Container Linking Notes

Scenario

Derby should be run in a separate container because we don't want monolithic containers. But I want to be able to run multiple Derby containers and yet the Derby container runs Derby on port 1527. I also need to be able to link the WebLogic container to the Derby container – ideally without exposing the Derby port to the outside world. Docker linking is the solution to this problem.

Demo

- First run Derby opening up port 1527 – note we can access this container from NetBeans on the Mac
  - docker run -p 1527:1527 –name derby -v ~/docker/derby-databases:/appl/derby/databases taylodl/derby &
    - Note we explicitly named the container
      - This allows us to easily refer to this container by name
      - This will be convenient later
  - docker ps
    - Note our Derby container is running and it's name is 'derby'
  - ifconfig
    - Get address of eth0
  - Access from NetBeans on Mac
- Show that we can't access the 'derby' container from inside taylodl/wls-mydomain
  - docker run -p 7001:7001 taylodl/wls-mydomain &
  - Launch browser & goto WebLogic admin console
    - Attempt to add Derby datasource
    - When testing the configuration you'll get a connection refused error
    - There's no amount of configuration that can resolve this issue
      - There's no route between the containers

- Stop and cleanup all containers
  - stop: docker stop `docker ps -q`
  - cleanup: docker rm `docker ps -aq`
- Run Derby without exposing port 1527 – note that we can't access it from NetBeans on the Mac
  - docker run –name derby -v ~/docker/derby-databases:/appl/derby/databases taylodl/derby &
  - Access from NetBeans on Mac
- Now link the taylodl/wls-mydomain container to the Derby container, first running bash and then running WebLogic
  - docker run -i -t –link derby:derby_server taylodl/wls-mydomain /bin/bash
    - env
      - Look at all the DERBY_SERVER entries in the environment
      - Notice too the IP addresses are all from Docker's 172.17.x.x subnet
    - cat /etc/hosts
      - Note derby_server has an entry
  - docker run -p 7001:7001 –link derby:derby_server taylodl/wls-mydomain &
    - Launch browser and go to WebLogic admin console
    - Attempt to add Derby datasource using 'derby_server' as the host
    - Successful!
- Stop all running taylodl/wls-mydomain instances, but leave Derby running
- Now run taylodl/wls-responsive-todo
  - docker run -p 7001:7001 –link derby:derby_server taylodl/responsive-todo &
  - Launch browser and go to WebLogic admin console
    - See that data source is already defined
    - Goto monitoring tab and see our database is up and running
  - docker ps inspect
    - running taylodl/wls-responsive-todo container
    - Note the links
- Now run taylodl/wls-responsive-todo with the application

- docker run -p 7001:7001 –link derby:derby_server -v ~/docker/wls-autodeploy:/appl/oracle/middleware/wls/wls12120/user_projects/domains/mydomain/autodeploy taylodl/wls-responsive-todo &
  - Launch browser and go to localhost:7001/responsive-todo

-