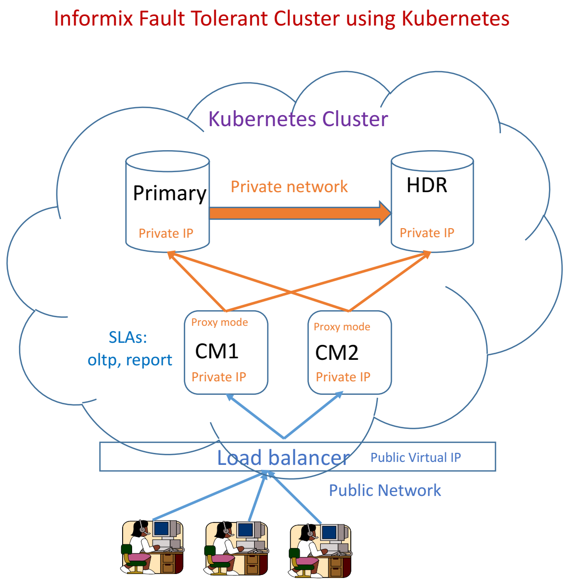
# Instructions for setting up Informix Cluster in Kubernetes (AWS using KOPS)

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 AWS using KOPS.

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



## Signup for AWS.

### 1.Download AWS CLI by following these instructions:

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

## 2) Login to AWS web console, using identify and access management service(IAM), create “kops” user with “AdminstratorAccess” privilege.

Note down its “Access Key ID” and “Secret access key”. Need it in the next step.

## 3) Login to AWS from your local host using aws CLI.

For this, first you need to get “Access Key ID” and “Secret access key” from AWS web console. Also choose region you want to create your kubernetes cluster.

```
$ aws configure
AWS Access Key ID [None]: XXXXXXXXXXXXXXXXXX
AWS Secret Access Key [None]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Default region name [None]: us-east-1
Default output format [None]:
```

## 4) Install kops from this web link: https://github.com/kubernetes/kops/releases

Install 1.6.0 or later version to work with 1.6.2 or later Kubernetes cluster version.
1.6.2 or later version is needed for Informix cluster to work in Kubernetes environment.
Note: In older versions below 1.6.x, reverse DNS lookup functionality do not work, this functionality is needed for trusted host configuration for Informix cluster nodes.

Rename downloaded executable to kops and make sure to change your PATH to include kops location.

## 5) Create route53 domain for your cluster by following instructions(step 2/5) from this URL: https://kubernetes.io/docs/getting-started-guides/kops/

## 6) Create an S3 bucket to store your clusters state by following instructions(step 4/5) from this URL: https://kubernetes.io/docs/getting-started-guides/kops/

For this exercise, I named the S3 bucket as “informix-kops-state”.

## 7) Set S3 bucket name in your shell environment

```
$ export KOPS_STATE_STORE=s3://informix-kops-state
```

## 8) Build your cluster configuration

```
$ kops create cluster --zones=us-east-1c kubernetes.informix.cloud
```

kubernetes.informix.cloud is my cluster domain name that I created in Route53.

10) Review output configuration and rerun same command with --yes option to create the Kubernetes cluster

```
$ kops create cluster --zones=us-east-1c kubernetes.informix.cloud --yes
```

11) Install kubectl by following instructions from here:
https://kubernetes.io/docs/tasks/tools/install-kubectl/

12) Wait for few minutes and run kubectl command to get cluster status

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.2",
GitCommit:"477efc3cbe6a7effca06bd1452fa356e2201e1ee", GitTreeState:"clean",
BuildDate:"2017-04-19T20:33:11Z", GoVersion:"go1.7.5", Compiler:"gc",
Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.2",
GitCommit:"477efc3cbe6a7effca06bd1452fa356e2201e1ee", GitTreeState:"clean",
BuildDate:"2017-04-19T20:22:08Z", GoVersion:"go1.7.5", Compiler:"gc",
Platform:"linux/amd64"}
```

```
$ kubectl get nodes
NAME                            STATUS         AGE       VERSION
ip-172-20-52-134.ec2.internal   Ready,node     23h       v1.6.2
ip-172-20-53-100.ec2.internal   Ready,node     23h       v1.6.2
ip-172-20-62-192.ec2.internal   Ready,master   23h       v1.6.2
```

Note: Make sure kubernetes cluster version is 1.6.2 or above.

13) Validate Kubernetes cluster using kops validate command

```
$ kops validate cluster
       Using cluster from kubectl context: kubernetes.informix.cloud

