Plan and incompatibility toward Embulk 🐟 v1.0

Dai Mikurube / July 9, 2020
5 years have passed since the first Embulk release

- Embulk is going to introduce lots of plugin incompatibility through v0.9 → v0.10 → v0.11 → v1.
- This presentation is to explain the plan
  - for users -- which versions to choose in production?
  - for plugin developers -- how to catch up?
  - inside the core -- why? (if time allows)
For Users (everyone)
Planned versions

v0.10 - "development"

We're here: v0.10.5

v0.11 - "(pre)stable"

v1.0.* - stable

v1.0 ~

For Users (everyone)
v0.9(.23)

- v0.9.23 - stable

- Stable (as of July 2020)
- Only one production-ready version as of July 2020
- All latest plugins (as of July 2020) should work with it
- No more updates are expected in v0.9
  - Unless a backport is really required
v0.10

- "Development" unstable versions -- tries-and-errors
  - Not for your production! (TD uses it, though)
- Introducing several plugin incompatibility
- Deprecating several plugin API/SPI
- Plugins should be able to catch up to work both for the latest v0.9 (≒ v0.9.23) and the latest v0.10
How Embulk and plugins would update

When a plugin is updated to catch up with the latest v0.10, the latest version of the plugin should work with:

- The latest v0.9 (= v0.9.23 if no backports)
- The latest v0.10

v0.10.* is developed to satisfy the condition for plugins to work both with the latest v0.9 and the latest v0.10.
v0.11

- "(Pre-)stable" versions -- should be production-ready
- v0.11.0 will be identical with the last v0.10
- v0.11.0 defines **v1-ready** API/SPI
- v0.11.1+ removes items deprecated in v0.10.*
  - Many of legacy plugins (for v0.9) stop working (TBD)
v1.0

- Stable!
- **v1.0.0** will be identical with the last **v0.11**
  - Released when **v0.11.*** settles down, and gets confirmed
## Plugin would / wouldn't work with vX.Y.Z?

<table>
<thead>
<tr>
<th>No catch-up</th>
<th>Catch-up in v0.10</th>
<th>v1-ready (v0.11+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ v0.9.23</td>
<td>✓ v0.9.23 (or latest)</td>
<td>? v0.9.23</td>
</tr>
<tr>
<td>? v0.10.*</td>
<td>? v0.10.*</td>
<td>? v0.10.*</td>
</tr>
<tr>
<td>? the la(te)st v0.10</td>
<td>✓ the la(te)st v0.10</td>
<td>✓ the la(te)st v0.10</td>
</tr>
<tr>
<td>? v0.11.0</td>
<td>✓ v0.11.0</td>
<td>✓ v0.11.0</td>
</tr>
<tr>
<td>✗ v0.11.1+</td>
<td>? v0.11.1+</td>
<td>✓ v0.11.1+</td>
</tr>
<tr>
<td>✗ v1.0</td>
<td>? v1.0</td>
<td>✓ v1.0</td>
</tr>
</tbody>
</table>
Other decided incompatibility for every user

● JRuby is to be optional (not embedded in the core)

```bash
# TBD
embulk -X jruby=file:///.../jruby-complete-9.1.15.0.jar run ???\.yml
```

○ Bundler and Liquid: users to install by themselves? (TBD)

● `selfupdate` never finds the latest

```bash
embulk selfupdate # It does not work since v0.10.0.
                # It does not update to v0.10.* from v0.9.

embulk selfupdate X.Y.Z # Specific version is always needed.
```
https://www.embulk.org/: Renewed!

Pluggable, Multi-source/destination, Open-sourced Bulk Data Loader
to/from databases, storages, file formats, cloud services, ...

Latest versions!

Quick Start

Linux & macOS

curl --create-dirs -o ~/.embulk/bin/embulk --header "https/chmod +x ~/.embulk/bin/embulk

Windows

PowerShell -Command "$([Net.ServicePointManager..."

Announcements (replacing embulk-announce ML)

Look into this article on the stable and development versions.

