

## Exercise #3:

# Set up simulator, Apache services and IDE environment

## Introduction

In this exercise, we are going to set up all the services required to run the Big Data / IoT real time (truck) event processing exercises. We will install NiFi onto our Hortonworks Sandbox while activating Kafka and Storm for later use in the exercise series. We will also walkthrough how to set up an IDE on our local machine for Storm development testing and deploy our Storm project to the Sandbox for further testing.

## Pre-requisites

- Read **Table 1** to figure out some basic details about your sandbox

Parameter	Value (VirtualBox)
Host Name	127.0.0.1
Port	2222
Terminal Username	root
Terminal Password	hadoop

**Table 1: Virtual Machine Information**

- Added `sandbox.hortonworks.com` to your `/private/etc/hosts` file (mac and linux users)
- Added `sandbox.hortonworks.com` to your `/c/Windows/System32/Drivers/etc/hosts` file (windows 7 users)

The following terminal commands in the exercise instructions are performed in VirtualBox Sandbox and Mac machine. For windows users, to run the following terminal commands, download [Git Bash](#).

If on mac or linux, to add `sandbox.hortonworks.com` to your list of hosts, open the terminal, enter the following command, replace {Host-Name} with the appropriate host for your sandbox:

```
echo '{Host-Name} sandbox.hortonworks.com' | sudo tee -a /private/etc/hosts
```

If on windows 7, to add `sandbox.hortonworks.com` to your list of hosts, open git bash, enter the following command, replace {Host-Name} with the appropriate host for your sandbox:

```
echo '{Host-Name} sandbox.hortonworks.com' | tee -a /c/Windows/System32/Drivers/etc/hosts
```

## Section #1: Setup Nifi Environment


### STEP 1: Install Nifi by Ambari Wizard

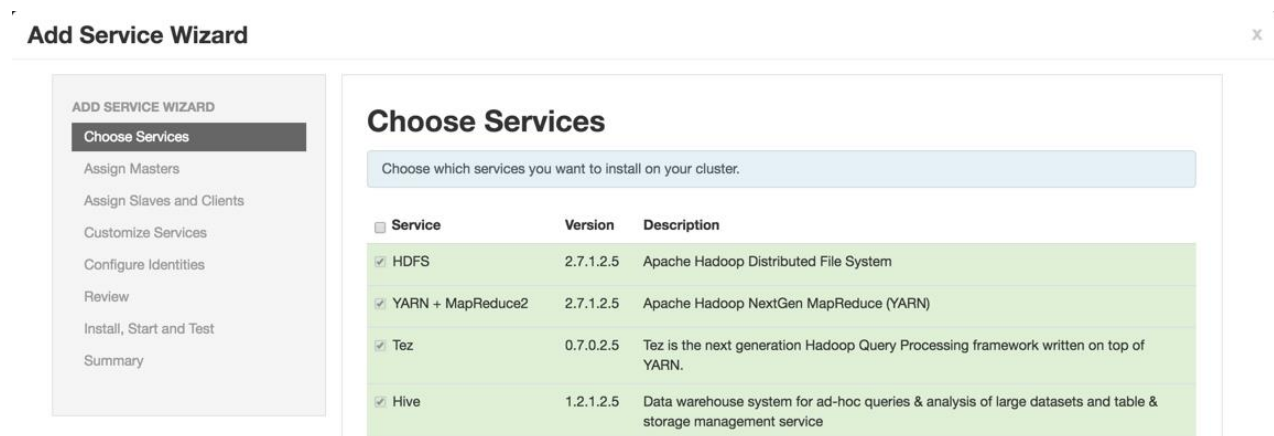
Use the following steps to perform the NiFi installation:

1. SSH into the Sandbox VM:

```
ssh root@sandbox.hortonworks.com -p 2222
```

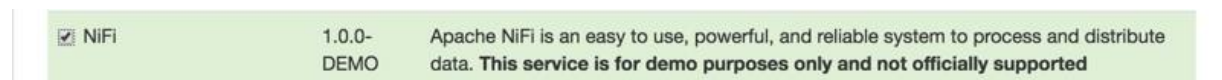
Note: You should receive a success message.

2. Login into Ambari as admin. In the left sidebar of services, click on the **Actions** button. A drop down menu appears, select the **Add Service** button . The **Add Service Wizard** window will appear.



<input type="checkbox"/> Service	Version	Description
<input checked="" type="checkbox"/> HDFS	2.7.1.2.5	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	2.7.1.2.5	Apache Hadoop NextGen MapReduce (YARN)
<input checked="" type="checkbox"/> Tez	0.7.0.2.5	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	1.2.1.2.5	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service

Choose the NiFi service:



<input checked="" type="checkbox"/> NIFI	1.0.0-DEMO	Apache NiFi is an easy to use, powerful, and reliable system to process and distribute data. <b>This service is for demo purposes only and not officially supported</b>
--	------------	---

3. Once NiFi box is checked, select the **Next** button. As the Ambari Wizard transitions to **Assign Masters**, you will see an **Error** window message ignore it. Click on the **OK** button. You will see an indicator that the **Assign Masters** page is loading, keep its default settings and click **Next**. As the wizard transitions to **Assign Slaves**

- and Clients, a **Validation Issues** window will appear, select **Continue Anyway**. The wizard will continue onto the next setup settings called **Customize Services**. Keep its default settings, click **Next**. A **Consistency Check Failed** window will appear, click **Proceed Anyway** and you will proceed to the **Review** section.
4. For the **Review** section, you will see a list of repositories, select **Deploy->**.

**Add Service Wizard**

ADD SERVICE WIZARD

- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Configure Identities
- Review**
- Install, Start and Test
- Summary

## Review

Please review the configuration before installation

**Admin Name :** admin  
**Cluster Name :** Sandbox  
**Total Hosts :** 1 (0 new)

**Repositories:**

- debian7 (HDP-2.5):  
http://s3.amazonaws.com/dev.hortonworks.com/HDP/debian7/2.x/BUILDS/2.5.1.0-19
- debian7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/debian6
- redhat6 (HDP-2.5):  
http://s3.amazonaws.com/dev.hortonworks.com/HDP/centos6/2.x/BUILDS/2.5.0.0-1245/
- redhat6 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos6
- redhat7 (HDP-2.5):  
http://s3.amazonaws.com/dev.hortonworks.com/HDP/centos7/2.x/BUILDS/2.5.1.0-19
- redhat7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos7
- suse11 (HDP-2.5):  
http://s3.amazonaws.com/dev.hortonworks.com/HDP/suse11sp3/2.x/BUILDS/2.5.1.0-19

← Back Print **Deploy →**

Note: you should see a preparing to deploy message.

