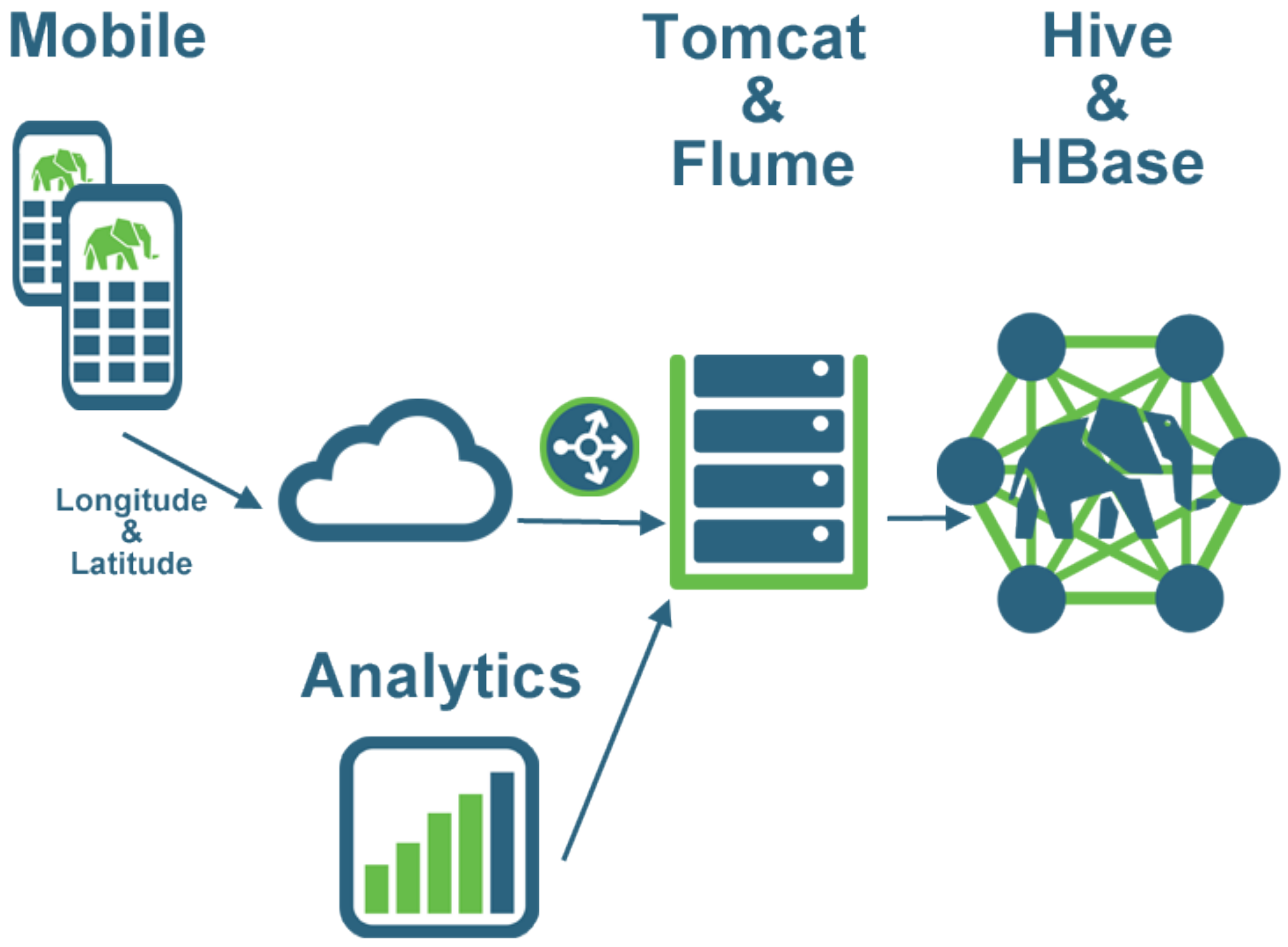
