

Compatibility Overview & Constraints

You are trying to make Scala + Hadoop + Hive + Spark all work together. The key things to coordinate are:

1. **Scala version** — must match (or be compatible with) the version Spark is built with
2. **Spark version** — needs to support Hive metastore, integrate with Hadoop
3. **Hive version** — the version of Hive metastore / SerDes / UDFs that Spark can interoperate with
4. **Hadoop version** — the version of HDFS / YARN / core libraries
5. **Java (JDK/JRE) version** — everything runs on the JVM, so Java version compatibility is critical

Mismatches in any one of these often lead to `NoClassDefFoundError`, method signature mismatches, or runtime module / reflection issues.

Recommended Combination (as of 2025)

Based on what is stable, well-supported, and less likely to run into nasty runtime issues, here is a recommended “stack”:

Component	Recommended Version / Range	Reason / Notes
Scala	2.12.x (or 2.13.x in some newer builds)	Spark 3.x is commonly built with Scala 2.12 (and newer Spark versions may also support 2.13). docs.qubole.com+3community.cloudera.com+3downloads.apache.org+3
Spark	3.2.x / 3.3.x / 3.4.x	These versions are mature, widely used, and have good support for Hive metastore compatibility.
Hive	Hive 2.3.x (or compatible with Spark's Hive bindings)	Spark SQL is designed to be compatible with Hive versions (metastore, SerDes, UDFs) especially around Hive 1.2 → 2.3 range. archive.apache.org+1
Hadoop	Hadoop 3.x (for example Hadoop 3.3.x)	Many Spark builds target Hadoop 3.x; ensures newer HDFS / YARN features.
Java (JDK/JRE)	Java 8 or Java 11 (or Java 17 with caution)	Spark supports Java 8, 11, and in recent versions also 17. issues.apache.org+3community.cloudera.com+3spark.apache.org+3 Scala (2.12.15+ / 2.13.6+) supports JDK 17. Scala Documentation

So a concrete example that is fairly safe would be:

- Scala **2.12.15**
- Spark **3.3.4** (built for Scala 2.12)
- Hive **2.3.9** (or another 2.3.x)
- Hadoop **3.3.1** (or a 3.x release that matches your cluster)
- Java **11**

This combination gives a good balance. Java 8 also works and is “classic,” but Java 11 is more future-oriented (LTS) and supported by recent Spark versions.

Caveats & Gotchas

- **Spark + Java 17 issues:** Some users have reported `IllegalAccessError` problems when running Spark 3.3+ on Java 17 (due to module / reflective access to certain `sun.nio.ch` internals) unless special JVM options (e.g. `--add-opens`) are used. [Stack Overflow+1](#)
- **Hive metastore version differences:** Spark includes a built-in Hive binding (often Hive 1.2.1 historically) but can talk to other Hive metastore versions. You have to ensure the classpath and versions used for the Hive client / metastore libraries align. [archive.apache.org+1](#)
- **Scala version mismatches:** If you compile your application in Scala 2.13 but your Spark cluster expects 2.12, you will see linkage errors. Always align the Scala binary version. [Stack Overflow+1](#)
- **Hadoop / Spark binary builds:** When you download Spark, often the binary is pre-built against a particular Hadoop version (e.g. “Prebuilt for Hadoop 3.2 and later”). Make sure the Spark binary you get is compatible with your Hadoop version.
- **Classpath conflicts:** Multiple versions of Hive / Hadoop jars on classpath cause conflicts. You’ll need to manage dependencies carefully (especially for Hive / Hadoop client jars).
- **Java module restrictions in newer JDKs:** With JDK 17+, stricter module encapsulation means some internal APIs are blocked unless you explicitly add `--add-exports` or `--add-opens` flags.

Final Suggestion & Version Summary

To “make everything work to gather (collect, query, process)” with minimal friction, I’d suggest:

- Scala: **2.12.15**
- Spark: **3.3.4** (or 3.4.x) built for Scala 2.12
- Hive: **2.3.x** (e.g. 2.3.9)
- Hadoop: **3.3.x**
- Java: **11**

If you prefer maximum stability / widest compatibility, you might also use Java 8 (though newer features are limited). But Java 11 is a safe middle ground for more modern setups.

