Javaアップグレードに関する苦労話: HadoopとそのディストリビューションBigtopの場合

2026/05/23

株式会社NTTデータ
岩崎 正剛

© 2026 NTT DATA Japan Corporation

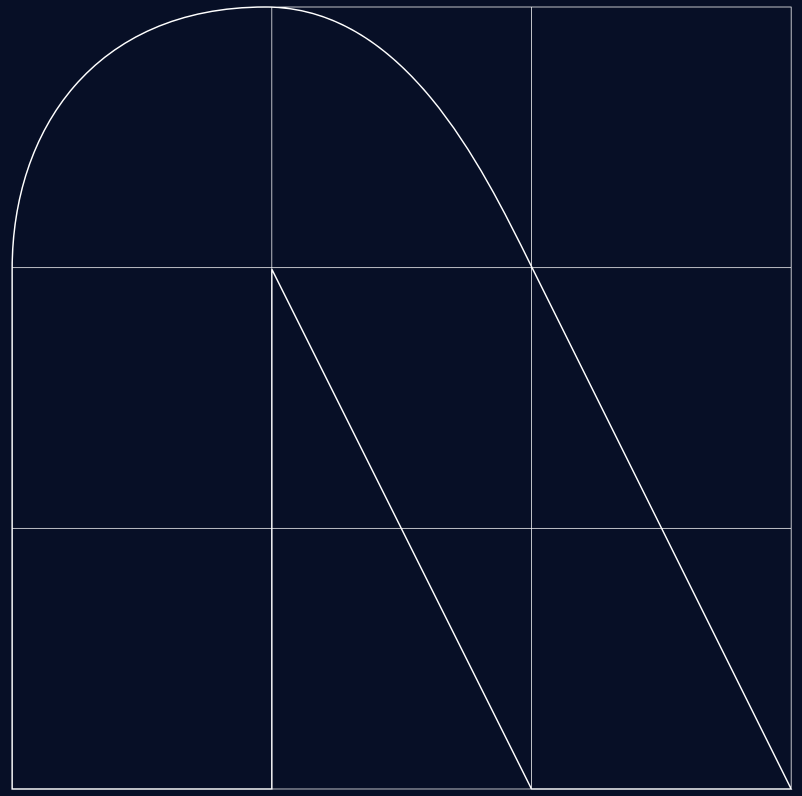
自己紹介

岩崎 正剛 / Masatake Iwasaki

Open source software developer / data engineer @NTT DATA Japan
Committer & PMC member of Apache Hadoop and Apache Bigtop
ASF member

© 2026 NTT DATA Japan Corporation

HadoopとBigtopの紹介



© 2026 NTT DATA Japan Corporation

Apache Hadoopとは何か?

- Hadoopは大規模データ処理のためのミドルウェア
- 汎用的なLinuxサーバ(～10000台)でクラスタを構成し、大量のデータを格納しつつ、並列分散処理することができる。
- 大きく分けて、以下の3種類の要素からなる。
  - 分散ファイルシステム(HDFS)
  - 計算リソース管理機構(YARN)
  - YARN上に実装された分散処理フレームワーク(MapReduce)

- Googleが自社の検索サービスに利用していた技術を、論文の形で紹介した内容を参考にして開発されたOSS
  - 論文の内容は、書籍『Googleを支える技術』などで紹介され、日本でも知られるところとなった
- 2006年に最初のバージョンが公開された
  - 「ビッグデータ」がbuzzwordだったのは2011～2014くらい?
  - その後、クラウド、コンテナ、機械学習、AIなどがキーワードとして注目されていく

- Javaで実装されている。

© 2026 NTT DATA Japan Corporation

Hadoopエコシステム

- Hadoopは実用上、多種多様な周辺ミドルウェアと組み合わせて利用される
- Hadoopエコシステムという呼称は、Hadoopと周辺ミドルウェア群を指す呼称
- 周辺ミドルウェアの代表的な例をあげると、以下のようなものがある。
  - Spark: 汎用的なAPIで並列分散処理アプリケーションを記述できるフレームワーク
  - Hive: SQL(ライクな)言語で並列分散処理アプリケーションを記述できる処理系
  - HBase: GoogleのBigtableを参考に実装された分散キーバリューストア
  - Ranger: アクセス制御のための管理基盤
- 周辺ミドルウェアの実装言語もJava/JVM言語が主流。

