OOM Geo-Red Active-Active via Affinity/AntiAffinity

Geo-Redundancy:-

Problem/Requirement:-

- As an ONAP operator- We require the capability of ONAP -K8 containers deployment in a Geo-Red fashion i.e. into specific zone/regions thus providing protection from a single cluster failure.
- Besides HA, We should have the ability to deploy any container(POD) with our choice with use cases as such if a container used Intel DPDK technology the pod may state that it has to be deployed to an Intel processor based host node, Also another use case could be to ensure placement of a DCAE complex close to the VNFs generating high volumes of traffic thus minimizing networking cost.

Solution Delivered (POC):-

- Affinity/Anti-Affinity is one of the K8 feature which enable us to define affinity-anti affinity rules to make sure the POD placement works extensively, So for POC, We used VNFSDK component with deploying it with certain conditions like:-

    i. No Application container replicas to be on the same node(host).
    ii. No two DB container replicas on the same node (host).
    iii. Both the APP and DB container for 1 replica set to be co located on a single node(host).

For achieving this we used a 4 node setup as shown below

```
NAME     STATUS   ROLES    AGE     VERSION
k8s-1    Ready    <none>   44d     v1.10.5-rancher1
k8s-2    Ready    <none>   44d     v1.10.5-rancher1
k8s-3    Ready    <none>   43d     v1.10.5-rancher1
k8s-4    Ready    <none>   44d     v1.10.5-rancher1
```

And we used the default labels provided by kubernetes to each nodes

```
kubectl get nodes --show-labels
NAME     STATUS   ROLES    AGE     VERSION          LABELS
k8s-1    Ready    <none>   44d     v1.10.5-rancher1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=rancher,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=Region1,failure-domain.beta.kubernetes.io/zone=FailureDomain1,io.rancher.host.docker_version=17.03,io.rancher.host.linux_kernel_version=4.4,io.rancher.host.os=linux,kubernetes.io/hostname=k8s-1,nodetype=east
k8s-2    Ready    <none>   44d     v1.10.5-rancher1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=rancher,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=Region1,failure-domain.beta.kubernetes.io/zone=FailureDomain1,io.rancher.host.docker_version=17.03,io.rancher.host.linux_kernel_version=4.4,io.rancher.host.os=linux,kubernetes.io/hostname=k8s-2,nodetype=north,select=test
k8s-3    Ready    <none>   43d     v1.10.5-rancher1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=rancher,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=Region1,failure-domain.beta.kubernetes.io/zone=FailureDomain1,io.rancher.host.docker_version=17.03,io.rancher.host.linux_kernel_version=4.4,io.rancher.host.os=linux,kubernetes.io/hostname=k8s-3,nodetype=south
k8s-4    Ready    <none>   44d     v1.10.5-rancher1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=rancher,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=Region1,failure-domain.beta.kubernetes.io/zone=FailureDomain1,io.rancher.host.docker_version=17.03,io.rancher.host.linux_kernel_version=4.4,io.rancher.host.os=linux,kubernetes.io/hostname=k8s-4,nodetype=west
```

AntiAffinity between DB Pods

Now lets see how the codes looks like for configuring the Anti-Affinity for VNFSDK-POSTGRESS PODs (just for the POC purpose we increased the replica count to 4)

```
affinity:

  podAntiAffinity:

    requiredDuringSchedulingIgnoredDuringExecution:

    - labelSelector:

        matchExpressions:

        - key: app
```

```
        operator: In
        values:
        - vnfsdk-postgres
      topologyKey: "kubernetes.io/hostname"
```

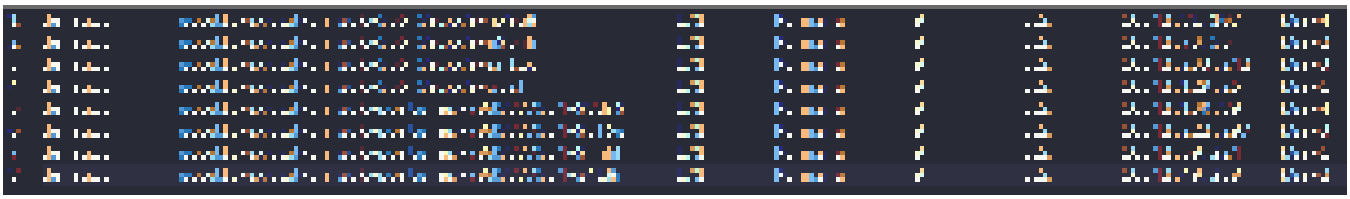
This snippet of values.yaml under vnfsdk-postgress dir ensures that the db pods of vnfsdk-postgress will never reside on the same nodes

Affinity for DB and App pods and Antiaffinity for APP pods

Now for the Affinity between the DBand the APPpods and also the antiaffinity between APP pods we used the below code snippet in values.yaml of vnfsdk

```
affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app
          operator: In
          values:
          - vnfsdk
      topologyKey: "kubernetes.io/hostname"
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app
          operator: In
          values:
          - vnfsdk-postgres
      topologyKey: "kubernetes.io/hostname"
```

When the code was deployed the result was as below



Replica sets on different nodes with the APP and DB on the same node while the APP and DB are also never colocated on the same node

K8s-1	k8s-2	k8s-3	k8s-4

goodly-squid-vnfsdk-556f59ccd9-xtzff	goodly-squid-vnfsdk-556f59ccd9-n95jh	goodly-squid-vnfsdk-556f59ccd9-snlzc	goodly-squid-vnfsdk-556f59ccd9-jx9q8
goodly-squid-vnfsdk-postgres-78d58775c4-9rnhh	goodly-squid-vnfsdk-postgres-78d58775c4-s4l8r	goodly-squid-vnfsdk-postgres-78d58775c4-9cf5g	goodly-squid-vnfsdk-postgres-78d58775c4-98dr9

So with this, We achieved a deployment Active-Active Geo-redundancy.

- Now for each and every component using specific DB/application need to be tested for Geo-redundancy like SDNC/Clamp already started.
- Federation feature can also be used for achieving HA across multi-cluster deployments.