       Validating cluster kubernetes.informix.cloud

       INSTANCE GROUPS
       NAME                 ROLE    MACHINETYPE    MIN     MAX     SUBNETS
       master-us-east-1c    Master  m3.medium      1       1       us-east-1c
       nodes                Node    t2.medium      2       2       us-east-1c

       NODE STATUS
       NAME                           ROLE    READY
       ip-172-20-52-134.ec2.internal       node    True
       ip-172-20-53-100.ec2.internal       node    True
       ip-172-20-62-192.ec2.internal       master  True

       Your cluster kubernetes.informix.cloud is ready
```

14.Start proxy to connect to Kubernetes control plane:

$ Kubectl proxy &

*Starting to serve on 127.0.0.1:8001*

Keep this proxy command running.

### 15.Open dashboard by navigating to http:/localhost:8001/ui  to get to Kubernetes Dashboard.

## 16.Install Docker

Open this URL and follow instructions to install Docker on your local host:
https://docs.docker.com/engine/installation/

## 17.Build Docker images

We  will be using AWS EC2 Container registry to store our Docker images.

Reference material : `http://blog.redspread.com/using-awss-ec2-container-registry-with-k8s/`

### 18) Login to EC2 Container registry

```
$ aws ecr get-login | sh -
```

### 19.Clone git project:

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

### 20. Get Informix server tar file:

### URL to download Informix Server Developer edition:

https://www.ibm.com/developerworks/downloads/im/informix/

Copy tar file to kubernetes-informix-cluster/docker/server_ctx/iif.12.10.tar.
Note: Make sure to rename target file to iif.12.10.tar. Dockerfile file in server_ctx directory refers to this file name.

### 21) Get Informix Client SDK tar file.

### URL to download Informix Client SDK  developer edition:

https://www-01.ibm.com/marketing/iwm/tnd/preconfig.jsp?id=2013-03-26+02%3A58%3A21.558674R&S_TACT=&S_CMP=

Copy tar file to kubernetes-informix-cluster/docker/cm_ctx/clientsdk.4.10.tar
Note: Make sure to rename target file to clientsdk.4.10.tar.  Dockerfile in cm_ctx directory refers to this file name.

## 22) Create repository for Informix server docker image

```
$ aws ecr create-repository --repository-name
kubernetes/informix
       {
           "repository": {
               "repositoryArn": "arn:aws:ecr:us-east-
       1:323253210322:repository/kubernetes/informix",
               "registryId": "###########",
               "repositoryName": "kubernetes/informix",
               "repositoryUri": "##########.dkr.ecr.us-east-
       1.amazonaws.com/kubernetes/informix",
               "createdAt": 1496558420.0
           }
       }
```

Note down the "repositoryUri" value.

## 23) Build Docker images for Informix server:

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

$ docker build -t ##########.dkr.ecr.us-east-
1.amazonaws.com/kubernetes/informix:v1 .
```

Use "repositoryUri" value here.

## 24) Push Informix server Docker image to EC2 container registry:

```
$ docker push ###########.dkr.ecr.us-east-
1.amazonaws.com/kubernetes/informix:v1
```

## 25) Create repository for Informix connection manager docker image

```
$ aws ecr create-repository --repository-name
kubernetes/informix_cm
       {
           "repository": {
               "repositoryArn": "arn:aws:ecr:us-east-
       1:323253210322:repository/kubernetes/informix_cm",
               "registryId": "###########",
               "repositoryName": "kubernetes/informix_cm",
               "repositoryUri": "##########.dkr.ecr.us-east-
       1.amazonaws.com/kubernetes/informix_cm",
               "createdAt": 1496558900.0
           }
       }
```

Note down the "repositoryUri" value.

### 26) Build Docker image for Informix Connection Manager:

```
$ cd kubernetes-informix-cluster/docker/cm_ctx/

$ docker build -t ############.dkr.ecr.us-east-
1.amazonaws.com/kubernetes/informix_cm:v1 .
```

### 27) Push Connection Manager Docker image to EC2 Container registry:

```
$ docker push #########.dkr.ecr.us-east-
1.amazonaws.com/kubernetes/informix_cm:v1
```

## Build Informix cluster using below kubernetes YAML file:

### 28) Create SSL keystore files using IBM Global Security Kit:

```
$ cd kubernetes-informix-cluster/kubernetes/
```

### Command to create keystore and SSL keys:

```
#Create keystore files
$ gsk8capicmd_64 -keydb -create -db informix.kdb -pw informix4k8 -type cms -expire 3650 -
stash

#create certificate
$ gsk8capicmd_64 -cert -create -db informix.kdb -pw informix4k8 -dn "CN=`hostname`" -size
2048 -label informix -default_cert yes
```

Informix.sth and Informix.kdb are required for SSL client connections.
For more details on IBM Global Security Kit, please refer to this URL:
https://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.sec.doc/ids_ssl_006.htm

Alternatively, you can use the Informix.sth and Informix.kdb files from GIT repository for your test cluster. You cannot use these files for your production cluster.