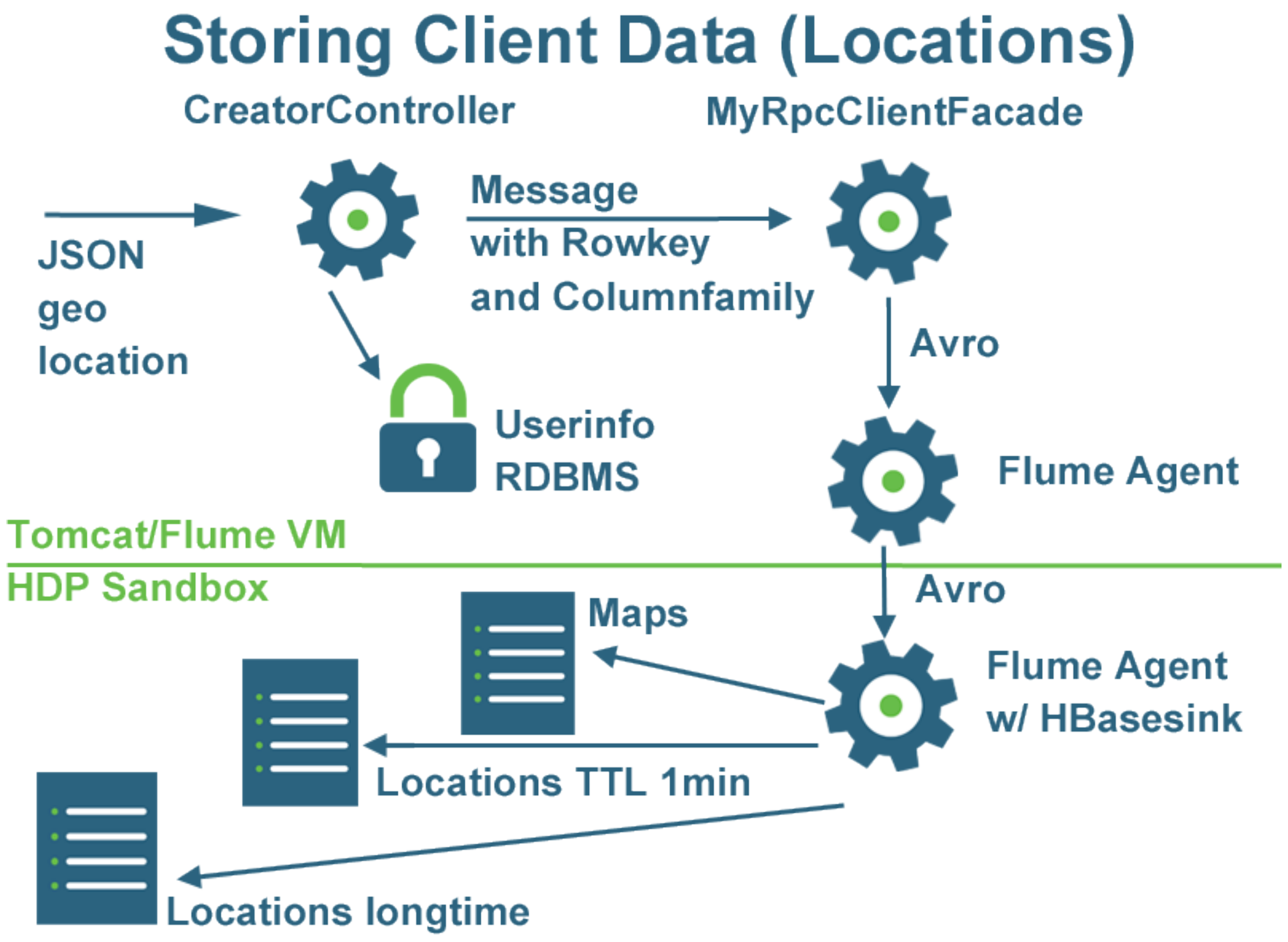
iiCaptain and the Data Ocean