For Users (everyone)
For Developers
Two choices for plugin development

- **Catch up accordingly with the latest Embulk v0.10.***
  - We (manage to) do it for [github.com/embulk](github.com/embulk)
  - TD members may post Pull Requests for some other plugins
  - Especially for plugins that TD uses (Thanks!)
  - Helpful for the Embulk core team
  - Feedbacks would be reflected to v0.11 and v1

- **Catch up with v1-ready API/SPI when v0.11.0 is out**
  - One-shot catch-up
0) Ruby plugins

TBD

- Sorry, but the situation is complicated with Ruby plugins
- I guess most of the existing Ruby plugins would work as-is
  - Not guaranteed... we may see some unavoidable incompatibility
- Tests -- not working fine
  - Helps are welcome
1) gradle-embulk-plugins

- [https://github.com/embulk/gradle-embulk-plugins](https://github.com/embulk/gradle-embulk-plugins)
  - Gradle plugin to build Java-based Embulk plugins

- Recommended to apply for all plugins **NOW**
  - Either catch-up accordingly v0.10.*, or once v0.11.0
  - It helps building and releasing pure-Java plugins (Maven)
  - It enables checking plugin's dependencies explicitly
    - By Gradle "dependency lock"
1) gradle-embulk-plugins

plugins {
    id "java"
    id "checkstyle"
    id "jacoco"
}

version = "X.Y.Z"

For Developers

plugins {
    id "java"
    id "checkstyle"
    id "jacoco"
    id "maven-publish"
    id "org.embulk.embulk-plugins"
    version "0.4.1"
}

group = "<your Maven group name>"

// group = "io.github.your-user-name"
// is typical
// group = "org.embulk"
// is only for github.com/embulk

version = "X.Y.Z"
description "<'description' of Gem>"
1) **gradle-embulk-plugins**

```gradle
configurations { provided }

dependencies {
    compile "org.embulk:embulk-core:.."
    provided "org.embulk:embulk-core:.."

    // May have conflict with embulk-core.
    compile "........."
}

task classpath(...) {
    from (configurations.runtime
         - configurations.provided
         + files(jar.archivePath))
    into "classpath"
}
```

For Developers

```gradle
dependencies {
    compileOnly "org.embulk:embulk-core:.."

    // Explicit conflict handling.
    compile(".........") {
        exclude group: "...", module: "...
    }
}
```
1) gradle-embulk-plugins

$ ./gradlew build
...

================================== WARNING ==================================
Following "runtime" dependencies are included also in "compileOnly" dependencies.

"com.fasterxml...:jackson-core:2.6.7"  // <= Conflicting with embulk-core's dependencies.

"compileOnly" dependencies are used to represent Embulk's core to be "provided" at runtime. They should be excluded from "compile" or "runtime" dependencies like the example below.

dependencies {
    compile("org.glassfish.jersey.core:jersey-client:2.25.1") {
        exclude group: "javax.inject", module: "javax.inject"
    }
}

=============================================================================
1) **gradle-embulk-plugins**

```groovy
// Applicable now immediately

```
1) gradle-embulk-plugins

publishing {
  publications {
    // Publish it with "publishEmbulkPluginMavenPublicationToMavenRepository".
    embulkPluginMaven(MavenPublication) {
      from components.java // Must be "components.java".
    }
  }
  repositories {
    maven {
      // Any Maven repository you want to release to!
      name = "mavenCentral"
      url "https://oss.sonatype.org/service/local/staging/deploy/maven2"
      credentials {
        username = project.hasProperty("ossrhUsername") ? ossrhUsername : ""
        password = project.hasProperty("ossrhPassword") ? ossrhPassword : ""
      }
    }
  }
}
1) **gradle-embulk-plugins**

```shell
$ ./gradlew dependencies --write-locks
...

$ cat gradle/dependency-locks/embulkPluginRuntime.lockfile

# This is a Gradle generated file for dependency locking.
# Manual edits can break the build and are not advised.
# This file is expected to be part of source control.
com.fasterxml.jackson.core:jackson-annotations:2.6.7
com.fasterxml.jackson.core:jackson-core:2.6.7
com.fasterxml.jackson.core:jackson-databind:2.6.7
com.jcraft:jsch:0.1.55
...

$ git add gradle/dependency-locks/embulkPluginRuntime.lockfile
```
2) Dependencies - source of incompatibility

If your Java plugin is (directly or indirectly) using

- Jackson, Guava, Apache Commons Lang 3, javax.validation
  - Include them explicitly in your plugin's dependencies (if used)
    - Choose **versions 100% same with embulk-core** till v0.11.0
    - gradle-embulk-plugins will warn, but you can ignore it
      - You can mark "ignored" explicitly in embulkPlugin { }

- JRuby, Joda-Time, Logback, Guice
  - Stop using them -- find an alternative, or give up using it
    - Ex. Joda-Time → java.time classes
2) Dependencies - source of incompatibility

```java
embulkPlugin {
    mainClass = "..."
    category = "input"
    type = "..."
    ignoreConflicts = [
        [ group: "com.fasterxml.jackson.core", module: "jackson-annotations" ],
        [ group: "com.fasterxml.jackson.core", module: "jackson-core" ],
        ...
    ]
}
```

