# Zookeeper in Docker Swarm

## Introduction

The goal is to get a cluster of zookeeper instances running inside of docker swarm, so that they all see each other and form a zookeeper cluster that could be used to support a kafka cluster.

## Result

The zookeeper cluster seemed to form, elect a leader and answer various commands from the test container. There seemed nothing wrong with the cluster. When the leader was manually failed, swarm restarted it and a new leader was elected.

## Compose file

The following compose file was used to create a three node zookeeper cluster and test whether it elects a leader and can then survive a failure of a node and then a recovery of the same node. Another option was to also map external ports to the internal port 2181 for clients to connect to. This is not needed as long as the containers are part of a single docker swarm and hence the network the zookeeper clients are on is also part of the same network the zookeeper cluster is on.

version: '3.1'

services:

zoo1:

image: zookeeper

restart: always

hostname: zoo1

networks:

- zoo\_net

environment:

ZOO\_MY\_ID: 1

ZOO\_SERVERS: server.1=0.0.0.0:2888:3888 server.2=zoo2:2888:3888 server.3=zoo3:2888:3888

zoo2:

image: zookeeper

restart: always

hostname: zoo2

networks:

- zoo\_net

environment:

ZOO\_MY\_ID: 2

ZOO\_SERVERS: server.1=zoo1:2888:3888 server.2=0.0.0.0:2888:3888 server.3=zoo3:2888:3888

zoo3:

image: zookeeper

restart: always

hostname: zoo3

networks:

- zoo\_net

environment:

ZOO\_MY\_ID: 3

ZOO\_SERVERS: server.1=zoo1:2888:3888 server.2=zoo2:2888:3888 server.3=0.0.0.0:2888:3888

test:

image: busybox:1.28

command: /bin/top

networks:

- zoo\_net

networks:

zoo\_net:

## Testing

The following tests were run from the test container, via docker exec -it test /bin/sh and then executing the following commands.

**Command**: echo srvr | nc zoo1 2181

**Results**:

echo srvr | nc zoo1 2181

Zookeeper version: 3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 04:05 GMT

Latency min/avg/max: 0/0/0

Received: 3

Sent: 2

Connections: 1

Outstanding: 0

Zxid: 0x0

Mode: follower

Node count: 4

**Command**: echo srvr | nc zoo2 2181

**Results**:

echo srvr | nc zoo2 2181

Zookeeper version: 3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 04:05 GMT

Latency min/avg/max: 0/0/0

Received: 2

Sent: 1

Connections: 1

Outstanding: 0

Zxid: 0x100000000

Mode: follower

Node count: 4

**Command**: echo srvr | nc zoo3 2181

**Results**:

echo srvr | nc zoo3 2181

Zookeeper version: 3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 04:05 GMT

Latency min/avg/max: 0/0/0

Received: 12

Sent: 11

Connections: 1

Outstanding: 0

Zxid: 0x100000000

Mode: leader

Node count: 4

Proposal sizes last/min/max: -1/-1/-1

**Command**: echo mntr | nc zoo3 2181

**Results**:

echo mntr | nc zoo3 2181

zk\_version 3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 04:05 GMT

zk\_avg\_latency 0

zk\_max\_latency 0

zk\_min\_latency 0

zk\_packets\_received 18

zk\_packets\_sent 17

zk\_num\_alive\_connections 1

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\_open\_file\_descriptor\_count 32

zk\_max\_file\_descriptor\_count 1048576

zk\_fsync\_threshold\_exceed\_count 0

zk\_followers 2

zk\_synced\_followers 2

zk\_pending\_syncs 0

zk\_last\_proposal\_size -1

zk\_max\_proposal\_size -1

zk\_min\_proposal\_size -1

## 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