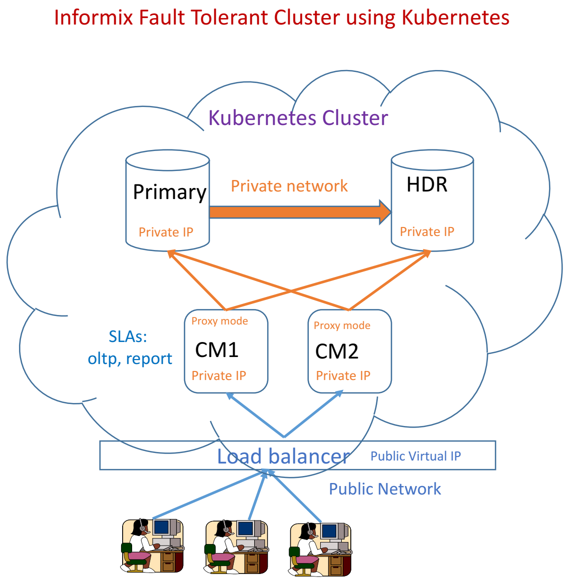
IIUG -2018 Conference Hands-On-Lab Instructions for setting up Informix Cluster in Kubernetes (Google Cloud Platform)

This project helps to setup a fault tolerant Informix cluster along with Connection Manager in Kubernetes container service.

For details on Kubernetes, please refer to https://kubernetes.io/docs/home/

This document helps you build Docker images for Informix server and connection manager, and deploy fault tolerant Informix cluster within google cloud platform kubernetes container service. Even though google cloud platform container services was used for testing Informix Kubernetes cluster, the artifacts in this project helps you deploy Informix cluster in any of the public cloud Kubernetes Container Services.

End goal of this project is to build a fault tolerant Informix cluster environment as shown in this below picture:



1.Login to google cloud VM instance

```
$ ssh -i id_rsa gcp4nag@35.225.226.51
```

Note: Path to ssh private key file and ip address will be provided in the lab

2. Update packages

```
$ sudo yum update -y
```

3. Install git

```
$ sudo yum install git -y
```

4. Install kubectl

```
$ sudo yum install kubectl -y
```

5. Install docker

```
$ sudo yum install docker -y
```

6. Enable docker for non-root user

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

7. Important note: Logout and login again

```
$ exit

$ ssh -i id_rsa gcp4nag@35.225.226.51
```

8. Initialize google cloud environment

```
$ gcloud init
```

```
Choose the account you would like to use to perform operations for
this configuration:
 [1] 598686203493-compute@developer.gserviceaccount.com
 [2] Log in with a new account
Please enter your numeric choice:  1

gcloud services operations describe operations/tmo-acf.0069c985-8676-4925-ac62-bb46b0a78b24
Pick cloud project to use:
 [1] fit-authority-167622
 [2] Create a new project
Please enter numeric choice or text value (must exactly match list
item): 1
Your current project has been set to: [fit-authority-167622].

Do you want to configure a default Compute Region and Zone? (Y/n)? Y

Please enter a value between 1 and 53, or a value present in the list:  8
```

9. Set default zone:

```
$ gcloud config set compute/zone us-central1-a
```

10. Configure kubectl command line access by running the following command:

```
$ gcloud container clusters get-credentials informix-cluster --zone us-central1-a --project fit-authority-167622
```

```
        Fetching cluster endpoint and auth data.
        kubeconfig entry generated for informix-cluster.
```

11. Create a uniqe namespace. From the below command replace my-namespace with your unique user name

```
$ kubectl create namespace my-namespace
```

12. To be sure things are right, let’s list all of the namespaces in our cluster.

```
$ kubectl get namespaces --show-labels
        NAME          STATUS   AGE    LABELS
        default       Active   156m   <none>
        kube-public    Active   156m   <none>
        kube-system    Active   156m   <none>
        my-namespace   Active   15s    <none>
```

13. Get current context

```
$ kubectl config current-context
        gke_fit-authority-167622_us-central1-a_informix-cluster
```

14. create new context using your new namespace.

