Security/Governance/GDPR Demo on CDP PvC Base 7.x

Summary

How to quickly setup Cloudera Security/Governance/GDPR (Worldwide Bank) demo using Cloudera Data Platform Data Center (CDP PvC Base). It can be deployed either on AWS using AMI or on your own setup via provided script

Recording

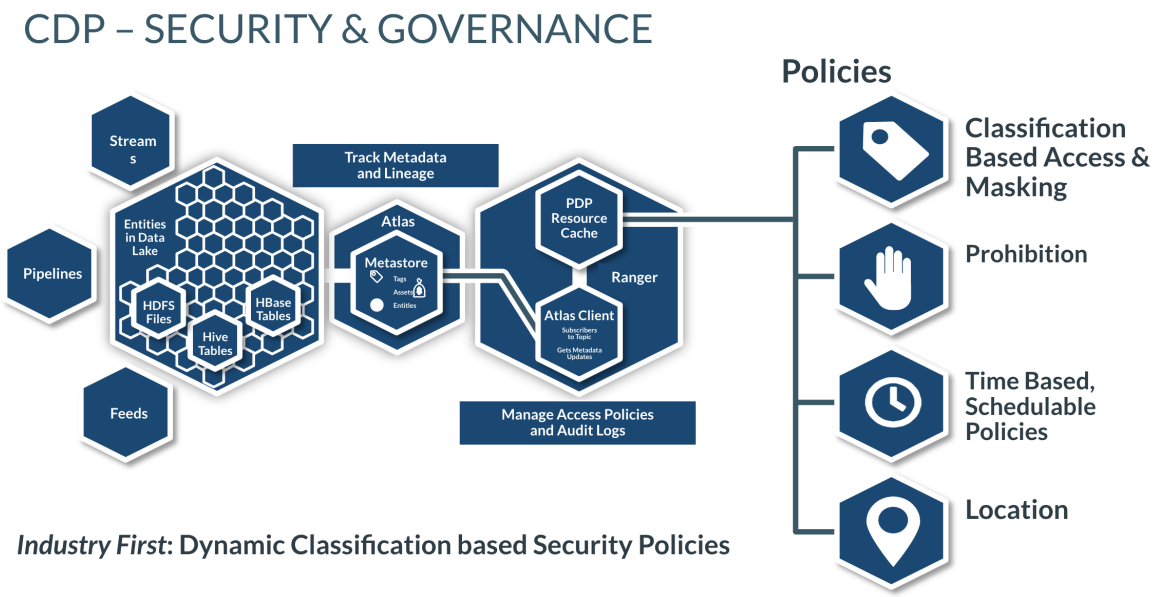
Cloudera Connect Showcase - CDP Demo on Security and Governance

What's included

- Single node CDP 7.1.4 including:
  - Cloudera Manager (60 day trial license included) - for managing the services
  - Kerberos - for authentication (via local MIT KDC)
  - Ranger - for authorization (via both resource/tag based policies for access and masking)
  - Atlas - for governance (classification/lineage/search)
  - Zeppelin - for running/visualizing Hive queries
  - Impala/Hive 3 - for Sql access and ACID capabilities
  - Spark/HiveWarehouseConnector - for running secure SparkSQL queries

- Worldwide Bank artifacts
  - Demo hive tables
  - Demo tags/attributes and lineage in Atlas
  - Demo Zeppelin notebooks to walk through demo scenario
  - Ranger policies across HDFS, Hive/Impala, Hbase, Kafka, SparkSQL to showcase:
    - Tag based policies across HDP components
    - Row level filtering in Hive columns
    - Dynamic tag based masking in Hive/Impala columns
    - Hive UDF execution authorization
    - Atlas capabilities like
      - Classifications (tags) and attributes
      - Tag propagation
      - Data lineage

        - Business glossary:categories and terms
    - GDPR Scenarios around consent and data erasure via Hive ACID
- Hive ACID / MERGE labs



Option 1: Steps to deploy on your own setup

- Launch a vanilla Centos 7 VM and set up a single node CDP cluster using this Github but instead of "base" CM template choose the "wwbank_krb.json" template:

```
yum install -y git
#setup KDC
curl -sSL https://gist.github.com/abajwa-hw/bca3d23fe146c3ebd59a9b5fd19480a3/raw | sudo -E sh

git clone https://github.com/fabiog1901/SingleNodeCDPCluster.git
cd SingleNodeCDPCluster
./setup_krb.sh gcp templates/wwbank_krb.json

#Setup worldwide bank demo using script
curl -sSL https://raw.githubusercontent.com/abajwa-hw/masterclass/master/ranger-atlas/setup-dc-703.sh | sudo -E bash
```

Once the script completes, you will need to restart Zeppelin once (via CM) for it to pick up the demo notebooks

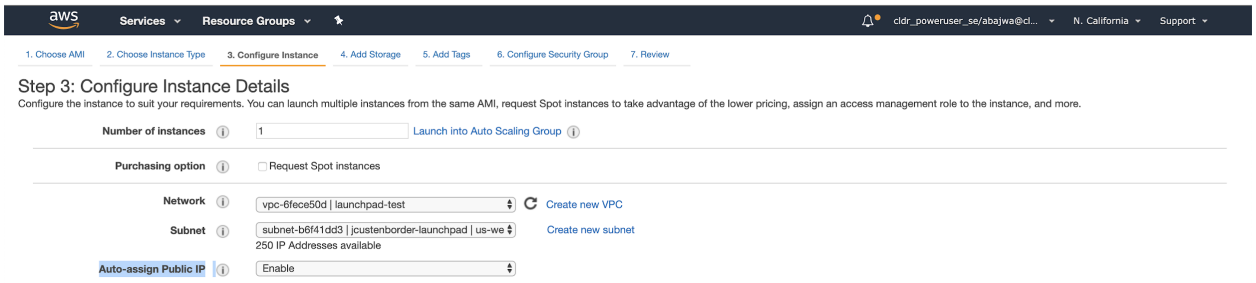
Option 2: Steps to launch prebuilt AMI on AWS

- 1. Login into the AWS EC2 console using your credentials
- 2. Select the AMI from ‘N. California’ region by clicking one the links below:
  - CDP 7.1.4 here (public)

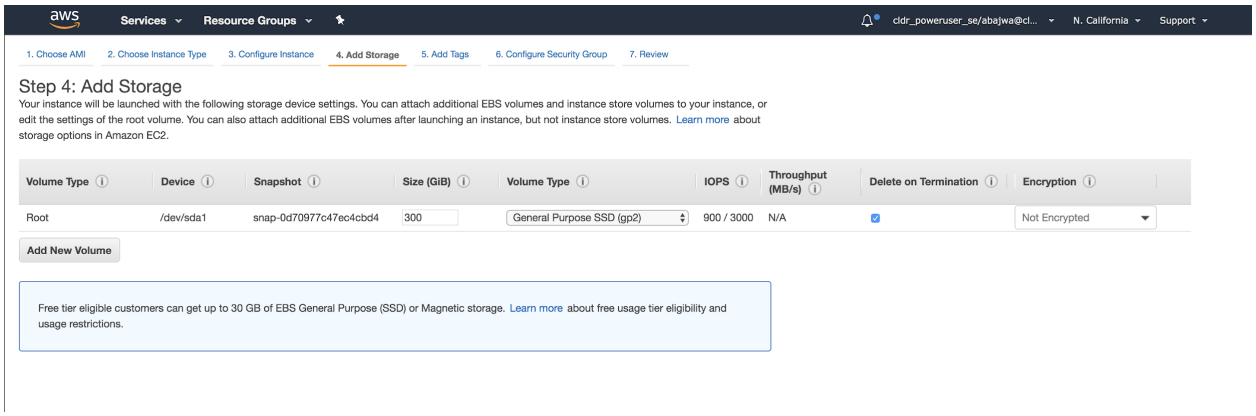
Now choose instance type: select ‘m4.4xlarge’ and click Next

Note: if you choose a smaller instance type from the above recommendation, not all services may come up

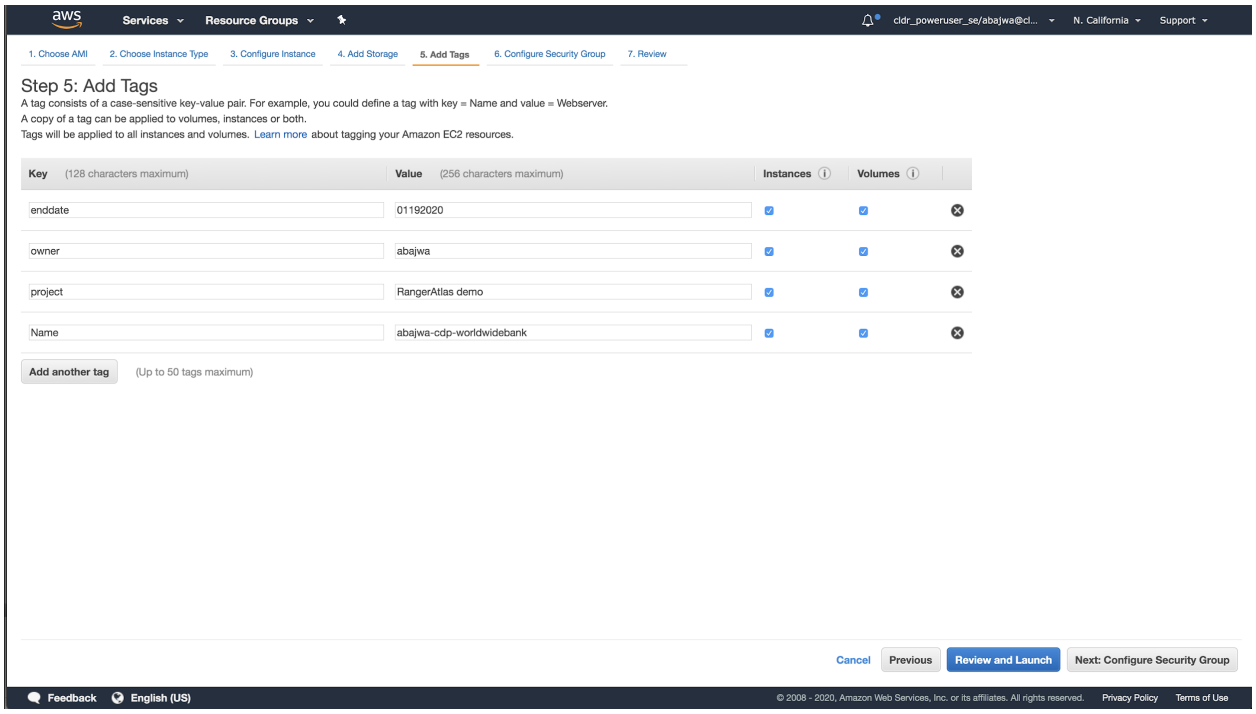
- 3.. Configure Instance Details: make sure "Auto-assign Public IP" is enabled and click ‘Next’



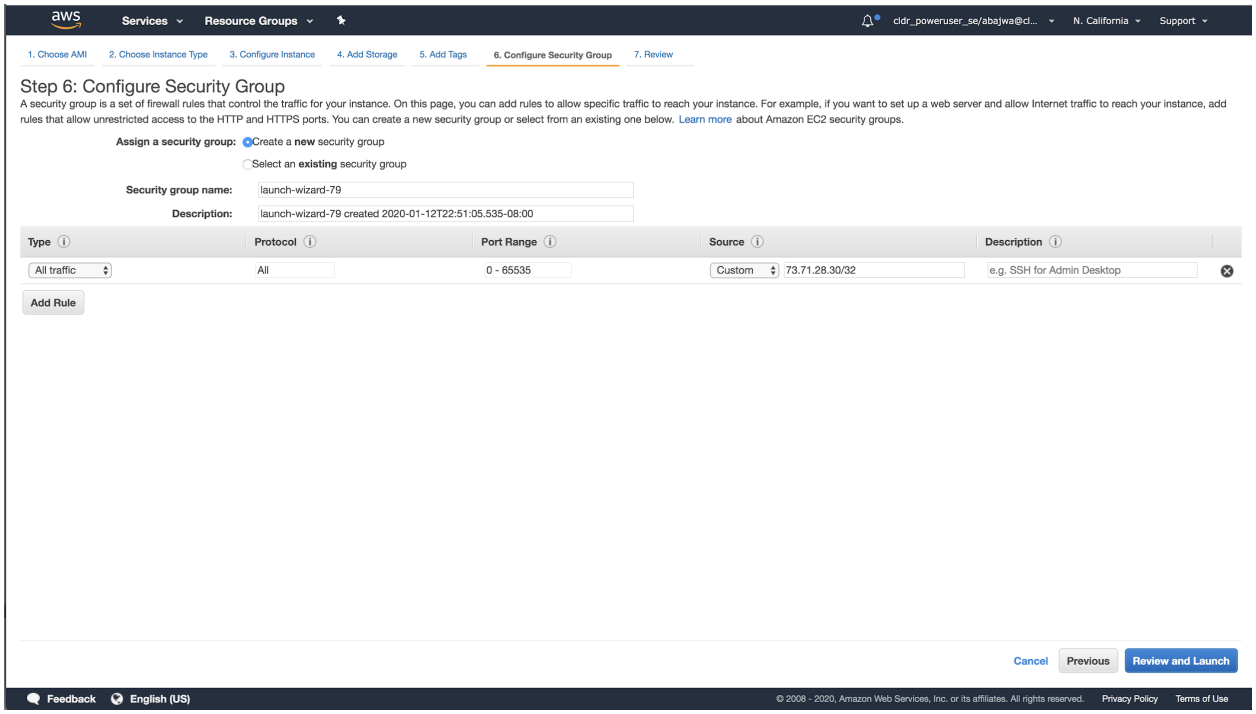
- 4. Add storage: use at least 100 GB and click ‘Next’



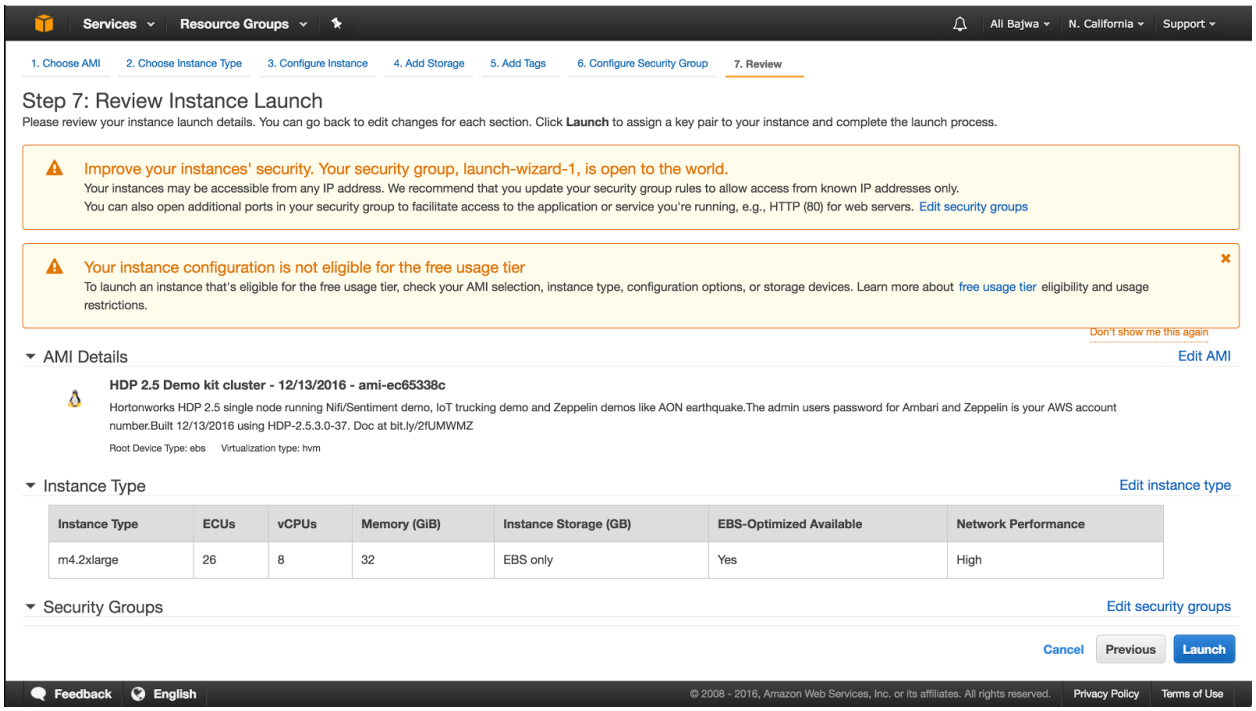
- 5. Add tags needed to prevent instances from being terminated. Then click ‘Next’



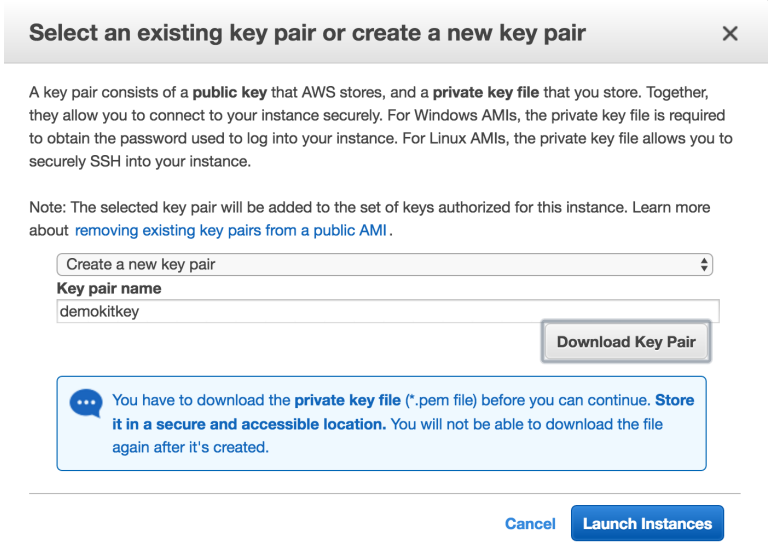
- 6. Configure security group: create a new security group and select ‘All traffic’ and open all ports to **only your IP**. Then click ‘Review and Launch’