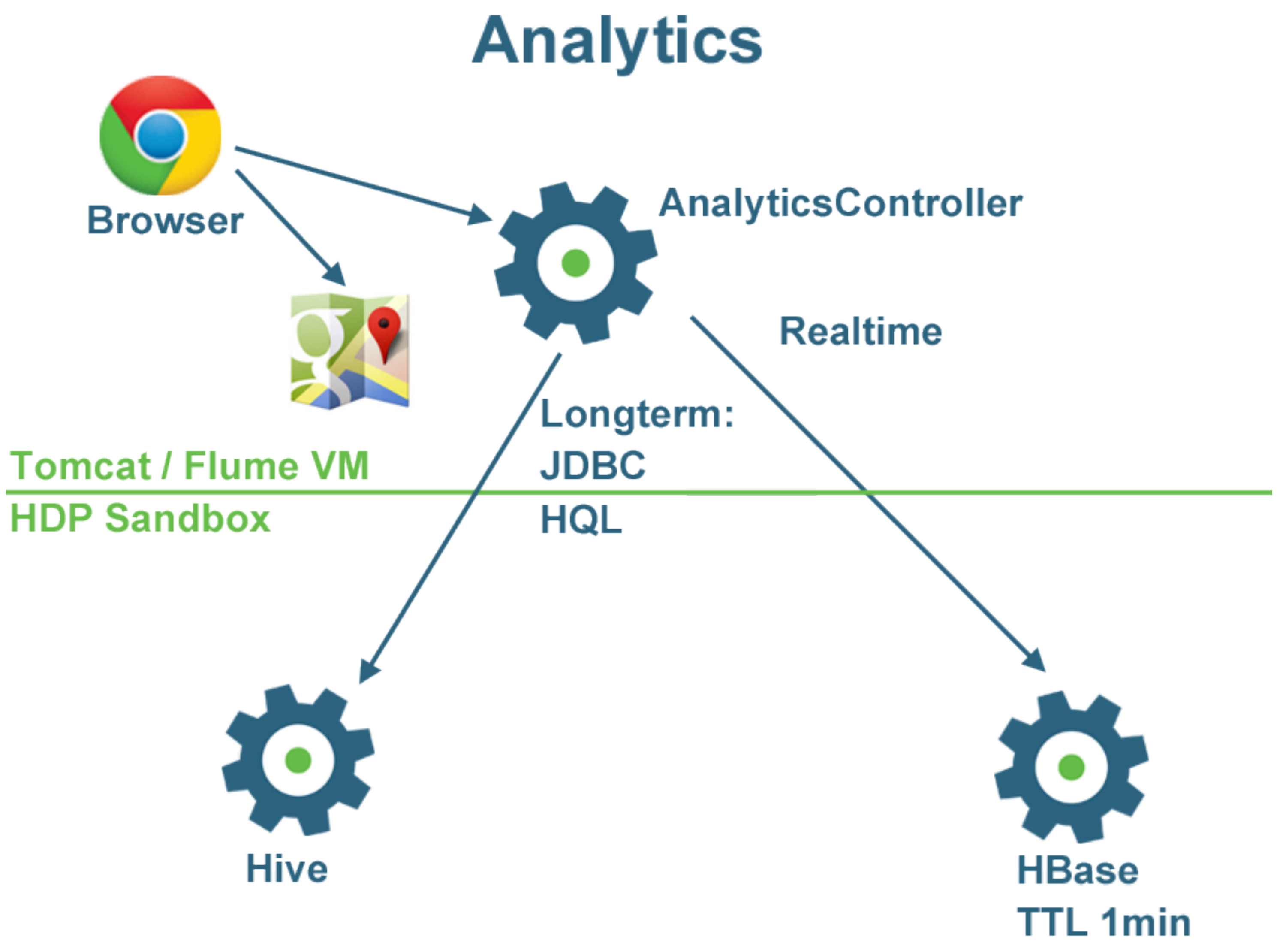
Before we start with the installation instructions please also have a look at the slide deck. To make things easier here are some illustrations showing what we want to achieve:
We want to create a platform dealing with all kinds of data from a mobile application. Furthermore we want to be able to answer questions like: Where are you users now? And where was the hotspot in the last 3 months? So we clearly have batch and non-batch requirements.