If you like, I can check *exact version compatibility tables* (Hive ↔ Spark ↔ Hadoop) for your target cluster (e.g. version numbers you plan) to give you the precise combo. Do you have particular versions in mind already (e.g. Hadoop 3.2.x or Hive 3.x)?

You said:

hadoop 3.3.6

ChatGPT said:

Got it 👍 You want to align everything around **Hadoop 3.3.6**.

That narrows things nicely, since Spark releases come pre-built against Hadoop **3.3+**.

Recommended Stack Around Hadoop 3.3.6

Component	Version
Hadoop	3.3.6
Spark	3.3.4 (pre-built for Hadoop 3.3+) OR Spark 3.4.x (also compatible)
Scala	2.12.x (e.g. 2.12.15 or 2.12.18) → matches Spark 3.3/3.4 default builds
Hive	2.3.x (e.g. 2.3.9) → stable with Spark’s Hive support
Java (JDK)	Java 11 (recommended) OR Java 8 (classic)

Why This Works

- **Spark ↔ Hadoop**
Spark **3.3.x** and **3.4.x** binaries are distributed as “prebuilt for Hadoop 3.3 and later” → perfect fit for **Hadoop 3.3.6**.
- **Scala ↔ Spark**
Spark 3.3.x and 3.4.x are compiled against **Scala 2.12**, with optional support for Scala 2.13 in newer builds.
To avoid headaches, stick with **Scala 2.12.x**.
- **Hive ↔ Spark**
Spark has built-in Hive support (historically Hive 1.2.1), but it can integrate with **Hive 2.3.x metastore** without issues.
Many production setups pair Spark 3.3 + Hive 2.3.9.
- **Java ↔ Spark/Scala**
 - Java 8: Always safe.
 - Java 11: Fully supported by Hadoop 3.3.x, Spark 3.3/3.4, Scala 2.12.
 - Java 17: Works in Spark 3.3+ but may need --add-opens flags (due to module access restrictions).

✅ So the clean combo with **Hadoop 3.3.6** is:

- **Hadoop 3.3.6**
- **Spark 3.3.4** (pre-built for Hadoop 3.3)
- **Scala 2.12.15** (to match Spark build)
- **Hive 2.3.9**
- **Java 11**

⚙️ Your Current Base

Component	Version
Hadoop	3.3.6
Spark	3.3.4 (Pre-built for Hadoop 3.3+)
Scala	2.12.15
Java	11
Hive	<i>(to be decided → 3.1.3 or 2.3.9)</i>

🧩 Hive 2.3.x vs Hive 3.x — Compatibility Summary

Area	Hive 2.3.x	Hive 3.x (e.g. 3.1.3)	Spark 3.3.4 Compatibility
Spark SQL Integration	✅ Fully compatible (officially tested with Spark 3.3)	⚠️ <i>Partially compatible</i> — Spark 3.3.x not officially tested with Hive 3.x	
Metastore Version	Uses older schema (v2)	Upgraded schema (v3) → requires metastore migration	Spark still expects v2 schema unless configured
Transactional Tables (ACID)	Limited support	✅ Native ACID table support	Spark 3.3.4 can read , but not fully write ACID tables

Storage Format	ORC/Parquet	ORC/Parquet (enhanced)	Works fine if using external tables
Auth / Ranger Integration	Stable	Enhanced (with LLAP, Tez, Ranger)	Works, but Spark ignores Hive's LLAP features
Metastore DB (MySQL/Postgres)	Compatible with Spark's Hive client jars	Hive 3.1.3 metastore JARs conflict with Spark's built-in Hive 1.2.1 JARs	⚠️ Must manually replace JARs to avoid NoSuchMethodError
Recommended for Spark	✅ Yes	❌ Not officially — use only if you can manage JAR conflicts manually	



What Actually Happens with Hive 3.x

- Spark's Hive integration uses **Hive 1.2.1 client libraries internally** (for catalog and UDFs).
→ These don't align perfectly with Hive 3.x JARs.
- If you connect Spark 3.3.4 to a **Hive 3.1.3 metastore**, you'll likely hit:

```
java.lang.NoSuchMethodError: org.apache.hadoop.hive.metastore.api.Table.isTemporary()Z
```