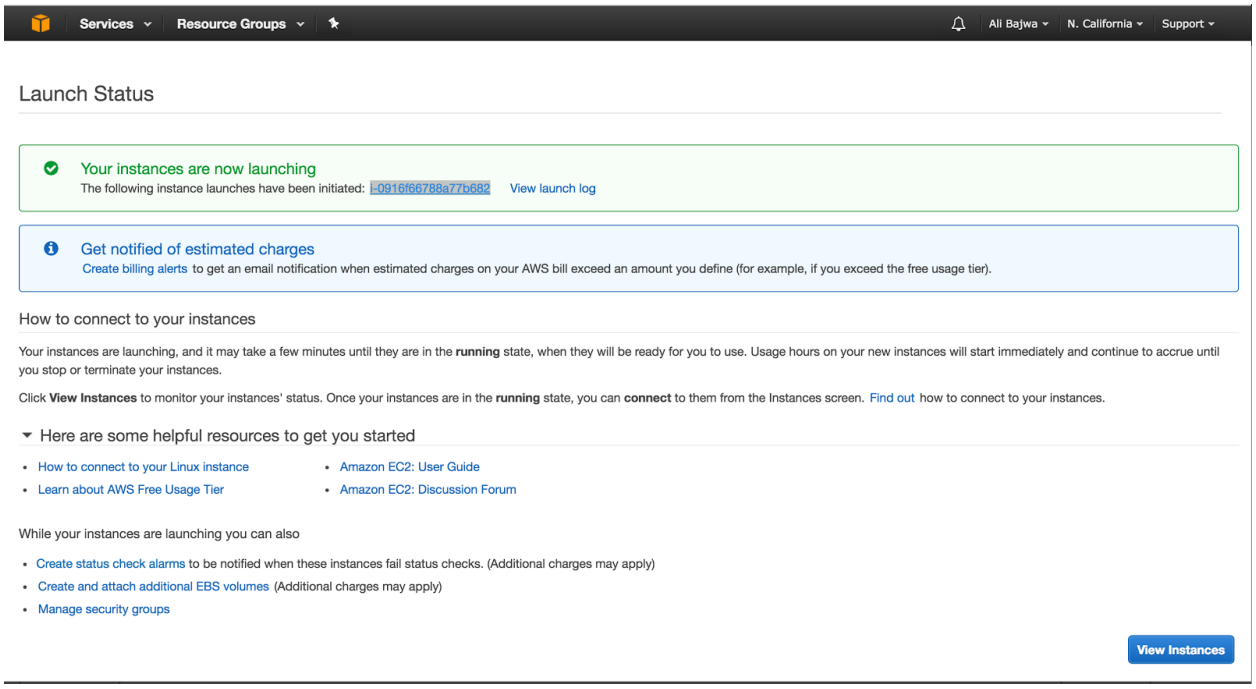
- 7. Review your settings and click Launch



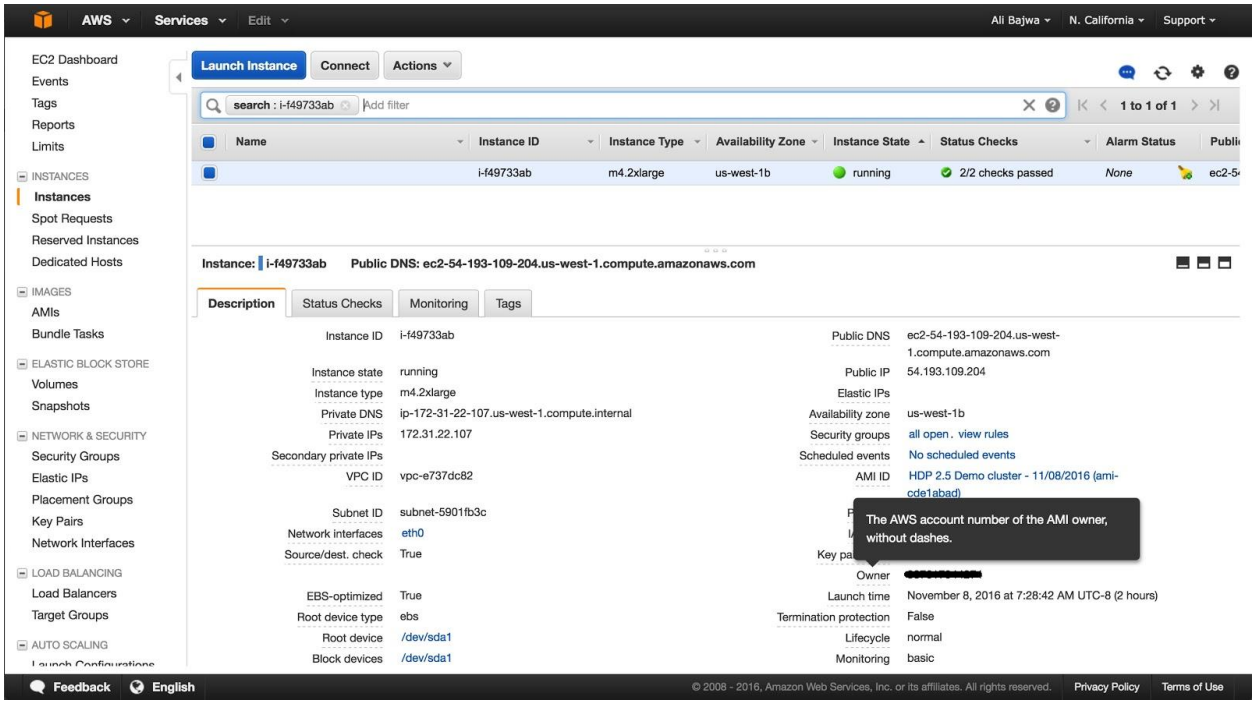
- 8. Create and download a new key pair (or choose an existing one). Then click ‘Launch instances’



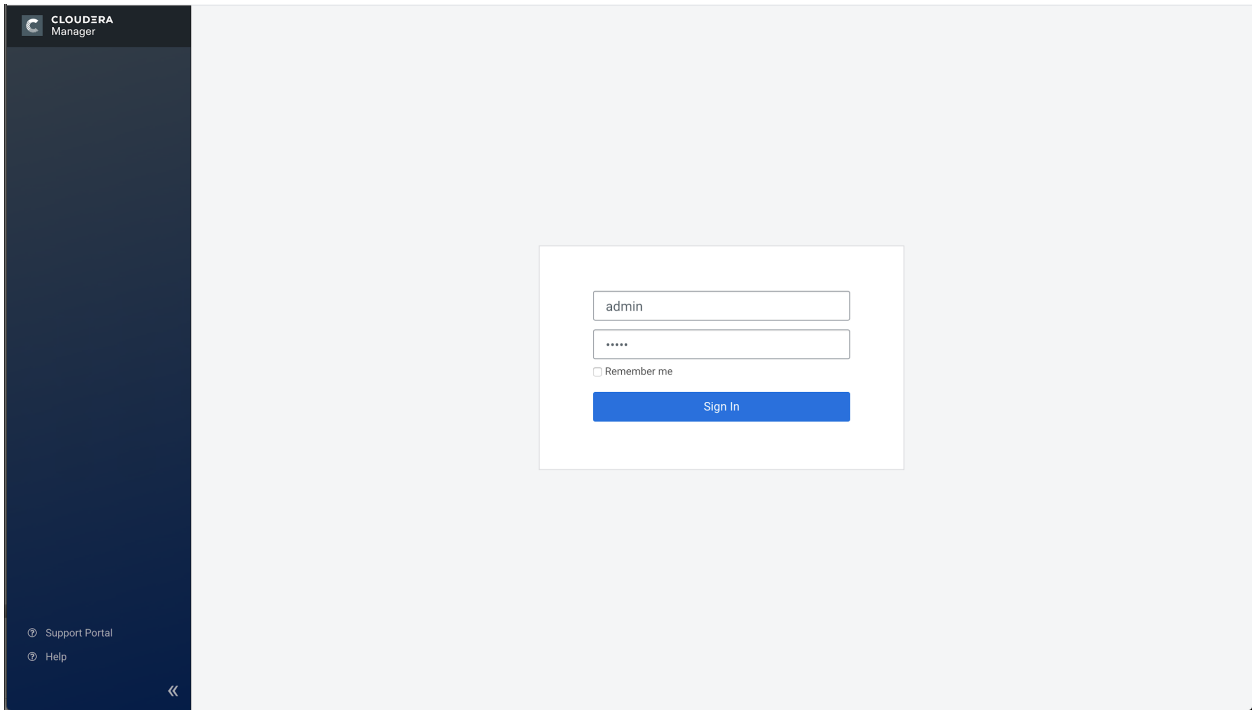
- 9. Click the shown link under ‘Your instances are now launching’



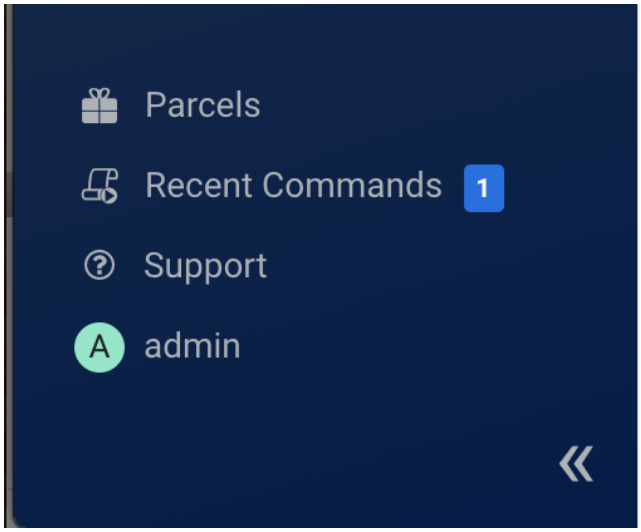
- 10. This opens the EC2 dashboard that shows the details of your launched instance



- 11. Make note of your instance's 'Public IP' (which will be used to access your cluster). If the 'Public IP' is blank, wait 1-2 minutes for this to be populated
- 12. After 5-10 minutes, open the below URL in your browser to access Cloudera Manager (CM) console: http://<PUBLIC IP>:7180. Login as admin/admin



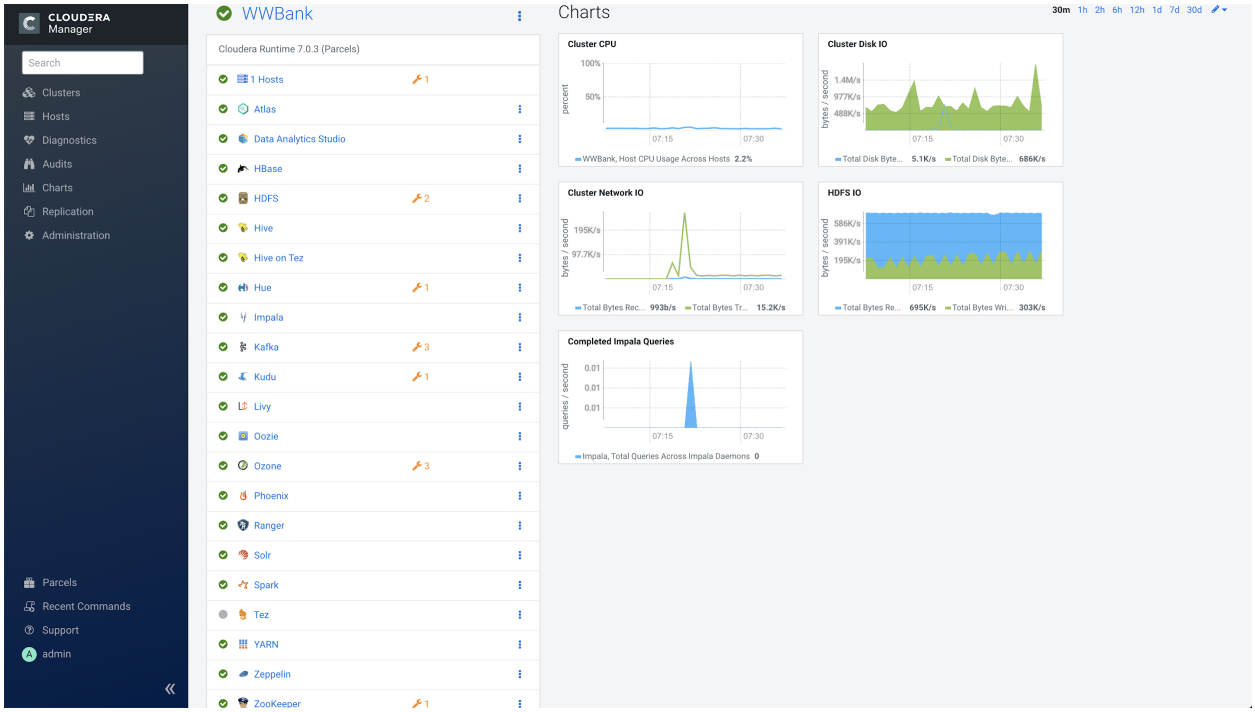
- 13. At this point, CM may still be in the process of starting all the services. You can tell by the presence of the blue operation notification near the bottom left of the page. If so, just wait until it is done.



(Optional) You can also monitor the startup using the log as below:

- Open SSH session into the VM using your key and the public IP e.g. from OSX:
  `ssh -i ~/.ssh/mykey.pem centos@<publicIP>`
- Tail the startup log:
  `tail -f /var/log/cdp_startup.log`
- Once you see “cluster is ready!” you can proceed

- 14. Once the blue operation notification disappears and all the services show a green check mark, the cluster is fully up.



If any services fail to start, use the hamburger icon next to SingleNodeCluster > Start button to start

Accessing cluster resources

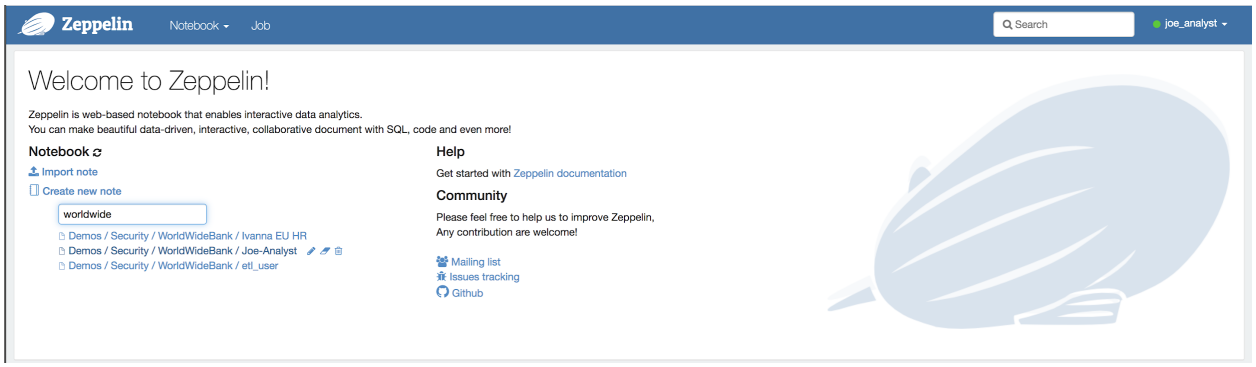
CDP urls

- Access CM at :7180 as admin/admin
- Access Ranger at :6080. Ranger login is admin/BadPass#1
- Access Atlas at :31000. Atlas login is admin/BadPass#1
- Access Zeppelin at :8885. Zeppelin users logins are:
  - joe_analyst = BadPass#1
  - ivanna_eu_hr = BadPass#1
  - etl_user = BadPass#1
- Access Hue at :8889. Hue users logins are:
- joe_analyst = BadPass#1
- ivanna_eu_hr = BadPass#1
- etl_user = BadPass#1