5. The wizard will transition to the **Install, Start and Test** section. Click **Next**.

## Add Service Wizard

The screenshot shows the 'Install, Start and Test' step of the Ambari Add Service Wizard. On the left sidebar, 'Install, Start and Test' is selected. The main content area has a title 'Install, Start and Test' and a message: 'Please wait while the selected services are installed and started.' Below this is a green progress bar showing '100 % overall'. A table displays the installation status for one host:

Host	Status	Message
sandbox.hortonworks.com	100%	Success

Below the table, it says '1 of 1 hosts showing - Show All' and 'Show: 25' with navigation arrows. A green message box at the bottom states 'Successfully installed and started the services.' and a 'Next ->' button is visible.

- At the Summary section, you see an **Important** alert, which states that we should restart services that contain **restart indicators** in the left sidebar of Ambari Services on the Ambari Dashboard. Click **Continue->**.

## Add Service Wizard

The screenshot shows the 'Summary' step of the Ambari Add Service Wizard. On the left sidebar, 'Summary' is selected. The main content area has a title 'Summary' and an **Important** alert: 'You may also need to restart other services for the newly added services to function properly (for example, HDFS and YARN/MapReduce need to be restarted after adding Oozie). After closing this wizard, please restart all services that have the restart indicator 🔄 next to the service name.' Below the alert, a message says 'Here is the summary of the install process.' and a box states 'The cluster consists of 1 hosts' and 'Installed and started services successfully on 1 new host'. A 'Complete ->' button is at the bottom right.

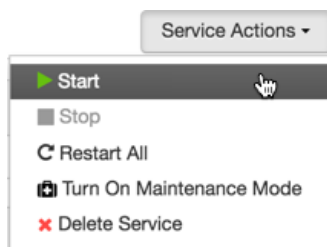
- Upon a successful installation, you should see a new service listed with a **green check symbol** next to the service name. This check symbol also indicates that NiFi is running.



## STEP 2: Start Nifi via Ambari Services

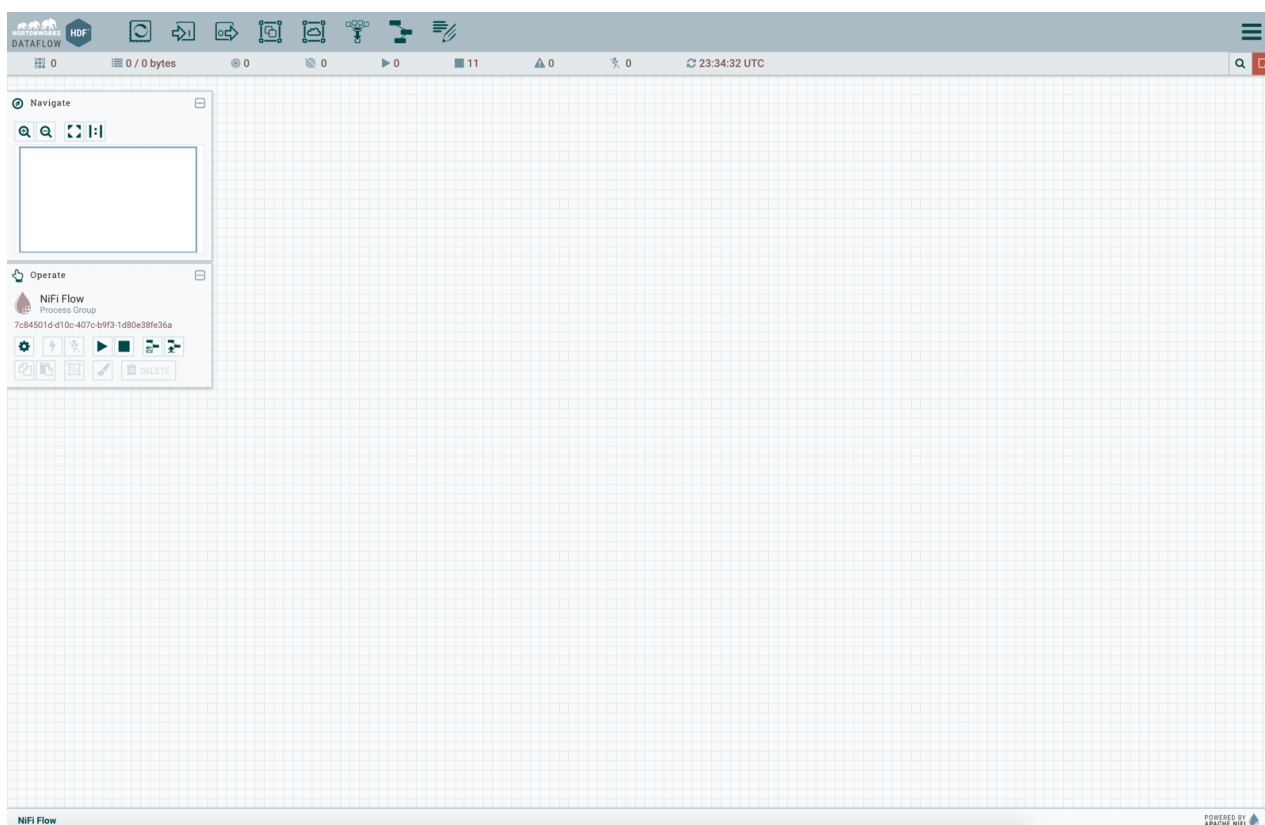
If NiFi is not already running, we will use Ambari Service Tool to launch NiFi.

1. Click on NiFi located in the left sidebar of Ambari Services on the Dashboard.
2. Select the **Action Services** button, click **Start** to activate NiFi.



Note: Once the service successfully started you should see a **green check symbol** next to the name of the service.

3. Open NiFi at `http://sandbox.hortonworks.com:9090/nifi/`. Wait 1 to 2 minutes for NiFi to load.



## Section #2: Setup Kafka Service

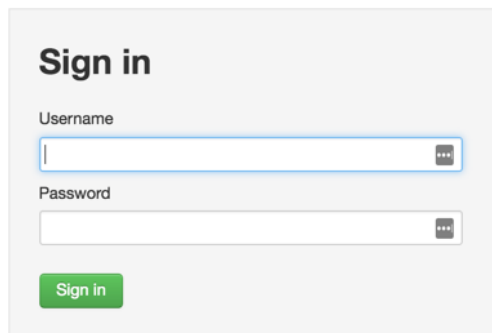
We need to setup Kafka because it will be used as secure cluster or the location where NiFi transports the data. Storm will pull that data from the cluster and push it into its topology(dataflow).