Most important components (Servlets, etc.):



We will visualize geo locations with Google Maps. So keep in mind the demo needs an internet connection.



Shopping list:
1. Hortonworks HDP Sandbox 2.0

2. A VM with Linux. 1GB Memory, 1 Core.

3. Tomcat

4. Flume binary

5. The App
   HTML5/Javascript App which will be baked into a native iOS or Android App through build.phonegap.com. We use Javascript to detect the geo location:

```
   navigator.geolocation.getCurrentPosition(showPosition);
```

6. Custom Flume HBase Sink

7. Loadtest tool and AppAnnie Helper

8. Apache Derby

Clone all three git repositories.
For building with Phonegap you need my credentials:
digitalemil@googlemail.com:Mm7496812abc Please add them to the build.xml (target phonegap): `<arg value="yourUsername:yourPassword" />`

Start the sandbox with 4GB main memory and the ip address to /etc/hosts on your Mac.

```
sudo vi /etc/hosts:
the.ip.ofyour.sandbox     sandbox sandbox.hortonworks.com
the.ip.ofyour.tomcatvm    tomcatvm
```

Make sure all 3 machines (Your local MacBook, Sandbox and the Tomcat VM have these lines in /etc/hosts)
Then:

```
ssh root@sandbox (Passwort: hadoop)
./start_hbase.sh
```

Give Hive access to HBase classes http://hortonworks.com/blog/hbase-via-hive-part-1/).

```
mkdir /usr/lib/hive/auxlib; cp /usr/lib/hive/lib/hive-hbase-handler-*.jar /usr/lib/hive/auxlib ;
cp /usr/lib/hbase/lib/hbase*jar /usr/lib/hive/auxlib/; cp /usr/lib/hbase/lib/htrace-core-*.jar
/usr/lib/hive/auxlib/; cp /etc/hbase/conf/hbase-site.xml /usr/lib/hive/auxlib/; cp
/usr/lib/hive/auxlib/* /usr/lib/hadoop/lib/
```

vi /etc/hive/conf/hive-site.xml and add:

```
<property>
    <name>hive.aux.jars.path</name>
    <value>file:///usr/lib/hive/auxlib</value>
</property>
```

For root (HiveServer2 run as root) we need to adjust the HADOOP_CLASSPATH

```
echo export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/usr/lib/hive/auxlib >>.bash_profile
su - hive
echo export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/usr/lib/hive/auxlib >>.bash_profile
exit
```

Then:

```
reboot
```

login again and:

```
su - hbase
```

```
hbase shell
create 'iicaptain-1min' , {NAME => 'l', VERSIONS => 1, TTL => 60}
create 'iicaptain' , {NAME => 'w', VERSIONS => 1}, {NAME => 'l', VERSIONS => 1}
exit
exit
```

Let's create the tables

```
su - hive
export HADOOP_CLASSPATH=${HADOOP_CLASSPATH}:/usr/lib/hive/auxlib/
hive
Create External Table iiCaptain (key string, user string, longitude string, latitude string,
timestamp decimal) Stored By 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH
SERDEPROPERTIES ("hbase.columns.mapping" = ":key,l:user,l:longitude,l:latitude,l:timestamp")
TBLPROPERTIES ("hbase.table.name" = "iicaptain");
Create External Table iiCaptain1Min (key varchar(128), user varchar(20), longitude varchar(20),
latitude varchar(20), timestamp decimal) Stored By
'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping"
= ":key,l:user,l:longitude,l:latitude,l:timestamp") TBLPROPERTIES ("hbase.table.name" =
"iicaptain-1min");
Create Table HiiCaptain (user string, longitude string, latitude string, timestamp decimal);
quit;
exit;
```