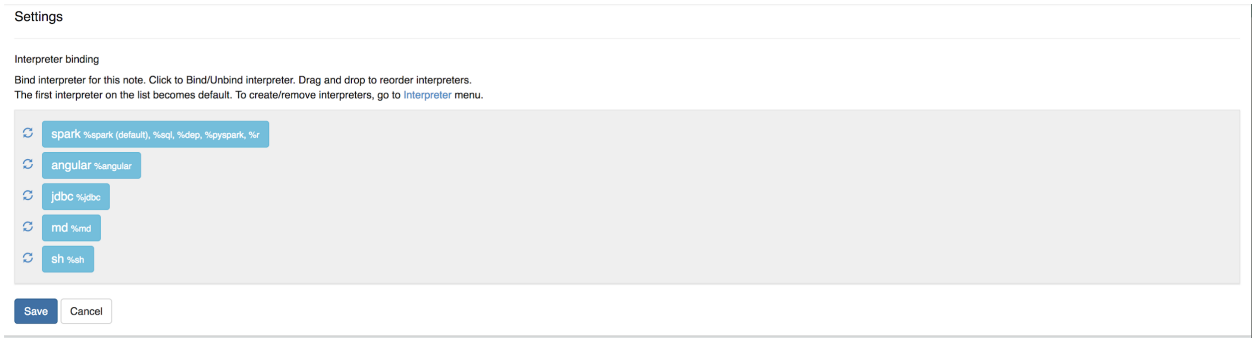
Demo walkthrough

Run queries as joe_analyst

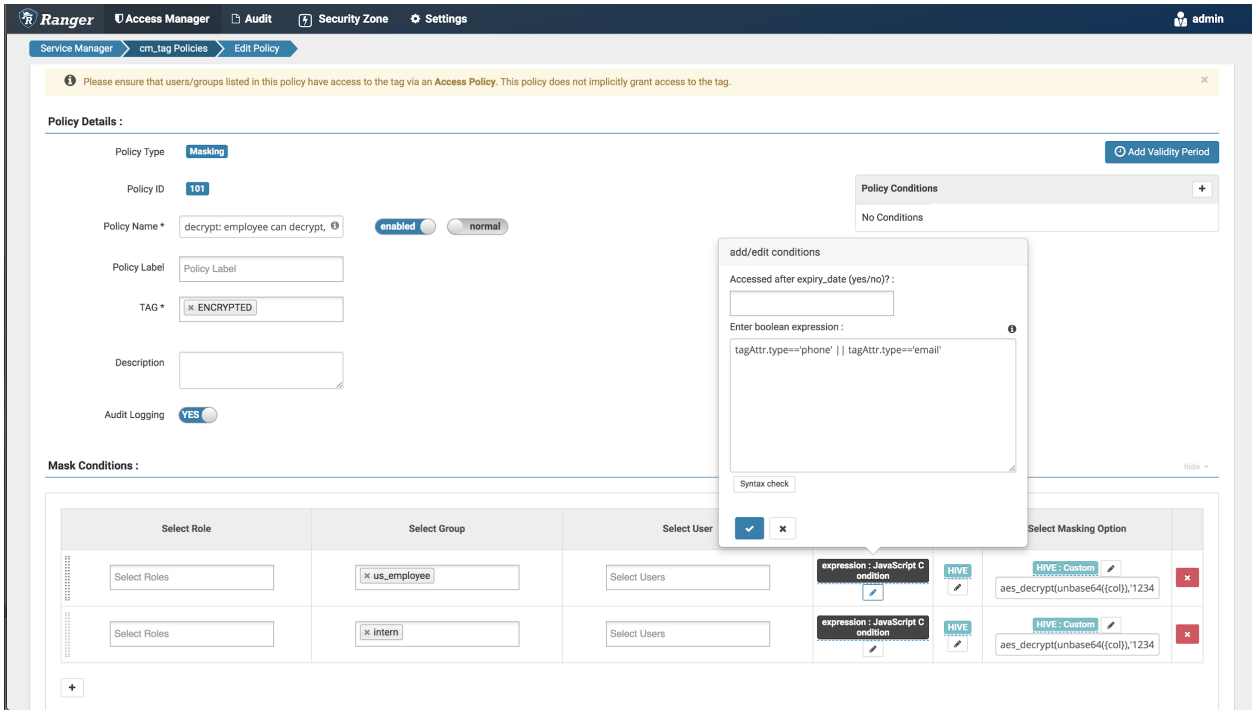
1. Open Zeppelin and login as joe_analyst. Find his notebook by searching for "worldwide" using the text field under ‘Notebook’ section. Select the notebook called: “Worldwide Bank - Joe Analyst”



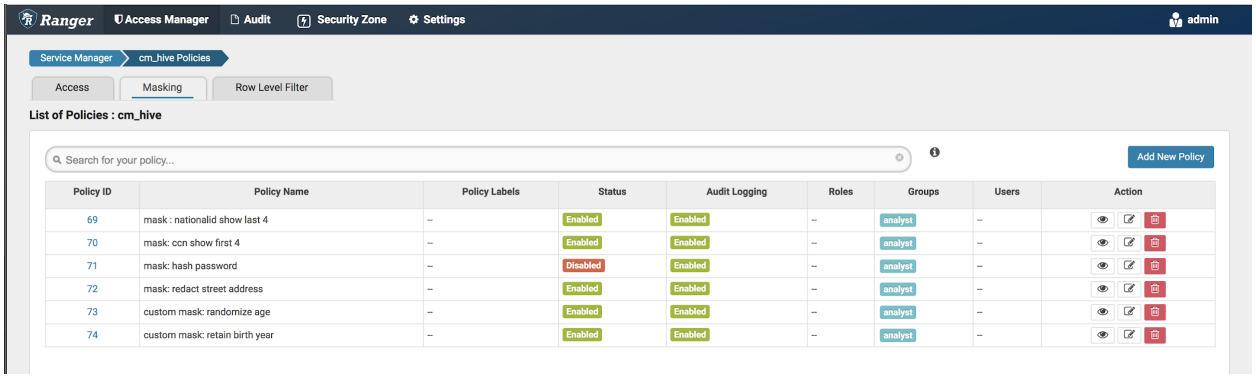
2. On the first launch of the notebook, you will be prompted to choose interpreters. You can keep the defaults but make sure you click Save button:



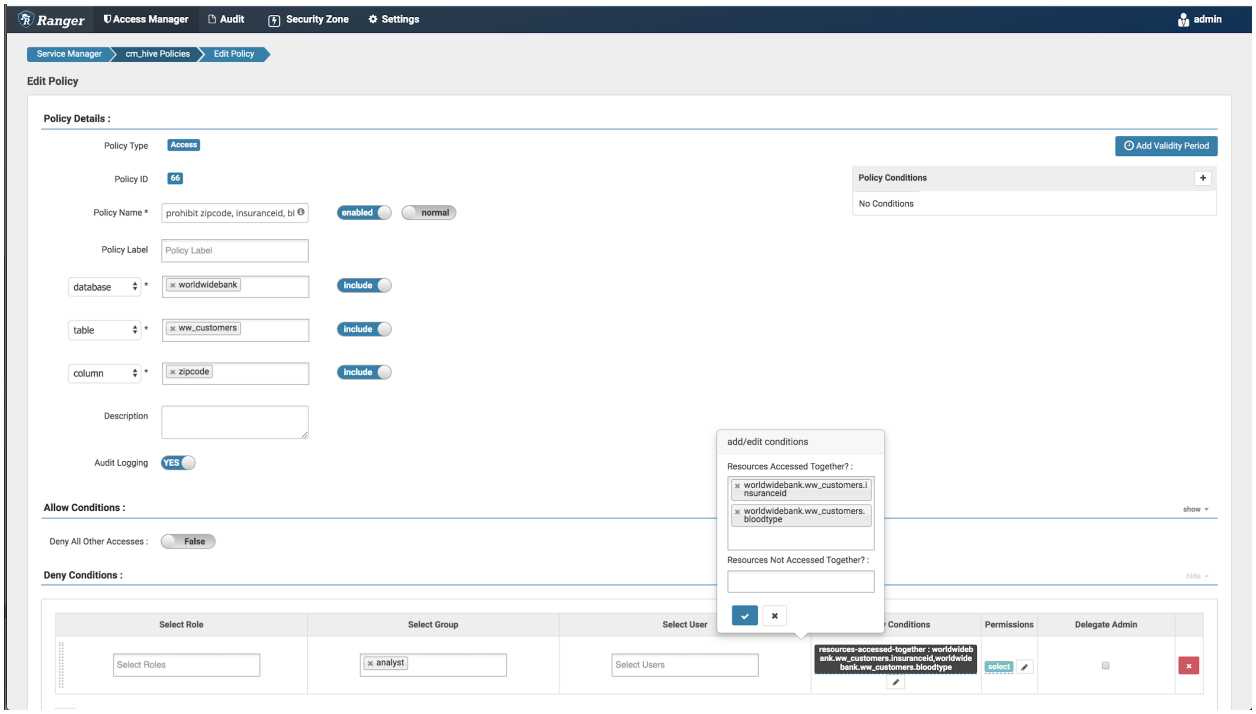
3. Run through the notebook. This notebook shows
a) MRN/password masked via tag policy. Here is the Ranger policy that enables this:



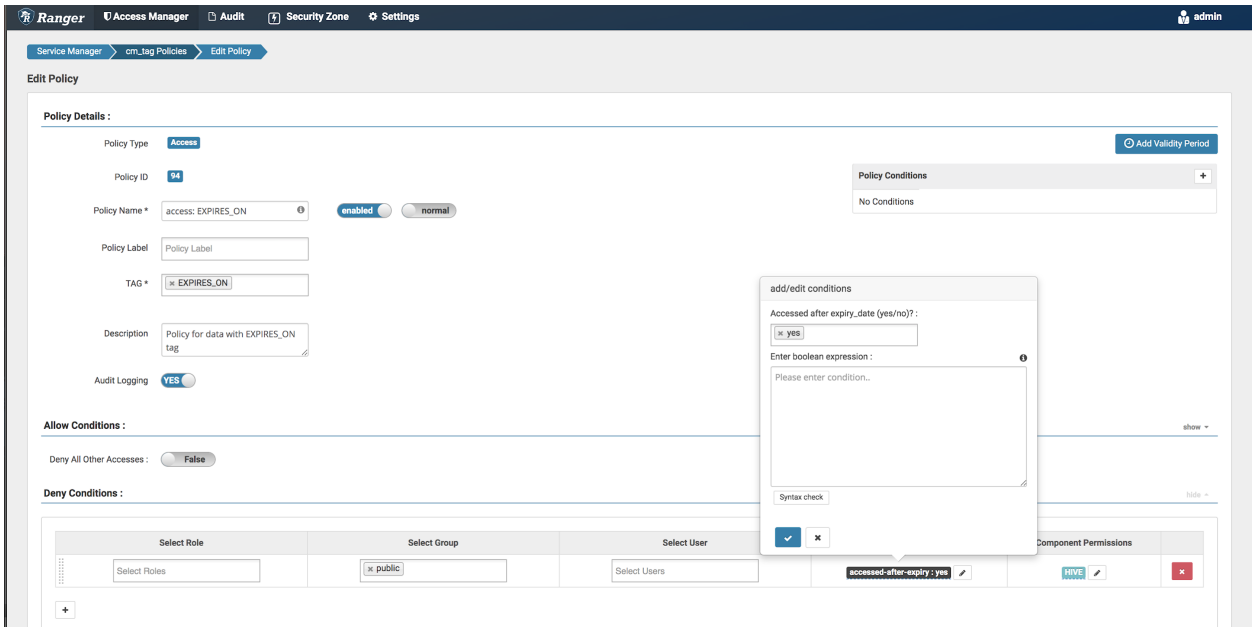
tb) dynamic column level masking
Address, nationalID, credit card number are masked via Hive column policies specified in Ranger. Notice that Birthday and age columns are masked using custom mask



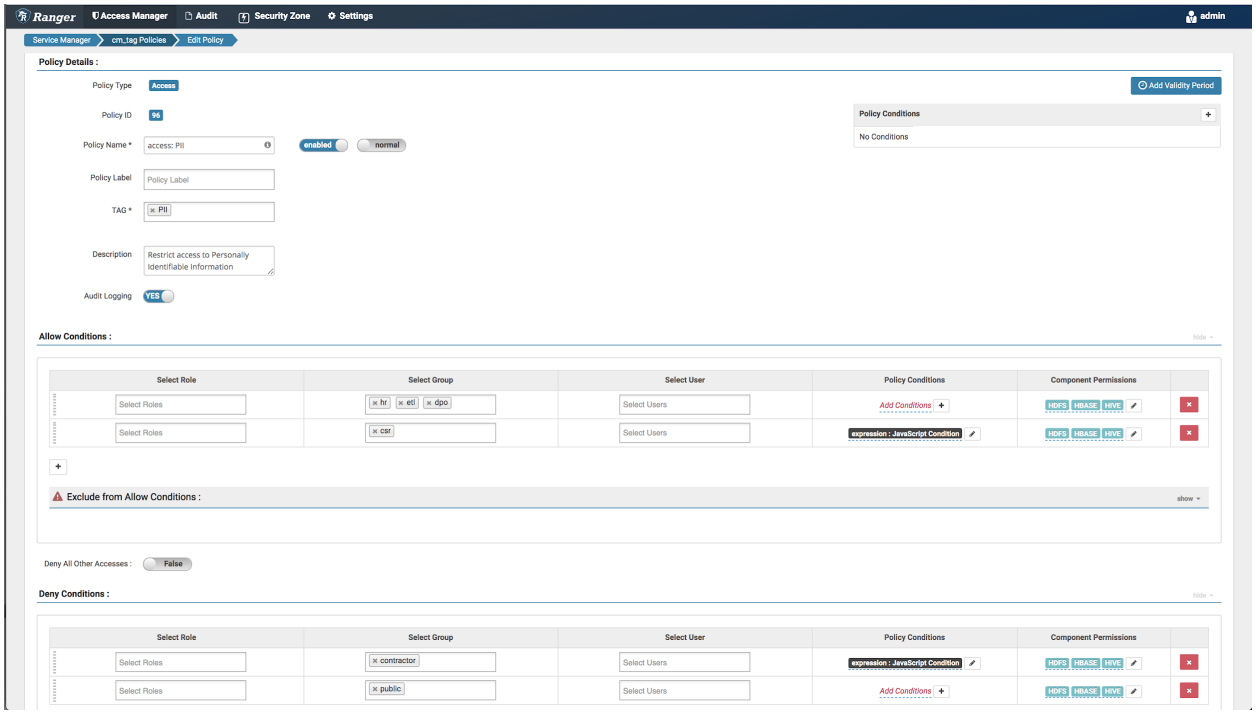
b) It also shows prohibition policy where zipcode, insuranceid and bloodtype can not be combined in a query



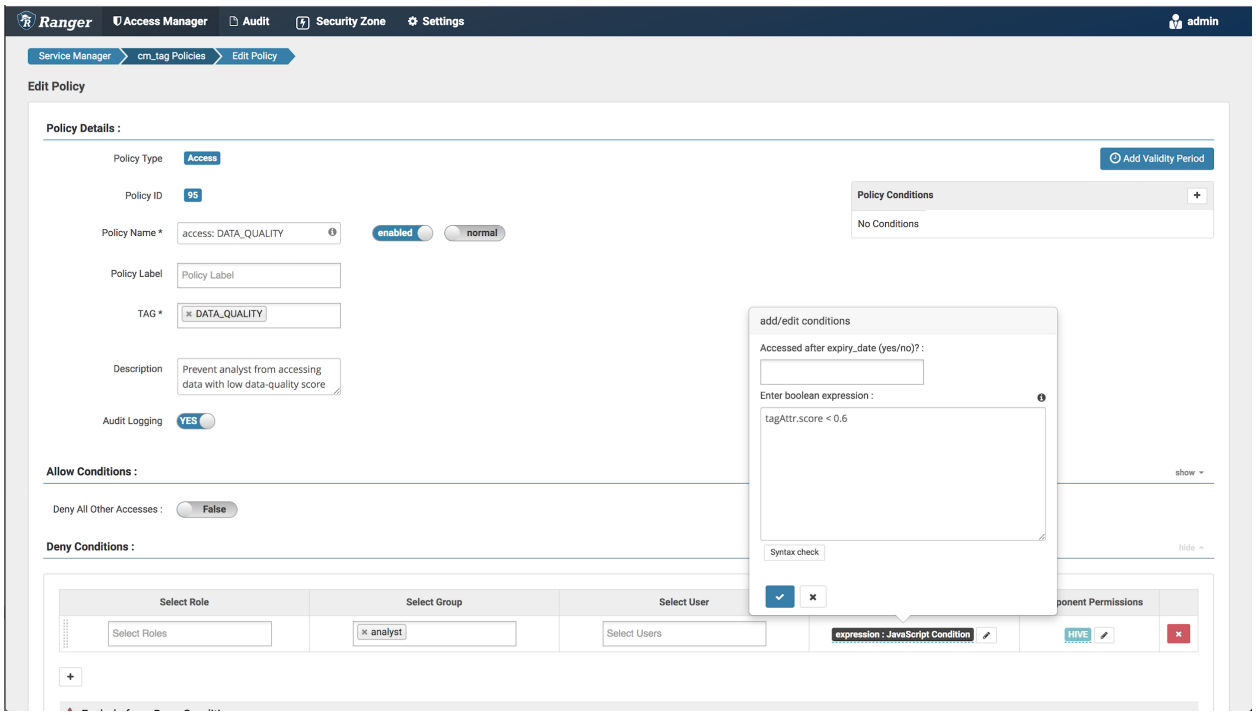
c) also shows tag based policies:
Attempts to access any object tagged with EXPIRES_ON accessed after expiry date, will be denied. As we will show later, the fed_tax column of tax_2015 table is tagged in Atlas as EXPIRED_ON with expiry date of 2016 so it should not be allowed to be queried



Also attempts to access object tagged with PII will be denied as per policy, only HR allowed. As we will show later, the ssn column of tax_2015 table is tagged as PII in Atlas.