### STEP 1: Start Kafka

#### 1.1 ACCESS AMBARI

Login to Ambari to activate Kafka. Enter the URL in your browser

```
http://sandbox.hortonworks.com:8080
```

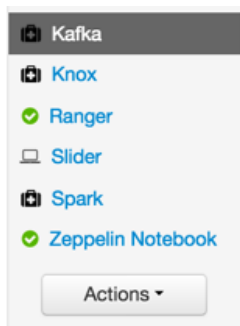


The image shows the Ambari 'Sign in' form. It has a title 'Sign in' in bold. Below the title are two input fields: 'Username' and 'Password'. Each field has a small icon on the right side. Below the password field is a green 'Sign in' button.

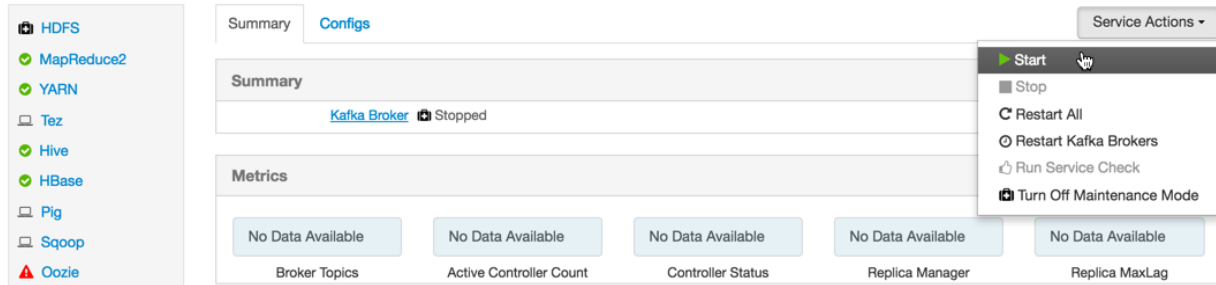
Note: username for admin is admin. Password is the password you defined.

#### 1.2 USE AMBARI TO ACTIVATE KAFKA

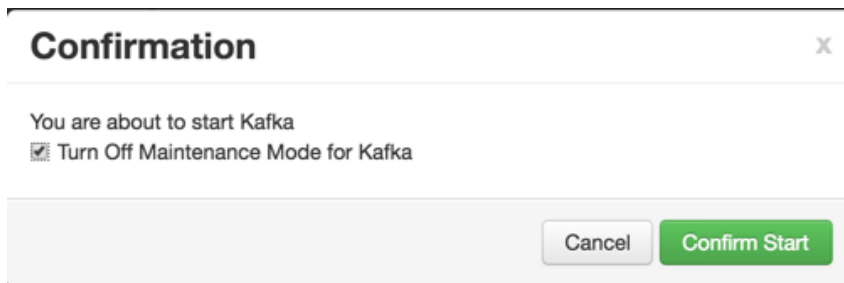
1. Click on Kafka located in the left sidebar list of installed services



2. Click on **Service Actions -> Start** located at the top left of the Kafka Services Page:



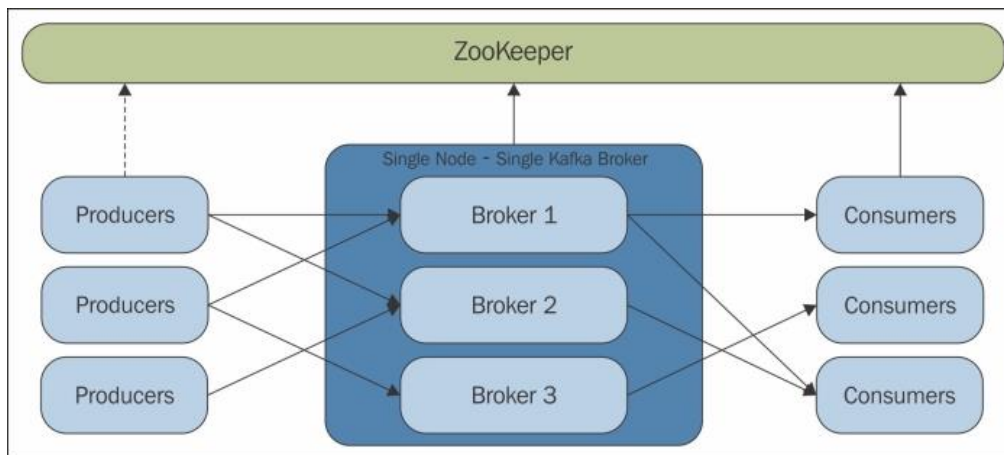
3. Check the **Turn off Maintenance Mode for Kafka** box and click on **Confirm Start**:



Wait for Kafka to start.

## STEP 2: Configure Kafka with ZooKeeper

ZooKeeper is the coordination interface between the Kafka broker and consumers:



The important Zookeeper properties can be checked in Ambari.

### 2.1 CONFIGURE ZOOKEEPER

Click on **ZooKeeper** in the list of services, then open the Configs tab. Verify ZooKeeper runs on port 2181:

The screenshot shows the Ambari interface for configuring the ZooKeeper service. The left sidebar lists various services, with ZooKeeper selected. The main panel shows the configuration for the ZooKeeper Server, including fields for host, directory, tick length, and port. The port field is highlighted with a green box.

If this port 2181 is busy or consumed by other processes, you can change the default port number of ZooKeeper to any other valid port number. If ZooKeeper is not running, you can start the Zookeeper service from Ambari:

The screenshot shows the Ambari interface for the ZooKeeper service. The 'Summary' tab is active, showing the service status as 'Started'. The 'Service Actions' dropdown menu is open, and the 'Start' button is highlighted with a green box.

## 2.2 CONFIGURE KAFKA

From the Kafka page, click on the **Configs** tab. Verify the `zookeeper.connect` property points to your ZooKeeper server name and port:



The screenshot shows the Ambari Services page for Kafka Broker configuration. The top navigation bar includes 'Ambari Sandbox', '0 ops', '0 alerts', 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user profile 'admin'. The left sidebar lists services: HDFS, MapReduce2, YARN, Tez, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Falcon, Storm, Flume, Ambari Metrics, Atlas, Kafka (selected), Knox, Ranger, Slider, and Spark. The main content area shows the 'Summary' and 'Configs' tabs. Below the tabs, there's a 'Group' dropdown set to 'Default (1)' and a 'Manage Config Groups' button. A notification bar shows 'V2' and 'V1' versions of the configuration, with a message 'admin authored on Tue, May 03, 2016 02:51'. The configuration table for 'Kafka Broker' is as follows:

Property	Value	Actions
Kafka Broker host	sandbox.hortonworks.com	
zookeeper.connect	sandbox.hortonworks.com:2181	🔒 🟢 🔄
log.dirs	/kafka-logs	🔒 🟢 🔄
log.retention.hours	168	🔒 🟢 🔄
log.roll.hours	168	🔒 🟢 🔄
listeners	PLAINTEXT://localhost:6667	🔒 🟢 🔄

## Section #3: Setup Storm & HBase Services

We need to setup the Storm & HBase services because storm will deploy a topology that is configured to send data to HBase and HBase is used to create the tables that storm populates.

