Resolving Elasticsearch 9.0.1 Upgrade Issues with Custom Plugin Compatibility

Daniel Gleim · 2026-07-14

Resolving Elasticsearch 9.0.1 Upgrade Issues with Custom Plugin Compatibility

Introduction

Upgrading Elasticsearch to a new major version is always exciting: you get access to the latest features, performance improvements, and security updates. However, major version upgrades can also surface compatibility issues, especially when you're running custom plugins. If you're planning to upgrade from Elasticsearch 8.x to 9.0, you'll want to understand how plugin compatibility works and what to do when things don't go as planned.

Important upgrade path note: To perform a major upgrade from 8.x to 9.x, you generally must first upgrade to **8.19.x**. The one exception is 8.18.x: since Elasticsearch 8.18.0 and 9.0.0 were released simultaneously, upgrading directly from 8.18.x to 9.0.x is also a supported path. If you are on any earlier 8.x version, upgrade to 8.19.x first.

Problem Description

Imagine you've just upgraded your Elasticsearch cluster from version 8.18.1 to 9.0.1, and suddenly your cluster won't start. The logs show a frustrating error message:

```
java.lang.IllegalArgumentException: Plugin [your-custom-plugin] was built for Elasticsearch
version 8.18.1 but version 9.0.1 is running
```

Your cluster is completely down, and you're staring at an error that points to plugin version compatibility. This is exactly what happened in a recent support case where a user's custom plugin prevented their freshly upgraded cluster from starting.

Root Cause Analysis

The root cause of this issue lies in Elasticsearch's strict plugin compatibility enforcement. When Elasticsearch starts up, it performs a version check on all installed plugins. Each plugin contains metadata that specifies exactly which version of Elasticsearch it was built for.

Here's what happens during the compatibility check:

- Elasticsearch reads the `plugin-descriptor.properties` file from each installed plugin
- It compares the `elasticsearch.version` property in the plugin descriptor with the running Elasticsearch version
- If there's a mismatch, Elasticsearch refuses to start and throws an `IllegalArgumentException`

This strict enforcement exists for good reason. Major version changes in Elasticsearch often include breaking API changes, internal restructuring, or modified interfaces that plugins depend on. A plugin built for version 8.x might use APIs or internal classes that have been removed or changed in version 9.x, potentially causing crashes or data corruption.

In this specific case, the custom plugin was compiled against Elasticsearch 8.18.1 APIs, but the cluster was now running 9.0.1. The version mismatch triggered the safety mechanism, preventing cluster startup.

Solution

The solution involves rebuilding your custom plugin against the new Elasticsearch version. Here's how to resolve this step by step:

Step 1: Update Your Plugin's Build Configuration

First, you need to update your plugin's build configuration to target Elasticsearch 9.0.1. If you're using Gradle (which most Elasticsearch plugins do), update your `build.gradle` file.

Important: Elasticsearch has bundled its own JDK since version 7.0, so you do not need to install a JDK to *run* Elasticsearch. However, to *build* your plugin, your build environment must use a compatible JDK. Elasticsearch 9.0 bundles JDK 24, so your plugin should be compiled against JDK 24 as well.

```
dependencies {
    compileOnly "org.elasticsearch:elasticsearch:9.0.1"
    // Update any other Elasticsearch dependencies to 9.0.1
}

// Ensure your build environment's Java version matches Elasticsearch 9.0's bundled JDK
java {
    sourceCompatibility = JavaVersion.VERSION_24
    targetCompatibility = JavaVersion.VERSION_24
}
```

Step 2: Review and Update Your Plugin Code

Before rebuilding, review your plugin code for potential compatibility issues. Major version upgrades often include breaking changes that might affect your plugin:

- Check the Elasticsearch breaking changes documentation for version 9.0
- Look for deprecated APIs from 8.x that have now been completely removed
- Update any import statements or method calls that have changed

Migrate from SecurityManager to Entitlements (critical for most plugins)

One of the most impactful breaking changes in Elasticsearch 9.0 for plugin developers is the **permanent replacement of the Java SecurityManager with Elasticsearch's own Entitlements system**. The Java SecurityManager was deprecated since Java 17 and is completely disabled in Java 24. To retain a comparable level of protection, Elasticsearch 9.0 introduced Entitlements as a permanent replacement.

