To properly configure Kafka settings in your atlas-application.properties file, ensure the atlas.kafka settings reflect your Kafka server details correctly. Since you've set the Kafka listener on localhost:9092, you should update the Kafka-related properties as follows:

**Adjusted Kafka settings in atlas-application.properties:**

**2. Kafka Zookeeper Configuration for Atlas:**

**atlas.kafka.bootstrap.servers=localhost:9092 # Kafka server and port updated**

atlas.kafka.zookeeper.session.timeout.ms=400

atlas.kafka.zookeeper.connection.timeout.ms=200

atlas.kafka.zookeeper.sync.time.ms=20

atlas.kafka.auto.commit.interval.ms=1000

atlas.kafka.hook.group.id=atlas

**Key points:**

1. atlas.kafka.bootstrap.servers should point to localhost:9092, which is your Kafka server address.
2. The rest of the Kafka properties like auto.commit.interval.ms, auto.offset.reset, and session.timeout.ms are standard and can be left as-is or adjusted based on your specific needs.

**Step2 :Update Kafka Zookeeper Settings**:

* Ensure the Kafka Zookeeper connection is pointing to the correct Zookeeper server and port. Example:

atlas.kafka.zookeeper.connect=localhost:2181 # Pointing to Zookeeper default port

**Step 3: Verify Kafka and HBase Configuration**

1. **Check Kafka Zookeeper Connection**:
   * Kafka relies on Zookeeper for managing its cluster, so ensure Kafka can connect to Zookeeper. In your Kafka configuration (config/server.properties), confirm that Zookeeper's address is correct:

zkCli.sh -server <zookeeper\_host>:2181

**Step 5: Verify Zookeeper Connectivity**

1. **Test Zookeeper with CLI**:
   * You can use the Zookeeper command-line interface (zkCli.sh) to ensure it’s accessible:

bash

zkCli.sh -server localhost:2181

restart kafka :

$KAFKA\_HOME/bin/kafka-server-start.sh $KAFKA\_HOME/config/server.properties

**Steps to Configure HBase Integration with Apache Atlas:**

-Ensure HBase is Configured Correctly in Hadoop:

hbase shell

status

-Edit the atlas-application.properties file:

vi /opt/apache-atlas-2.4.0/conf/atlas-application.properties

-add or update the following properties to enable the HBase hook:

properties

# HBase related configuration for Atlas hook

atlas.hbase.hook.enabled=true

atlas.hbase.zk.quorum=localhost:2181 # Zookeeper quorum

atlas.hbase.zk.client-port=2181 # Zookeeper client port

atlas.hbase.table.metadata.namespace=atlas # Optional: define namespace for metadata storage

**:Copy hbase-site.xml to Atlas Configuration**

1. **Copy the hbase-site.xml file from your HBase configuration directory to the Apache Atlas conf/hbase directory:**

**cp /opt/hbase-2.5.5-hadoop3/conf/hbase-site.xml /opt/apache-atlas-2.4.0/conf/hbase/**

Ensure that the Atlas conf/hbase directory contains the hbase-site.xml file. If you want, you can verify the configuration content by opening it:

nano /opt/apache-atlas-2.4.0/conf/hbase/hbase-site.xml

Steps to disable the embedded HBase in Atlas:

**Edit the atlas-env.sh file**:

* Locate the atlas-env.sh file in the Atlas configuration directory (/opt/apache-atlas-2.4.0/conf/).
* Open it and find the variable ATLAS\_EMBEDDED\_HBASE. If it is set to true, change it to false.

ATLAS\_EMBEDDED\_HBASE=false

**Restart Atlas**:

* After disabling the embedded HBase and configuring the external one, restart Apache Atlas to apply the changes:

./bin/atlas\_stop.py

./bin/atlas\_start.py

To verify if **Apache Atlas** is actually using the **HBase instance from Hadoop**, you can follow these steps:

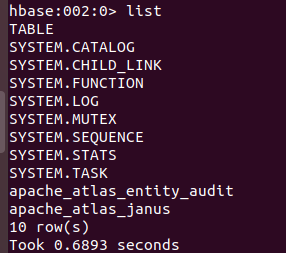
**1. Check the HBase Tables Used by Atlas**

**Atlas stores metadata in HBase. You can check if the expected tables are created in your external HBase instance by running the following command:**

**hbase shell**

**Then, list all tables to see if Atlas-related tables exist:**

**List**



✅ **2. Check Atlas Logs for HBase Connection**

If Atlas is correctly using your **external HBase**, there should be no errors related to HBase in the Atlas logs.