### STEP 1: Start Storm & HBase

1. View the HBase Services page

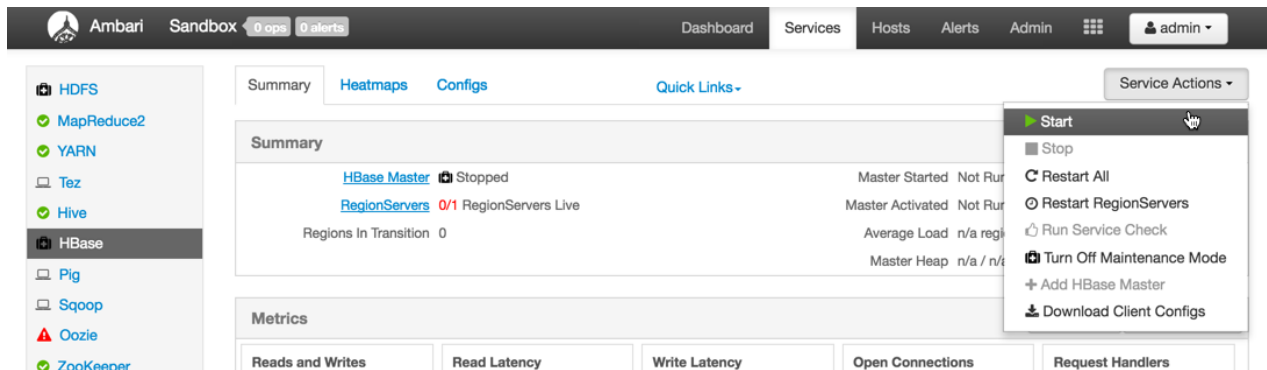
From the previous exercises: HDFS, Hive, YARN and Kafka should already be running but HBase may be down. From the Dashboard page of Ambari, click on HBase from the list of installed services.

The screenshot shows the Ambari Services page with a list of installed services. The services listed are HDFS, MapReduce2, YARN, Tez, Hive, and HBase. HBase is highlighted with a dark background, indicating it is the selected service.

#### 1.1 SETUP HBASE

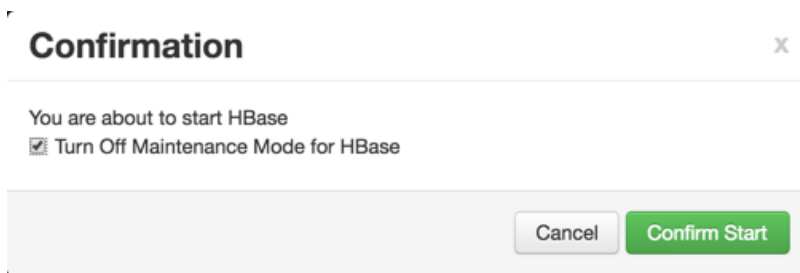
## 2. Start HBase

From the HBase page, click on **Service Actions -> Start**



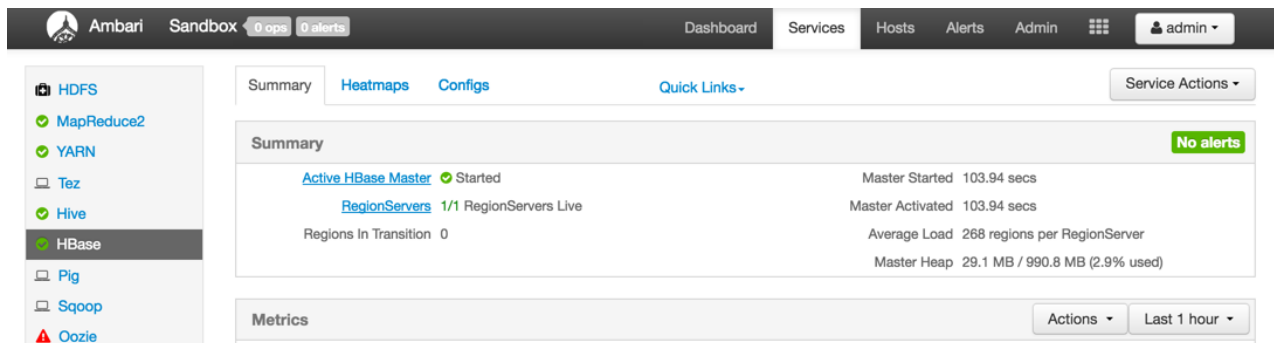
The screenshot shows the Ambari interface for the HBase service. The 'Service Actions' dropdown menu is open, showing options: Start, Stop, Restart All, Restart RegionServers, Run Service Check, Turn Off Maintenance Mode, Add HBase Master, and Download Client Configs. The 'Start' option is highlighted. The main content area shows the HBase service status as 'Stopped' with a 'Stop' icon. The 'RegionServers' are listed as '0/1 RegionServers Live'. The 'Metrics' section shows 'Reads and Writes', 'Read Latency', 'Write Latency', 'Open Connections', and 'Request Handlers'.

Check the box and click on **Confirm Start**:



The screenshot shows a 'Confirmation' dialog box. The title is 'Confirmation'. The text inside says 'You are about to start HBase'. There is a checked checkbox labeled 'Turn Off Maintenance Mode for HBase'. At the bottom, there are two buttons: 'Cancel' and 'Confirm Start'.

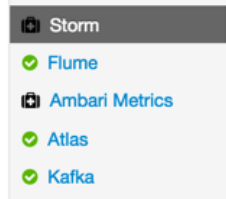
Wait for HBase to start (It may take a few minutes to turn green)



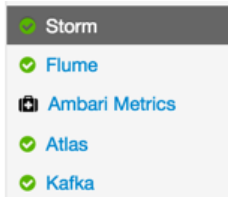
The screenshot shows the Ambari interface for the HBase service after it has started. The 'Service Actions' dropdown menu is now closed. The main content area shows the HBase service status as 'Started' with a green checkmark icon. The 'RegionServers' are listed as '1/1 RegionServers Live'. The 'Metrics' section shows 'Reads and Writes', 'Read Latency', 'Write Latency', 'Open Connections', and 'Request Handlers'. A 'No alerts' badge is visible in the top right corner of the summary section.

### 1.2 SETUP STORM

3. Start Storm the same way we started HBase in the previous steps. We will need it later for streaming real-time event data.



4. After starting storm, a green check symbol will be present:



Now that we have storm activated, we need to download a storm demo project for later when we use Storm's Visualization feature.