If your plugin previously declared permissions via a `plugin-security.policy` file or made any calls to `SecurityManager` APIs, you must migrate to the new Entitlements-based approach. This is often the most time-consuming part of a plugin upgrade to 9.0, and it should be addressed before attempting a rebuild.

Refer to the Elasticsearch plugin developer documentation for guidance on how to declare entitlements in your plugin.

Step 3: Rebuild Your Plugin

Once you've updated the dependencies and code, rebuild your plugin:

```
./gradlew build
```

This creates a new plugin ZIP file in the `build/distributions/` directory, now compiled against Elasticsearch 9.0.1.

Step 4: Install the Updated Plugin

Note: Modifying plugins requires the Elasticsearch node to be stopped. In our crash scenario, the node is already down, but keep this in mind if you are doing this proactively.

Remove the old plugin version and install the newly built one:

```
# Remove the old plugin
bin/elasticsearch-plugin remove your-custom-plugin

# Install the updated plugin
bin/elasticsearch-plugin install file:///path/to/your-updated-plugin.zip
```

Step 5: Start Your Cluster

With the updated plugin installed, you can now start your Elasticsearch node:

```
bin/elasticsearch
```

The version compatibility check should now pass, and your cluster will start successfully.

Why There Is No Temporary Workaround

A common question during stressful upgrade outages is: *"Can I just set a flag to bypass the version check temporarily?"*

The short answer is **No**. Unlike some other software, Elasticsearch does not offer a startup parameter (like `-Des.plugin.version.check=false`) to disable this check. The enforcement is deeply baked into the core startup process.

The only "hack" to bypass this is to unzip the plugin, manually change the `elasticsearch.version` in the `plugin-descriptor.properties` file, and zip it back up. **You should absolutely never do this during a major version upgrade (like 8.x to 9.x).**

Because major upgrades update core dependencies (like Lucene), remove deprecated APIs, and, as noted above, completely replace the SecurityManager with Entitlements, tricking Elasticsearch into loading an old plugin will almost certainly result in a hard crash (e.g., `ClassNotFoundException` or `NoSuchMethodError`) the moment the plugin is executed. The strict version check isn't an annoyance; it is a critical safety net protecting your JVM from crashing and your data from corruption.

Takeaways

Here are the key lessons to remember when dealing with Elasticsearch plugin compatibility during upgrades:

Verify Your Upgrade Path First: Not all 8.x versions support a direct upgrade to 9.0. Only 8.18.x and 8.19.x can upgrade directly to 9.x (note that 8.18.x is only a supported starting point for 9.0.x). To upgrade to 9.1.0 or any later 9.x version, you must start from 8.19.x. If you are on an earlier 8.x version, upgrade to 8.19.x first.

Plan for Plugin Updates: When upgrading Elasticsearch major versions, always budget time for updating custom plugins. This isn't optional, it's a required part of the upgrade process.

Migrate from SecurityManager to Entitlements: If your plugin uses `plugin-security.policy` or SecurityManager APIs, this migration is mandatory for Elasticsearch 9.0. Plan accordingly, as it may require significant code changes.

Java Version Requirements for Building: Elasticsearch 9.0 bundles JDK 24. While you don't need an external JDK to *run* Elasticsearch, your plugin's build environment must use JDK 24 to compile against the 9.0 APIs.

Test in Development First: Never upgrade production clusters without first testing the entire stack, including custom plugins, in a development environment.

Monitor Breaking Changes: Keep track of Elasticsearch's breaking changes documentation for each release. This helps you understand what might need updating in your plugins.

Consider Plugin Alternatives: If maintaining custom plugins becomes burdensome, evaluate whether existing community plugins or native Elasticsearch features can meet your needs.

The strict plugin compatibility checking in Elasticsearch might seem frustrating when you're trying to upgrade quickly, but it's there to protect your cluster from potentially serious issues. Taking the time to properly rebuild and test your plugins ensures a smooth upgrade experience and maintains the stability of your search infrastructure.

Conclusion

Upgrading to Elasticsearch 9.0 is well worth the effort, but custom plugins require careful attention, especially with the introduction of the new Entitlements system. The version compatibility check is not an obstacle; it is your first line of defense against a broken cluster. With the right preparation, a verified upgrade path, and thoroughly tested plugins, the transition from 8.x to 9.0 can be smooth and straightforward. When in doubt, always test in a staging environment before touching production.

This article was published on the Search Guard blog: https://search-guard.com/blog/resolving-elasticsearch-9-0-1-upgrade-issues-with-custom-plugin-compatibility/