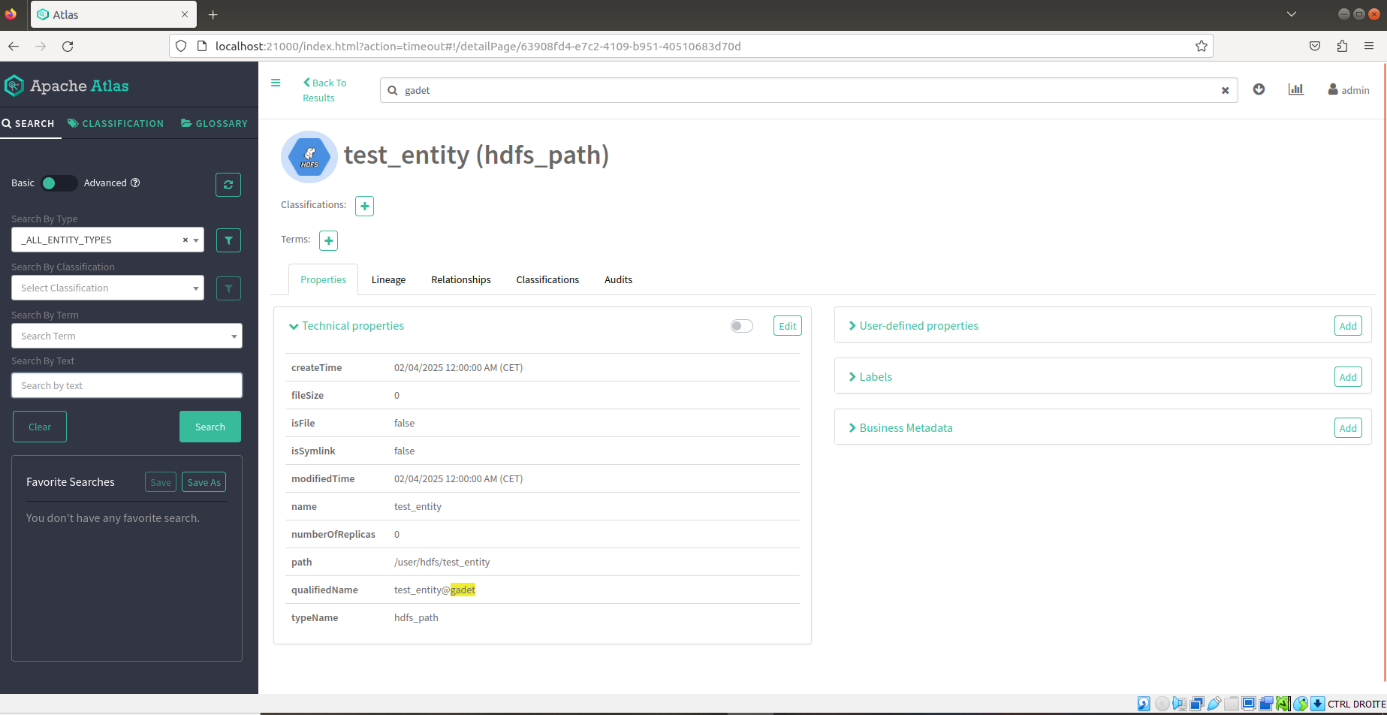
tail -f /opt/apache-atlas-2.4.0/logs/application.log

loook for logs indicating a successful connection to HBase, such as:

INFO - Successfully connected to HBase at hdfs://node:9000/hbase

**✅ 3. Create a Metadata Entity in Atlas and Check HBase**

* In the **Atlas UI**, manually create a metadata entity (e.g., a new database/table).
* Check HBase (hbase shell -> scan 'ATLAS\_ENTITY\_AUDIT\_EVENTS') to see if it reflects the changes.



**Create a Test Entity in Apache Atlas**

You can do this via the **Atlas UI** or **API**:

**🔹 Using UI** (Easiest)

* Go to **Apache Atlas Web UI** → **Create a new entity** (e.g., a "Table" under hive\_table).
* Save it.

**🔹 Using API** (If you prefer CLI)  
Run this curl command to create a dummy entity:

* bash
* CopyEdit
* curl -X POST -u admin:admin -H "Content-Type: application/json" -d '{
* "entities": [
* {
* "typeName": "hive\_table",
* "attributes": {
* "qualifiedName": "test\_table@default",
* "name": "test\_table"
* }
* }
* ]
* }' <http://localhost:21000/api/atlas/v2/entity>

**2️⃣ Check the HBase Audit Table Again**

Run this in HBase shell:

scan 'apache\_atlas\_entity\_audit'

If you see new rows, **Atlas is successfully using your external HBase** 🎉. If not, we might need to check logs or configs.