5. Let's SSH into the sandbox by shell:

```
ssh root@sandbox.hortonworks.com -p 2222
```

6. We will download the demo project because it contains the necessary code and configuration properties required for storm to deploy the topology.

```
cd ~  
git clone -b hdp25experiment https://github.com/james94/iot-truck-streaming
```

7. Navigate to the iot-truck-streaming folder, which is the location where the appropriate storm configuration files reside for configuring our topology.

```
cd ~/iot-truck-streaming
```

8. Since we are at the base of our project, let's export our storm demo configurations. We will create a new folder **storm\_demo** as the new location where storm will look for instructions on configuring the storm topology:

```
sudo mkdir /etc/storm_demo  
sudo cp config.properties /etc/storm_demo/  
sudo cp -R storm-demo-webapp/routes/ /etc/storm_demo/
```

Note: Storm will refer to the properties in the config.properties file to configure the topology.

You can use the Ambari dashboard to check status of other components too. If **HDFS**, **Hive**, **YARN** are down, you can start them in the same way: by selecting the service and then using the Service Actions to start it. The remaining components do not have to be up. (Oozie can be stopped to save memory, as it is not needed for this exercise)

## Section #4: Run the simulator by terminal

The simulator must be setup in order for NiFi to pull data from it and push that data into the dataflow.

### Stream Simulator

The stream simulator is a lightweight framework that generates truck event data. The simulator uses [New York City Truck Routes \(kml\)](#) which defines driver road paths with Latitude and Longitude information.

The simulator uses [Akka](#) to simplify concurrency, messaging and inheritance. It has two [Plain Old Java Objects \(POJOS\)](#), one for Trucks and another for Drivers that generate the events. Consequently, the `AbstractEventEmitter` class becomes extended while the `onReceive` method generates events, creates new [Actors](#), sends messages to Actors and delivers those events to an `EventCollector` class. This class's purpose is to collect events generated from the domain objects and print them to standard output. Let's run the simulator through the terminal and see the events that are generated.

### STEP 1: Run the simulator by shell

Before we run the simulator, let's install and download the simulator.

#### 1.1 SETUP THE SIMULATOR

1. If you have not already, ssh into sandbox shell, type the command to access the sandbox by shell:

```
ssh root@sandbox.hortonworks.com -p 2222
```

2. Install Apache Maven. We will use it to compile the simulator code, so we can activate the simulator by shell or NiFi later. Execute the command:

```
cd ~/iot-truck-streaming
./setup/bin/install_maven.sh
```

Note: You will be prompted to allow maven to install, type 'y' for yes

After your maven package installs, you should obtain the message: Complete!

3. For maven to run, it needs to detect the pom.xml file. Rename pom25.xml to pom.xml, copy/paste the commands:

```
mv -f storm-streaming/pom25.xml storm-streaming/pom.xml
/usr/maven/bin/mvn clean package -DskipTests
```

Note: You should receive that all sub projects compiled successfully.

## 1.2 RUN THE SIMULATOR

1. To test the simulator, run `generate.sh` script.

```
cd stream-simulator
chmod 750 *.sh
./generate.sh
```

Note: press **ctrl+c** stop the simulator

You should see message data generated. The data in the image includes logs as can be seen in the top portion and truck events bottom portion. We will use NiFi to separate this data.

```
jmedel — root@sandbox:~/iot-truck-streaming/stream-simulator — ssh root@127.0.0.1 -p 22...
[Jun 01 02:25:56] INFO (RouteProvided.java) - Going Original Direction...
[Jun 01 02:25:56] INFO (Truck.java) - Route has ended for Driver[32] on Truck[88]
[Jun 01 02:25:56] INFO (Truck.java) - The Driver[Ryan Templeton] has new Truck[86] with[Des Moines to Chicago Route 2] traversed 3 times.
[Jun 01 02:25:56] INFO (RouteProvided.java) - Reverting Direction..
[Jun 01 02:25:56] INFO (Truck.java) - Route has ended for Driver[32] on Truck[86]
[Jun 01 02:25:56] INFO (Truck.java) - The Driver[Ryan Templeton] has new Truck[88] with[Des Moines to Chicago Route 2] traversed 4 times.
[Jun 01 02:25:56] INFO (Truck.java) - Truck[88] with Driver[Ryan Templeton] has stopped its route
2016-06-01 02:25:56.701|40|11|Jamie Engesser|1384345811|Joplin to Kansas City|Normal|38.65|-90.2|1000
2016-06-01 02:25:56.74|105|13|Joe Niemiec|24929475|Peoria to Ceder Rapids|Normal|41.62|-93.58|1000
2016-06-01 02:25:56.74|98|21|Jeff Markham|1961634315|Saint Louis to Memphis|Normal|37.03|-94.58|1000
2016-06-01 02:25:56.777|66|10|George Vetticaden|1390372503|Saint Louis to Tulsa|Normal|38.64|-90.18|1000
2016-06-01 02:25:56.777|25|23|Adam Diaz|160779139|Des Moines to Chicago Route 2|Normal|40.7|-89.52|1000
2016-06-01 02:25:56.797|89|17|Eric Mizell|1927624662|Springfield to KC Via Columbia|Normal|38.65|-90.21|1000
2016-06-01 02:25:56.797|87|28|Olivier Renault|137128276|Springfield to KC Via Hannibal Route 2|Normal|39.78|-89.66|1000
2016-06-01 02:25:56.829|16|24|Don Hilborn|1090292248|Peoria to Ceder Rapids Rout
```

Note: generate.sh runs java source code located at `iot-truck-`

`streaming/stream-`

`simulator/src/main/java/com/hortonworks/streaming/impl/collectors/StdOutEventCol`

`lector.java` . If you would like to see modify/run the code.

## Section #5: Setup IntelliJ IDE locally and run topologies on the sandbox

We will use IntelliJ as our editor to write and change storm code on our local computer and then send our storm project to the sandbox.

### STEP 1: Install IntelliJ locally

If you have not installed **IntelliJ**, refer to JetBrains [Installing and Launching](#) instructions.

### STEP 2: Download trucking demo for development with IDE

Earlier we cloned the `iot-truck-streaming` project from github onto our sandbox. Now we will clone it onto our local machine.