- プロダクト間には依存関係があり、あるプロダクトに非互換な変更が入ると、それに依存するプロダクト側でも修正が必要。
- 基本的には、各プロダクトはそれぞれ独立の開発者グループによって維持され、プロダクト間の互換性は時々壊れる。
- どのバージョンを組み合わせると機能するかは、Hadoopエコシステムの開発者自身にとっても、自明ではない。

© 2026 NTT DATA Japan Corporation

Apache Bigtopとは何か

公式サイト (https://bigtop.apache.org/) より:

- "Bigtop is an Apache Foundation project for Infrastructure Engineers and Data Scientists looking for comprehensive packaging, testing, and configuration of the leading open source big data components."
- 『Bigtopは、先進的なビッグデータ関連OSSのパッケージングやテスト、設定を求めているインフラエンジニアやデータサイエンティストのためのApacheソフトウェア財団のプロジェクトです。』

Hadoopエコシステムの製品を中心とした大規模データ処理基盤を容易に構築するためのプロジェクト

Hadoopディストリビューションを提供するCloudera社がASFに寄贈したパッケージングのコードが元になっている

© 2026 NTT DATA Japan Corporation

Hadoop as a services

- Hadoopエコシステムの製品を組み合わせた大規模データ処理基盤は、各種主要クラウドサービスでも提供されている。
  - Amazon EMR
  - Azure HDInsight
  - Google Cloud Dataproc
- 特に各サービスのデータストアと組み合わせて利用しやすい
  - Amazon S3
  - Azure Data Lake Storage
  - Google Cloud Storage

- Bigtopはそういった基盤を自分で作りたい人向けのもの。

- (200x年当時はまだ、マネージドサービス上で気軽に大規模データを処理できる状況ではなかった)

© 2026 NTT DATA Japan Corporation

データ基盤構築のためにBigtopが提供する機能

1. 要件を満たすソフトウェアの選定と、それらの間の相互運用性の確認
   - →相互運用性を確認済みの幅広いビッグデータ関連ソフトウェアを、debやrpmとしてパッケージングするための資材および、ビルド済みパッケージを提供

2. ソフトウェアパッケージをサーバ群にインストールし、各製品を適切に設定
   - →各ソフトウェアのデプロイを自動化するための Puppet マニフェストを提供

3. 設定を施したサーバ群が、クラスタとして正しく動作することを確認
   - →各ソフトウェアの主要な機能の動作確認のための、スモークテストを提供

4. ローカル環境で、デプロイ資材の開発、各製品の試用を実現
   - →Dockerコンテナ上でPuppetマニフェストの適用、スモークテストの実行や試用をするためのProvisionerを提供

© 2026 NTT DATA Japan Corporation

Apache Bigtopの歴史

	1.0.0	1.1.0	1.2.1	1.3.0	1.4.0	1.5.0
Flink			1.1.3	1.4.2	1.6.4	1.6.4
Hadoop	2.6.0	2.7.1	2.7.3	2.8.4	2.8.5	2.10.1
HBase	0.98.12	0.98.12	1.1.9	1.3.2	1.3.2	1.5.0
Hive	1.0.0	1.2.1	1.2.1	2.3.3	2.3.3	2.3.6
Kafka	0.8.1.1	0.8.1.1	0.10.1.1	0.10.2.2	0.10.2.2	2.4.0
Spark	1.3.1	1.5.1	2.1.1	2.2.1	2.2.3	2.4.5
Zeppelin		0.5.6	0.7.2	0.7.3	0.7.3	0.8.2
Zookeeper	3.4.6	3.4.6	3.4.6	3.4.6	3.4.6	3.4.13

