CoreOS Blog

Running etcd in Docker Containers

December 13, 2013 • By Rob Szumski

Update: This blog post refers to functionality in an older version of etcd, 0.4.x. Check out the updated Docker guide (https://coreos.com/etcd/docs/latest/docker_guide.html) for up-to-date information.

etcd (https://github.com/coreos/etcd) is a highly-available key value store for shared configuration and service discovery. Running etcd within a docker container is a convenient way to deploy etcd or test out a sample cluster. Etcd containers are very easy to run — you just need to manipulate a few of the flags to make the networking work correctly. You can grab the etcd container (https://quay.io/repository/coreos/etcd) or have docker download it automatically.

Start Containers

We're going to start three containers that use port `500x` for the client communication and port `800x` for the intra-cluster/raft communication. A few of the flags need to be modified per container:

- `-name` needs to be unique
- `-peer-addr` and `-addr` need to contain the IP where the machine can be reached on the outside of the container and a unique port number

- `-peers` needs to contain a list of current members of the cluster and the port for server/raft communication (`800x`). The first container doesn't need the `-peers` list since it will be the master.

In this example, all three of these containers will be running on the same machine. Here are the three docker run commands that will get the cluster up and running. First, export a variable with your public IP address:

```
export PUBLIC_IP=54.196.167.255
```

And then start up our leader + two followers:

```
docker run -d -p 8001:8001 -p 5001:5001 quay.io/coreos/etcd:v0.4.6 -peer-addr ${PUBLIC_IP}:8001 -addr ${PUBLIC_IP}:5001 -name etcd-node1
```

```
docker run -d -p 8002:8002 -p 5002:5002 quay.io/coreos/etcd:v0.4.6 -peer-addr ${PUBLIC_IP}:8002 -addr ${PUBLIC_IP}:5002 -name etcd-node2 -pee
rs ${PUBLIC_IP}:8001,${PUBLIC_IP}:8002,${PUBLIC_IP}:8003
```

```
docker run -d -p 8003:8003 -p 5003:5003 quay.io/coreos/etcd:v0.4.6 -peer-addr ${PUBLIC_IP}:8003 -addr ${PUBLIC_IP}:5003 -name etcd-node3 -pee
rs ${PUBLIC_IP}:8001,${PUBLIC_IP}:8002,${PUBLIC_IP}:8003
```

Testing the Cluster

You should be able to test the cluster by curling any of the machines in the cluster or viewing the dashboard in a browser. Let's request `/stats/leader` which should indicate that all of our followers connected successfully:

```
curl -L 54.196.167.255:5001/v2/stats/leader
```

You should see something that looks like:

```
{"leader":"etcd-node1","followers":{"etcd-node2":{"latency":{"current":1.33342,"average":1.6575929235264224,"standardDeviation":0.85967933750
97724,"minimum":1.034932,"maximum":27.428239},"counts":{"fail":4,"success":3936}},"etcd-node3":{"latency":{"current":1.448973,"average":1.745
7679780163597,"standardDeviation":2.3964311316740465,"minimum":1.063433,"maximum":86.745084},"counts":{"fail":14,"success":3912}}}}
```

As you can see, `etcd-node1` is the current leader and `etcd-node2` and `etcd-node3` are successfully connected as followers. You can check out the dashboard module at `/mod/dashboard` but you'll need to launch etcd with `-cors='*'` to make the cross-origin requests work.

Flags

Below is a quick description of the flags we used in this example and some other common ones:

FLAG	DESCRIPTION
peers	Comma-separated list of cluster members. Any valid IP address will work. Use server/raft port for each member.
peers-file	A file containing the list of cluster members
name	The unique name of the node.
addr	Address that the etcd clients will connect to.
bind-addr	Interface that the etcd server will listen for client connections.
peer-addr	Address that the etcd server will advertise for intra-cluster/raft communication.
peer-bind-addr	Interface that the etcd server will listen for intra-cluster/raft connections.
data-dir	Path to the directory that stores data
ca-file	Path to a CA file. Used for SSL client certificates.
cert-file	Path to a certificate file. Used for SSL client certificates.
key-file	Path to a key file. Used for SSL client certificates.
peer-ca-file	Path to a CA file. Used for intra-cluster SSL communication.
peer-cert-file	Path to a certificate file. Used for intra-cluster SSL communication.
peer-key-file	Path to a key file. Used for intra-cluster SSL communication.
cors	White-listed addresses that cross-origin requests are allowed from .

OS (https://www.coreos.com)

COMPANY

About (https://coreos.com/about/)