UPDATED for Apache Flink v2.1 + Apache Iceberg v1.10 + AWS Glue: Get Your JAR Versions Right!

If you're diving into the cutting edge of **streaming + lakehouse integration**, this update is for you. Apache Flink **v2.1** introduces next-gen streaming semantics, while **Apache Iceberg v1.10** brings advanced table evolution, write performance, and AWS Glue catalog improvements. But there's a catch—**none of it works smoothly unless your JAR versions are perfectly aligned**.

Getting Flink, Iceberg, and AWS Glue to cooperate isn't just about adding dependencies—it's about **matching compatible versions across three evolving ecosystems**. A single mismatch between `flink-runtime`, `flink-table-runtime`, or `iceberg-flink-runtime` can trigger classpath conflicts, serialization errors, or cryptic `NoSuchMethodError` exceptions.

Here's the hard truth: Before you run a single Flink job or register a single Iceberg table, you need to know exactly which JARs to pull—and why. This version validation process often takes hours (or days) of trial and error. Been there. Done that. Learned it the hard way.

That's why I'm sharing my Gradle `build.gradle.kts`, so you don't have to. Because when your versions align, **Flink and Iceberg don't just run—they sing!**

Take a look

For those who'd rather skip the explanations and get straight to it — here's the full Gradle build script:

```
plugins {
    application
    id("com.gradleup.shadow") version "9.2.2"
}

// --- Read the Gradle properties file
val appMainClass: String? by project

repositories {
    mavenCentral()
    maven {
        url = uri("https://packages.confluent.io/maven/")
    }
}

// --- Dependency version numbers
val flinkVersion: String = "2.1.0"
val hadoopVersion: String = "3.4.2"
val kafkaVersion: String = "4.0.1"
val awssdkVersion: String = "2.35.10"
var icebergVersion: String = "1.10.0"
var confluentKafkaVersion: String = "8.0.0"
var jacksonVersion: String = "2.18.1"

// --- Comprehensive exclusion of all metrics libraries to avoid conflicts
with Flink
configurations.all {
    exclude(group = "io.dropwizard.metrics", module = "metrics-core")
    exclude(group = "io.dropwizard.metrics", module = "metrics-jvm")
    exclude(group = "io.dropwizard.metrics", module = "metrics-json")
    exclude(group = "io.dropwizard.metrics", module = "metrics-graphite")
    exclude(group = "io.dropwizard.metrics", module = "metrics-jmx")
    exclude(group = "io.dropwizard.metrics")
    exclude(group = "com.codahale.metrics", module = "metrics-core")
    exclude(group = "com.codahale.metrics")
}

dependencies {
    // --- Logging dependencies
    implementation("org.slf4j:slf4j-api:2.0.17")
    runtimeOnly("ch.qos.logback:logback-classic:1.5.12")

    // --- Kafka, Avro and JSON dependencies
    implementation("org.apache.kafka:kafka-clients:${kafkaVersion}")
    implementation("org.apache.avro:avro:1.12.0")
    implementation("io.confluent:kafka-avro-
serializer:${confluentKafkaVersion}")
    implementation("io.confluent:kafka-schema-registry-
client:${confluentKafkaVersion}")
    implementation("tech.allegro.schema.json2avro:converter:0.3.0")
    implementation("org.json:json:20250517")
    implementation("com.fasterxml.jackson.core:jackson-
databind:${jacksonVersion}")
    implementation("com.fasterxml.jackson.dataformat:jackson-dataformat-
avro:${jacksonVersion}")
```

```
    // --- Hadoop dependencies (with metrics exclusions)
    implementation("org.apache.hadoop:hadoop-common:${hadoopVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }

    // --- Flink dependencies
    compileOnly("org.apache.flink:flink-core:${flinkVersion}")
    implementation("org.apache.flink:flink-runtime:${flinkVersion}")
    compileOnly("org.apache.flink:flink-streaming-java:${flinkVersion}")
    compileOnly("org.apache.flink:flink-table-common:${flinkVersion}")
    compileOnly("org.apache.flink:flink-table-runtime:${flinkVersion}")
    compileOnly("org.apache.flink:flink-table-api-java-
bridge:${flinkVersion}")
    compileOnly("org.apache.flink:flink-metrics-
dropwizard:${flinkVersion}")
    implementation("org.apache.flink:flink-clients:${flinkVersion}")
    compileOnly("org.apache.flink:flink-connector-base:${flinkVersion}")
    implementation("org.apache.flink:flink-connector-kafka:4.0.1-2.0")
    implementation("org.apache.flink:flink-connector-
datagen:${flinkVersion}")
    implementation("org.apache.flink:flink-avro-confluent-
registry:${flinkVersion}")
    implementation("org.apache.flink:flink-json:${flinkVersion}")

    // --- AWS SDK v2 dependencies
    implementation("software.amazon.awssdk:sdk-core:${awssdkVersion}")

implementation("software.amazon.awssdk:secretsmanager:${awssdkVersion}")
    implementation("software.amazon.awssdk:ssm:${awssdkVersion}")
    implementation("software.amazon.awssdk:glue:${awssdkVersion}")
    implementation("software.amazon.awssdk:kms:${awssdkVersion}")
    implementation("software.amazon.awssdk:s3:${awssdkVersion}")
    implementation("software.amazon.awssdk:sts:${awssdkVersion}")
    implementation("software.amazon.awssdk:dynamodb:${awssdkVersion}")

    // --- Iceberg dependencies (with explicit metrics exclusions)
    runtimeOnly("org.apache.iceberg:iceberg-core:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }
    implementation("org.apache.iceberg:iceberg-api:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }
    runtimeOnly("org.apache.iceberg:iceberg-common:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }
    runtimeOnly("org.apache.iceberg:iceberg-aws:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }
    implementation("org.apache.iceberg:iceberg-
snowflake:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }
    implementation("org.apache.iceberg:iceberg-flink-
2.0:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }

    // --- Iceberg Flink Runtime (with comprehensive metrics exclusion)
    implementation("org.apache.iceberg:iceberg-flink-runtime-
2.0:${icebergVersion}") {
        exclude(group = "io.dropwizard.metrics")
        exclude(group = "com.codahale.metrics")
    }

    // --- Snowflake JDBC driver
    implementation("net.snowflake:snowflake-jdbc:3.27.0")

    // --- Flink test dependencies
    testImplementation("org.apache.flink:flink-test-
utils:${flinkVersion}")
    testImplementation("org.apache.flink:flink-test-utils-
junit:${flinkVersion}")

    // --- JUnit Jupiter for testing
    testImplementation(libs.junit.jupiter)
    testRuntimeOnly("org.junit.platform:junit-platform-launcher")

    // --- This dependency is used by the application.
    implementation(libs.guava)
}

// --- If the version is not provided, use the default
version = "dev-SNAPSHOT"

description = rootProject.name

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

application {
    // --- If the main class is not provided, use the default
    if (appMainClass.isNullOrEmpty()) {
        mainClass.set("kickstarter.AvroDataGeneratorApp")
    } else {
        mainClass.set("kickstarter." + appMainClass)
    }
}
```

```
tasks.withType<Zip> {
    isZip64 = true
}

tasks.shadowJar {
    archiveBaseName.set(rootProject.name)
    archiveClassifier.set("")
    mergeServiceFiles()

    // --- Comprehensive exclusion of metrics libraries from shadow JAR
    exclude("io/dropwizard/metrics/**")
    exclude("com/codahale/metrics/**")
    exclude("META-INF/services/io.dropwizard.metrics.*")

    // --- Relocate problematic dependencies to avoid conflicts
    relocate("com.google.common", "shaded.com.google.common")

    manifest {
        attributes(
            "Main-Class" to application.mainClass.get(),
            "Implementation-Title" to rootProject.name,
            "Implementation-Version" to project.version
        )
    }
}

tasks.build {
    dependsOn(tasks.shadowJar)
}

tasks.named<Test>("test") {
    useJUnitPlatform()
    jvmArgs = listOf(
        "--add-opens", "java.base/java.util=ALL-UNNAMED",
        "--add-opens", "java.base/java.time=ALL-UNNAMED",
        "--add-opens", "java.base/java.lang.invoke=ALL-UNNAMED"
    )
}
```