Install Flume:

```
yum clean all
yum install flume
yum install ntp
ntpdate -s time.nist.gov
```

Copy flume configs and start scripts from your Mac to Sandbox:

```
cd iiCaptainFlume
mvn clean package
scp flume/flumeToHBase.conf root@sandbox:/root
scp flume/startHBaseSink.sh root@sandbox:/root
scp target/iiCaptainFlume-0.8.1.jar root@sandbox:/usr/lib/flume/lib
scp flume/tomcatToFlume.conf user@tomcatvm:/yourpathtoflume
scp flume/startAvroSink.sh root@sandbox:/yourpathtoflume
```

Back to Sandbox:

```
chmod +x startHBaseSink.sh
./startHBaseSink.sh
```

Tomcat/Flume VM:

```
ssh horton@tomcatvm (what have you)
...Install tomcat and flume...
cd /yourpathtoflume
chmod +x startAvroSink.sh
./startAvroSink.sh
```

Initially I started with an in-memory db as part of the tomcat process as there is not really a lot in it anyway. But to be able to play around with sqoop I wanted to have a real one. I'm using derby but mysql or something else should also work. ssh to the tomcatvm and download Apache Derby and extract.

```
cd /opt
sudo tar xvf db-derby-10.10.1.1-bin.tar
sudo mkdir iicaptainDB
sudo chown -R derbyuser
su - derbyuser
cd /opt/iicaptain
../db-derby-10.10.1.1-bin/bin/startNetworkServer -h tomcatvm &
```

Now we need to copy the JDBC driver to sandbox:

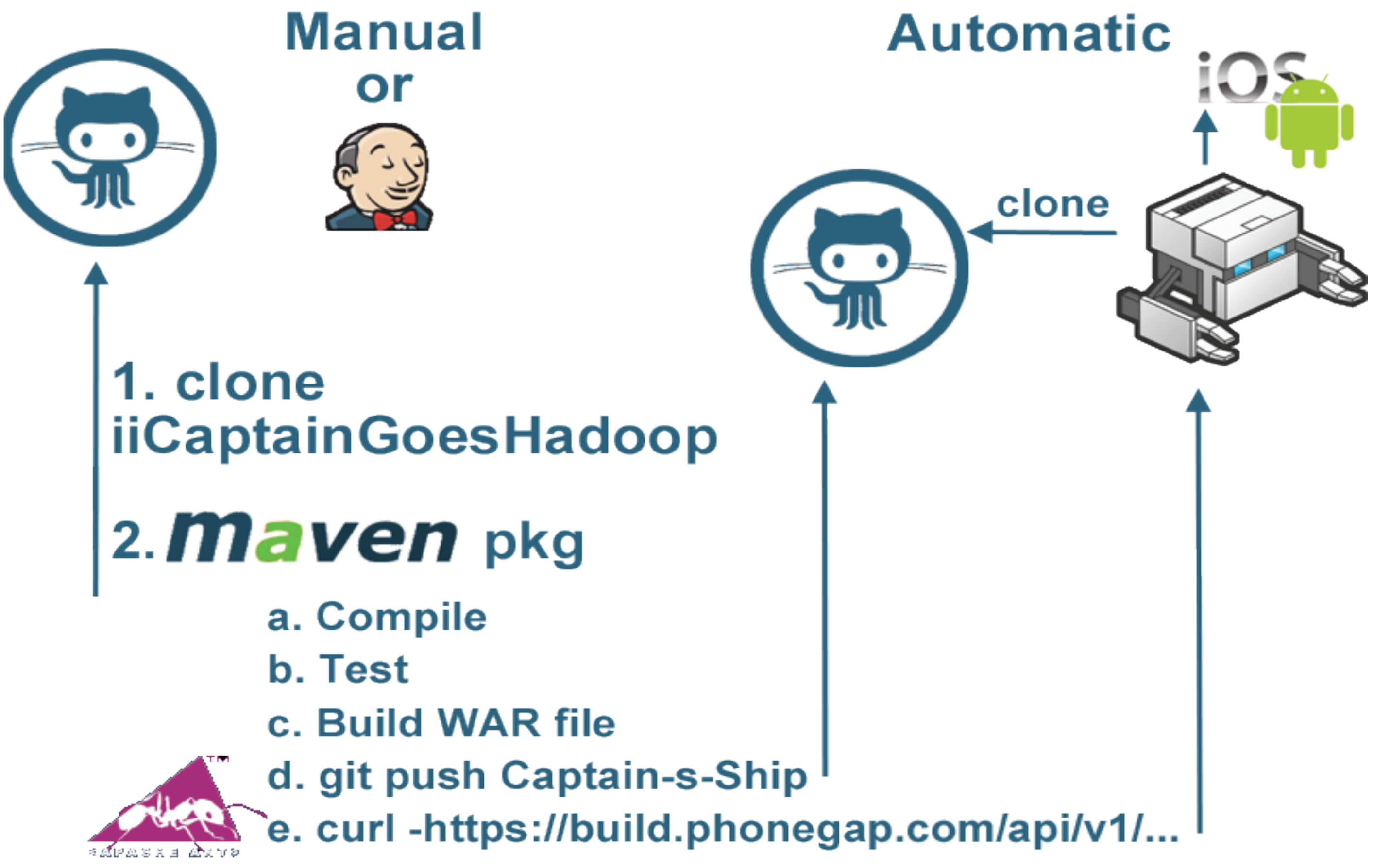
```
scp /opt/db-derby-10.10.1.1-bin/lib/derbyclient.jar root@sandbox:/usr/lib/sqoop/lib/
```