Make sure to replace "my-namespace" with namespace that you created above.

```
$ kubectl config set-context dev --namespace=my-namespace --cluster=gke_fit-authority-167622_us-central1-a_informix-cluster --user=gke_fit-authority-167622_us-central1-a_informix-cluster
```

```
        Context "dev" created.
```

15. Let’s switch to operate in the development namespace.

```
$ kubectl config use-context dev
```

You can verify your current context by doing the following:

```
$ kubectl config current-context
        dev
```

Note: At this point, all requests we make to the Kubernetes cluster from the command line are scoped to the development namespace.

16. Start proxy to connect to Kubernetes control plane:

```
$ kubectl proxy &
        Starting to serve on 127.0.0.1:8001
```

Keep this proxy command running.

17. Verify cluster by running hello-world program

```
$ kubectl run hello-node --image=gcr.io/google-samples/node-hello:1.0 --port=8080
$ kubectl expose deployment hello-node --type="LoadBalancer"
```

```
$ kubectl get service hello-node
        NAME        TYPE          CLUSTER-IP     EXTERNAL-IP   PORT(S)         AGE
        hello-node  LoadBalancer  10.7.241.134   <pending>     8080:31613/TCP  22s
```

```
#Wait for few seconds till external IP address is assigned
```

```
$ kubectl get service hello-node
          NAME CLUSTER-IP EXTERNAL-IP PORT(S) AGE
          hello-node 10.107.246.252 104.197.98.97 8080:30075/TCP 45s
```

#open this below web page
http://104.197.98.97:8080

18. Delete service

```
$ kubectl delete service hello-node
```

19. Start docker daemon

```
$ sudo systemctl start docker
```

20. Check docker status and make sure it shows "active"

```
$ sudo systemctl status docker
```

21. Clone git project

```
$ cd ~/
$ git clone https://github.com/nagaraju-inturi/kubernetes-informix-cluster.git

        Cloning into 'kubernetes-informix-cluster'...
        remote: Enumerating objects: 92, done.
        remote: Total 92 (delta 0), reused 0 (delta 0), pack-reused 92
        Unpacking objects: 100% (92/92), done.
```

```
$ ls
kubernetes-informix-cluster
```

22. Copy server and clientsdk tar files to build docker images

```
$ cp iif.12.10.tar ~/kubernetes-informix-cluster/docker/server_ctx/

$ cp clientsdk.4.10.tar ~/kubernetes-informix-cluster/docker/cm_ctx/
```

23. Configure Docker to use gcloud

```
$ gcloud auth configure-docker

        The following settings will be added to your Docker config file
        located at [/home/gcp4nag/.docker/config.json]:
         {
```

```
   "credHelpers": {
     "gcr.io": "gcloud",
     "us.gcr.io": "gcloud",
     "eu.gcr.io": "gcloud",
     "asia.gcr.io": "gcloud",
     "staging-k8s.gcr.io": "gcloud",
     "marketplace.gcr.io": "gcloud"
   }
  }

  Do you want to continue (Y/n)?  Y
  Docker configuration file updated.
```

24. Build Docker images for Informix server:

```
$ cd ~/kubernetes-informix-cluster/docker/server_ctx/

# Note: From below command replace "nagaraju" with your unique username.
$ docker build -t gcr.io/fit-authority-167622/informix-nagaraju:v1 .
```

25. Push Informix server Docker image to google container registry:

```
# Note: From below command replace "nagaraju" with your unique username.
$ docker push gcr.io/fit-authority-167622/informix-nagaraju:v1
```

26. Build Docker image for Informix Connection Manager:

```
$ cd ~/kubernetes-informix-cluster/docker/cm_ctx/
$ docker build -t gcr.io/fit-authority-167622/informix_cm-nagaraju:v1 .
```

27. Push Connection Manager Docker image to Container registry:

```
$ docker push  gcr.io/fit-authority-167622/informix_cm-nagaraju:v1
```

28. Verify container images uploaded to google container registry:

```
$ gcloud container images list
      gcr.io/fit-authority-167622/informix
      gcr.io/fit-authority-167622/informix-nagaraju
      gcr.io/fit-authority-167622/informix_cm
      gcr.io/fit-authority-167622/informix_cm-nagaraju
```