1. Perform the `git clone` command on the local machine. Feel free to clone it in any directory, just remember the location. In the exercise, let's clone it in our Documents folder.

```
cd ~/Documents
git clone -b hdp25experiment https://github.com/james94/iot-truck-streaming.git
```

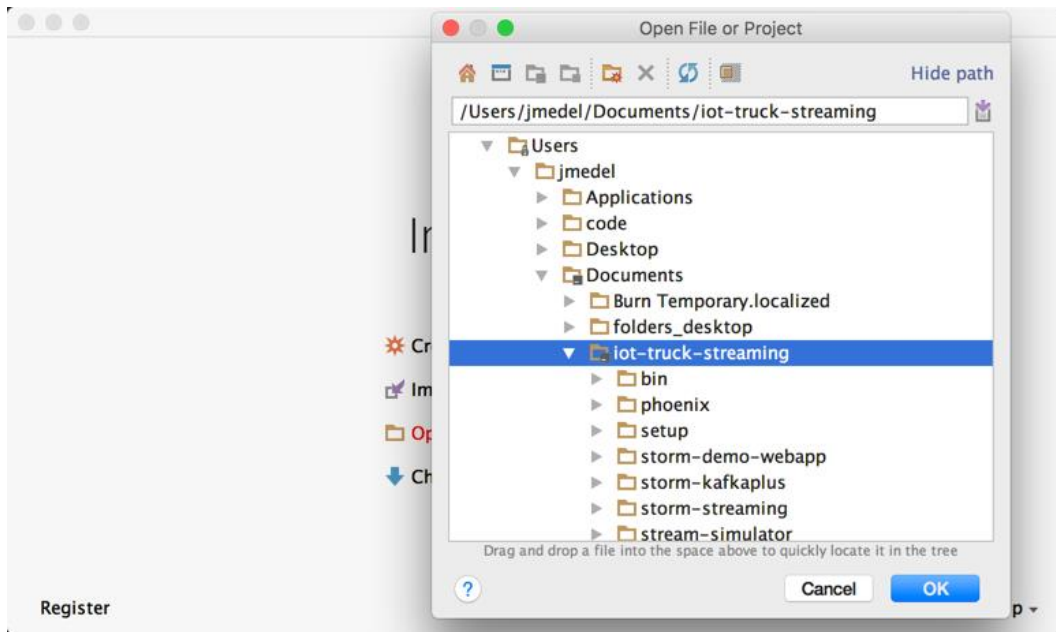
Note: You may receive an error if you don't have git installed on your local machine.

### STEP 3: Open trucking demo in IntelliJ

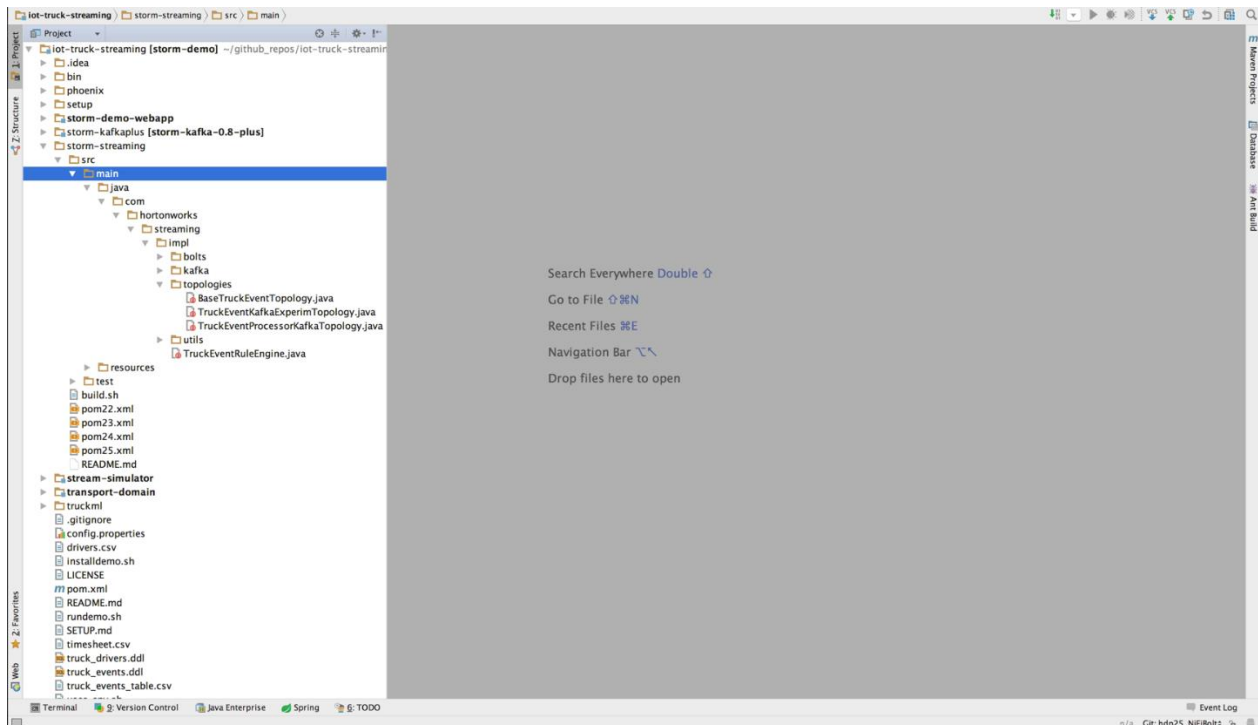
1. Open IntelliJ. On the welcome screen, click on the Open button. We will open our **iot-truck-streaming** project.



2. Select **iot-truck-streaming** project.



3. IntelliJ will display the project in its IDE.



#### STEP 4: Configure IntelliJ to recognize Maven project

You'll notice, the image icons next to the code files have small red circles on them. IntelliJ does not recognize the Trucking Demo is a Maven Project. The solution we will use is to run `mvn cleanpackage` from the command line and cause IntelliJ to warn us



that Maven Projects need to be imported to be recognized by the IDE. When the **Enable Auto-Import** box appears in IntelliJ, we will activate for Maven projects.

1. Let's begin the process to run maven against our Trucking Demo Project.

#### 4.1 SPECIFY POM.XML FOR MAVEN

- For maven to work, we have to tell it which pom file to use from the storm-streaming directory. Let's rename pom25 to pom using shell.

```
mv iot-truck-streaming/storm-streaming/pom25.xml iot-truck-streaming/storm-streaming/pom.xml
```

#### 4.2 COMPILE AND PACKAGE DEMO FILES INTO JAR

- Now that we specified the pom for maven, it knows how to build the project. Let's execute the command:

```
cd iot-truck-streaming
mvn clean package -DskipTests
```

Output should show success for each sub project within the overall project.

```
/target/storm-kafka-0.8-plus-0.4.jar mvn clean package -DskipTests
[INFO]
[INFO] -----
[INFO] Building Storm Demo Parent Project 1.0
[INFO] -----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ storm-demo ---
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] transport-domain ..... SUCCESS [ 1.279 s]
[INFO] stream-simulator ..... SUCCESS [ 2.346 s]
[INFO] storm-streaming ..... SUCCESS [01:31 min]
[INFO] storm-demo-webapp ..... SUCCESS [ 4.735 s]
[INFO] storm-kafka-0.8-plus ..... SUCCESS [ 2.857 s]
[INFO] Storm Demo Parent Project ..... SUCCESS [ 0.010 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:42 min
[INFO] Finished at: 2016-09-14T12:20:14-07:00
[INFO] Final Memory: 142M/1034M
[INFO] -----
HW12388:iot-truck-streaming jmedel$
```

#### APACHE MAVEN COMMAND:

mvn clean deletes everything in the target folder. For example, if this was the second time you ran the mvn command from `iot-truck-streaming` folder, the `storm-streaming` folder as well as other folders that contain `pom.xml` files will have their target folders impacted. Clean part of the command removes the old target folder, while the package part of the command compiles the code and packages it into jar files per the pom file.