Attempts to access cost_savings.claim_savings table as analyst will fail because there is a policy that minimum of 60% data quality score is required for analysts. As we will see, this table is tagged in Atlas as having score of 51%

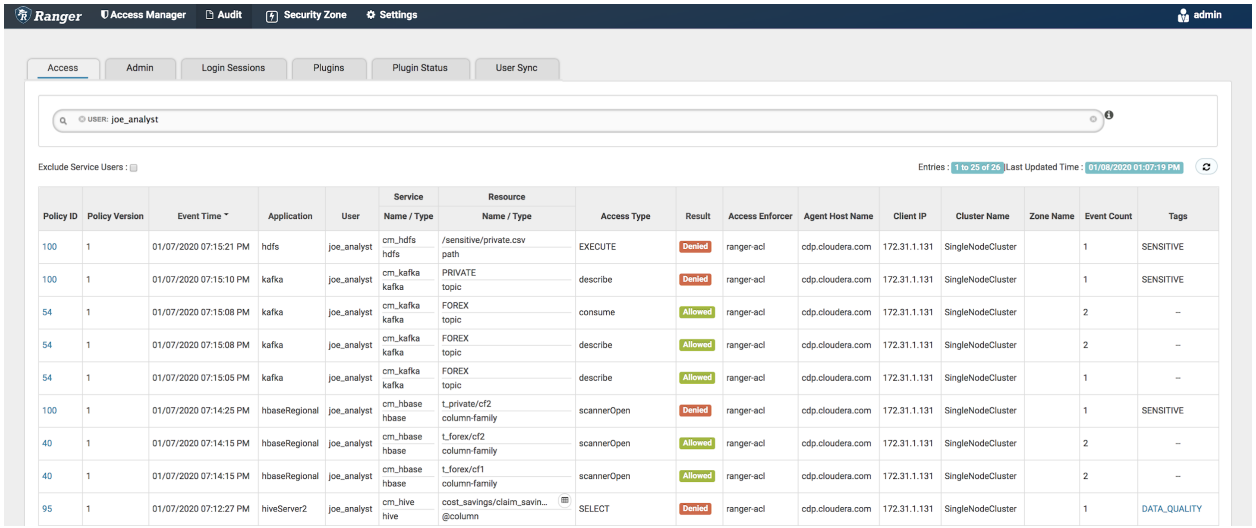


The same queries can also be run via SparkSQL via spark-shell (as described above). Sample queries for joe_analyst:

```
hive.execute("SELECT surname, streetaddress, country, age, password, nationalid, ccnumber,
mrn, birthday FROM worldwidebank.us_customers").show(10)

hive.execute("select zipcode, insuranceid, bloodtype from
worldwidebank.ww_customers").show(10)

hive.execute("select * from cost_savings.claim_savings").show(10)
```

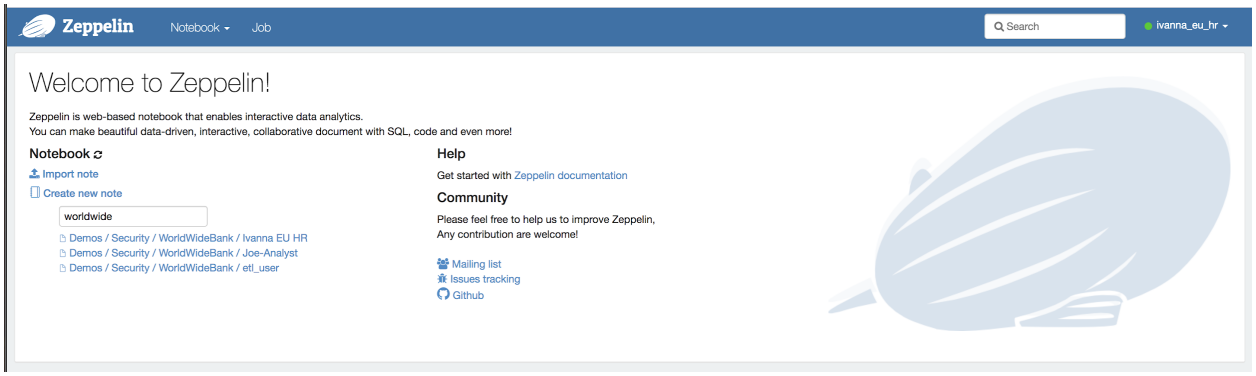
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0.7.0.0.0-462
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import com.hortonworks.hwc.HiveWarehouseSession
import com.hortonworks.hwc.HiveWarehouseSession

scala> import com.hortonworks.hwc.HiveWarehouseSession._
import com.hortonworks.hwc.HiveWarehouseSession._

scala> val hive = HiveWarehouseSession.session(spark).build()
hive: com.hortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl = com.hortonworks.spark.sql.hive.llap.HiveWarehouseSessionImpl@2ee4c028

