting prometheus and m3db

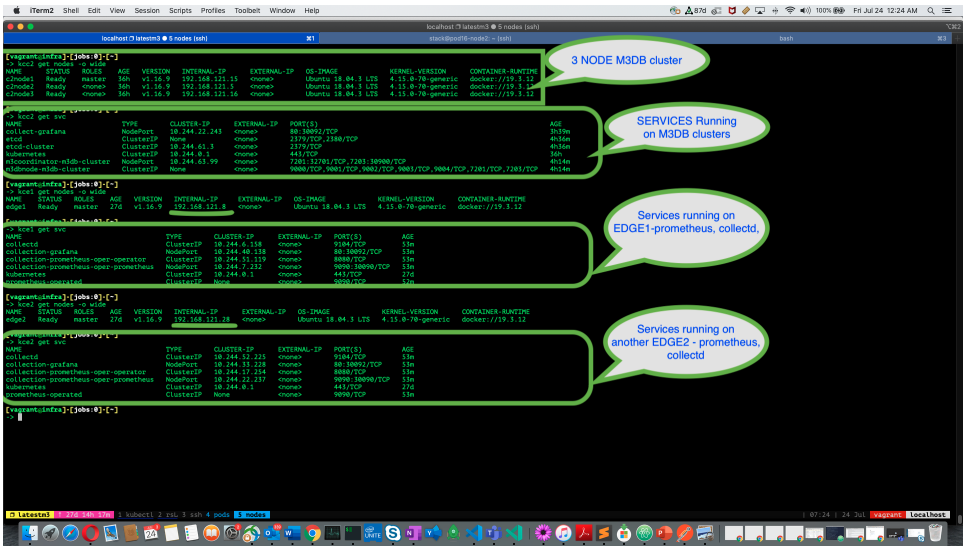
```
    remoteWrite:
    # - url: http://remote1/push
      - url: "http://192.168.121.12:31772/api/v1/prom/remote/write"
        writeRelabelConfigs:
          - targetLabel: metrics_storage
            replacement: m3db_remote
```

- NOTE , in the above 192.168.121.12 is the ip of the cluster node where m3db is running. 31772 is the NodePort of the m3coordinator.
- After running the following pods and services shall be present in the nodes edge1 and edge2

collectd-prometheus

```
vagrant@emco:~/multicloud-k8s/kud/tests$ kce1 get pods
NAME                                                     READY   STATUS    RESTARTS   AGE
collection-collectd-62z6z                                1/1     Running   0          19m
collection-grafana-5ff455f786-bcm9m                      2/2     Running   0          19m
collection-prometheus-oper-operator-b55765c57-2n7cn      1/1     Running   0          19m
prometheus-collection-prometheus-oper-prometheus-0       3/3     Running   1          19m
vagrant@emco:~/multicloud-k8s/kud/tests$ kce2 get pods
NAME                                                     READY   STATUS    RESTARTS   AGE
collection-collectd-jdvl9                                1/1     Running   0          20m
collection-grafana-5ff455f786-q9rrn                      2/2     Running   0          20m
collection-prometheus-oper-operator-b55765c57-tzchg      1/1     Running   0          20m
prometheus-collection-prometheus-oper-prometheus-0       3/3     Running   1          19m
vagrant@emco:~/multicloud-k8s/kud/tests$ kce1 get svc
NAME                                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
collectd                                 ClusterIP   10.244.20.184   <none>        9104/TCP         20m
collection-grafana                       NodePort    10.244.17.32    <none>        80:30092/TCP     20m
collection-prometheus-oper-operator      ClusterIP   10.244.24.248   <none>        8080/TCP         20m
collection-prometheus-oper-prometheus    NodePort    10.244.14.32    <none>        9090:30090/TCP   20m
kubernetes                               ClusterIP   10.244.0.1      <none>        443/TCP          2d22h
prometheus-operated                      ClusterIP   None            <none>        9090/TCP         20m
vagrant@emco:~/multicloud-k8s/kud/tests$ kce2 get svc
NAME                                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
collectd                                 ClusterIP   10.244.13.251   <none>        9104/TCP         20m
collection-grafana                       NodePort    10.244.41.253   <none>        80:30092/TCP     20m
collection-prometheus-oper-operator      ClusterIP   10.244.10.220   <none>        8080/TCP         20m
collection-prometheus-oper-prometheus    NodePort    10.244.31.123   <none>        9090:30090/TCP   20m
kubernetes                               ClusterIP   10.244.0.1      <none>        443/TCP          2d22h
prometheus-operated                      ClusterIP   None            <none>        9090/TCP         20m
```

- Once all pods are deployed correctly, the topology should be like below:



Manual steps for getting the m3db up and running:

As pointed out earlier, we are working on some steps which are not automated for the deployment of the stack.
In due course of time, these might as well be automated. But these are steps till then :

1. Three node cluster on which m3db needs to be deployed have to be labelled, before the m3db script is run. The commands :

NodeLabelling

```
NODES=($(kubectl get nodes --output=jsonpath={.items..metadata.name}))
kubectl label node/${NODES[0]} failure-domain.beta.kubernetes.io/region=us-west1-a
kubectl label node/${NODES[1]} failure-domain.beta.kubernetes.io/region=us-west1-b
kubectl label node/${NODES[2]} failure-domain.beta.kubernetes.io/region=us-west1-c
kubectl label node/${NODES[0]} failure-domain.beta.kubernetes.io/zone=us-west1-a --overwrite=true
kubectl label node/${NODES[1]} failure-domain.beta.kubernetes.io/zone=us-west1-b --overwrite=true
kubectl label node/${NODES[2]} failure-domain.beta.kubernetes.io/zone=us-west1-c --overwrite=true
```

2. Create db namespace and bootstrap m3 db - only if required

Bootstrap M3db

```
kubectl -n training port-forward svc/m3coordinator-m3db-cluster 7201

curl -vvv -X POST http://localhost:7201/api/v1/database/create -d '{
  "type": "cluster",
  "namespaceName": "collectd",
  "retentionTime": "168h",
  "numShards": "64",
  "replicationFactor": "3",
  "hosts": [
        {
            "id": "m3db-cluster-rep0-0",
            "isolationGroup": "us-west1-a",
            "zone": "embedded",
            "weight": 100,
            "address": "m3db-cluster-rep0-0.m3dbnode-m3db-cluster:9000",
            "port": 9000
        },
        {
            "id": "m3db-cluster-rep1-0",
            "isolationGroup": "us-west1-b",
            "zone": "embedded",
            "weight": 100,
            "address": "m3db-cluster-rep1-0.m3dbnode-m3db-cluster:9000",
            "port": 9000
        },
        {
            "id": "m3db-cluster-rep2-0",
            "isolationGroup": "us-west1-c",
            "zone": "embedded",
            "weight": 100,
            "address": "m3db-cluster-rep2-0.m3dbnode-m3db-cluster:9000",
            "port": 9000
        }
    ]
}'
```

3. Connecting the prometheus and m3coordinator service.

After the pods for m3db and prometheus, we need to make m3coordinator service a NodePort in case, we are not using the loadbalance.
This can be done by kubectl edit command :

Make m3coordinator service a NodePort

```
kubectl edit svc/m3coordinator-m3-cluster
Edit type to be NodePort instead of the "ClusterIP" that is present there.
```