01B: Spark on Zeppelin – custom Dockerfile

Posted on July 22, 2018

Pre-requisite: Docker is installed on your machine for Mac OS X (E.g. $ brew cask install docker) or Windows 10. Docker interview Q&As.

What is Apache Zeppelin?

Zeppelin is a web based notebook to execute arbitrary code in Scala, SQL, Spark, etc. You can mix languages. Apache Zeppelin helps data analysts, data scientist, and business users to get better understanding of data. As described below you can quickly explore data, create visualizations and share

300+ Java Interview FAQs

- 300+ Java FAQs
- 16+ Java Key Areas Q&As
- 150+ Java Architect FAQs
- 80+ Java Code Quality Q&As
- 150+ Java Coding Q&As

300+ Big Data Interview FAQs

- 300+ Big Data FAQs
- Tutorials - Big Data
  - TUT - Starting Big Data
  - TUT - Starting Spark & Scala

their insights, as web pages, with various stakeholders. For example

1) Prepare data using Shell by say downloading files with curl/wget, and then inject to HDFS.

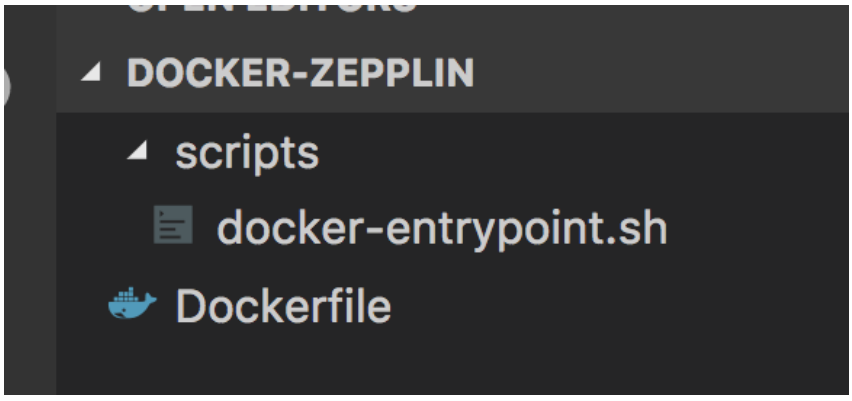
2) Perform data analytics with Spark (i.e Scala) or pyspark (i.e. Python).

3) Perform simple visualizations in SQL.

4) Export the results with Shell, and publish to create graphs.

How to install Apache Zeppelin on Docker?

You need around ~4GB disk space to create a Docker container with Ubuntu OS and Apache Zeppelin.



Apache Zeppelin on Docker

Step 1: Create a folder say “docker-zepplin” under a folder named “projects”. Within the folder docker-zepplin, create a file named “**Dockerfile**” and it should have the following contents. It installs Java on Ubuntu and then the Zeppelin note book.

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

The Dockerfile shown below was simplified from the image "apache/zeppelin", which is already available to be used from the Docker Hub. It installs all the conda packages, Python related packages, and "R" language related packages.

```
FROM ubuntu:16.04
MAINTAINER java-success.com

ENV ZEPPELIN_PORT 8080
ENV ZEPPELIN_HOME /usr/zeppelin

EXPOSE $ZEPPELIN_PORT

#install java
RUN apt-get update && \
  apt-get install -y software-properties-common &&
  add-apt-repository ppa:openjdk-r/ppa && \
  apt-get update && \
  apt-get install -y openjdk-8-jdk

#install other
RUN apt-get install -y \
  npm \
  vim \
  wget

#install Zeppelin
RUN wget http://apache.mirror.amaze.com.au/zeppel
  tar -zxf zeppelin-0.8.0-bin-all.tgz -C /usr/ &&
  mv /usr/zeppelin* $ZEPPELIN_HOME

WORKDIR $ZEPPELIN_HOME

# Create a Zeppelin user
RUN useradd --home $ZEPPELIN_HOME --shell /bin/ba
RUN chown zeppelin:zeppelin -R $ZEPPELIN_HOME


COPY scripts/docker-entrypoint.sh /docker-entrypo
RUN chmod 777 /docker-entrypoint.sh

ENTRYPOINT ["/docker-entrypoint.sh"]

```

The https://zeppelin.apache.org/download.html has the link to Zeppelin downloads. Get the binary package with all interpreters as shown above.

Step 2: Create a "scripts" folder, and create the "**docker-entrypoint.sh**" file with the following contents.

```
#!/bin/bash

# Start Zeppelin as "zeppelin" user.
su - zeppelin -c "$ZEPPELIN_HOME/bin/zeppelin-daem

# Non-ending command to keep the container alive.
tail -F n0 /dev/null

```

Step 3: Create a docker image with the following command. The docker command looks for a "**Dockerfile**" in the current (i.e ".") folder.

```
$ docker build -t zeppelin-simple:0.8.0 .

```

This may take a while to download the 1 GB zeppelin-0.8.0-bin-all.tgz. Once the image is created, you can verify it with:

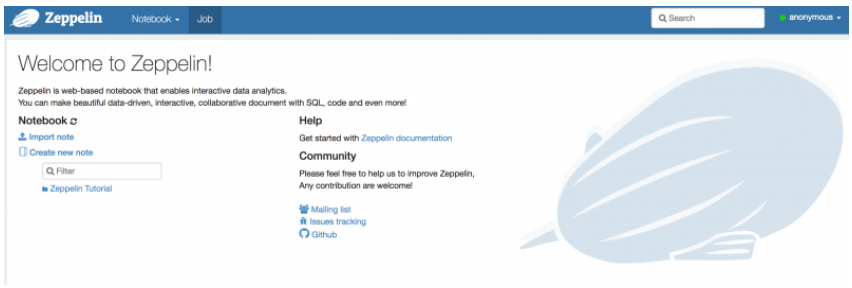
```
$ docker images

```

```

REPOSITORY          TAG
zeppelin-simple     0.8.0

```

Step 4: You can now run a docker container with this image as shown below:

```
$ docker run -it -p 8080:8080 zeppelin-simple:0.8.(

```

Step 5: Go to a browser and type: “http://locahost:8080”.



Apache Zeppelin UI

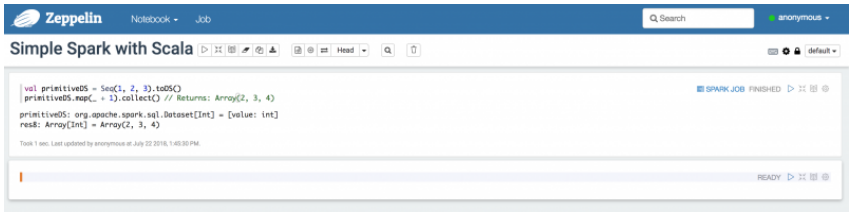
Step 6: Select the link “Create new note”, and name it “Simple Spark with Scala” and select the **interpreter** as “spark”.

Type the following simple Spark code to add 1 to the given set of numbers.

```
1 val primitiveDS = Seq(1, 2, 3).toDS()
2 primitiveDS.map(_ + 1).collect() // Returns: Array(
3
```

Press the play button, and the output will be:

```
1 Array(2, 3, 4)
2
```



Spark using Scala

‹ Event sourcing & CQRS interview Q&As

01A: Spark on Zeppelin – Docker pull from Docker hub ›

Disclaimer