Iteration #4: Docker 1.9 upgrade, Swarm, and Secret Data

Background

In this iteration the team is planning to upgrade Docker to the latest stable release on all of the POC hosts, as well as, the build/package server. In order to do this, the Systems team will have to deviate from the Red Hat packaged Docker, which is not currently supporting Docker 1.9.x. Once the infrastructure has been upgraded, the team will build a Docker Swarm using the three docker-web hosts. Once the Swarm is up and running, we will design an application to test various deployment scenarios. For example, we will attempt to test communication across multiple hosts using Docker's new network overlay features. Additionally, we will test Docker's ability to scale out services across multiple hosts.

Also in this iteration, the team will begin looking into the workflow associated with moving code through the various software development lifecycle steps. The team will research options for storing configuration information. For example, if we use different databases for different environments, do we have to have a separate Dockerfile/image for each environment? Or, is this information passed in as environmental variables or configuration files? The team will report back on the security implications associated with the various approaches. As part of this research, the team will also look at how a tool like Jenkins could be coupled with Docker to automate parts of the workflow and assist with deploying an application into each dev/test/prod environment.

Goals

1. Upgrade Docker to 1.9.x on docker-web1, docker-web2, docker-web3, docker-app1, docker-app2, docker-app3, and dockerpackage (Mike L., Patrice)
2. Create a Docker Swarm with docker-web1, docker-web2, and docker-web3. Make docker-web3 the Swarm Master. (John, Alex T., Mike L.)
3. Create test application that demonstrates swarm capabilities (John, Alex T., Paul)
4. Setup a Docker private registry on docker-app3 (John, Alex, and Mike L.)
5. Research and determine the proper place to store sensitive configuration information (i.e. db connection information, passwords, etc) for build (Mike M., Mike P., Alex T, John)
6. Determine Docker workflow for migrating code from development to production exploring opportunities for automation (Mike M., Mike P., Alex T, John)
7. Experiment with Dockerizing more complicated applications using Docker Compose (Mike M., Mike P., Alex T, John)

Schematics

- Insert diagrams here

Artifacts

- Insert links here

For step 5, storing sensitive information:

1 – Information about all the successive layers from a Docker image can be obtained using dfimage (3rd party tool) or by analyzing the images in Docker hub.

2 – In order to protect sensitive information we can use docker-compose:

[root@docker-web1 ~]# docker-compose -v

docker-compose version 1.5.2, build 7240ff3

3 – Docker-compose uses docker-compose.yml as its configuration file.

Information that can be added to docker-compose.yml is shown below:

environment

Add environment variables. You can use either an array or a dictionary.

Environment variables with only a key are resolved to their values on the machine Compose is running on, which can be helpful for secret or host-specific values.

environment:

  RACK_ENV: development

  SESSION_SECRET:

env_file

Add environment variables from a file. Can be a single value or a list.

Environment variables specified in environment override these values.

env_file:

  - .env

RACK_ENV: development

4 – Following the creation of docker-compose.yml, we run:

Docker-compose build

5 – We then run:

Docker-compose up

Summary

Potential Next Steps

1. Upgrade dockerpackage to 1.9.1 (Mike Lewis)
2. Finish building swarm cluster (John Wang, Paul, Mike Lewis)
3. Determine difference between Dockerfile and Docker-Compose (Alex)
4. Iterate on the best practices for storing secret information with Docker (Alex, Mike M.)
5. Use Jenkins to build and deploy WebEase (Daniel, Mike M.) - assuming Brad is okay with this