Then make sure these JARs are installed in your Apache Flink's installation `<FLINK_HOME>/lib/` Folder:

> Replace `<FLINK_HOME>` with your Apache Flink installation home directory, e.g., `/opt/flink`.

```
curl -L "https://repo1.maven.org/maven2/org/apache/flink/flink-s3-fs-hadoop/2.1.0/flink-s3-fs-hadoop-2.1.0.jar" -o "/opt/flink/lib/flink-s3-fs-hadoop-2.1.0.jar"
curl -L "https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-common/3.4.2/hadoop-common-3.4.2.jar" -o "/opt/flink/lib/hadoop-common-3.4.2.jar"
curl -L "https://repo1.maven.org/maven2/org/apache/iceberg/iceberg-flink-runtime-2.0/1.10.0/iceberg-flink-runtime-2.0-1.10.0.jar" -o "/opt/flink/lib/iceberg-flink-runtime-2.0-1.10.0.jar"
curl -L "https://repo1.maven.org/maven2/org/apache/flink/flink-metrics-dropwizard/2.1.0/flink-metrics-dropwizard-2.1.0.jar" -o "/opt/flink/lib/flink-metrics-dropwizard-2.1.0.jar"
curl -L "https://repo1.maven.org/maven2/org/io/dropwizard/metrics/metrics-core/4.2.37/metrics-core-4.2.37.jar" -o "/opt/flink/lib/metrics-core-4.2.37.jar"
curl -L "https://repo1.maven.org/maven2/org/apache/flink/flink-shaded-hadoop-2-uber/2.8.3-10.0/flink-shaded-hadoop-2-uber-2.8.3-10.0.jar" -o "/opt/flink/lib/flink-shaded-hadoop-2-uber-2.8.3-10.0.jar"
```

The goal was to run Flink apps with the latest Flink and Iceberg versions backed by AWS Glue as the metastore (a.k.a., catalog). Using a fine-tuned Gradle build and the right JARs in Flink's `lib` directory, I achieved full compatibility and stable runtime performance across all components.

I'll stop here for now

It took some trial and error to find the right combination of JARs and dependencies, so I'll let this sit for a bit—especially for anyone looking to replicate the exact setup. In **Part II**, I'll walk through how, with these JARs in place, you can start building some truly powerful Flink apps using Java.

But if you can't wait, I've got you covered—check out the hands-on code here. I'll dive deeper into how everything works in **Part II**. Stay tuned!