scala>

scala> hive.execute("SELECT surname, streetaddress, country, age, password, nationalid, ccnumber, mrn, birthday FROM worldwidebank.us_customers").show(10)
+---------+--------------------+-------+---+--------------------+----------+----------+----+----------+
|  surname|       streetaddress|country|age|            password| nationalid|   ccnumber| mrn|  birthday|
+---------+--------------------+-------+---+--------------------+----------+----------+----+----------+
|   Powers| nnnn Xxxxxxxx Xxxxx|     US| 54|76a3fe33eb676cb12...|xxx-xx-3016|5.49xxxx+xx|null|01-01-1973|
|  Whitman|  nnnn Xxxxxxx Xxxxx|     US| 59|6103bc600fc877e5c...|xxx-xx-8255|4.55xxxx+xx|null|01-01-1977|
|   Marone|      nnn Xxxxx Xxxxx|     US| 44|da01407d5922624b1...|xxx-xx-6142|4.53xxxx+xx|null|01-01-1977|
|     Harp|  nnnn Xxx Xxxx Xxxx|     US| 30|717051c13b6bda179...|xxx-xx-4018|5.20xxxx+xx|null|01-01-1997|
|  Pereira|nnnn Xxxxxxxx Xxxxxx|     US| 93|47d2ebe1a1c114669...|xxx-xx-1927|5.38xxxx+xx|null|01-01-2038|
|Blackburn|     nnn Xxxxxxx Xxxx|     US| 59|127473e0e3d200b19...|xxx-xx-7903|5.53xxxx+xx|null|01-01-1965|
| Gonzalez|      nnnn Xxxxx Xxxx|     US| 28|4082d6873762eaf3b...|xxx-xx-2582|4.71xxxx+xx|null|01-01-1995|
|   Helton|  nnnn Xxxxxxx Xxxxx|     US| 58|5a5e10b606eb7599b...|xxx-xx-4291|5.25xxxx+xx|null|01-01-1975|
|      Mei|     nnn Xxxxx Xxxxxx|     US| 77|79027eba8d2ae53c9...|xxx-xx-8366|5.20xxxx+xx|null|01-01-1941|
| Goldberg|     nnnn Xxxxx Xxxxx|     US| 81|478f719c117d97df4...|xxx-xx-4937|4.55xxxx+xx|null|01-01-1944|
+---------+--------------------+-------+---+--------------------+----------+----------+----+----------+
```

4. Confirm using Ranger audits that the queries ran as joe_analyst. Also notice that column names, masking types, IPs and policy IDs were captured. Also notice tags (e.g. DATA_QUALITY or PII) are captured along with their attributes. Also notice that these audits were captured for operations across Hive, Hbase, Kafka and HDFS

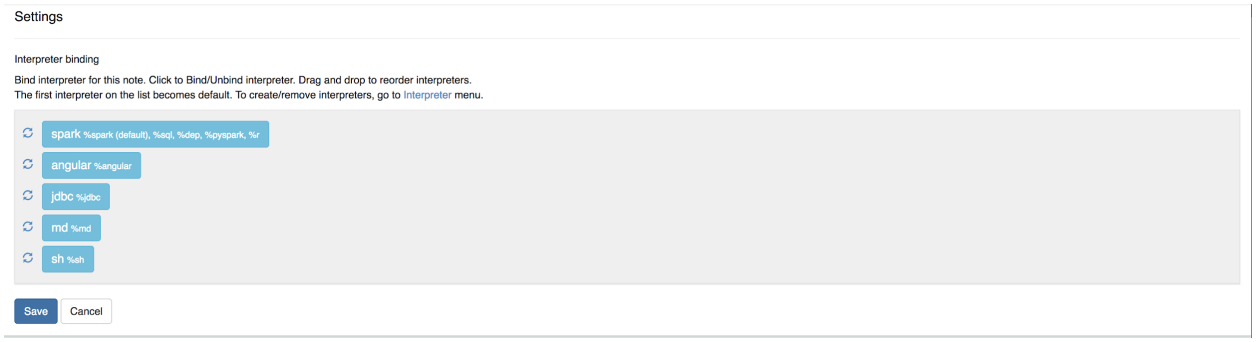


Run queries as ivanna_eu_hr

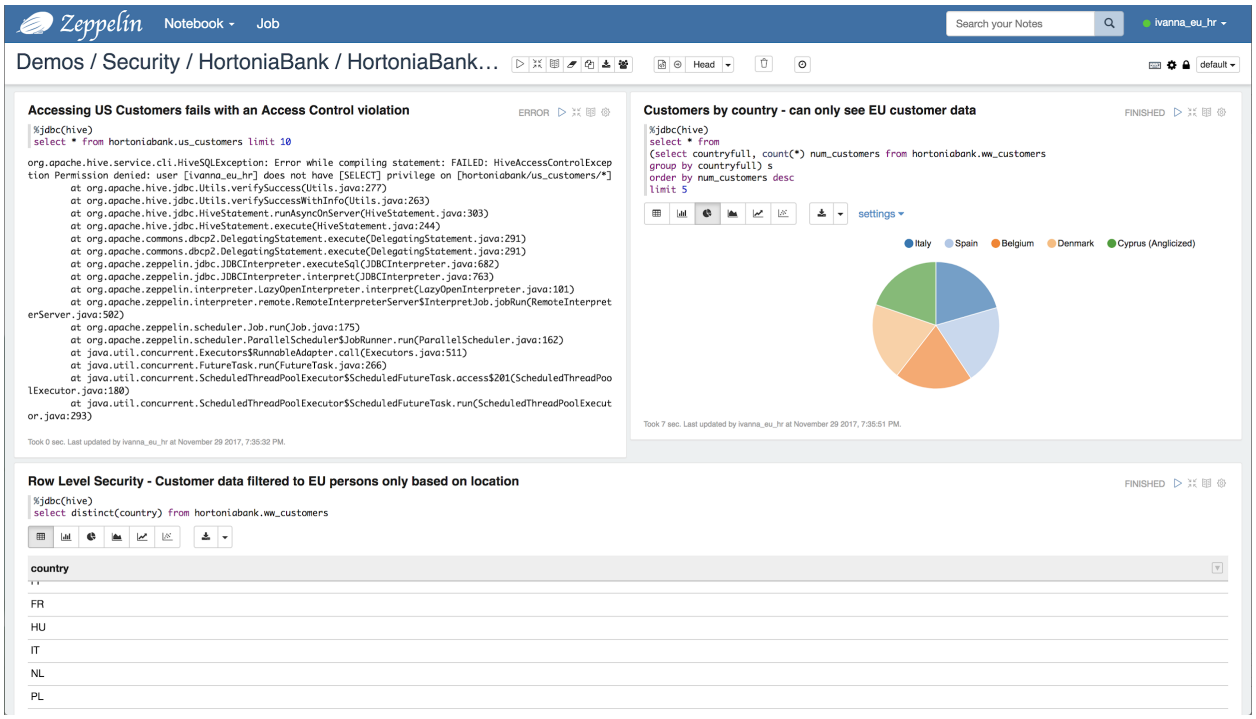
5. Once services are up, open Ranger UI and also login to Zeppelin as ivanna_eu_hr and find her notebook by searching for "hortonia" using the text field under the ‘Notebook’ section. Select the notebook called: “Worldwide Bank - Ivana EU HR”



6. On first launch of the notebook, you may be prompted to choose interpreters. You can keep the defaults but make sure you click Save button:

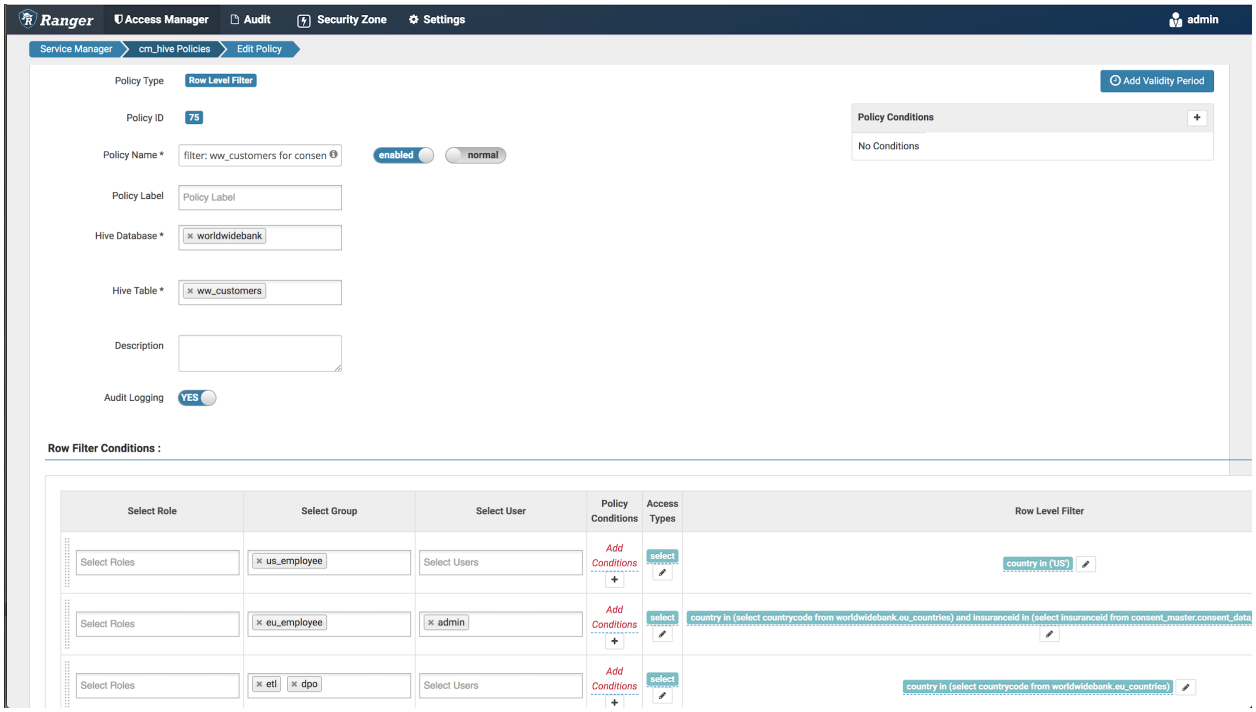


7. Run through the notebook cells using Play button in top right of each cell (or Shift-Enter)



This notebook highlights:

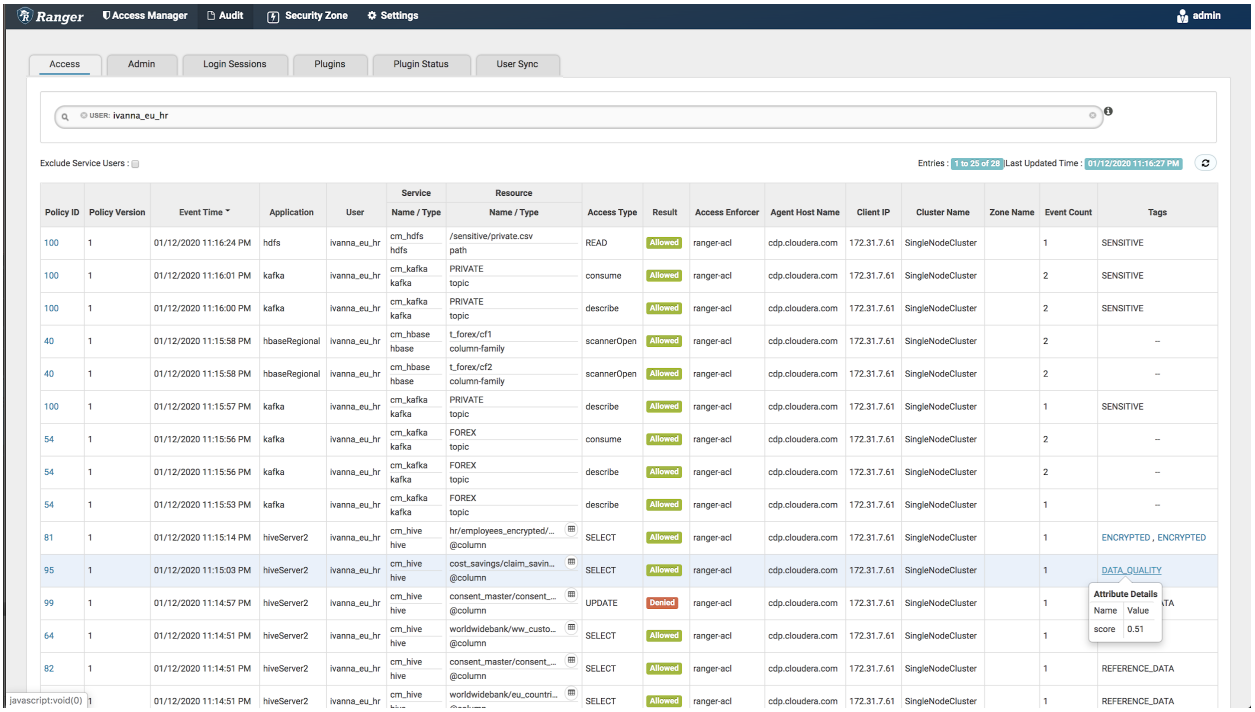
a) Row level filtering: as Ivana can only see data for European customers who have given consent (even though she is querying ww_customers table which contains both US and EU customers). Below is the Ranger hive policy that enables this feature:



b) it also shows that since Ivana is part of HR group, there are no policies that limit her access: so she can see raw passwords, nationalIDs, credit card numbers, MRN #, birthdays etc

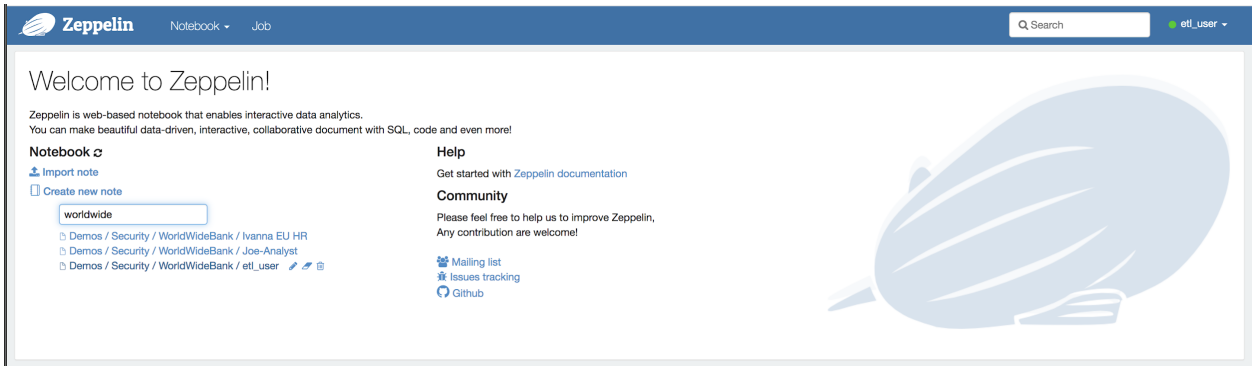
c) The last cells show that tag based policies d

3. Once you successfully run the notebook, you can open the Ranger Audits to show the policies and that the queries ran as her and that row filtering occurred (notice ROW_FILTER access type):



Run queries as etl_user

8. Similarly, you can login to Zeppelin as etl_user and run his notebook as well

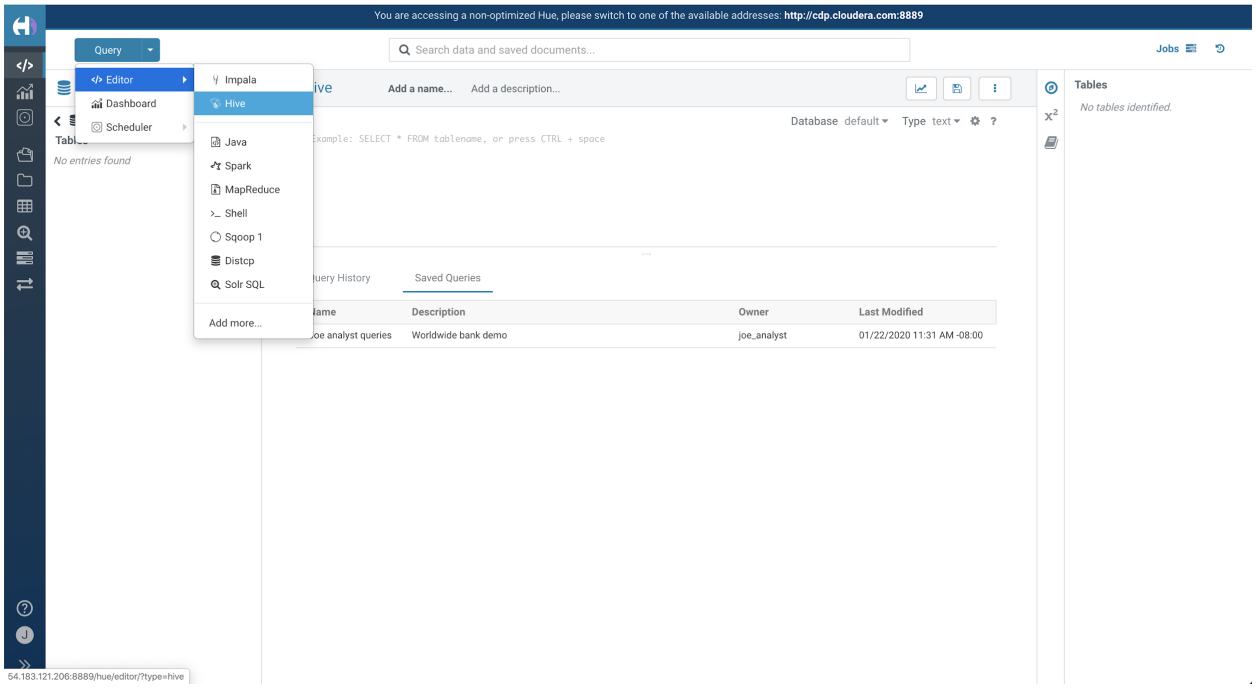


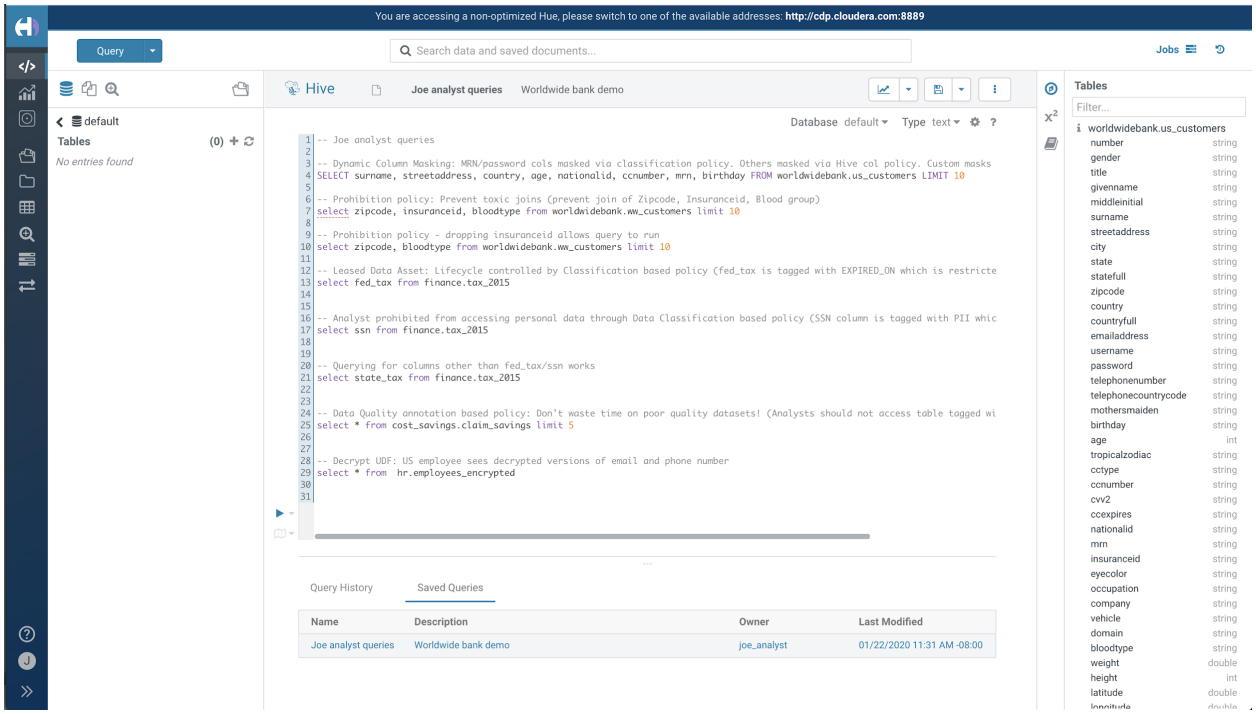
This notebook shows how an admin would handle GDPR scenarios like below using Hive ACID capabilities

- when a customer withdraws consent (so they no longer appear in searches)
- when a customer requests their data to be erased

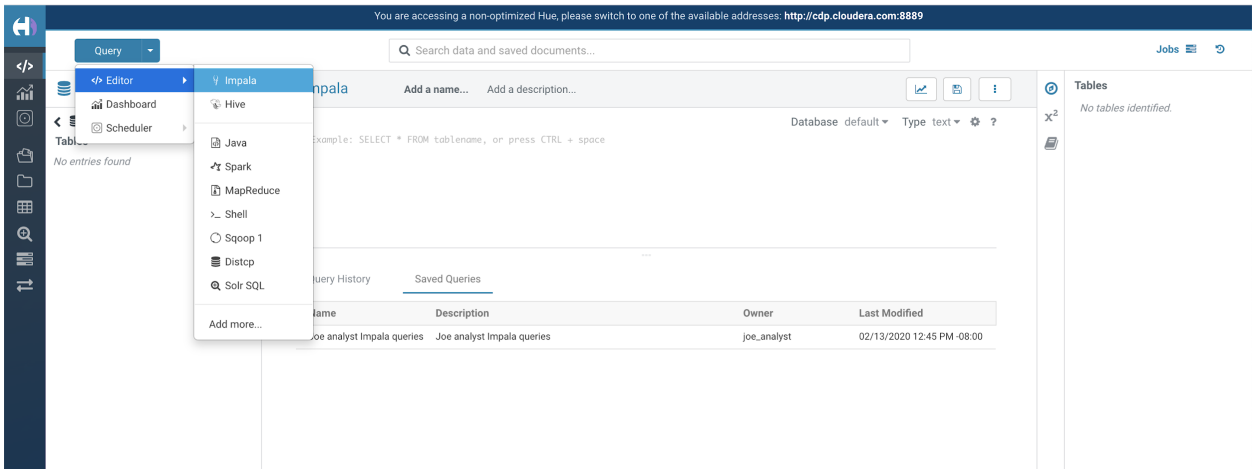
Run Hive/Impala queries from Hue

Alternatively you can login to Hue as joe_analyst and select Query > Editor > Hive and click "Saved queries" to run Joe's sample queries via Hive:

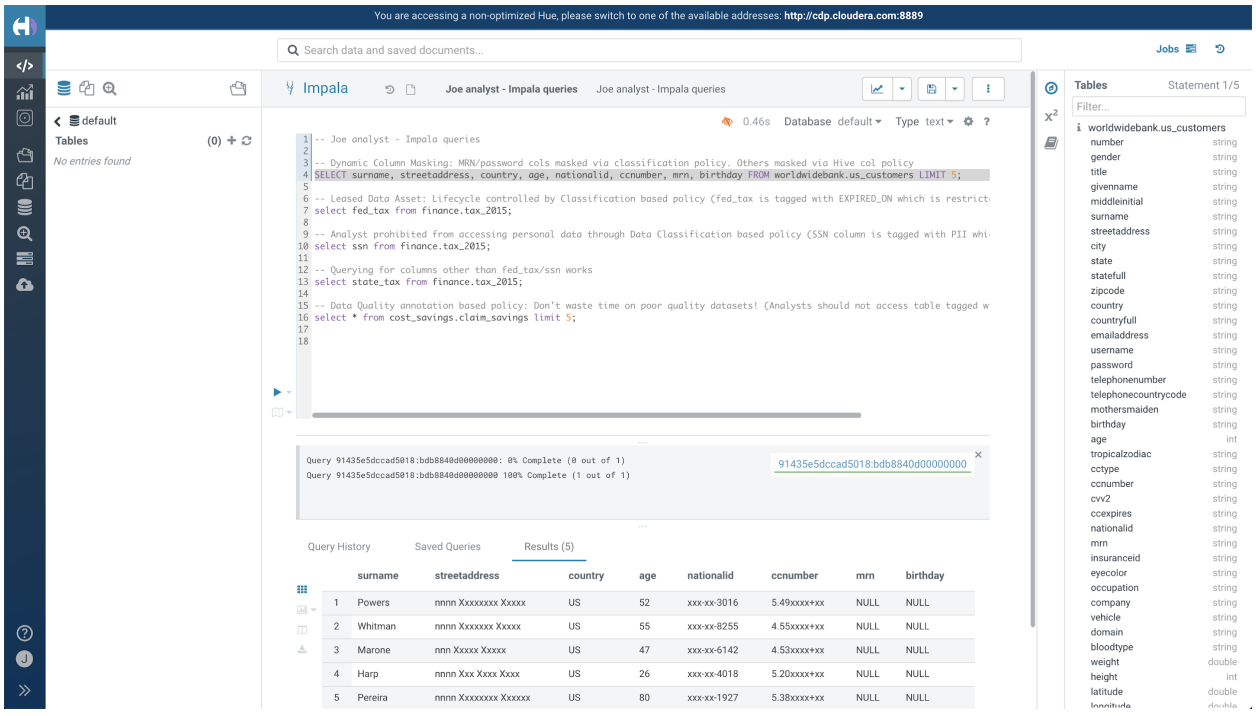




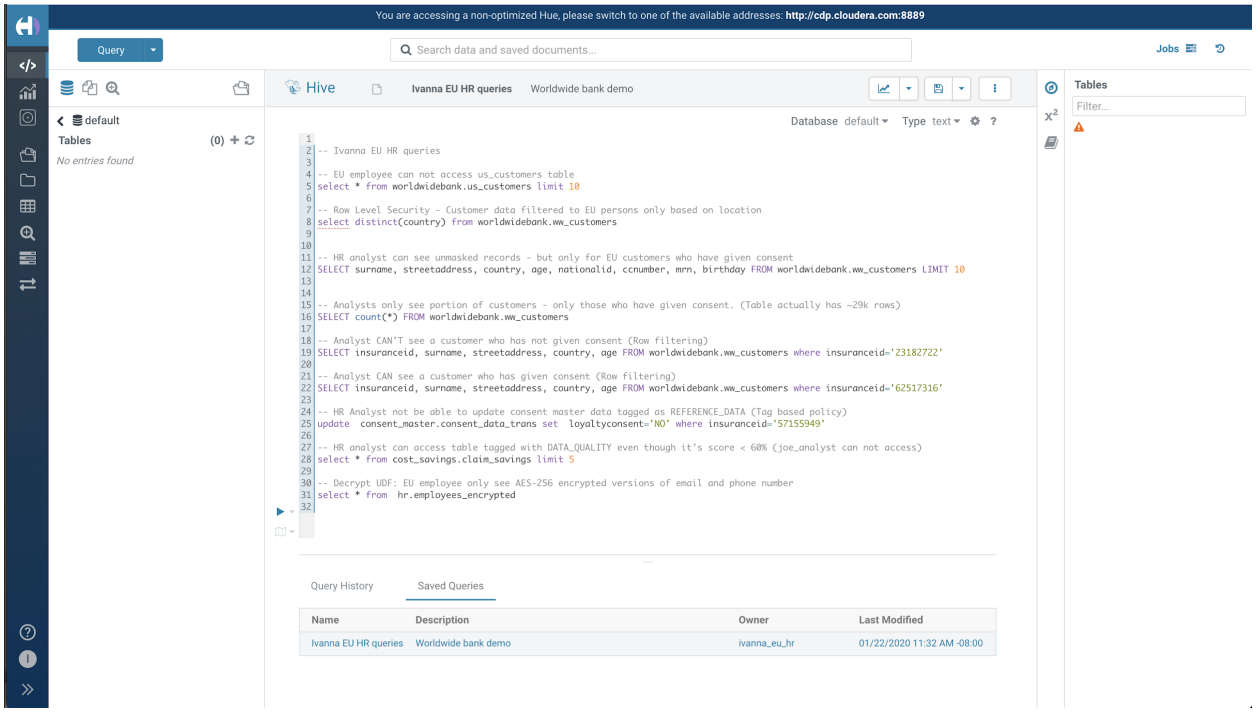
You can also switch the editor to Impala to run Joe's sample queries via Impala to show tag based access policy working for Impala:



In 7.1.x, Impala also supports column based masking



Alternatively you can login to Hue as ivanna_eu_hr and click "Saved queries" to run Ivanna's sample queries via Hive:



Run SparkSQL queries via HWC

To run secure SparkSQL queries (using Hive Warehouse Connector)

- connect to instance via SSH using your keypair
- authenticate as the user you want to run queries as via keytabs
  - `kinit -kt /etc/security/keytabs/joe_analyst.keytab joe_analyst/$(hostname -f)@CLOUDERA.COM`
- start SparkSql using HiveWarehouseConnector

```
spark-shell --jars
    /opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-assembly*.jar
    --conf spark.sql.hive.hiveserver2.jdbc.url="jdbc:hive2://$(hostname
    -f):10000/default;"    --conf
    "spark.sql.hive.hiveserver2.jdbc.url.principal=hive/$(hostname
    -f)@CLOUDERA.COM"    --conf
    spark.security.credentials.hiveserver2.enabled=false
```

- import HWC classes and start session
  - `import com.hortonworks.hwc.HiveWarehouseSession`
  - `import com.hortonworks.hwc.HiveWarehouseSession._`
  - `val hive = HiveWarehouseSession.session(spark).build()`

- run queries using hive.execute() e.g.
  - `hive.execute("select * from cost_savings.claim_savings").show(10)`

- Sample script to automate above for joe_analyst here:
  - /tmp/masterclass/ranger-atlas/HortoniaMunichSetup/run_spark_sql.sh

Troubleshooting Zeppelin

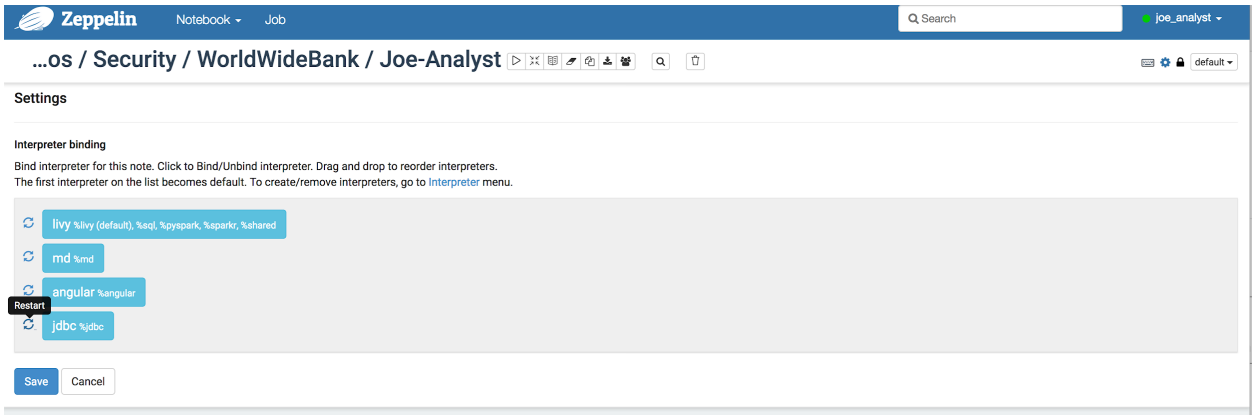
In case you encounter Thrift Exception below, it's likely the session was expired:

Prohibition policy - dropping insuranceid allows query to run