29. Create kubernetes secret for keystore files

```
$ kubectl create secret generic ssl-key-secret --from-file=ssl-kdb=/home/gcp4nag/kubernetes-informix-cluster/kubernetes/informix.kdb --from-file=ssl-sth=/home/gcp4nag/kubernetes-informix-cluster/kubernetes/informix.sth
```

30. Edit /home/gcp4nag/kubernetes-informix-cluster/kubernetes/informix-k8.yaml to replace gcr.io/fit-authority-167622/informix:v1 and gcr.io/fit-authority-167622/informix_cm:v1 to your image names.

```
$ vi /home/gcp4nag/kubernetes-informix-cluster/kubernetes/informix-k8.yaml
- image: gcr.io/fit-authority-167622/informix:v1
- image: gcr.io/fit-authority-167622/informix_cm:v1
```

31. Build Informix cluster using below kubernetes YAML file:

```
$ kubectl create -f /home/gcp4nag/kubernetes-informix-cluster/kubernetes/informix-k8.yaml
```

Wait for up to 5 minutes and check the cluster status:

32. Check statefulsets

```
$ kubectl get statefulsets
NAME      DESIRED   CURRENT   AGE
cm        2         2         1d
informix  2         2         1d
```

33. List PODS

```
$ kubectl get pods
NAME        READY    STATUS    RESTARTS   AGE
cm-0        1/1      Running   0          1d
cm-1        1/1      Running   0          1d
informix-0  1/1      Running   0          1d
informix-1  1/1      Running   0          1d
```

34. List Persistent Volume Claims

```
$ kubectl get pvc
NAME            STATUS   VOLUME                                     CAPACITY   ACCESSMODES   STORAGECLASS   AGE
data-informix-0 Bound    pvc-a20da4c9-4362-11e7-832e-42010a80007f   10Gi       RWO           standard       1d
data-informix-1 Bound    pvc-c418adf9-4362-11e7-832e-42010a80007f   10Gi       RWO           standard       1d
```

35. List Persistent Volumes

```
$ kubectl get pv
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS   CLAIM                     STORA
GECLASS   REASON   AGE
pvc-a20da4c9-4362-11e7-832e-42010a80007f   10Gi       RWO           Delete          Bound    default/data-informix-
0   standard             1d
pvc-c418adf9-4362-11e7-832e-42010a80007f   10Gi       RWO           Delete          Bound    default/data-informix-
1   standard             1d
```

36. List services to get external IP address for client connections:

```
$ kubectl get services
NAME        CLUSTER-IP    EXTERNAL-IP     PORT(S)                                                     AGE
cm          None          <none>          50000/TCP,50001/TCP,50002/TCP,50003/TCP,50004/TCP,50005/TCP
            1d
informix    None          <none>          60000/TCP,60001/TCP,60002/TCP                               1d
informix-
cm   10.107.243.88   104.198.172.24   50000:32201/TCP,50001:31096/TCP,50002:32722/TCP,50003:30588/TCP,50004:32
642/TCP,50005:32267/TCP   1d
kubernetes   10.107.240.1   <none>        443/TCP                                                     8d
```

Note down external ip address from ‘kubectl get services’ command for “cm” service and connect to the Informix cluster.

37: External Port numbers for client connections:

Connection Manager SLA	PORT	Description
OLTP	50000	This port connects to current primary server
REPORT	50001	This port connects to any of the secondary servers
OLTP_SSL		
REPORT_SSL		
OLTP_DRDA	50004	This DRDA port connects to current primary server
REPORT_DRDA	50005	This DRDA port connects to any of the secondary servers

Logging-in to Docker Containers:

38. Command to login to primary server informix-0 container:

```
$ kubectl exec -it informix-0 -- /opt/ibm/boot.sh --shell /bin/bash
```