---------- WARNING ----------
Following "runtime" dependencies are included also in "compileOnly" dependencies.

```text
([IGNORED] "com.fasterxml.jackson.core:jackson-annotations:2.6.7"
([IGNORED] "com.fasterxml.jackson.core:jackson-core:2.6.7"
...
```
2) Dependencies - source of incompatibility

- Dependencies of **embulk-core** is now visible from plugins
  - Jackson, Guava, Joda-Time, JRuby, ...
- Some of them could be passed from the core to plugins
  - It has caused problems between the core and plugins
    - Conflicts, broken when upgrading a library, ...

```java
interface PluginTask extends Task {
    @Config("example")
    JsonNode getExample();
}
```

```java
public ConfigDiff transaction(
    ConfigSource config, ...)
{ 
    ObjectNode object = config.getObjectNode();
}
```
2) Dependencies - source of incompatibility

- **embulk-core** will keep using those dependencies internally, but they will be hidden from plugins during v0.10.
2) Dependencies - source of incompatibility

- Dependencies below will stay visible from plugins:
  - `slf4j-api` (no logger implementation -- Logback)
  - `javax.inject` (not Guice)
  - `msgpack-core` -- TBD: maybe only its model classes
3) Prepare for Java 9+ (TBD: target → 11?)

- **JEP 320**: `javax.xml` (& more) is removed from JRE since 11
- **embulk-core** will **NOT** provide it in place of JRE
  - A plugin using `javax.xml` will need to include them by itself

```java
dependencies {
  ...

  compile "javax.activation:javax.activation-api:..."
  compile "javax.xml.bind:jaxb-api:..."
  compile "com.sun.xml.bind:jaxb-core:..."
  compile "com.sun.xml.bind:jaxb-impl:..."
}
```
4) Depend only on `embulk-api/spi`

- Plugins would **NOT** depend on `embulk-core`, but:
  - `embulk-api` *(Started v0.10.1 ~ work-in-progress)*
  - `embulk-spi` *(To start v0.10.6 or later)*
- Moving API/SPI from `embulk-core` to them
- `embulk-api/spi` will be **Documented**
- Others remaining in `embulk-core` will be **Undocumented**

```java
dependencies {
    compileOnly "...:embulk-core:...
}
```

```java
dependencies {
    compileOnly "...:embulk-api:...
    compileOnly "...:embulk-spi:...
}
```
5) Core features exported to libraries

Along with `embulk-api/spi`, utility classes for plugins are exported out of `embulk-core`, into external libraries:

- **TimestampFormatter / TimestampParser**
  - `embulk-util-timestamp` ([Javadoc](#))

- **ConfigSource#loadConfig / TaskSource#loadTask**
  - `embulk-util-config` ([Javadoc](#))

- ... to be continued
6) Miscellaneous deprecation

- `Exec.getLogger` → `org.slf4j.LoggerFactory.getLogger`
- Guava `Optional` → `java.util.Optional`
- Guava `Throwables` → `throw RuntimeException` (or else)
- `Timestamp` → `java.time.Instant`
  - `embulk-util-timestamp` handles `java.time.Instant`
  - `Timestamp` would remain in some interfaces, though
- `@ConfigInject` → Use `Exec.get???(())` instead
- `ModelManager` → Build your own `ObjectMapper`
Join Slack *embulk-dev*

- **No user support**
  - Only for plugin (or core) developers

- Announcement, discussion, and Q&A for v0.10+ catch-ups

- To join:
  - Leave a message at: [https://github.com/embulk/embulk/issues/1222](https://github.com/embulk/embulk/issues/1222)
    - Nice to leave a note about your plugins
    - I'll invite your GitHub email address (in commits)
  - Or, contact [@dmikurube](https://github.com/dmikurube) directly in some way
Appendix

Inside the Core
What's the problem? (Trivial case)

Using Jackson expecting 2.6.7
What's the problem? (Trivial case)

Using Jackson expecting 2.6.7
What's the problem? (Trivial case)

Inside the Core

```
embulk-core

jackson-annotations:2.6.7
jackson-core:2.6.7
jackson-databind:2.6.7

Using Jackson expecting 2.11.1

embulk-input-???
jackson-dataformat-xml:2.11.1
```
What's the problem? (Trivial case)

Inside the Core

Using Jackson expecting 2.6.7

Standoff (deadlock)
What's the problem? (Trivial case)

Unrealistic to test all.
What's wanted to achieve

Inside the Core

Maintain their own dependencies.

- Easier to test each.
What's wanted to achieve

Loose coupling.
⇒ Easier to test each.
What's wanted to achieve

Less dependent (to core).

⇦ Easier to test each.

Inside the Core
What's wanted to achieve

embulk-core

Compact!
Maintainable!

embulk-api/embulk-spi

utils

embulk-input-???

utils

jackson-core:3.X.Y