# Etcd in Docker Swarm

## Introduction

The goal is to get a cluster of etcd instances running inside of docker swarm and for a cluster that can be used as a simple fault tolerant key value store.

## Result

The etcd cluster seemed to form, elect a leader and answer various commands from with the containers in the cluster. Simple tests were performed such as creating a key, updating a key, watching a key from another node while it was updated. All the tests were successful.

## Compose file

The following compose file was used to create a three node etcd cluster. There are no external ports which means that any code wanting to access the cluster should be added to the network. This allows multiple clusters to be created within swarm without interfering with each other.

version: '3.4'

services:

etcd-00:

image: quay.io/coreos/etcd:v3.2.17

hostname: etcd-00

command:

- etcd

- --name=etcd-00

- --data-dir=data.etcd

- --advertise-client-urls=http://etcd-00:2379

- --listen-client-urls=http://0.0.0.0:2379

- --initial-advertise-peer-urls=http://etcd-00:2380

- --listen-peer-urls=http://0.0.0.0:2380

- --initial-cluster=etcd-00=http://etcd-00:2380,etcd-01=http://etcd-01:2380,etcd-02=http://etcd-02:2380

- --initial-cluster-state=new

- --initial-cluster-token=${ETCD\_TOKEN}

volumes:

- etcd-00vol:/data.etcd

networks:

- etcd\_net

deploy:

replicas: 1

etcd-01:

image: quay.io/coreos/etcd:v3.2.17

hostname: etcd-01

command:

- etcd

- --name=etcd-01

- --data-dir=data.etcd

- --advertise-client-urls=http://etcd-01:2379

- --listen-client-urls=http://0.0.0.0:2379

- --initial-advertise-peer-urls=http://etcd-01:2380

- --listen-peer-urls=http://0.0.0.0:2380

- --initial-cluster=etcd-00=http://etcd-00:2380,etcd-01=http://etcd-01:2380,etcd-02=http://etcd-02:2380

- --initial-cluster-state=new

- --initial-cluster-token=${ETCD\_TOKEN}

volumes:

- etcd-01vol:/data.etcd

networks:

- etcd\_net

deploy:

replicas: 1

etcd-02:

image: quay.io/coreos/etcd:v3.2.17

hostname: etcd-02

command:

- etcd

- --name=etcd-02

- --data-dir=data.etcd

- --advertise-client-urls=http://etcd-02:2379

- --listen-client-urls=http://0.0.0.0:2379

- --initial-advertise-peer-urls=http://etcd-02:2380

- --listen-peer-urls=http://0.0.0.0:2380

- --initial-cluster=etcd-00=http://etcd-00:2380,etcd-01=http://etcd-01:2380,etcd-02=http://etcd-02:2380

- --initial-cluster-state=new

- --initial-cluster-token=${ETCD\_TOKEN}

volumes:

- etcd-02vol:/data.etcd

networks:

- etcd\_net

deploy:

replicas: 1

volumes:

etcd-00vol:

driver: local

etcd-01vol:

driver: local

etcd-02vol:

driver: local

networks:

etcd\_net:

driver: overlay

driver\_opts:

encrypted: "true"

internal: true

## Testing

Get access to one of the etcd containers using the following command but changed for the particular cluster names created.

$ docker exec -it jon\_etcd-02.1.tmedsevoh7653h1nryb0ai6ae /bin/sh

Once at the prompt inside one of the containers, execute the following command to set the version of the API to version 3.

# export ETCDCTL\_API=3

Once this is done then the following commands will work.

**Command**: etcdctl member list

Lists all the members of the cluster

**Results**:

1c2164feaff355dc, started, etcd-02, http://etcd-02:2380, http://etcd-02:2379

70f71ecf7dfb681a, started, etcd-01, http://etcd-01:2380, http://etcd-01:2379

f05ec268fbc96c37, started, etcd-00, http://etcd-00:2380, http://etcd-00:2379

**Command**: etcdctl put my-key 'this is the value for my-key'

Stores a key with a value

**Results**:

OK

**Command**: etcdctl get my-key

Gets the value for a specified key

**Results**:

my-key

this is the value for my-key

**Command**: etcdctl check perf

Runs a test to check the speed of the cluster

**Results**:

PASS: Throughput is 150 writes/s

PASS: Slowest request took 0.041751s

PASS: Stddev is 0.003887s

PASS

## Challenges

Placement of zookeeper nodes so that each node is on a different VM or physical machine is problematic. Either the compose file has to know about the environment or the environment has to be altered to know about the zookeeper nodes being put on it. One option is to have the compose file deploy->placement option state the docker node name it should be placed on. Another option is to have the compose file use placement based on a label and then set each node with a different label.

So far, the placement logic inside the compose file is not sophisticated enough to define the scenario to prevent zookeeper instances potentially ending up on the same nodes.

Zookeeper Four Letter Commands

### ZooKeeper Commands: The Four Letter Words

ZooKeeper responds to a small set of commands. Each command is composed of four letters. You issue the commands to ZooKeeper via telnet or nc, at the client port.

Three of the more interesting commands: "stat" gives some general information about the server and connected clients, while "srvr" and "cons" give extended details on server and connections respectively.

conf

**New in 3.3.0:** Print details about serving configuration.

cons

**New in 3.3.0:** List full connection/session details for all clients connected to this server. Includes information on numbers of packets received/sent, session id, operation latencies, last operation performed, etc...

crst

**New in 3.3.0:** Reset connection/session statistics for all connections.

dump

Lists the outstanding sessions and ephemeral nodes. This only works on the leader.

envi

Print details about serving environment

ruok

Tests if server is running in a non-error state. The server will respond with imok if it is running. Otherwise it will not respond at all.

A response of "imok" does not necessarily indicate that the server has joined the quorum, just that the server process is active and bound to the specified client port. Use "stat" for details on state wrt quorum and client connection information.

srst

Reset server statistics.

srvr

**New in 3.3.0:** Lists full details for the server.

stat

Lists brief details for the server and connected clients.

wchs

**New in 3.3.0:** Lists brief information on watches for the server.

wchc

**New in 3.3.0:** Lists detailed information on watches for the server, by session. This outputs a list of sessions(connections) with associated watches (paths). Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.

wchp

**New in 3.3.0:** Lists detailed information on watches for the server, by path. This outputs a list of paths (znodes) with associated sessions. Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.

mntr

**New in 3.4.0:** Outputs a list of variables that could be used for monitoring the health of the cluster.

$ echo mntr | nc localhost 2185

zk\_version 3.4.0

zk\_avg\_latency 0

zk\_max\_latency 0

zk\_min\_latency 0

zk\_packets\_received 70

zk\_packets\_sent 69

zk\_outstanding\_requests 0

zk\_server\_state leader

zk\_znode\_count 4

zk\_watch\_count 0

zk\_ephemerals\_count 0

zk\_approximate\_data\_size 27

zk\_followers 4 - only exposed by the Leader

zk\_synced\_followers 4 - only exposed by the Leader

zk\_pending\_syncs 0 - only exposed by the Leader

zk\_open\_file\_descriptor\_count 23 - only available on Unix platforms

zk\_max\_file\_descriptor\_count 1024 - only available on Unix platforms

zk\_fsync\_threshold\_exceed\_count 0

The output is compatible with java properties format and the content may change over time (new keys added). Your scripts should expect changes.

ATTENTION: Some of the keys are platform specific and some of the keys are only exported by the Leader.

The output contains multiple lines with the following format:

key \t value

Here's an example of the **ruok** command:

$ echo ruok | nc 127.0.0.1 5111

imok