```
%jdbc(hive)
select zipcode, bloodtype from worldwidebank.ww_customers
limit 10
```

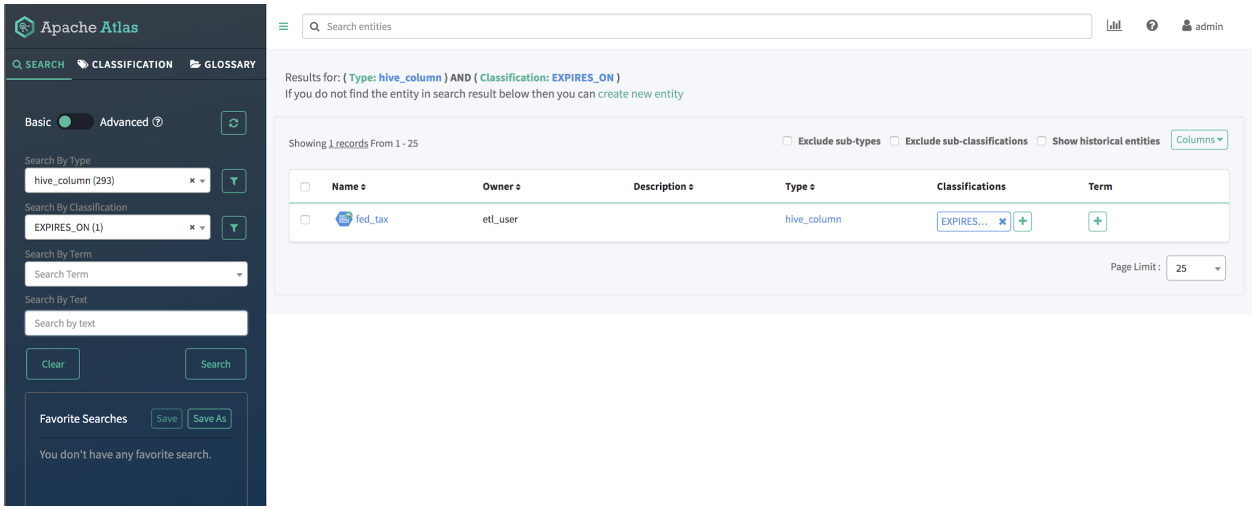
```
java.sql.SQLException: org.apache.hive.org.apache.thrift.TException: Error in calling method ExecuteStatement
        at org.apache.hive.jdbc.HiveStatement.runAsyncOnServer(HiveStatement.java:334)
        at org.apache.hive.jdbc.HiveStatement.execute(HiveStatement.java:265)
        at org.apache.commons.dbcp2.DelegatingStatement.execute(DelegatingStatement.java:291)
        at org.apache.commons.dbcp2.DelegatingStatement.execute(DelegatingStatement.java:291)
        at org.apache.zeppelin.jdbc.JDBCInterpreter.executeSql(JDBCInterpreter.java:718)
        at org.apache.zeppelin.jdbc.JDBCInterpreter.interpret(JDBCInterpreter.java:801)
        at org.apache.zeppelin.interpreter.LazyOpenInterpreter.interpret(LazyOpenInterpreter.java:103)
        at org.apache.zeppelin.interpreter.remote.RemoteInterpreterServer$InterpretJob.jobRun(RemoteInterpreterServer.java:633)
        at org.apache.zeppelin.scheduler.Job.run(Job.java:188)
        at org.apache.zeppelin.scheduler.ParallelScheduler$JobRunner.run(ParallelScheduler.java:162)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(ScheduledThreadPoolExecutor.java:180)
```

Just scroll to the top and click the gears icon (near top right) to display the interpreters and restart the jdbc one

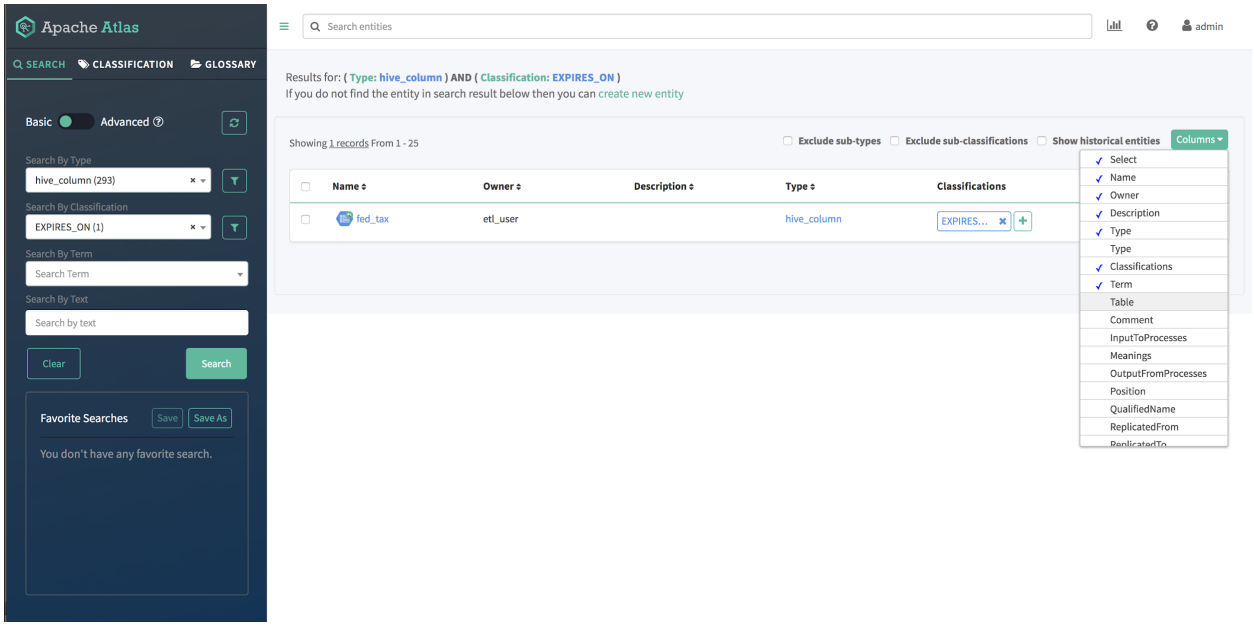


Atlas walk through

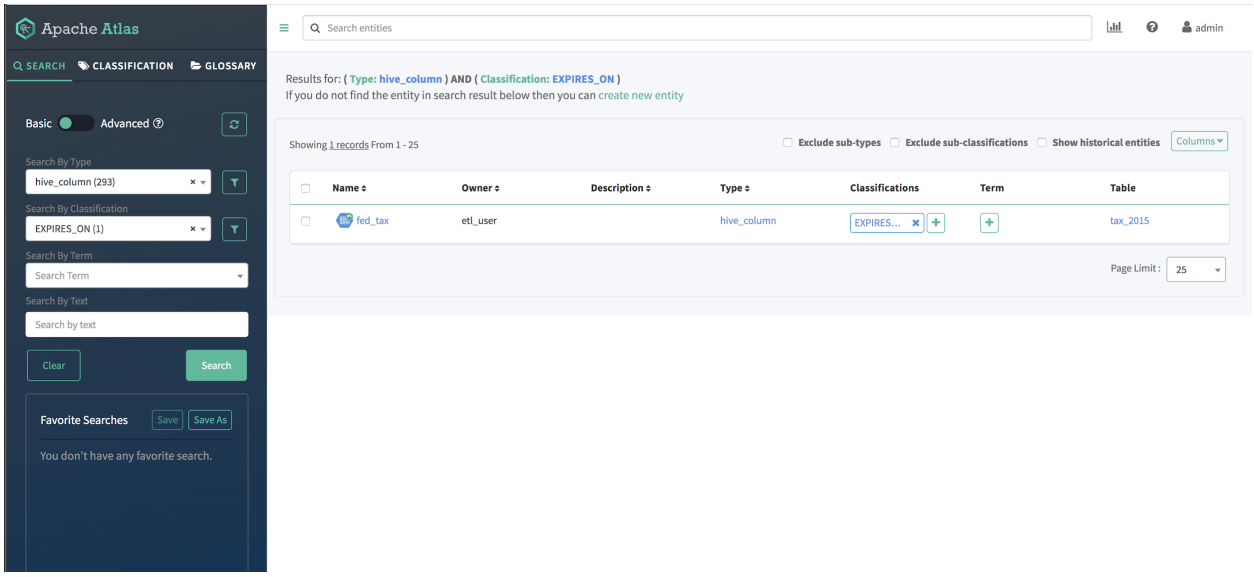