Now let's import the user data via sqoop (better: schedule by oozie):

```
ssh root@sandbox
hadoop fs -rmr /user/root/myusers
sqoop import --driver org.apache.derby.jdbc.ClientDriver --connect
jdbc:derby://tomcatvm:1527/iicaptain --table myusers -m 1 --hive-import
```

Deploy the app in Tomcat. As mentioned in the slide deck we need two apps:
iiCaptainGoesHadoop/delivery/iicaptain-hbase.war is part of the repo.
The Hive variant we need to build:
mvn package
I sometimes had problem with both apps in one tomcat instance (might be memory related as they are rather fat) so I'm deploying each app into it's own tomcat instance. When you make a copy from the tomcat folder to create the other one you have to modify the ports of one of them. Edit server.xml in the conf folder and modify the connector port (e.g. 8080->8088), shutdown port (e.g. 8005->8006) and ajp1.3 connector port (e.g. 8009->8010) The sources are configered for Hive. If you want to build the HBase version you need to modify: pom.xml (remove Hive dep, add HBase dep), src/main/webapp/META-INF/spring/hadoop.xml and src/main/java/net/iicaptain/homeport/web/AnalyticsController.java
Deploy both versions under iicaptain

Illustration of the build process:



You can also import the project into eclipse and deploy it from there.
Access the app: iiCaptain as guest/guest
Click on "Settings" and insert the ip/hostname of your tomcat VM.
Click "Start Journey".If your're allowing it your browser will give the app access to your geo location. You might need to modify your browser settings: [For Chrome]
Start anaytics (assuming you deployed the app under the contexts '/' and '/iicaptain'):
Realtime (HBase): Click Me!
Longterm (Hive): Me Too!

Cool stuff. Isn't it?