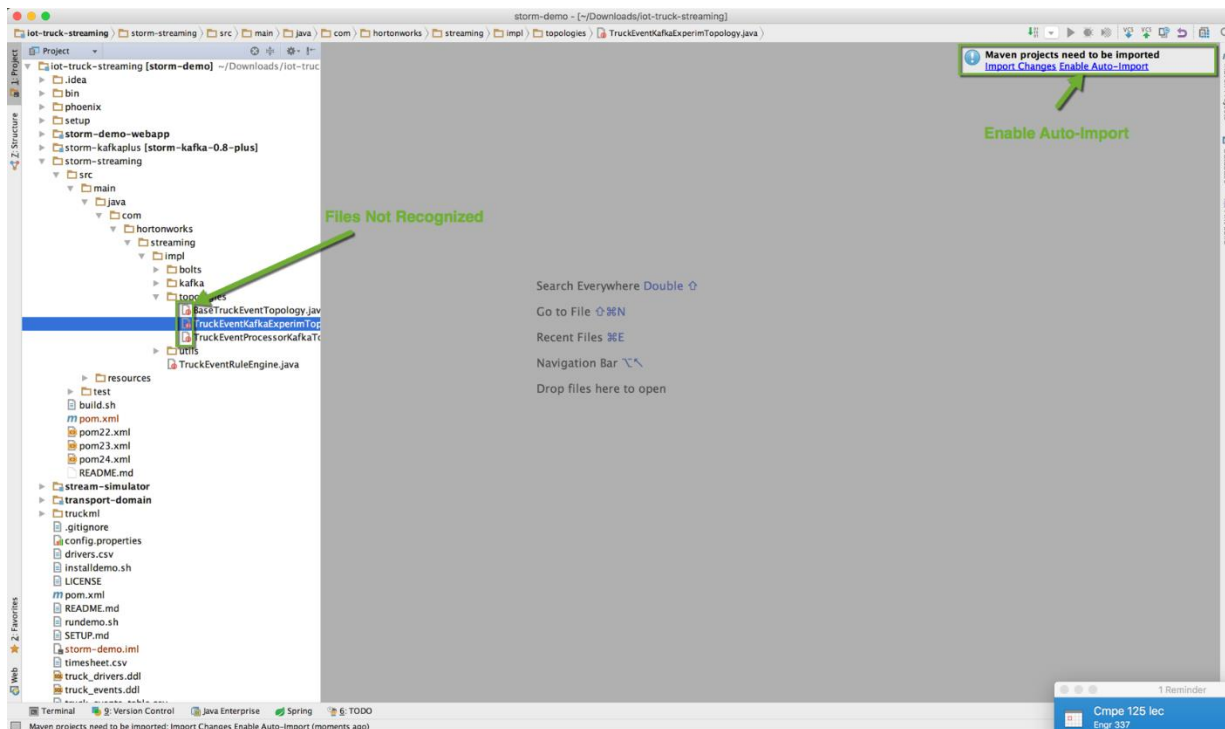
Note: packaging may take around 9 minutes. Add `-DskipTests` to the end of mvn command to speed up process.

### Why is Maven important for development testing?

It enables developers to test their large software projects that contain many java files at fast speeds.

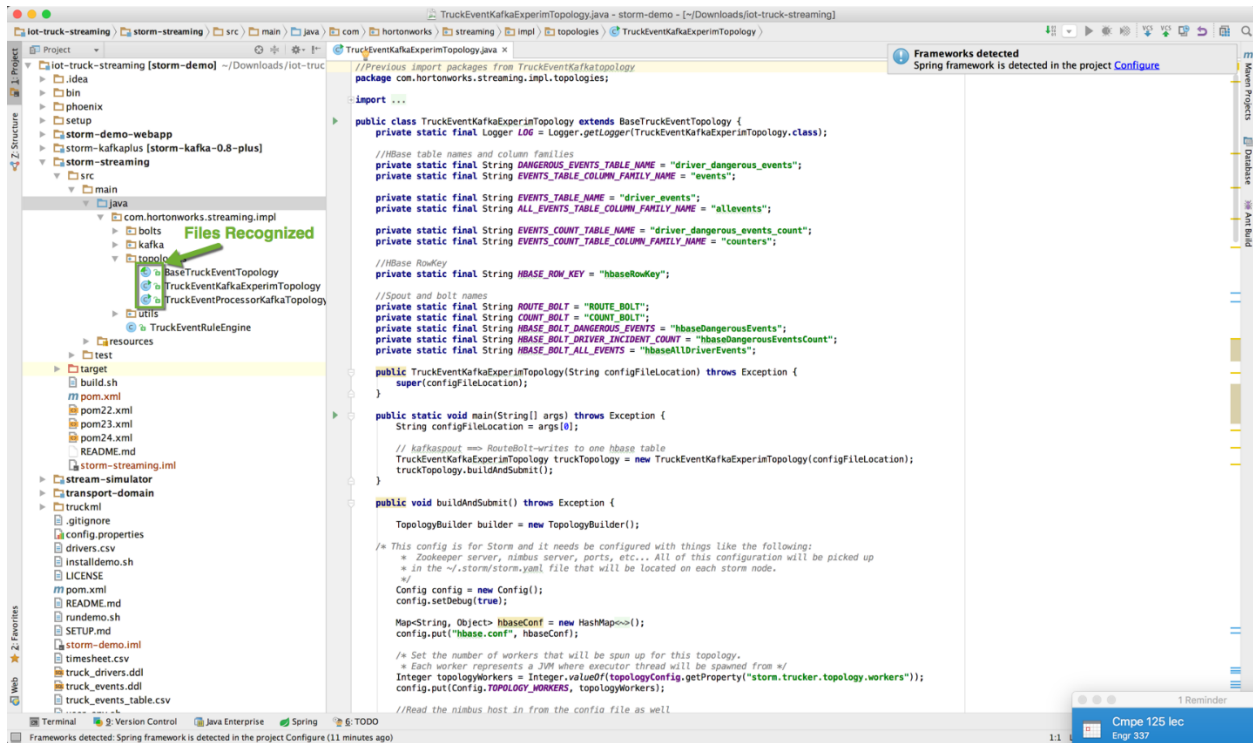
#### 4.3 INSTRUCT INTELLIJ TO ENABLE AUTO-IMPORT FOR MAVEN PROJECTS

1. Switch to the IntelliJ IDE to see if it recognizes the Maven Project.
2. As you run maven, you will see in IntelliJ that a box in the top right corner appears and states **Maven projects need to be imported**. Click on **Enable Auto-Import**.