9. Login to Atlas and show the Hive columns tagged as EXPIRES_ON



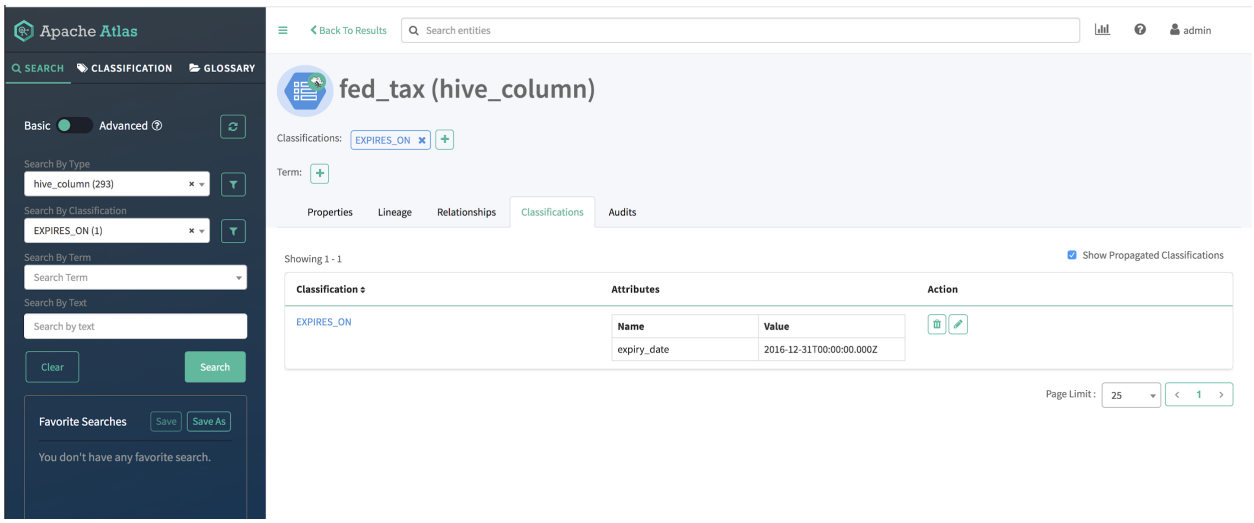
To see the table name, you can select Table in the Column dropdown



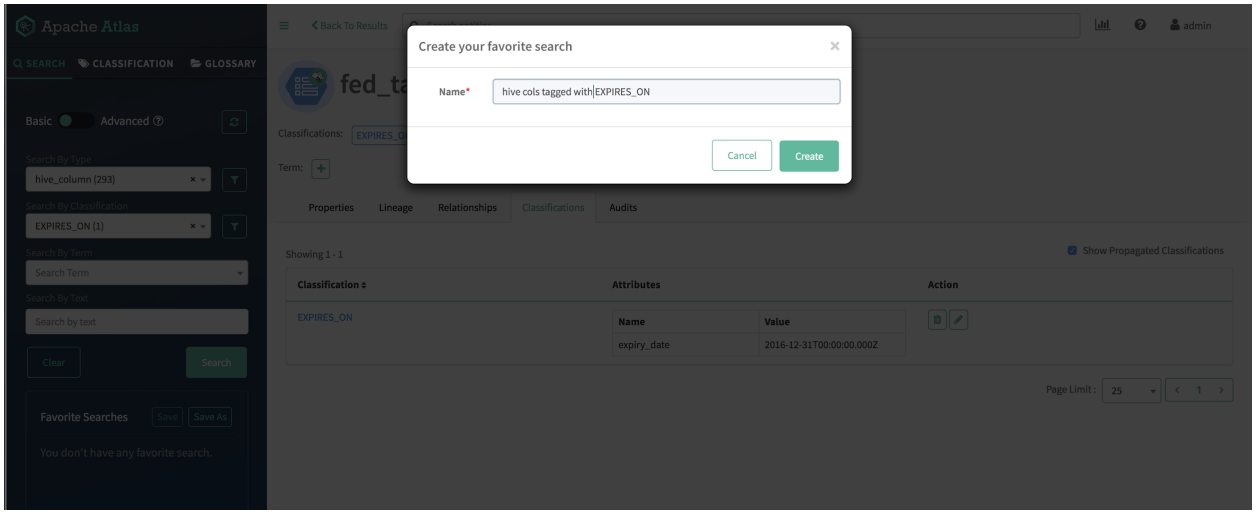
Now notice the table name is also displayed



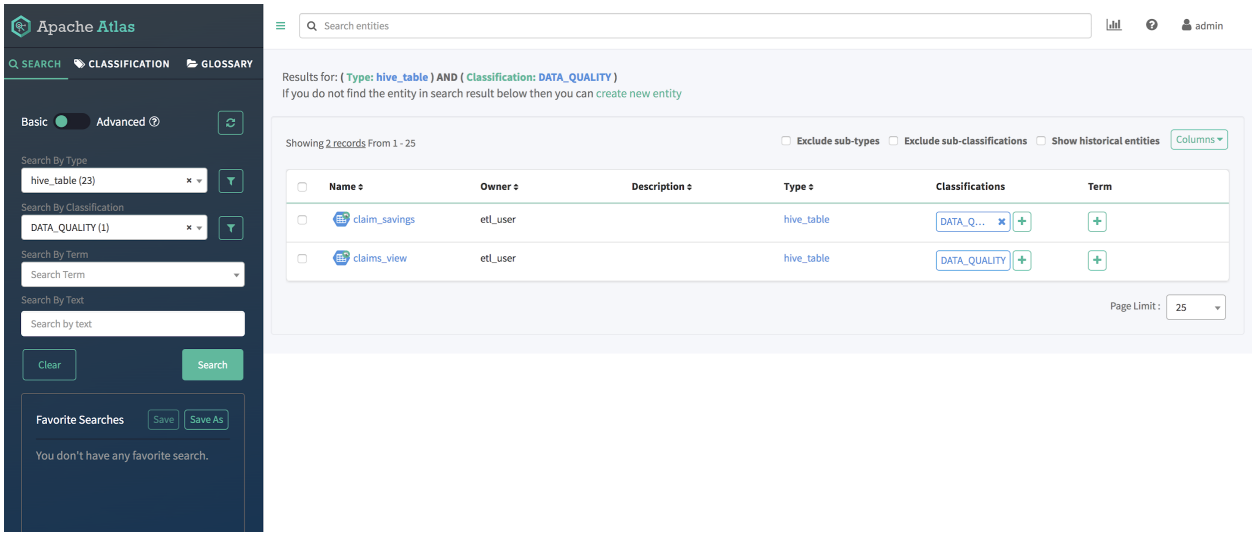
Selecting the fed_tax column and opening the 'Classifications' tab shows the attributes of tag (expiry_date) and value



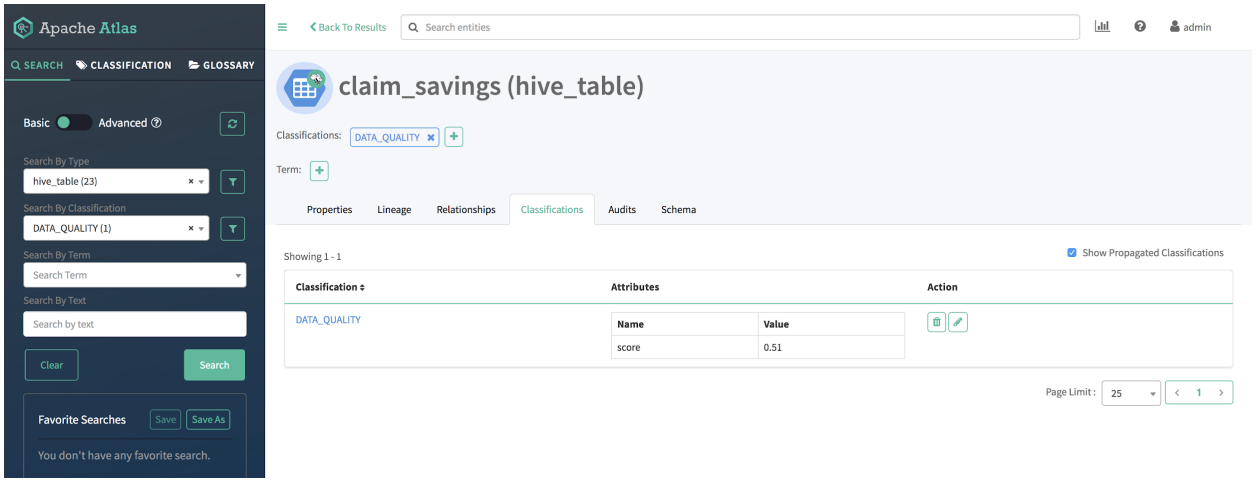
To save this search, you can click the “Save As” button near bottom left:



Similarly you can query for hive tables tagged with DATA_QUALITY...

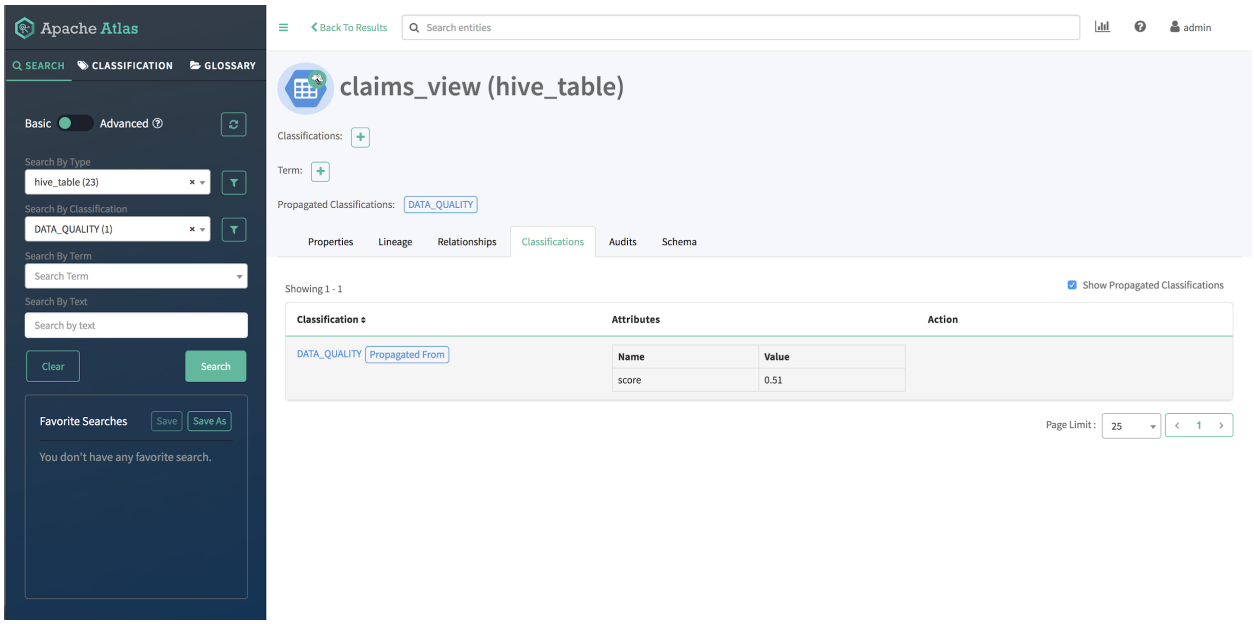


...and click on claim_savings to see that the quality score associated with this table is less than 60%

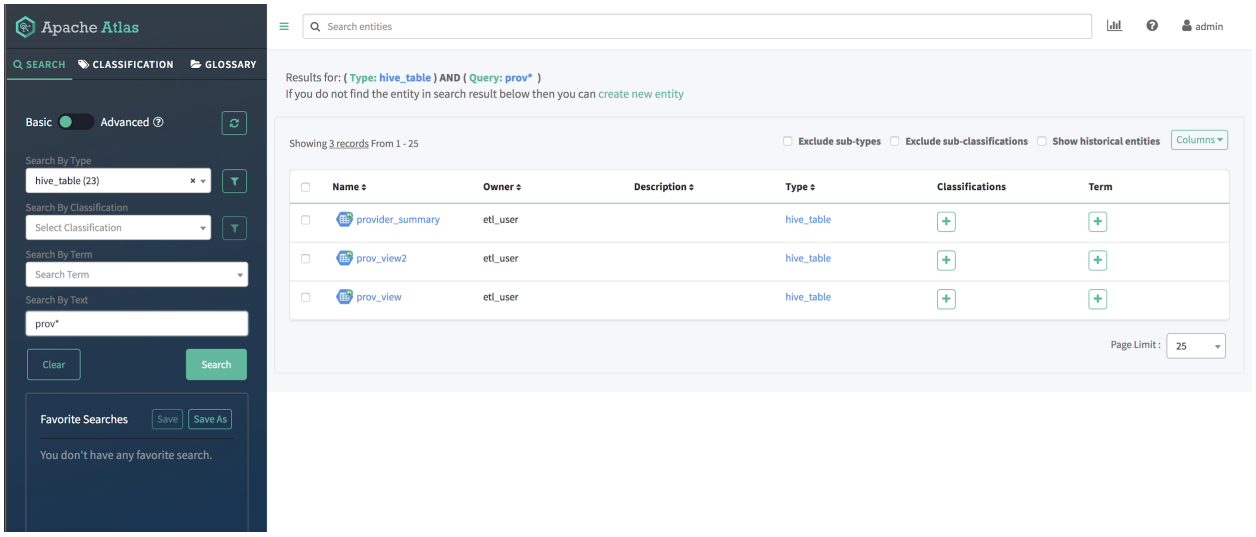


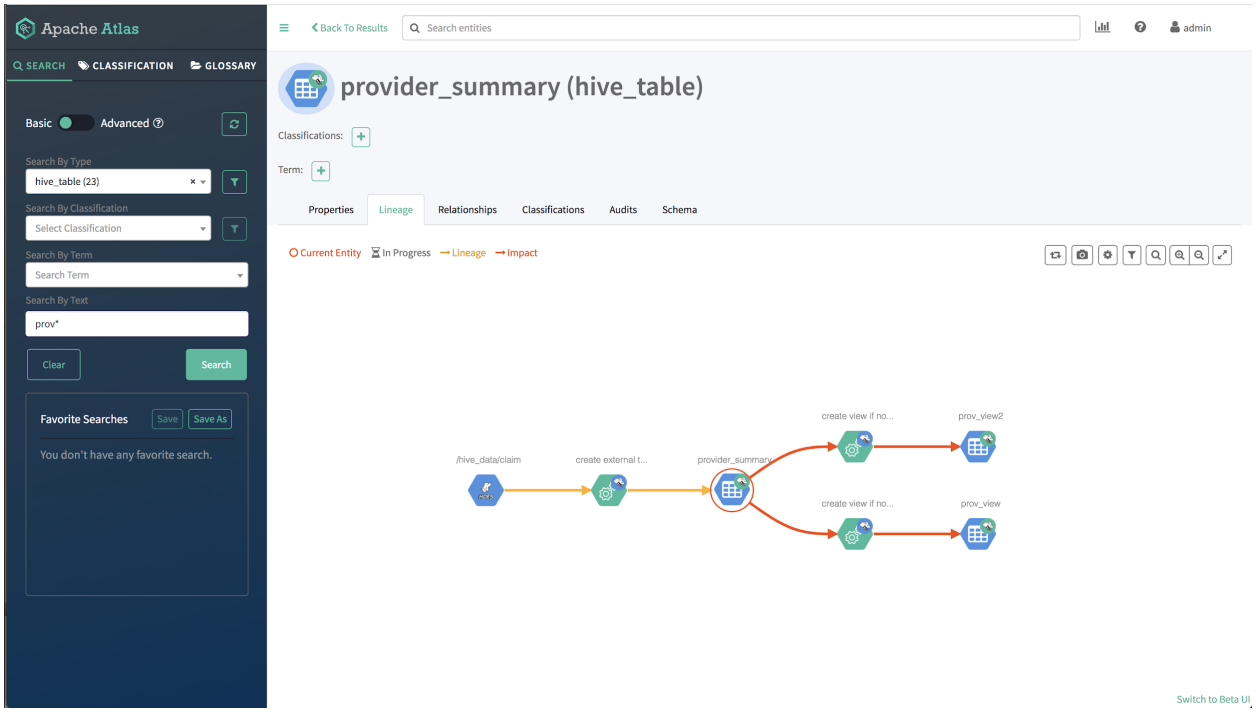
Click back, and select claims_view table instead and click the lineage tab. This shows that this table was derived from the claims_saving table

Click on the Classifications tab and notice that because the table claims_view table was derived from (claims_savings) had a DATA_QUALITY tag, the tag was automatically propagated to claims_view table itself (i.e. no one had to manually tag it)

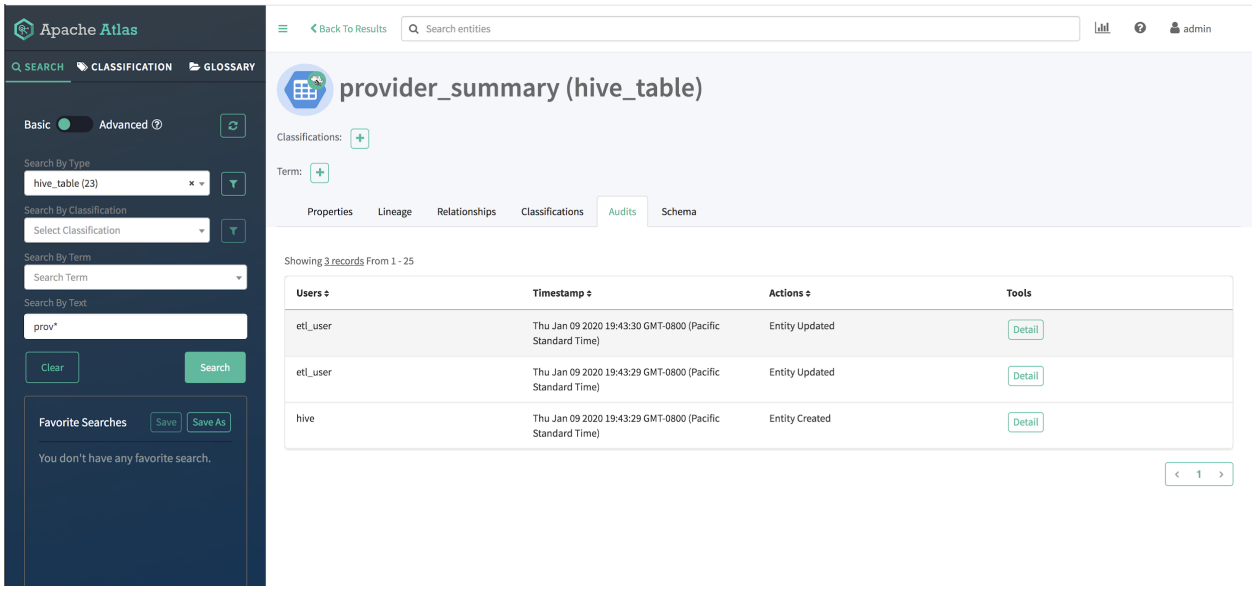


10. Use Atlas to query for hive_tables and pick provider_summary to show lineage and impact

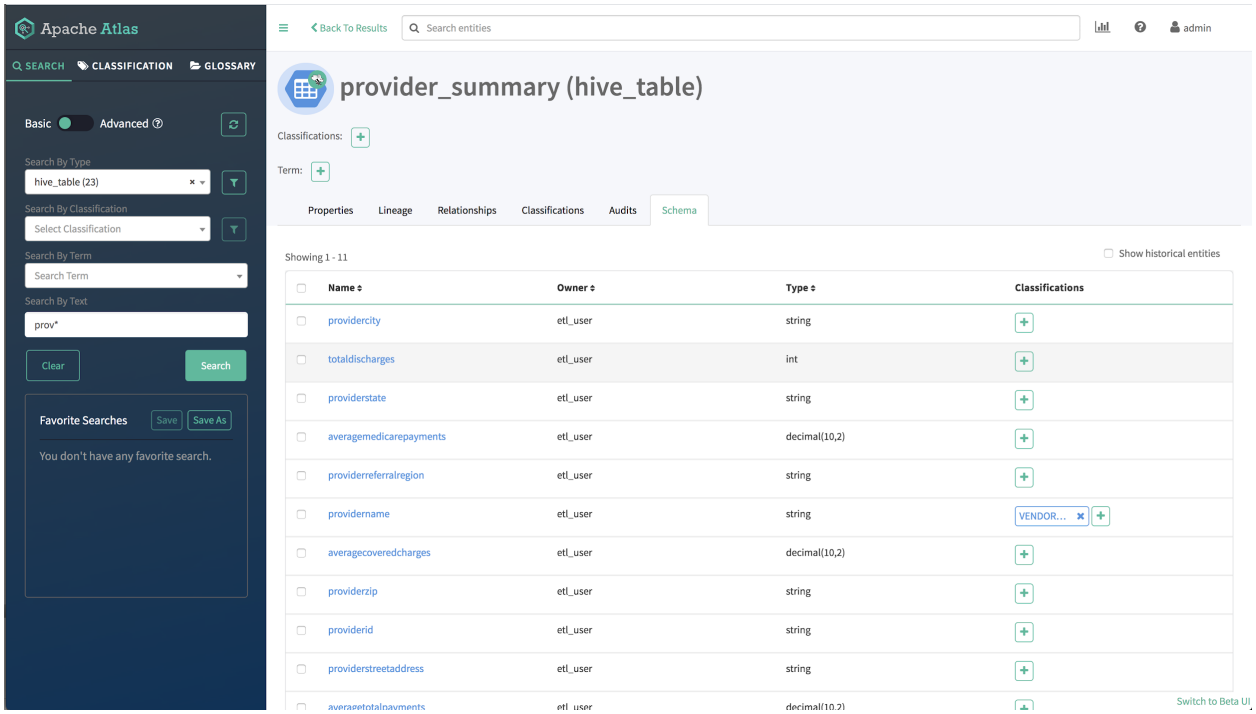




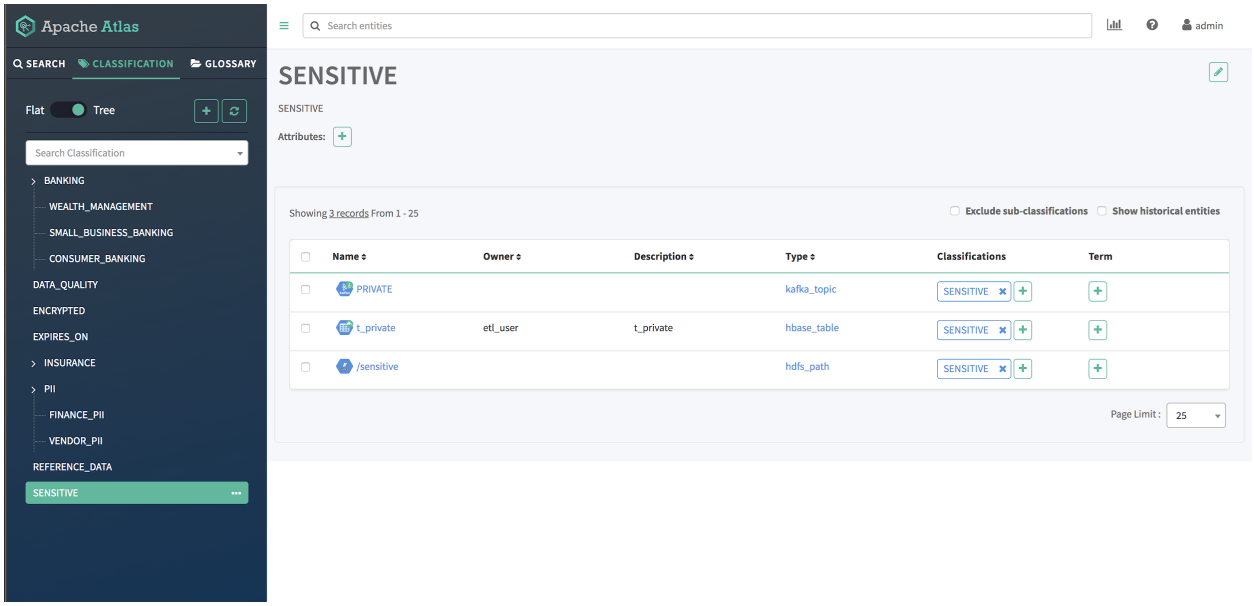
You can use the Audits tab to see audits on this table

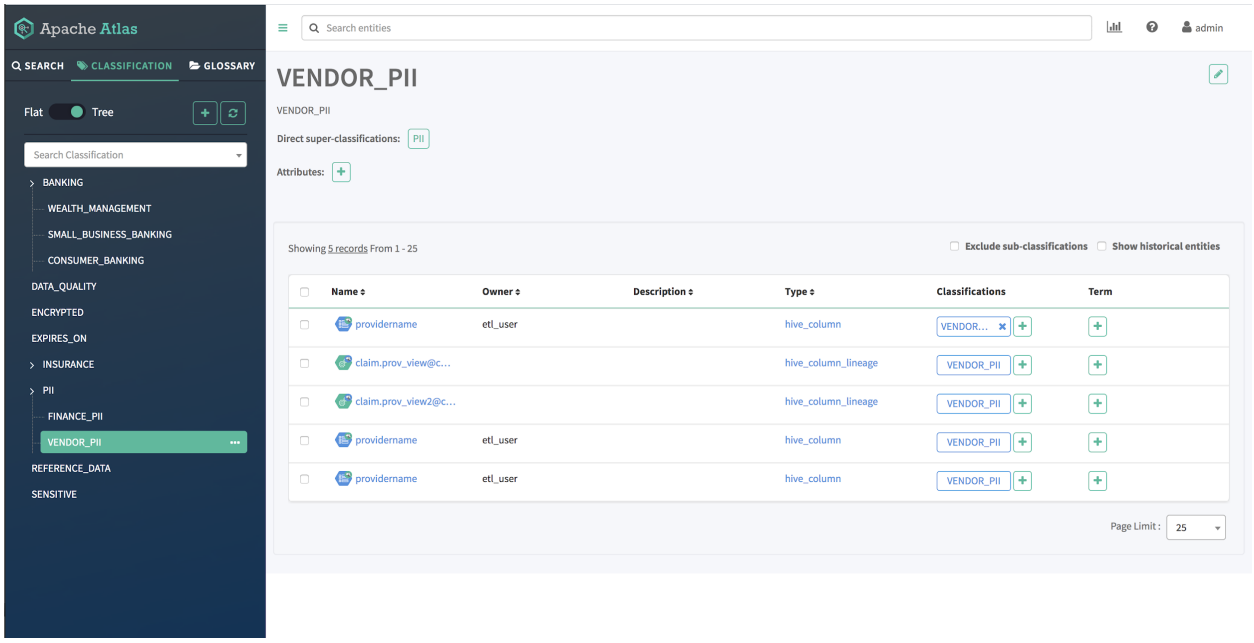