To switch user to informix:

```
$ su informix
```

```
[informix@informix-0 ibm]$ onstat -

IBM Informix Dynamic Server Version 12.10.FC9 -- On-Line (Prim) -- Up 2 days 01:58:37 -- 172660 Kbytes

[informix@informix-0 ibm]$  onstat -g dri

        IBM Informix Dynamic Server Version 12.10.FC9 -- On-Line (Prim) -- Up 00:02:54 -- 164468 Kbytes

        Data Replication at 0x45a3b028:
         Type         State       Paired server       Last DR CKPT (id/pg)    Supports Proxy Writes
         primary       on         informix1                4 / 104        NA

         DRINTERVAL   0
         DRTIMEOUT    30
         DRAUTO       0
         DRLOSTFOUND  /opt/ibm/informix/etc/dr.lostfound
         DRIDXAUTO    0
         ENCRYPT_HDR  0
         Backlog      0
         Last Send    2018/10/16 21:57:14
         Last Receive 2018/10/16 21:57:14
         Last Ping    2018/10/16 21:56:51
         Last log page applied(log id,page): 4,105

[informix@informix-0 ibm]$  onstat -g cmsm
       IBM Informix Dynamic Server Version 12.10.FC9 -- On-Line (Prim) -- Up 00:03:07 -- 164468 Kbytes
       Unified Connection Manager: cm0                    Hostname: cm-0.cm.my-namespace.svc.cluster.local
```

```
CLUSTER        informix_cluster       LOCAL
        Informix Servers: informix0,informix1
        SLA                  Connections   Service/Protocol   Rule
        oltp                           0     50000/onsoctcp   DBSERVERS=primary MODE=proxy
        report                          0     50001/onsoctcp   DBSERVERS=(HDR,RSS),primary MODE=proxy
        oltp_drda                        0     50004/drsoctcp   DBSERVERS=primary MODE=proxy
        report_drda                       0     50005/drsoctcp   DBSERVERS=(HDR,RSS),primary MODE=proxy

        Failover Arbitrator: Failover is disabled
        ORDER=SDS,HDR,RSS PRIORITY=100 TIMEOUT=1

Unified Connection Manager: cm1                    Hostname: cm-1.cm.my-namespace.svc.cluster.local

CLUSTER        informix_cluster       LOCAL
        Informix Servers: informix0,informix1
        SLA                  Connections   Service/Protocol   Rule
        oltp                           0     50000/onsoctcp   DBSERVERS=primary MODE=proxy
        report                          0     50001/onsoctcp   DBSERVERS=(HDR,RSS),primary MODE=proxy
        oltp_drda                        0     50004/drsoctcp   DBSERVERS=primary MODE=proxy
        report_drda                       0     50005/drsoctcp   DBSERVERS=(HDR,RSS),primary MODE=proxy

        Failover Arbitrator: Failover is disabled
        ORDER=SDS,HDR,RSS PRIORITY=101 TIMEOUT=1
```

39. Command to login to Connection manager container:

```
$ kubectl exec -it cm-0 -- /opt/ibm/boot.sh --shell /bin/bash
```

Connection manager log file at $INFORMIXDIR/tmp/cm.log

Scaling up Connection manager statefulset instances/pods:

40. Run the following command increase number of connection manager pods:

```
$ kubectl scale --replicas=3 statefulset cm
```

This above command makes sure that minimum three connection manager instances/pods running within the cluster.

```
$ kubectl get pods
NAME         READY    STATUS    RESTARTS   AGE
cm-0         1/1      Running   0          7m
cm-1         1/1      Running   0          7m
cm-2         0/1      Running   0          53s
informix-0   1/1      Running   0          7m
informix-1   1/1      Running   0          6m
```

41. Scaling up Informix server statefulset instances/pods:

```
$ kubectl scale --replicas=3 statefulset informix
```

The above command creates new pod with Informix RSS server.

```
$ kubectl get pods
NAME         READY    STATUS    RESTARTS   AGE
cm-0         1/1      Running   0          7m
```

```
cm-1        1/1      Running   0         7m
cm-2        0/1      Running   0         3m
informix-0   1/1      Running   0         7m
informix-1   1/1      Running   0         6m
```
informix-2   1/1   Running   0   2m

Note: Current logic in Informix docker image boot.sh script only supports up to three nodes (Primary, HDR and RSS) for Informix cluster:

Verify fault tolerant nature of Kubernetes cluster:

42. Delete cm-1 pod from cm statefulset:

```
$ kubectl get pods
NAME        READY     STATUS    RESTARTS   AGE
cm-0        1/1       Running   0          1d
cm-1        1/1       Running   0          1d
informix-0   1/1       Running   0          1d
informix-1   1/1       Running   0          1d

$ kubectl delete pod cm-1
pod "cm-1" deleted
```

43. After few seconds verify pods again:

```
$ kubectl get pods
NAME        READY     STATUS        RESTARTS   AGE
cm-0        1/1       Running       0          1d
cm-1        0/1       Terminating   0          18s
informix-0   1/1       Running       0          1d
informix-1   1/1       Running       0          1d

$ kubectl get pods
NAME        READY     STATUS        RESTARTS   AGE
cm-0        1/1       Running       0          1d
cm-1        0/1       Running       0          50s
informix-0   1/1       Running       0          1d
informix-1   1/1       Running       0          1d
```

Kubernetes recreates the pod.

Same thing can be done for Informix statefulset as well.

44. Delete informix-1(HDR server) and check what happens.

Note: You may need to wait for atleast 3 minutes HDR to reconnect to primary server. Kubernetes DNS service discovery takes around 90 seconds to recognize new pod ip address.

```
$ kubectl get pods
NAME        READY     STATUS    RESTARTS   AGE
cm-0        1/1       Running   0          1d
cm-1        1/1       Running   0          1d
```

```
cm-2         1/1       Running   0         1d
informix-0   1/1       Running   0         1d
informix-1   1/1       Running   0         1d
informix-2   1/1       Running   0         1d

$ kubectl delete pod informix-1
pod "informix-1" deleted
```

45. After few minutes verify pods again:

```
$ kubectl get pods
NAME         READY     STATUS    RESTARTS   AGE
cm-0         1/1       Running   0          1d
cm-1         1/1       Running   0          1d
cm-2         1/1       Running   0          1d
informix-0   1/1       Running   0          1d
informix-1   1/1       Running   0          1d
informix-2   1/1       Running   0          1d
```

46. Delete informix-0 (primary server) and check what happens.

Note: You may need to wait for atleast 3 minutes primary server to reconnect to HDR and RSS servers. Kubernetes DNS service discovery takes around 90 seconds to recognize new pod ip address.

```
$ kubectl get pods
NAME         READY     STATUS    RESTARTS   AGE
cm-0         1/1       Running   0          1d
cm-1         1/1       Running   0          1d
cm-2         1/1       Running   0          1d
informix-0   1/1       Running   0          1d
informix-1   1/1       Running   0          1d
informix-2   1/1       Running   0          1d

$ kubectl delete pod informix-0
pod "informix-0" deleted
```

47. After few minutes verify pods again:

```
$ kubectl get pods
NAME         READY     STATUS    RESTARTS   AGE
cm-0         1/1       Running   0          1d
cm-1         1/1       Running   0          1d
cm-2         1/1       Running   0          1d
informix-0   1/1       Running   0          1d
informix-1   1/1       Running   0          1d
informix-2   1/1       Running   0          1d
```

Kubernetes Pods and Controllers for Informix cluster (Informix-k8.yaml file review):

Informix Kubernetes Cluster yaml file creates these following kubernetes pods and controllers:

Statefulsets: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Important Note on StatefulSets:

Kubernetes StatefulSets gets you predictable host names, and external storage(volumes) are bound to the pods(containers) in StatefulSets till the life of StatefulSets. These properties of StatefulSets helps build database cluster which require persistent state.

Host names within StatefulSets pods starts with <setname>-0, <setname>-1, <setname>-2 and so on.

Informix serve Docker image is constructed -- check logic with in boot.sh script -- to start primary server on informix-0, HDR on informix-1, and RSS on informix-2. Note: “informix” is the statefulset name for Informix cluster.

Pods: https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

Services: https://kubernetes.io/docs/concepts/services-networking/service/

Persistent Volumes: https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Dynamic Provisioning for Persistent Volumes:
http://blog.kubernetes.io/2016/10/dynamic-provisioning-and-storage-in-kubernetes.html

Secrets: https://kubernetes.io/docs/concepts/configuration/secret/