Author: Chamikara Jayalath
First proposed: 05/31/2018
Status: Stable

This document was migrated to Apache Beam Website:
https://beam.apache.org/contribute/dependencies/

This document describes a proposal for keeping Beam dependencies up to date.

Recently we have run into many issues due to Beam dependencies being significantly out of date. For examples, see [1], [2] and [3]. Old dependencies cause user pain and can result in a system being unusable for some users. Many users do not use Beam in isolation, and bundle other dependencies in the same deployment. These additional dependencies might pull in incompatible dependencies to the user's environment which can again result in broken Beam pipelines, sometimes with undefined behavior. To prevent this, users will have to update their deployment environment or worse yet may end up not being able to use Beam along with some of the other dependencies at all.

Current status

Currently, Beam Java SDK's Gradle and Maven builds define a set of top level dependencies and various components (runners, IO connectors, etc) can choose to include these dependencies. Components usually use the version defined at the top level but may choose to override the version.

If a component *X* chooses to override the version of a dependency *D* from *a* to *b* and another component *Y* is incompatible with version *b* of *D*, deployment of a user that uses both components *X* and *Y* will end up in a broken state.

A similar issue could arise if two dependencies of Beam depend on a common library but use incompatible versions of that library.

Also, users might not use Beam in isolation, a user that depends on Beam as well as other libraries in the same environment might run into similar issues if Beam and another library share a dependency while using incompatible versions.

Beam Python SDK handles dependencies slightly differently, all dependencies are defined in a single setup.py [4] file and are grouped. One of the groups describe required dependencies while other groups are for dependencies for various optional features. All Python modules have to use the versions of dependencies defined in setup.py file. Additionally, for most of the dependencies, Python SDK allows automatic upgrades upto next major version. Because of this setup, Python SDK currently does not run into component conflicts but other two forms of dependency conflicts described above can still occur.

This picture can become more complicated during runtime. Runner specific code might be incompatible with dependencies included by certain modules, and if these dependencies leak into runtime, a pipeline might end up in a broken state.

The overall issue is not common to Beam and well known in the industry as the *Diamond Dependency* problem (or *Dependency Hell*).

One common solution for the diamond dependency problem is semantic versioning [5]. The basic idea is that dependencies will be versioned in the form x.y.z where x is the major version, y is the minor version, and z is the patch version. A major version change may be backwards incompatible and is expected to be rare. A minor versions may be released more regularly but are expected to be backwards compatible. But in practice, important fixes (such as security patches) might get released in the form of minor version updates and it will be healthy for the Beam project to depend on recently released minor versions of dependencies.

Identifying outdated dependencies

A big part of keeping dependencies up to date involves identifying the outdated dependencies of Beam that the community should try to upgrade.

Yifan Zou introduced a proposal [6] that achieves this purpose. The basic idea is to run a per-SDK Jenkins job that identifies and flags outdated dependencies. I refer you to Yifan's proposal for details on the exact methodology.

In addition to this, Beam community members might identify other critical dependency updates that have to be manually triggered. For example,

(1) A minor release of a dependency due to a critical security vulnerability.
(2) A dependency conflict that was was triggered by a minor version release of a Beam dependency (this does not apply to Java SDK that depends on exact minor versions of dependencies).

Upgrading identified outdated dependencies

After outdated dependencies are identified, Beam community has to act to upgrade the dependencies regularly. I believe we have to introduce several policy modifications to properly achieve this task. I propose following.

- **Human readable reports on status of Beam dependencies are generated weekly by an automated Jenkins job and shared with the Beam community through the dev list.**

  These reports should be concise and should highlight the cases where the community

has to act on. See [6] for more details on this.

- **Beam components should define dependencies and their versions at the top level. There can be rare exceptions, but they should come with explanations.**

  Components include various Beam runners, IO connectors, etc. Component-level dependency version declarations should only be performed in rare cases and should come with a comment explaining the reasoning for overriding the dependency. For example, dependencies specific to a runner that are unlikely to be utilized by other components might be defined at the runner.

- **A significantly outdated dependency (identified manually or through the automated Jenkins job) should result in a JIRA that is a blocker for the next release. Release manager may choose to push the blocker to the subsequent release or downgrade from a blocker.**

  This will be a blocker for next major and minor version releases of Beam. JIRA may be created automatically or manually. I recommend methodology defined in [6] to automatically identify high priority dependency updates.
  For manually identified critical dependency updates, Beam community members should create blocking JIRAs for next release. In addition to this Beam community members may trigger patch releases for any critical dependency fixes that should be made available to users urgently.

- **Dependency declarations may identify owners that are responsible for upgrading the respective dependencies.**

  Owner can be mentioned in a comment. Blocking JIRAs will be initially assigned to these owners (if available). Release manager may choose to re-assign these JIRAs. A dependency may have more than one declared owner and in this case the JIRA will be assigned to the first owner mentioned.

- **Dependencies of Java SDK components that may cause issues to other components if leaked should be vendored.**

  This should be done on a case-by-case basis since shading can result in fat jars. See [7] for a thread on this.

Dependency updates and backwards compatibility

Beam releases adhere to semantic versioning [5]. Hence, community members should take care when updating dependencies. Minor version updates to dependencies should be backwards compatible in most cases. Some updates to dependencies though may result in backwards compatible API or functionality changes to Beam. PR reviewers and committers should take care to detect any dependency updates that could potentially introduce backwards incompatible changes to Beam before merging and PRs that update dependencies should include a statement regarding this verification in the form of a PR comment. Dependency updates that result in backwards incompatible changes to non-experimental features of Beam should be held till next major version release of Beam. Any exceptions to this policy should only occur in extreme cases (for example, due to a security vulnerability of an existing dependency that is only fixed in a subsequent major version) and should be discussed in the Beam dev list. Note that backwards incompatible changes to experimental features may be introduced in a minor version release.

[1] https://issues.apache.org/jira/browse/BEAM-3098
[2] https://issues.apache.org/jira/browse/BEAM-3991
[3] https://issues.apache.org/jira/browse/BEAM-4229
[4] https://github.com/apache/beam/blob/master/sdks/python/setup.py
[5] https://semver.org/
[6] https://docs.google.com/document/d/1rqr_8a9NYZCgeiXpTIwWLCL7X8amPAVfRXsO72BpBwA/edit#bookmark=id.rl13gopyrbku
[7] https://lists.apache.org/thread.html/12383d2e5d70026427df43294e30d6524334e16f03d86c9a5860792f@%3Cdev.beam.apache.org%3E