**Directory**

* API Source: mock API data based on sampling
* Flume: Flume config
* Spark: Spark streaming processing code + mock file data based on random generation
* vagrant: Vagrant config
* Zeppelin: exported Zeppelin notebook

**Setup**

1. Setup VM

* Copy vagrant folder to desired location, this folder will be the shared VM folder
* Install Vagrant (<https://www.vagrantup.com/downloads.html>)
* Modify Windows host file (run Notepad with Admin right, open C:\Windows\System32\drivers\etc\hosts, add a record in new line: 127.0.0.1 [lambda-pluralsight](http://lambda-pluralsight/))
* Ensure only there is only one mapping to 127.0.0.1
* (Optional) Install Cygwin (<https://www.cygwin.com/>)
* Open Cygwin terminal, navigate to the vagrant folder, type command: **vagrant up** (this will download the VM from online repo and set it up in VirtualBox/VMWare based on the Vagrantfile config)
* To stop VM, use **vagrant halt**

2. Setup Flume

* In the vagrant folder, use command: **vagrant ssh** to ssh into VM
* Install Flume following Tun Lin’s guide (but taking the flume.conf from the Flume folder)

3. Setup IDE

* Install IntelliJ Community Edition
* Import the Spark/pom.xml as a Maven project
* Check “Search for projects recursively” and “Import Maven projects automatically” checkboxes
* Select “bead:lambda:1.0-SNAPSHOT” to import
* Unzip Spark/WinUtils/bin/unzip-here.zip for debugging inside IDE
* To debug in IDE, update the path in src/main/scala/utils/SparkUtils.scala on hadoop.home.dir to point to the WinUtils folder
* Search for “maven-antrun-plugin” in the root pom.xml, update the copy JAR location to the shared Vagrant folder
* Wait for IntelliJ to restore the packages (auto restore based on pom.xml, will take some time for the first setup)
* To build, open the Maven Projects panel on the right, open yelp/Lifecycle and double click install, this will build an output and copy the final JAR to the VM shared folder

4. Submit Job in VM

**/pluralsight/spark/bin/spark-submit --class streaming.StreamingJob /vagrant/yelp-1.0-SNAPSHOT-shaded.jar**

**/pluralsight/spark/bin/spark-submit --class streaming.StreamingJob --master yarn --deploy-mode cluster /vagrant/yelp-1.0-SNAPSHOT-shaded.jar** (this is the cluster mode version)

5. Start Flume in VM

**/vagrant/apache-flume-1.9.0-bin/bin/flume-ng agent –conf /vagrant/apache-flume-1.9.0-bin/conf/ -f /vagrant/apache-flume-1.9.0-bin/conf/flume.conf -Dflume.root.logger=DEBUG,console -n agent\_review**

6. Generate API stream (this completes the Flume API stream)

* Run review\_source\_api.py

7. Start Zeppelin

* [http://localhost:8988](http://localhost:8988/)
* Import Zeppelin notebook from Zeppelin folder
* Run the desired sections in the notebook
* Start file stream streaming context
* 8. Generate file stream (from IntelliJ, this completes the file stream)
* Update paths in src/main/resources/application.conf
* Run src/main/scala/reviewstream/ReviewProducer.scala

**Useful Links from Host**

<http://localhost:50070> (HDFS browser, Utilites->Browse File System)

<http://localhost:8988> (Zeppelin)

<http://localhost:8088> (YARN UI)

**Useful HDFS Command from VM**

hdfs dfs -list /

hdfs dfs -rm -r /flume

hdfs dfs -cat /flume/yelp/2019-04-20/events-0409.1555733355201 (update name accordingly)

**Scala Classes**

* Settings: binding configs from application.conf
* domain: all domain case classes
* functions: the two state mapping functions (input to MapWithState)
* ReviewProducer: standalone random file generator
* StreamingJob: main Spark streaming job
* SparkUtils: Spark context creation infrastructure code (checkpointing and restore etc)

**Miscellaneous**

* ShadedJar: all-in-one JAR with all the dependencies