Narrative – Advanced Docker

Slide 1 – Introduction

- General introduction
- Note I'm a JEE guy
- Work for American Electric Power
- Will be approaching this subject from an enterprise perspective

Slide 2 – Private repository

- Many enterprises aren't going to use images from the Docker public repository
- Yet you need to be able to transfer images from one machine to another throughout the enterprise
- You need a private repository
- There's a Docker image having everything you need
  - registry
- Get the public registry image from the public GitHub repository and then push it to your private public registry
- Goto https://github.com/docker/docker-registry
  - You can store your images locally or in the cloud of your choosing
  - Makes sense to use Docker to run a Docker registry!

Slide 3 – Setting up your private registry

- The easiest thing is to use the registry image from the Docker public repository
- Being an enterprise and all I'm assuming you have a storage frame and you're not going to be using local storage
  - Besides, at AEP we have the registry running on a small virtual machine in our test lab
- Our virtual machine is running Red Hat Enterprise Linux
- Basic strategy
  - Using local storage on the storage frame
  - Mount the location to the storage frame in your local filesystem
  - Run the registry image from the Docker public registry
    - You need to open port 5000
    - You need to mount the directory where you mounted the local storage frame into your container
    - You need to set the STORAGE_PATH environment variable to the location where you mounted the local storage frame into your container
    - Since you probably don't want to continue running the registry image from the Docker public registry you can push it to your (now running) private registry
    - Now you can re-run the registry image from your private registry
      - This is your confirmation your private registry is up and running properly

Slide 4 – Debugging Docker

- Simplest technique is to run /bin/bash inside your image
  - Usually /bin/bash is available

    - Some people use /bin/sh
    - Whatever you got is what you gotta use
    - This is a static technique of debugging
    - You can inspect environment variables
    - You can inspect file contents
    - Basically it's useful for checking out newly-created images
- Docker cp
    - Copy files out of a running container
    - Useful for copying application log files for further examination
- Docker logs
    - Somewhat of a misnomer
    - The “logs” are stdout/stderr
    - You can tail the “logs” to keep tabs on a container
    - You can look at what was written in the past too
- Docker exec
    - Run a command inside an already running container
    - You can run interactively, too
        - Meaning you can run bash
    - You can run ps too
        - There's a lot of information you can discover
            - CPU usage
            - memory usage

Slide 5 – Docker in Docker

- Sometimes you may want to run Docker inside Docker
    - Let's not consider why
- You need 3 things to make it happen
    - Your image must have the libcontainer library installed
        - taylodl/centos does
    - You need to make the Docker client available in the container
        - -v `which docker`:/usr/bin/docker
    - You need to make the Docker socket available in the container
        - -v /var/run/docker.sock:/var/run/docker.sock
    - Then you're set!

Slide 6 – Linking containers

Proceed with Container Linking Notes

Slide 7 – Volume containers

Proceed with Container Volume Notes

Slide 8 – Fig

- Their byline says “Fast, isolated development environments using Docker”
    - whatever
    - It's about container coordination
- We've seen taylodl/wls-mydomain running

- But what self-respecting enterprise application doesn't use a database?
- Where is it?
- taylodl/derby
- Now we have to startup taylodl/derby before starting taylodl/wls-mydomain
  - there's a dependency, since this domain configures a Derby data connection
- We could use a shell script
  - As we'll see this isn't as powerful
- See Fig Notes

Slide 9 – Creating centos image

- Display 'Creating Centos Image' document
  - Adjust font size for readability
- I used febootstrap
  - supermin is available now
  - I haven't used supermin
- Basically you include all your packages
  - Can include groups, too
- I put the result in ~/centos65 directory
  - Go ahead and look at that directory
- We use tar to create a tarball
  - For those not familiar with Linux, Java's 'jar' command is modeled after unix's 'tar' command and has the same options except for 'z', which is gzipping
- Now we import the tarball into a new Docker image
  - Docker import works kinda funny
  - The first parameter is an URL
    - Use '-' to read from STDIN instead of an URL
    - Then we mount the tarball onto STDIN
- Tag the image as taylodl/centos:6.5
  - The taylodl/centos image we just made will have been tagged as 'latest'
    - taylodl/centos:latest
  - We want it tagged as taylodl/centos:6.5
    - So we can make a taylodl/centos:7.0 if we want
      - And distinguish between the two
- Then we push both images into the Docker Hub repository

Slide 10 – Docker errata

- 'docker run' creates containers, try not to keep re-creating a container
  - docker stop/start instead
  - Naming containers also makes it easy to see what the container is and how it's being used
    - Also gives you an easy way to refer to the container
  - Naming containers are critical when working with volume containers
- 'docker save' and 'docker load' allow you to transfer images without using a repository
  - Really, seeing how easy it is to setup a private repository and how easy it is to setup a DockerHub account to create your own pubic repository – why would you ever really need to do this?
- 'docker export' and 'docker import' are great for copying data to/from a volume container

    - better than 'docker cp' because it already puts the data into a tarball
    - it's also a good way to get data for creating a volume image
        - taylodl/derby-tododb, taylodl/wls-autodeploy-responsive-todo were created using this technique
            - taylodl/derby-tododb
                - run taylodl/derby-volume
                - create your database
                - export the database
                - create a new docker image from taylodl/derby-volume adding the exported data into the image
                - See taylodl/derby-tododb Dockerfile for details
            - taylodl/wls-autodeploy-responsive-todo
                - It's probably easier to use 'docker cp' to copy the WAR file out of the image and into your local directory
- 'docker rm' gets rid of a container
    - generally faster to stop/start a container
    - a volume container will lose all it's data
        - sort of
            - the last container referencing the mounted volume that's removed will remove the data
            - oh yeah – volumes can be chained though containers.
    - Good strategy is to keep containers around once they've been created
        - took me a while to get into a better habit of doing that
- Fig is the best way to coordinate multiple containers
    - Docker purchased Fig
    - They've announced plans to do even more