Java 7

Java 7 / 8

© 2026 NTT DATA Japan Corporation

Apache Bigtopの歴史

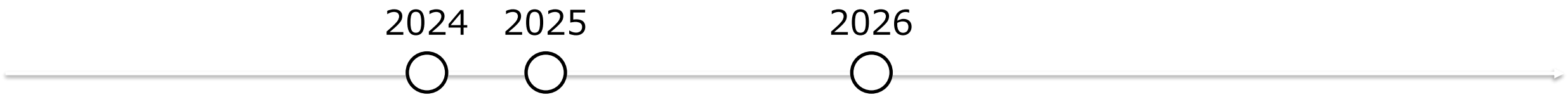
	3.0.0	3.0.1	3.1.0	3.1.1	3.2.0	3.2.1
Flink	1.11.3	1.11.6	1.11.6	1.11.6	1.15.0	1.15.3
Hadoop	3.2.2	3.2.2	3.2.3	3.2.4	3.3.4	3.3.5
HBase	2.2.6	2.2.6	2.4.8	2.4.8	2.4.13	2.4.13
Hive	3.1.2	3.1.2	3.1.2	3.1.2	3.1.3	3.1.3
Kafka	2.4.1	2.4.1	2.8.1	2.8.1	2.8.1	2.8.1
Spark	3.0.1	3.0.1	3.1.2	3.1.2	3.2.1	3.2.1
Zeppelin	0.9.0	0.10.0	0.10.0	0.10.0	0.10.1	0.10.1
Zookeeper	3.4.14	3.4.14	3.5.9	3.5.9	3.5.9	3.5.9

Java 8

Java 8 / 11

© 2026 NTT DATA Japan Corporation

Apache Bigtopの歴史



	3.3.0	3.4.0	3.5.0	3.6.0(仮)	3.7.0(仮)
Flink	1.16.2	1.20.0	1.20.1	1.20.1	
Hadoop	3.3.6	3.3.6	3.3.6	**3.4.3**	**3.5.x**
HBase	2.4.17	2.6.1	2.6.2	2.6.5	
Hive	3.1.3	4.0.1	4.0.1	4.0.1	
Kafka	2.8.2	3.4.1	3.4.1	3.4.1	
Spark	3.3.4	3.5.3	3.5.6	3.5.8	
Zeppelin	0.11.0	0.11.2	0.11.2	0.11.2	
Zookeeper	3.7.2	3.8.4	3.8.4	3.8.4	

Java 8 / 11 / 17

Java 17

© 2026 NTT DATA Japan Corporation

Javaのリリースサイクル

- (Java 9以降)半年ごとにfeature releaseを出す
- 約2年ごとにLTS(Long Term Support)版を出す

バージョン	GA
6	2006-12-12
7	2011-07-11
8 (LTS)	2014-03-18
9	2017-09-21
10	2018-03-20
11 (LTS)	2018-09-25
12	2019-03-19
13	2019-09-17
14	2020-03-17
15	2020-09-15
16	2021-03-16
17 (LTS)	2021-09-14
18	2022-03-22
19	2022-09-20
20	2023-03-21
21 (LTS)	2023-09-19
22	2024-03-19
23	2024-09-17
24	2025-03-18
25 (LTS)	2025-09-16
26	2026-03-17

© 2026 NTT DATA Japan Corporation

OpenJDKのディストリビューション

- (基本的には)LTS版をベースに、OSディストリビュータなどがOpenJDKディストリビューションを提供

ディストリビューション名	提供主体	特徴
Red Hat build of OpenJDK	Red Hat	RHELにバンドルし、企業向けに提供。8/11/17/21/25 を長期メンテナンス対象とするライフサイクルを公開。
Eclipse Temurin	Eclipse Adoptium プロジェクト	旧 AdoptOpenJDK。コミュニティベースの無償ビルド。8/11/17/21 をLTSとして長期提供。
Amazon Corretto	AWS (Amazon Web Services)	AWS 環境での利用を前提とした案内が多く、8/11/17/21 をLTSとして長期サポート。
Microsoft Build of OpenJDK	Microsoft	Azureなど Microsoft 環境向けに提供。11/17/21 をLTSとして長期サポートすることを明示。短期版は長期サポート対象外。
Oracle JDK	Oracle	Oracleが提供する商用ディストリビューション。利用には有償サブスクリプションが必要。8/11/17/21/25 をLTSとして Premier / Extended Support を設定。
Azul Zulu	Azul	Azul Platform Core の一部として提供。8/11/17/21/25 などを中心に長期サポート(有償)。無償の Zulu Community 版はLTS対象外。