You can use Schema tab to inspect table schema

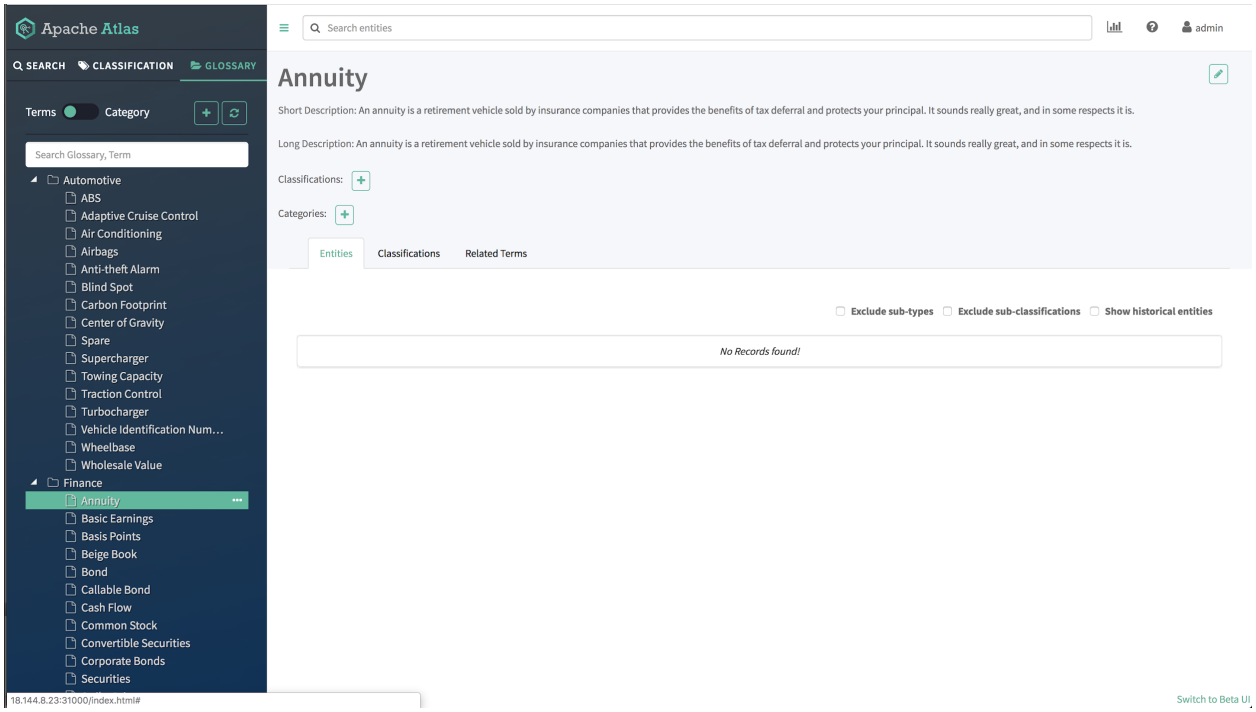


11. Navigate to the Classification tab to see how you can easily see all entities tagged with a certain classification (across Hive, Hbase, Kafka, HDFS etc)

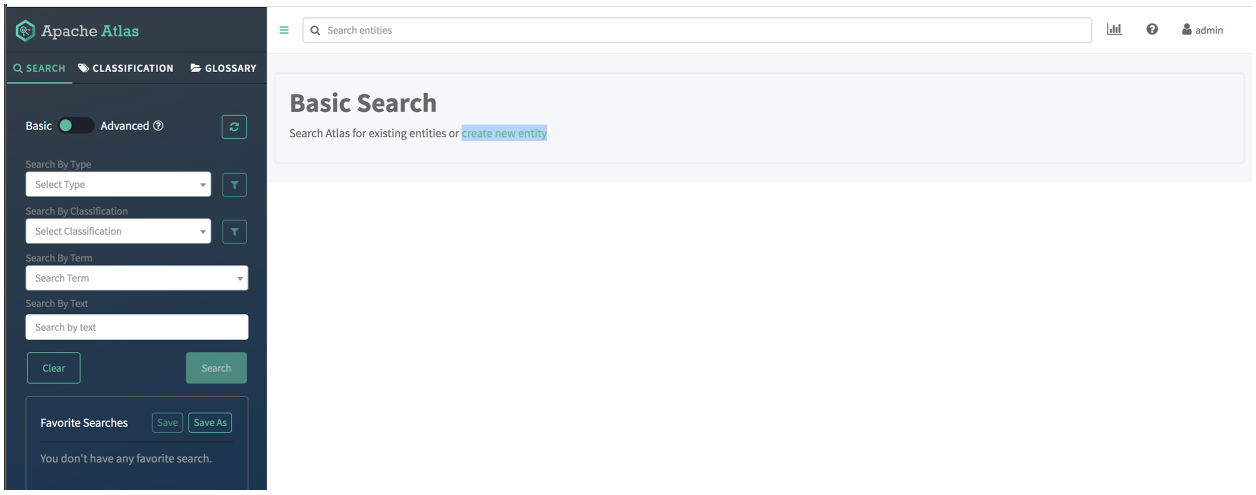




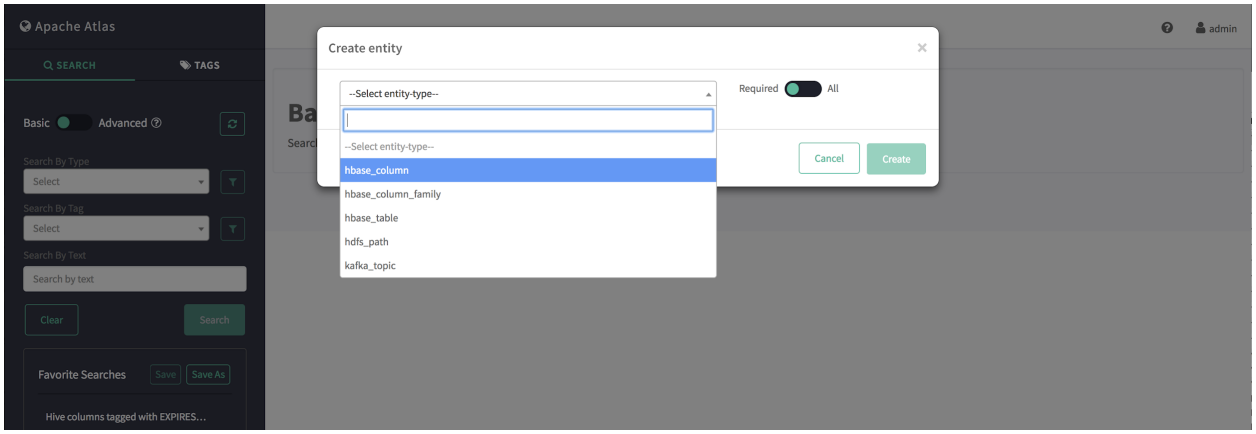
Navigate to the Glossary tab to see how you can define Glossary categories and terms, as well as search for any entities associated with those terms:



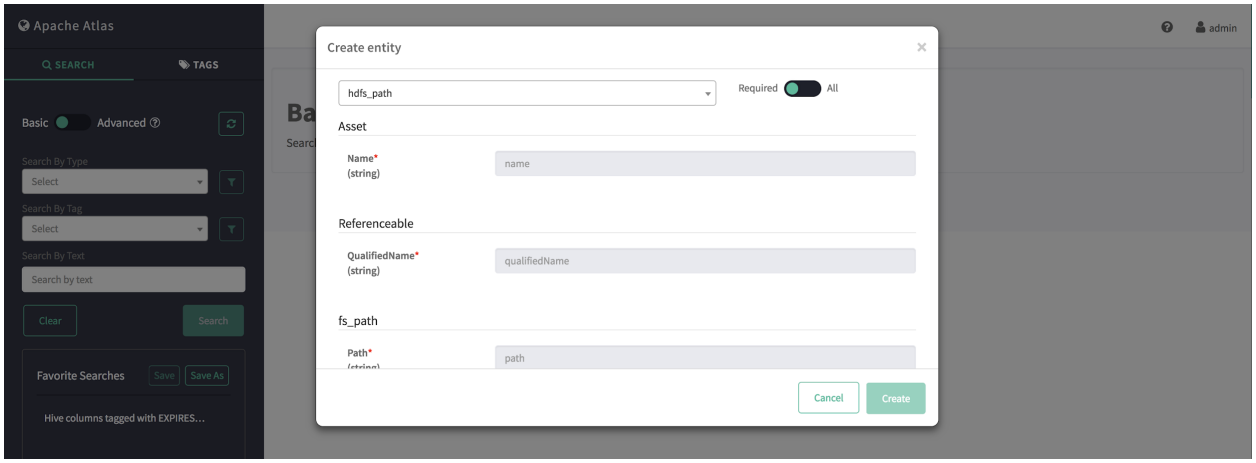
12. Navigate to Atlas home page and notice the option to create a new entity



Sample out of the box entity types that you can create shown below:

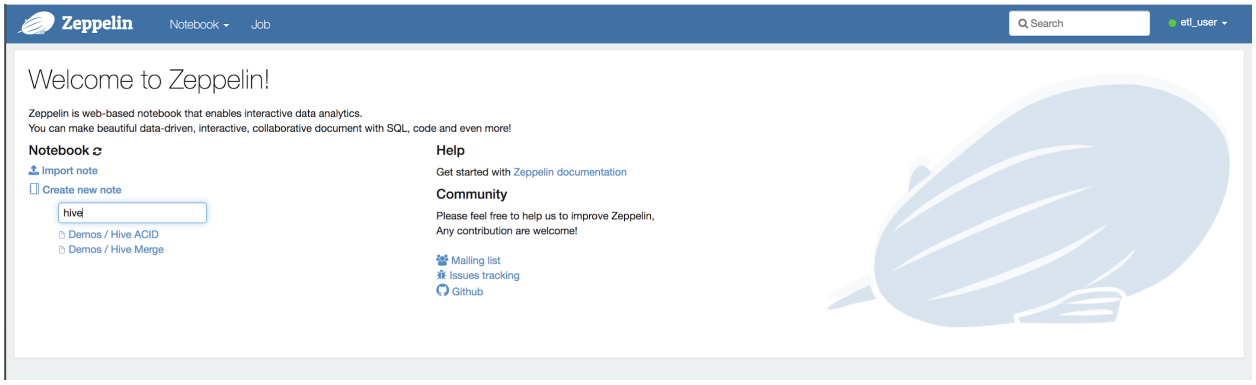


Selecting an entity type (e.g. hdfs_path), shows what required and optional fields you would need to provide to manually create the new entity

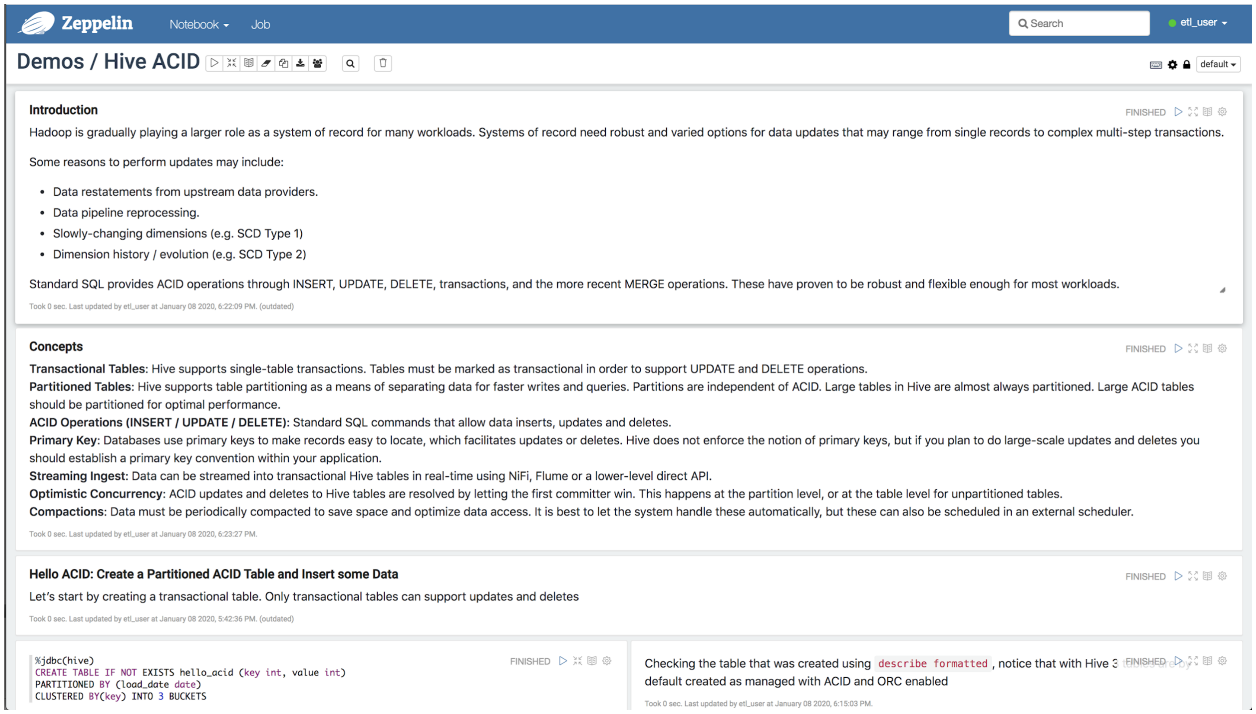


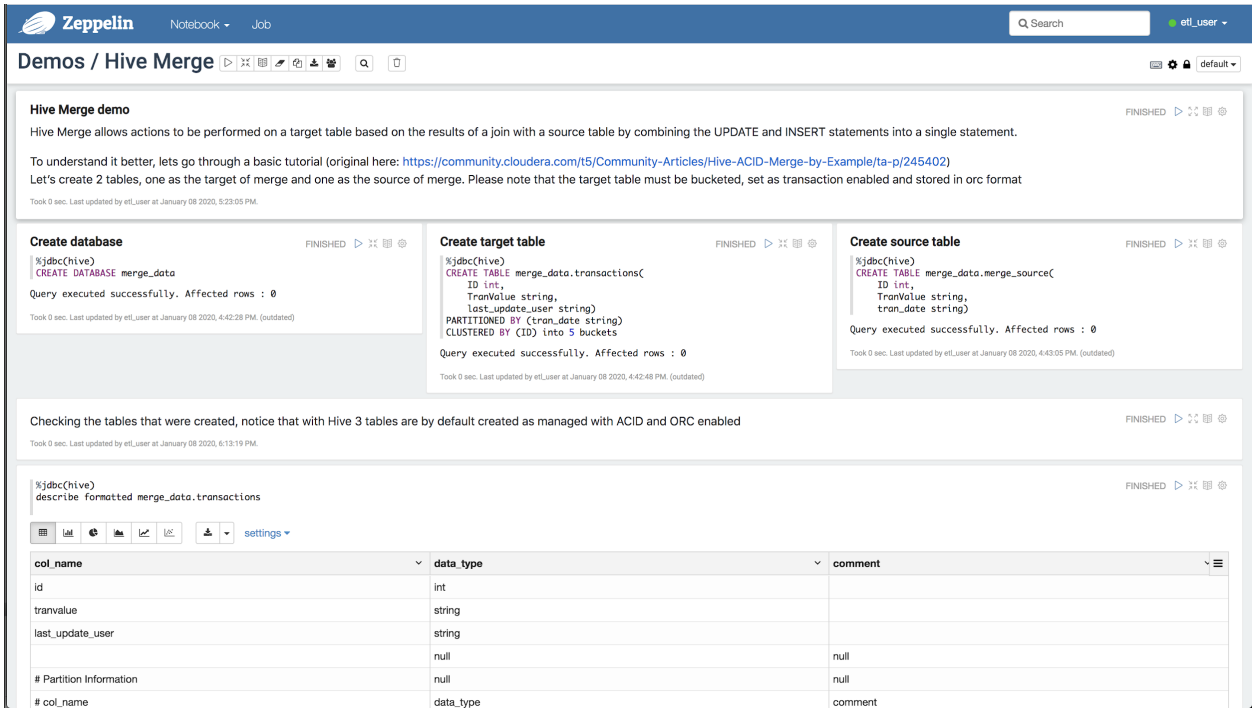
Hive ACID/Merge walk through

13. In Zeppelin there are two Hive related notebooks provided to demonstrate Hive ACID and MERGE capabilities. Login to Zeppelin as etl_user to be able to run these:



The notebooks contain tutorials that walk through some of the theory and concepts before going through some basic examples:





Appendix: Older AMIs

Older AMI links (for HDP releases) can be found below:

- For HDP 3.1.4: click here
- For HDP 3.1 with Knox SSO: click here
- For HDP 2.6.5 with Knox SSO: click here
- For HDP 2.6.5: click here