SHIFTING LEFT FOR CONTINUOUS QUALITY IN DATA

credit: www.cloudtestsoftware.com
ABOUT ME

- M.S in Electrical and Computer Engineering from the State University of New York, Stony Brook
- 12+ years in software engineering focusing on quality
- Currently Staff Software Engineer in Quality at Intuit
OVERVIEW

WHAT WILL BE COVERED:

- Data pipeline overview
- Testing ETL's
  - Why you should test and validate your ETL’s
  - Testing journey
  - “Unit” testing and integration testing
- Using Docker to shift left for continuous quality
- Why Docker for Data Engineering?
  - Docker ecosystem architecture
  - Docker networking, volumes and persistent storage
- Docker for Testing
Why don’t we write automated tests?

- It’s time consuming
- Easier to test manually
- Setting up dummy data is hard
- This feature should have gone out yesterday
- No mature testing frameworks
- The data domain is different
WHY SHOULD WE TEST IN A AUTOMATED FASHION? (YES, DATA PROJECTS AS WELL!)

- Makes you a better developer
  - Do not want PagerDuty pinging you at 2am!

- Makes your consumers happy
  - Letting customers find issues is a really long development cycle

- Helps with your 360 degree feedback from the team!
FUNCTIONAL TESTING: ITERATION 0

- Manual testing

- Pros
  - Easy to get started

- Cons
  - Not scalable
  - Time consuming
  - Have to maintain test documentation
FUNCTIONAL TESTING: ITERATION 1

- Shell script based testing
- Pros
  - Moderately easy to get started
  - Somewhat automated
- Cons
  - shell scripting, ugh!
  - Not extensible
FUNCTIONAL TESTING: ITERATION 2

- Java, TestNG based tests

Pros
- Automated, Repeatable
- Based on community supported testing frameworks

Cons
- Difficult to test edge cases
- Hard to read and maintain
- Developer adoption
UNIT TESTING

FUNCTIONAL TESTING: ITERATION 3

- Embedded Unit Tests in SQL/ETL files
- Setting up local VMs with required databases

Pros
- Easier to understand ETL flow
- SQL queries and test assertions are paired
- Easier developer adoption

Cons
- Local setup of required databases
- Need to fake test data
OPERATING SYSTEM VIRTUALIZATION

- Isolated user space instances
- Isolated view of processes and file system
- But, shares the host Linux kernel
**ADVANTAGES**

- Application and dependencies packaged together
  - **Dockerfile**: Single deployable artifact
- Docker images are portable and shareable
  - DockerHub, Amazon EC2 Container Registry, private registries
- Isolate software from hardware resources
FROM avinashp22/hadoop:2.7.3
MAINTAINER avinashp22 <avinash_padmanabhan@intuit.com>

ENV HIVE_VERSION 1.2.1

# setup dependencies, extract and install hive
RUN apt-get install -y debconf-utils && \
    /bin/bash -c "debconf-set-selections <<< 'mysql-server mysql-server/root_password rd password test1234'" && \
    /bin/bash -c "debconf-set-selections <<< 'mysql-server mysql-server/root_password rd_again password test1234'" && \
    apt-get install -y mysql-server libmysql-java && \
    wget http://apache.claz.org/hive/stable/apache-hive-$HIVE_VERSION-bin.tar.gz && \
    tar -xzvf apache-hive-$HIVE_VERSION-bin.tar.gz && \
    rm apache-hive-$HIVE_VERSION-bin.tar.gz && \
    mv apache-hive-$HIVE_VERSION-bin /usr/local/hive && \
    apt-get clean

# setting up hive_home and path
ENV HIVE_HOME /usr/local/hive
ENV PATH $PATH:$HIVE_HOME/bin

# copying hive configuration from local
"hadoop-docker/hive/Dockerfile" 37L, 1499C written
FROM centos:centos6.6
MAINTAINER avinashp22 <avinash_padmanabham@intuit.com>

# Set java environment variables
ENV JAVA_VERSION 1.7.0
ENV JAVA_HOME /usr/lib/jvm/java-1.7.0-openjdk.x86_64

# Set Maven environment variables
ENV M2_HOME /usr/local/maven
ENV PATH=${M2_HOME}/bin:${PATH}

# Update image, install Java, Maven and dependencies
RUN yum install -y java-"${JAVA_VERSION}"-openjdk-devel tar.x86_64 wget && \ 
    wget http://mirror.cc.columbia.edu/pub/software/apache/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-bin.tar.gz && \ 
    tar xzf apache-maven-3.0.5-bin.tar.gz -C /usr/local && \
    cd /usr/local && \
    ln -s apache-maven-3.0.5 maven && \
    yum -y install dialog which openssl openssh-server openssh-clients mcelog gdb s \
    ystat sudo && \
    yum clean all

ENV LANG en_US.utf8
ENV TZ "US/Western"
:set nonu
DOCKER ECOSYSTEM

docker build
docker pull
docker run

Docker daemon

Containers

Images

Registry

Credit: microsoft.com
DOCKER VOLUMES

- Data Volumes are on local file systems and retain data beyond the life of the container
- Mounting the volume allows you to share data between the local file system and the container file system
- Mount volume to local Github repo from your container
  - `docker run -p 5433:5433 -v local_github_repo_path:container_path`
DOCKER NETWORKING

- Docker creates 3 networks on the docker host
  - bridge, host, none
- Docker also allows you to create user defined networks
  - Can be used to link containers which are part of the data pipeline

Credit: developerblog.info
Running multi-container applications

- Using a single command, you manage all the services in a single host.

- You just need to create configuration file called docker-compose.yml, which contains the specification to create and run a set of docker containers.
version: '2'
services:
  vertica:
    image: avinashp22/vertica:7.1.2-11
    ports:
      - "5433:5433"
  depends_on:
    - hive
  hive:
    image: avinashp22/hive:1.2.1
    ports:
      - "50070:50070"
DEMO
#!/bin/bash

trap clean_exit 2

function clean_exit() {
    echo "Got SIGINT, exiting..."
    docker kill hive vertica
    docker rm hive vertica
    trap - 2
    kill -s 2 "$$"
}

# Start the docker containers
docker run -d --net sbgdata --name hive -p 50070:50070 avinashp22/hive:1.2.1
docker run -d --net sbgdata --name vertica -p 5433:5433 avinashp22/vertica:7.1.2-11

# Run the unit tests
sleep 37
mvn clean test -Dtest=sampleUnitTests#testSampleProduct -DdataSet=$1

# Clean up
docker kill hive vertica
docker rm hive vertica

? ?
? ?
FUTURE WORK

- Ability to fake testing data at test run time
  - How do you really use “big data” for testing?
- A way to test “code coverage” for SQL scripts
- As we move to real time ingestion, look into running spark jobs within docker containers
CONCLUSION

▸ Automated unit and functional testing for your data projects

▸ Start your testing as early as possible, testing locally whenever possible

▸ Bring development best practices in to your data projects