Create some load:

```
cd iiCaptainDataGenerator
vi generateLoad.sh (Point to your tomcatVM)
./generateLoad.sh
```

Now let's integrate AppAnnie:

```
ssh root@sandbox
mkdir /opt/appannie
chown oozie /opt/appannie
su - oozie
cd /opt/appannie
mkdir tmp
mkdir -p bin/de/digitalemil/iicaptain/appannie
hadoop fs -mkdir /user/hive/appannie
chmod 777 /opt/appannie/tmp
su - hue
```

```
hive
CREATE TABLE StoreData (Platform STRING, StartDate DATE, GeoStartTimeStamp BIGINT,
GeoEndTimeStamp BIGINT, CountryCode STRING, UnitsDownloads INT, UnitsUpdates INT, UnitsRefunds
INT, UnitsPromotions INT, RevenueDownloads DECIMAL, RevenueRefunds DECIMAL, RevenueUpdates
DECIMAL, RevenuePromotions DECIMAL) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS
TEXTFILE;
quit;
exit;
```

Now copy the scripts from the github repository to the sandbox:

```
cd iiCaptainDataGenerator
scp bin/de/digitalemil/iicaptain/appannie/XMLParser.class
root@sandbox:/opt/appannie/bin/de/digitalemil/iicaptain/appannie
scp import.sh root@sandbox:/opt/appannie
scp historic.sh root@sandbox:/opt/appannie
scp loaddata.* root@sandbox:/opt/appannie
scp refine.sh root@sandbox:/opt/appannie
```

The import.sh script downloads data from Apple, Google and Amazon for yesterday and should therefore be scheduled to run in the afternoon (PST) when Appannie has ingested all data from the day before. Afterwards the data needs to be loaded into HDFS (e.g. /apps/hive/warehouse/rawstoredata and refined. This will be done with Pig) and loaded into Hive. Download histroic data (do it just once) by executing the historic.sh script in /opt/appannie. Let first create some HCatalog Tables. One for the raw store data and one for the refined store data:

```
hcat -e "create table rawstoredata(appstore string, startdate string, country string,
unitsdownloads bigint, unitsupdates bigint, unitsrefunds bigint, unitspromotions bigint,
revenuedownloads double, revenueupdates double, revenuerefunds double, revenuepromotions double)
row format delimited fields terminated by ',' stored as textfile"
hcat -e "create table refinedstoredata(appstore string, startdate string, geostartdate bigint,
geoenddate bigint, country string, unitsdownloads bigint, unitsupdates bigint, unitsrefunds
bigint, unitspromotions bigint, revenuedownloads double, revenueupdates double, revenuerefunds
double, revenuepromotions double) row format delimited fields terminated by ',' stored as
textfile"
```

Then copy the csv files from the tmp folder onto HDFS:

```
hadoop fs -put tmp/*csv /apps/hive/warehouse/rawstoredata
```

Now run the following pig script to refine the data (also in refine.sh):

```
storeData = LOAD 'default.rawstoredata' USING org.apache.hcatalog.pig.HCatLoader();

refinedStoreData= FOREACH storeData GENERATE appstore, startdate,
        MilliSecondsBetween(ToDate(startdate), ToDate('1970-01-01', 'yyyy-mm-dd'))-978307200000L
AS geostartdate,
    MilliSecondsBetween(ToDate(startdate), ToDate('1970-01-01', 'yyyy-mm-
dd'))-978307200000L+86400000L AS geoenddate,
    country, unitsdownloads, unitsupdates, unitsrefunds, unitspromotions,
    revenuedownloads, revenueupdates, revenuerefunds, revenuepromotions;

STORE refinedStoreData INTO 'default.refinedstoredata' USING
org.apache.hcatalog.pig.HCatStorer();
```

Load the historic data into Hive:

```
su - hive
hive
insert into table storedata Select * from refinedstoredata;
quit;
exit;
```

You can create an Oozie workflow in Hue to do that. You can find mine in iiCaptainDataGenerator/appannieoozie.xml If you want to ingest historic data (last 30 days) use the script historic.sh. It downloads the data but the upload to HDFS and loading it into Hive is a manual step.
This leaves us with Sentiment Analysis but this is business as usual. Just configure the flume agent on the sandbox to read the tweets you're interested in (keyword: iiCaptain), store on HDFS, read in from Hive via JSON SerDe.