# Kafka in Docker Swarm

## Introduction

The goal is to get a cluster of kafka instances running inside of docker swarm, so that they all participate with each other and handle messages correctly. The first thing to note is that the confluent documentation states ‘Bridge networking is currently only supported on a single host. For multiple hosts, you must use overlay networks which are not currently supported’. This is from the following link: <https://docs.confluent.io/current/installation/docker/docs/installation/index.html>.

## Result

The kafka cluster started and accepted various commands, see below. If only 2 nodes were started it would not work correctly due to a configuration parameter that was set to replication factor of 3. Once three nodes were in the kafka cluster everything seemed to work fine.

The compose file was tested with two different stacks running to simulate different environments. Both stacks worked correctly and independently, allowing different users or different environments to be created independently from one another.

More extensive testing needs to be carried out to ensure the failover & resiliency works.

## Compose file

The following compose file was used to create a three node kafka cluster. No host network was used and no ports were visible outside of the defined network.

version: '3.1'

services:

kafka1:

image: confluentinc/cp-kafka:latest

hostname: kafka1

networks:

- zoo\_net

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka1:19092

KAFKA\_LISTENERS: PLAINTEXT://kafka1:19092

extra\_hosts:

- "moby:127.0.0.1"

kafka2:

image: confluentinc/cp-kafka:latest

hostname: kafka2

networks:

- zoo\_net

environment:

KAFKA\_BROKER\_ID: 2

KAFKA\_ZOOKEEPER\_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka2:19092

KAFKA\_LISTENERS: PLAINTEXT://kafka2:19092

extra\_hosts:

- "moby:127.0.0.1"

kafka3:

image: confluentinc/cp-kafka:latest

hostname: kafka3

networks:

- zoo\_net

environment:

KAFKA\_BROKER\_ID: 3

KAFKA\_ZOOKEEPER\_CONNECT: zoo1:2181,zoo2:2181,zoo3:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka3:19092

KAFKA\_LISTENERS: PLAINTEXT://kafka3:19092

extra\_hosts:

- "moby:127.0.0.1"

networks:

zoo\_net:

## Testing

The examples of tests below need the network name changed to fit with the stack name. The examples below used a stack name of jon and a topic name of jon. With a stack name of dev the network would be changed to dev\_zoo\_net. The topic name could be anything.

### Create topic

Creates a specific topic that spans all of the three nodes.

docker run \

--net=jon\_zoo\_net \

--rm \

confluentinc/cp-kafka:latest \

kafka-topics --create --topic jon --partitions 6 --replication-factor 3 --if-not-exists --zookeeper zoo1:2181,zoo2:2181,zoo3:2181

### Describe topic

Gets the topic information from the cluster to check that it has the correct replication factor and number of partitions.

docker run --net=jon\_zoo\_net \

--rm confluentinc/cp-kafka:latest \

kafka-topics --describe --topic jon --zookeeper zoo1:2181

### Create messages

Creates a series of messages containing a single number with an increasing value. Tests the sending capability of the cluster.

docker run \

--net=jon\_zoo\_net \

--rm confluentinc/cp-kafka:latest \

bash -c "seq 42 | kafka-console-producer --broker-list kafka1:19092,kafka2:19092,kafka3:19092 --topic jon && echo 'Produced 42 messages.'"

### Read messages

Reads messages from the topic. The messages are a single number.

docker run \

--net=jon\_zoo\_net \

--rm \

confluentinc/cp-kafka:latest \

kafka-console-consumer --bootstrap-server kafka1:19092,kafka2:19092,kafka3:19092 --topic jon --from-beginning

## Challenges

Placement of kafka 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 kafka 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.