alexellis / **arm.md**

Last active a month ago • Report gist

Test swarm connectivity

arm.md

Regular PC / cloud architecture (64-bit)

See the snippets "guide.md" and "redis.md" below.

Swarm on a Raspberry Pi

If you're wanting to run Docker Swarm on your Raspberry Pi checkout these instructions:

- Live Deep Dive - Docker Swarm Mode on the Pi

A test repo is available here with custom images for the ARM platform:

- Github repo: Swarmmode tests

guide.md

Basic connectivity tests

Kernel check

When setting up Swarm on a new image (even if it has the same name as elsewhere) it's important to check all the Kernel-level configuration.

Use the check-config script from Docker:

```
curl -sSL https://raw.githubusercontent.com/docker/docker/master/contrib/check-config.sh | bash
```

Look out for red errors - especially anything referring to IP or VLAN.

Try round-robin / swarm overlay networking

On the manager:

```
$ docker network create --driver=overlay --attachable=true testnet
$ docker service create --network=testnet --name web --publish 80 --replicas=5 nginx:latest
```

Or force Nginx to only run on the workers:

```
$ docker service create --network=testnet --name web --publish 80 --constraint='node.role==worker' --
replicas=5 nginx:latest
```

Now use `curl` to test connectivity from within a container.

```
$ docker run --name alpine --net=testnet -ti alpine:latest sh

# apk add --no-cache curl
# curl web
# curl web
# curl web
# curl web
# curl web
```

You should see 5 responses, once you've seen this try connecting from your PC/Laptop. If there is noticable lag between any of the curl commands you may have to move onto debugging your open ports and `advertise-addr`.

From the Docker Swarm Docs:

> The following ports must be available. On some systems, these ports are open by default.

- TCP port 2377 for cluster management communications
- TCP and UDP port 7946 for communication among nodes
- UDP port 4789 for overlay network traffic

Debugging

- Make sure you pass `--advertise-addr` to the docker `swarm init` and `swarm join` commands alike.

redis.md

Testing inter-service connectivity

This test uses the redis server and client. You will schedule redis to run on a worker, then log into the manager and connect with a redis client image.

On the manager:

```
root@scw-93f02d:~# docker service create --network=testnet --name redis --publish 6379 --replicas=1 --
constraint='node.role != manager' redis:latest
xa5f3uiseu7k6rz45x41a3nem
```

Check service started:

```
root@scw-93f02d:~# docker service ps redis
ID            NAME     IMAGE         NODE    DESIRED STATE  CURRENT STATE          ERROR  PORTS
psrpw5akjjl5  redis.1  redis:latest  worker  Running        Running 2 seconds ago
```

From the manager: run a container to attach to redis:

```
root@scw-93f02d:~# docker run --net=testnet -ti redis:latest redis-cli -h redis
redis:6379> incr docker-awesome
(integer) 1
redis:6379>
```

Control + D to exit, then check swarm status:

```
root@scw-93f02d:~# docker service ls
ID            NAME   MODE        REPLICAS  IMAGE
sjkhmujg1ehi  web    replicated  5/5       nginx:latest
xa5f3uiseu7k  redis  replicated  1/1       redis:latest


root@scw-93f02d:~# docker node ls
ID                           HOSTNAME    STATUS  AVAILABILITY  MANAGER STATUS
39kw5i5aix9mxzw06h6y3v07e    worker      Ready   Active
z4vpwtut5v66nwi0jijco6mjn *  scw-93f02d  Ready   Active        Leader
root@scw-93f02d:~#
```

developius commented on Mar 28

Yup, that works