© 2026 NTT DATA Japan Corporation

Red Hat build of OpenJDKのサポートライフサイクル

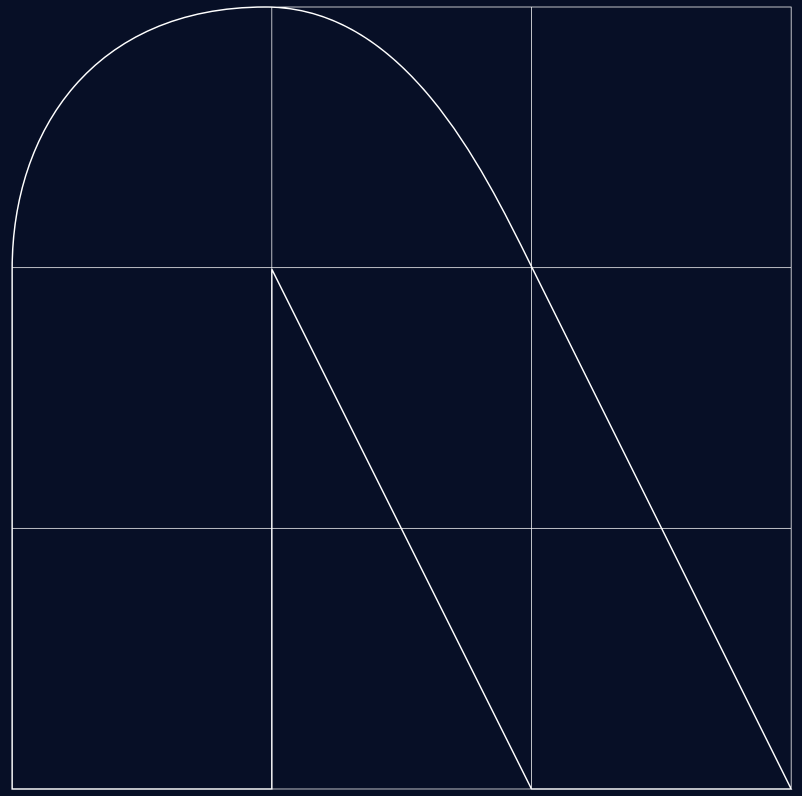
- OS(Red Hat Enterprise Linux)とは独立のサポートライフサイクルがある
- OpenJDK 8のEOLが11より長い
  - 何度か延長された結果
  - Java 8から11へのギャップが大きく、移行に苦労してるミドルウェア/ライブラリが多いことを反映?
- https://access.redhat.com/articles/1299013

OpenJDK on RHEL Support Table

	Supported RHEL Versions	End of Full Support	End of ELS-1
OpenJDK 6 (1.6)	RHEL 5.3+, 6.0+, 7.0+	December 31, 2016	N/A
OpenJDK 7 (1.7)	RHEL 5.9+, 6.3+, 7.0+	June 30, 2020	N/A
OpenJDK 8 (1.8)	RHEL 6.6*, 7.1+, 8.0+, 9.0+*	**November 30, 2026***	**December 31, 2030**
OpenJDK 11	RHEL 7.6+, 8.0+, 9.0+	**October 31, 2024**	**October 31, 2027**
OpenJDK 17	RHEL 8.4+, 9.0+	**December 31, 2027**	N/A
OpenJDK 21	RHEL 8.9+, 9.3+, 10.0+	**December 31, 2029**	N/A
OpenJDK 25	RHEL 9.7+, 10.1+	**December 31, 2030**	N/A

© 2026 NTT DATA Japan Corporation

HadoopのJavaマイグレーション



© 2026 NTT DATA Japan Corporation

Javaのアップグレードに必要な作業

- コンパイルエラーになるHadoop自体のコードの修正
- エラーになるテストケースの確認と対応
- ビルドツールをJavaバージョン対応版にアップグレード
  - Mavenおよびそのプラグイン
- 依存ライブラリを新しいJavaバージョン対応版にアップグレード
  - JUnit、Mockit、Jersey
- 開発者/CI環境向けのDockerfileの更新
- ドキュメント中のJavaに関する記述の更新

- 必要な修正量が多いパターンの例
  - これまでwarningだったJavadocの書き方がerrorになる
  - エラー時の挙動の変化で、通らなくなるテストの修正
    - 例: テスト上期待されている例外のメッセージの文言が変わった
  - JUnitのアップグレード: JUnitのテストケースを4の流儀から5に書き換え
    - JUnit 5上で古いAPIを使う仕組み(junite-vintage-engine)があるので、書き換えを後回しにしてゆっくりやれる
  - Jerseyのアップグレード: REST APIを定義するコードの修正