If you do not want SSL configuration, update Informix-k8.yaml file and change SSLCONFIG value to “false” for both Informix server and connection manager statefulsets, and create empty dummy files for Informix.sth and Informix.kdb files.

### 29) (Important step) Create kubernetes secret for keystore files. Name secret object as ssl-key-secret

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

Note: Make sure to input correct path for Informix.kdb and Informix.sth for the above command.

## 30) Update kubernetes-informix-cluster/kubernetes/informix-k8.yaml file to change (project id) container image name for both Informix server and connection .

```
- image: ###########.dkr.ecr.us-east-1.amazonaws.com/kubernetes/informix:v1
- image: #########.dkr.ecr.us-east-1.amazonaws.com/kubernetes/informix_cm:v1
```

## 31) Build Informix cluster using below kubernetes YAML file:

```
$ cd kubernetes-informix-cluster/kubernetes/
```

```
$ kubectl create -f 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   ACCESSMOD
ES   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                        STORAGECLASS   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

```
$ kubectl get services
NAME          CLUSTER-IP        EXTERNAL-
IP         PORT(S)
                AGE
cm            None              <none>            50000/TCP,50001/TCP,50002/TCP,50003/TCP,5
0004/TCP,50005/TCP                                       1d
informix      None              <none>            60000/TCP,60001/TCP,60002/TCP
                                                        1d

informix-
cm    100.67.141.243   a3b84c113494f...   50000:30880/TCP,50001:31552/TCP,50002:31902/TCP,50
003:30552/TCP,50004:31849/TCP,50005:31953/TCP   1d
kubernetes    100.64.0.1        <none>            443/TCP
                                                        1d
```

## 37) Get load balancer name from informix-cm service:

```
$ kubectl describe services informix-cm|grep -i Ingress
LoadBalancer Ingress:   a3b84c113494f11e7ae1412d5086efd8-535866110.us-east-
1.elb.amazonaws.com
```

Use this as host name to connect to Informix cluster.

## 38) 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 | 50002 | This SSL port connects to current primary server |
| REPORT_SSL | 50003 | This SSL port connects to any of the secondary servers |
| OLTP_DRDA | 50004 | This DRDA port connects to current primary server |
| REPORT_DRDA | 50005 | This DRDA port connects to any of the secondary servers |

Note: For SSL port to work, you need to either copy Informix.sth to client.sth, Informix.kdb to client.kdp and copy these files to $INFORMIXDIR/etc/
Or
Create client.kdb and client.sth files by creating keystore using "`gsk8capicmd_64 –keydb –create`" `command`, extract public key from Informix.kdb file and import the key to client.kdb file.

## Logging-in to Docker Containers:

### 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]$
```

### 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:

### 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
```

## 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:

### 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
```

### 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.**

Delete informix-1 and check what happens.

**Note**: Informix Kubernetes cluster do not automatically restart failed primary server as this may cause split brain situation, this operation requires DBA intervention. However, Informix Kubernetes cluster automatically restarts secondary server instance without DBA intervention.

## 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/

## Informix statefulset with Informix server Docker image:

```
#
# StatefulSet for Informix cluster.
# StatefulSet get predictable hostnames, and external storage is bound
# to the pods within StateFulSets for the life.`
# Replica count configures number of Informix Server containers.
#
 apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: informix
spec:
  serviceName: "informix"
  replicas: 2
  template:
    metadata:
      labels:
        app: informix
    spec:
      containers:
      - image: gcr.io/fit-authority-167622/informix:v3
        name: informix
        env:
        - name: SSLCONFIG
          value: "true"
        ports:
        - containerPort: 60000
          name: informix
        - containerPort: 60001
          name: informixssl
        - containerPort: 60002
          name: informixdrda
        volumeMounts:
        - name: data
          mountPath: /opt/ibm/data
        - name: vsslkeysecret
          mountPath: /etc/sslkeysecret
          readOnly: true
        livenessProbe:
          exec:
            command: ["/opt/ibm/chk4live.sh", "/tmp"]
            #command: ["ls", "/tmp"]
          initialDelaySeconds: 300
          timeoutSeconds: 5
        readinessProbe:
          exec:
            #command: ["/opt/ibm/chkinformix.sh", "/tmp"]
            command: ["ls", "/tmp"]
          initialDelaySeconds: 30
          timeoutSeconds: 30
      volumes:
      - name: data
```

```
        persistentVolumeClaim:
          claimName: data
      - name: vsslkeysecret
        secret:
          secretName: ssl-key-secret
          defaultMode: 256
  volumeClaimTemplates:
  - metadata:
      name: data
      annotations:
        volume.alpha.kubernetes.io/storage-class: anything
    spec:
      accessModes: ["ReadWriteOnce"]
      resources:
        requests:
          storage: 10Gi
```