#### 4.4 INTELLIJ RECOGNIZES TRUCKING DEMO IS MAVEN PROJECT

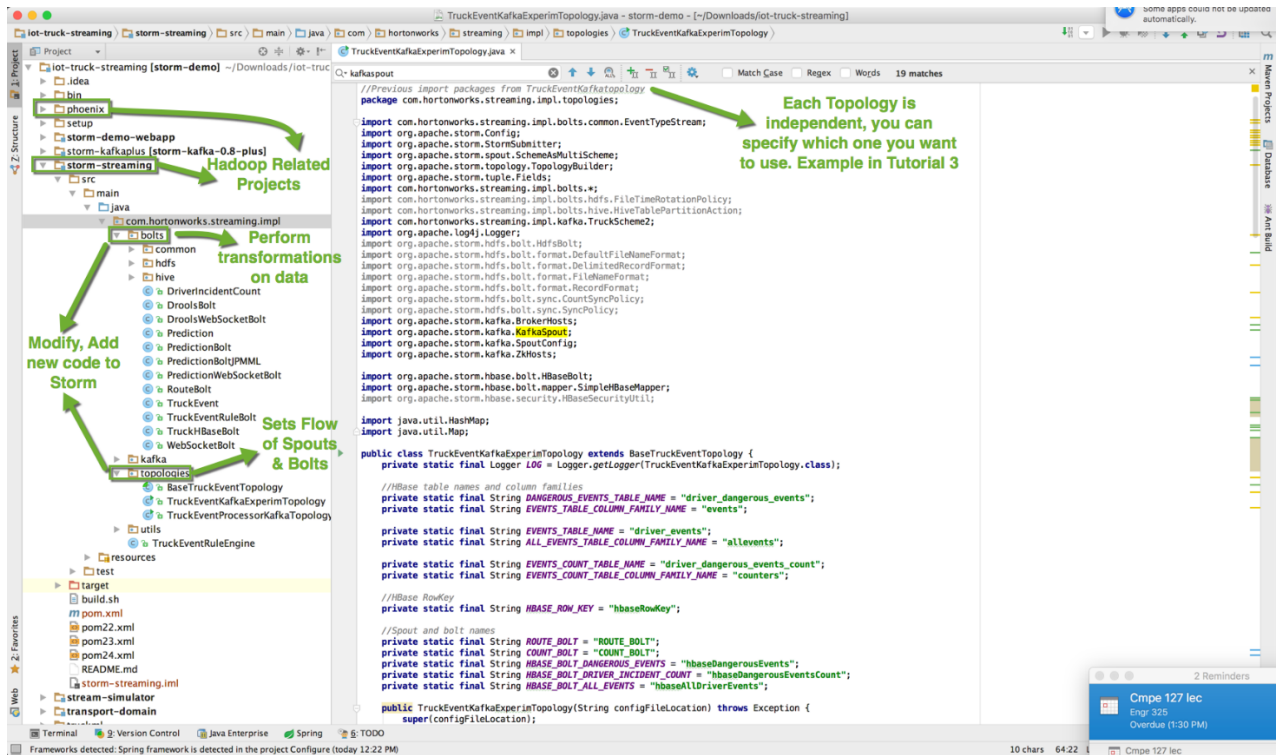
- Notice that the icons next to the code files changed to blue circles.



Note: If you want to make commits to github from IntelliJ, feel free to try out IntelliJ's git feature, else ignore the "Add Files to Git" Window if it appears.

#### 4.5 INTELLIJ SETUP TO DEVELOP HADOOP PROJECTS LOCALLY

1. Now we can develop our storm code from the storm-streaming folder directly on our local machine. Feel free to explore the code, modify or add bolts, spouts, or topology.



2. Once you've added or modified the code, we can run mvn command used earlier to package our storm project into a jar.

```
cd ~/Documents/iot-truck-streaming
mvn clean package -DskipTests
```

Note: If you want to add an enhancement to the demo, all you need to do is re-execute the above steps and the new modifications will be added to the jar.

3. As you will see, mvn generates a target folder in every sub project folder; for instance, let's view the project we are working on, storm-streaming, it's target folder contains:

```
ls -ltr storm-streaming/target
```

```
HW12388:iot-truck-streaming jmedel$ ls -ltr storm-streaming/target/
total 324216
drwxr-xr-x 3 jmedel staff 102 Sep 14 12:19 maven-status
drwxr-xr-x 8 jmedel staff 272 Sep 14 12:19 classes
drwxr-xr-x 3 jmedel staff 102 Sep 14 12:19 test-classes
-rw-r--r-- 1 jmedel staff 75791 Sep 14 12:19 original-storm-streaming-1.0-SNAPSHOT.jar
drwxr-xr-x 3 jmedel staff 102 Sep 14 12:19 maven-archiver
-rw-r--r-- 1 jmedel staff 165908231 Sep 14 12:20 storm-streaming-1.0-SNAPSHOT.jar
-rw-r--r-- 1 jmedel staff 10580 Sep 14 12:20 dependency-reduced-pom.xml
HW12388:iot-truck-streaming jmedel$
```

Notice the target folder contains **storm-streaming-1.0-SNAPSHOT.jar** file. This jar is a collection of java classes for our storm project. When you add an enhancement to the demo and maven is executed, the jar file will be removed and replaced with a new version.

4. Let's send our jar to the sandbox for later use in Day 3.

```
scp -P 2222 ~/Documents/iot-truck-streaming/storm-streaming/target/storm-streaming-1.0-SNAPSHOT.jar root@sandbox.hortonworks.com:/root/iot-truck-streaming/storm-streaming/target
```

Note: Each time we update the demo, we have to transport the latest jar file to the sandbox.

## SUMMARY

Since we can build jars on our local machine and have an instance of the **iot-truck-streaming** project on our sandbox with the appropriate configurations, we can make changes to our code locally, and then send the jar to any directory on our sandbox. In our case, we will send it to `storm-streaming/target` folder. This approach makes it easy to develop storm topologies locally, and test it on an HDP environment. Therefore, we can test if our code performs as expected by viewing the topology visually through the Storm View's Visualization. For example, did we connect our spouts and bolts properly. We can also use HBase to view if Storm sending the proper data to the tables. We will **perform these tests in Day 3**.