© 2026 NTT DATA Japan Corporation

HADOOP-15984: Update jersey from 1.19 to 2.x

- JAX-RS (REST APIのための標準/仕様)の実装であるJerseyのアップグレード
- 解決までに5年以上経過した。
- 難しい部分:
  - REST APIは各デーモンで実装されていて、影響範囲が多い。
  - バージョン差分で使い方が変わる部分があり、コード修正が避けられない。
  - Jersey 1と2を共存させて、サブモジュール単位で徐々に移行していく戦略がとれない。

Jerseyを利用してREST APIを実装するコードの例:

```
@GET
@Path("{" + UriFsPathParam.NAME + ":.*}")
@Produces({MediaType.APPLICATION_OCTET_STREAM + "; " + JettyUtils.UTF_8,
    MediaType.APPLICATION_JSON + "; " + JettyUtils.UTF_8})
public Response get(
    @Context final UserGroupInformation ugi,
    @Context final UriInfo uriInfo,
    @QueryParam(DelegationParam.NAME) @DefaultValue(DelegationParam.DEFAULT)
        final DelegationParam delegation,
    @QueryParam(UserParam.NAME) @DefaultValue(UserParam.DEFAULT)
        final UserParam username,
    @QueryParam(DoAsParam.NAME) @DefaultValue(DoAsParam.DEFAULT)
        final DoAsParam doAsUser,
    @QueryParam(GetOpParam.NAME) @DefaultValue(GetOpParam.DEFAULT)
        final GetOpParam op,
    ...
```

© 2026 NTT DATA Japan Corporation

巨大なパッチ

- Hadoopは4つのサブプロジェクト(Common, HDFS, YARN, MapReduce)の集まり
  - 過去に分割するかどうかの議論はあったが、しないことになった
- Mavenのサブモジュール数としては100以上
- 各種デーモンはREST APIを持ってるので、多くのサブモジュールが影響を受ける
- 各モジュールで開発が並行しているため、巨大なパッチは、すぐにconflictしがち
- 関連するテストをすべて流しきるのに数時間以上かかる

© 2026 NTT DATA Japan Corporation

Jersey 1.xと2.xを併用できない問題

- Javaでは、同一クラスローダ上に同じクラスの異なるバージョンは並存できない
  - Mavenにおいて、同一artifactの異なるバージョンを併用できない
- Jersey 1.x と 2.xはパッケージ名は変わっているので、Jerseyのコード自体は併用できるはず
  - com.sun.jersey.* -> org.glassfish.jersey.*
- Jersey 1.xはJAX-RS 1.1の実装なのに対して、Jersey 2.xはJAX-RS 2.0の実装
  - javax.ws.rs.* の異なるバージョンに依存する
    - (jsr311.jarとrs-api.jarが共存できない)
  - 関係するすべて(JAX-RSとJerseyとJackson?)をrelocation(後述)して共存させ、使い分けるのは困難

- Jersey 2.xをスキップしてJersey 3.x(JAX-RS 3.xの実装)に移行するのはどうか?
  - OracleからEclipse FoundationにJava EEが移管され、Jakarta EEになった結果、APIのパッケージ名が変わった
    - javax.ws.rs.* -> Jakarta.ws.rs.*
  - Jersey 3.xにするためには、Jettyを11にアップグレードする必要があり、影響範囲が大きくなってしまう

© 2026 NTT DATA Japan Corporation

解決へのステップ

- 部分的に実施可能な修正を先に行い、ある程度パッチのサイズを減らす
  - Jerseyの依存ライブラリのバージョン差分に起因する修正
  - テストコードから本質的には必要ないJersey依存部分を消す
- 実行環境 でJava 11や17対応するために、標準ライブラリから消えてしまった依存ライブラリ(JAXB)の代用品を足す
  - Hadoop 3.3はJava 11を、Hadoop 3.4はJava 17を、実行環境としてサポート

- 最終的には巨大なパッチをマージ
  - 最終的には: 392 files changed, 32897 insertions(+), 26930 deletions(-)
  - .diffのサイズとしては25kBくらい?
- Hadoop 3.5はビルドも実行環境もJava 17前提になった

- 時間経過により、開発コミュニティの活動量/コード修正ペースも落ち着き、大きなパッチがマージしやすくなった?
- いまならLLMの登場と進化により、力業が容易になった
  - ただ、HADOOP-15984のコミット時点ではまだ使われていなかった(はず)