Replicas specifies the number of pods/containers.
Containers section specify the docker image location for the pods.
VolumeMounts specifies details about the type of external disk being mounted and location of the mount point.

Volumes specify the details about kubernetes persistent volume claims.

volumeClaimTemplates refers to dynamic provisioning feature of kubernetes clusters. For more details on this feature please refer to this web page:
[http://blog.kubernetes.io/2016/10/dynamic-provisioning-and-storage-in-kubernetes.html](http://blog.kubernetes.io/2016/10/dynamic-provisioning-and-storage-in-kubernetes.html)

## Connection manager (cm) statefulset with Informix connection manager docker image:

```
#
# StatefulSet for Informix connection manager group.
# Replica count configures the number of CM containers/pods.
#
 apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: cm
spec:
  serviceName: "cm"
  replicas: 2
  template:
    metadata:
      labels:
        app: cm
    spec:
      containers:
      - image: gcr.io/fit-authority-167622/informix_cm:v4
        name: cm
        env:
        - name: SSLCONFIG
```

```
          value: "true"
        ports:
        - containerPort: 50000
          name: oltp
        - containerPort: 50001
          name: report
        - containerPort: 50002
          name: oltpssl
        - containerPort: 50003
          name: reportssl
        - containerPort: 50004
          name: oltpdrda
        - containerPort: 50005
          name: reportdrda
        volumeMounts:
        - name: vsslkeysecret
          mountPath: /etc/sslkeysecret
          readOnly: true
        livenessProbe:
          exec:
            command: ["/opt/ibm/chk4live.sh", "/tmp"]
            #command: ["ls", "/tmp"]
          initialDelaySeconds: 30
          timeoutSeconds: 5
        readinessProbe:
          exec:
            #command: ["/opt/ibm/chkinformix.sh", "/tmp"]
            command: ["ls", "/tmp"]
          initialDelaySeconds: 30
          timeoutSeconds: 30
      volumes:
      - name: vsslkeysecret
        secret:
          secretName: ssl-key-secret
          defaultMode: 256
```

## Headless services for Informix statefulset:

```
#
# Headless service for Informix cluster statefulset
# Headless service with clusterIP set to NULL
# create DNS records for Informix cluster hosts.
#
apiVersion: v1
kind: Service
metadata:
  name: informix
  labels:
    app: informix
spec:
  ports:
    - port: 60000
      name: informix
    - port: 60001
      name: informixssl
    - port: 60002
      name: informixdrda
  clusterIP: None
  selector:
    app: informix
```

## Headless service for connection manager statefulset:

```
#
# Headless service for Informix Connection Manager statefulset.
# Headless service with clusterIP set to NULL
# create DNS records for Informix Connection Manager hosts.
#
apiVersion: v1
kind: Service
metadata:
  name: cm
  labels:
    app: cm
spec:
  ports:
    - port: 50000
      name: oltp
    - port: 50001
      name: report
    - port: 50002
      name: oltpssl
    - port: 50003
      name: reportssl
    - port: 50004
      name: oltpdrda
    - port: 50005
      name: reportdrda
  clusterIP: None
  selector:
    app: cm
```

## Informix-cm service to get external IP address and to add load balancer for client connections:

```
#
# Connection manager client service along with loadbalancer.
# This service gets external ip address for applications
# to connect to Informix cluster over the internet.
#
apiVersion: v1
kind: Service
metadata:
  name: informix-cm
  labels:
    app: cm
spec:
  ports:
  - name: oltp
    port: 50000
    targetPort: 50000
  - name: report
    port: 50001
    targetPort: 50001
  - name: oltpssl
    port: 50002
    targetPort: 50002
  - name: reportssl
```

```
    port: 50003
    targetPort: 50003
  - name: oltpdrda
    port: 50004
    targetPort: 50004
  - name: reportdrda
    port: 50005
    targetPort: 50005
  type: LoadBalancer
  selector:
    app: cm
```