© 2026 NTT DATA Japan Corporation

(参考)MavenのTransitive dependencies

- https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Transitive_Dependencies
- 依存ライブラリの依存ライブラリも依存ライブラリ
- 依存関係ツリー上に複数のバージョンがある場合、近いものが勝つ (dependency mediation)
- mediationの結果、問題なくビルドできて動くという保証はない
- Hadoopのクライアントを使うミドルウェアの依存関係ツリーはとても深い
- ミドルウェアとして、無駄に依存関係を増やさない/外に見せないべきという意識は、広がった気がする
  - 今日では、LLMで車輪の再発明的なコードを作って抱えるのも容易?

hadoop-commonのcommon-loggingに関するdependency mediation:

```
$ mvn dependency:tree -Dmaven-dependency-plugin.version=2.10 -Dverbose
...
[INFO] org.apache.hadoop:hadoop-common:jar:3.4.0-SNAPSHOT
...
[INFO] +- org.apache.httpcomponents:httpclient:jar:4.5.13:compile
[INFO] |  +- org.apache.httpcomponents:httpcore:jar:4.4.13:compile
[INFO] |  +- (commons-logging:commons-logging:jar:1.1.3:compile - version managed from 1.2; omitted for duplicate)
...
[INFO] +- commons-logging:commons-logging:jar:1.1.3:compile
...
[INFO] +- commons-beanutils:commons-beanutils:jar:1.9.4:compile
[INFO] |  +- (commons-logging:commons-logging:jar:1.1.3:compile - version managed from 1.2; omitted for duplicate)
[INFO] |  ¥- (commons-collections:commons-collections:jar:3.2.2:compile - omitted for duplicate)
[INFO] +- org.apache.commons:commons-configuration2:jar:2.1.1:compile
[INFO] |  ¥- (commons-logging:commons-logging:jar:1.1.3:compile - version managed from 1.2; omitted for duplicate)
```

© 2026 NTT DATA Japan Corporation

maven-shade-pluginを利用したパッケージ名の置換

- Javaの仕組み上、同一パッケージ名のライブラリの、異なるバージョンを併用することはできない。
- maven-shade-pluginの機能で、既存ライブラリのパッケージ名を置き換えたshadedライブラリを作ることができる。
- Hadoopでは、他のミドルウェアやユーザアプリケーションとの依存ライブラリのバージョン競合を防ぐ目的で利用されている。
- https://github.com/apache/hadoop-thirdparty

hadoop-thirdparty/hadoop-shaded-protobuf_3_25のpom.xml から抜粋(した内容を分かりやすさのために変数展開したもの):

```
<relocation>
  <pattern>com/google/protobuf</pattern>
  <shadedPattern>org.apache.hadoop.thirdparty.protobuf</shadedPattern>
</relocation>
```

© 2026 NTT DATA Japan Corporation

同一ライブラリの新旧バージョンの併用

- shadedライブラリと元のライブラリは共存できるので、同じライブラリの新旧バージョンを併用できる。
- 併用状態にすることで、サブモジュール単位で、バージョン競合に関する問題をい、段階的に修正できる。
- (Jerseyの場合は、JAX-RSのバージョン競合に起因する問題で、この手法が簡単に適用できなかった)

新旧ライブラリを併用:

```
<dependencies>
   <dependency>
     <groupId>org.apache.hadoop.thirdparty</groupId>
     <artifactId>hadoop-shaded-protobuf_3_25</artifactId>
   </dependency>
   ...
   <dependency>
     <groupId>com.google.protobuf</groupId>
     <artifactId>protobuf-java</artifactId>
     <version>2.5.0</version>
     <scope>provided</scope>
   </dependency>
```

relocateされたライブラリを明示的に利用:

```
import org.apache.hadoop.thirdparty.protobuf.BlockingService;
```

© 2026 NTT DATA Japan Corporation

hadoop-client-apiとhadoop-client-runtime (aka shaded client)

hadoop-clientのtransitive dependenciesを隠したもの
Hadoop 3.0.0で登場 (HADOOP-11804)
maven-shade-pluginのrelocation機能を利用

- hadoop-clientの依存ライブラリと、その呼び出し箇所のパッケージ名をバイトコード上書き換え
- 書き換え後の依存ライブラリをhadoop-client内部用に内包

hadoop-client-api: relocateされたorg.apache.hadoop.*が入ったfat jar
hadoop-client-runtime: relocateされた依存ライブラリが入ったfat jar

hadoop-client-apiとhadoop-client runtime:

```
$ ls -lh hadoop-client-api-3.4.0-SNAPSHOT.jar
-rw-rw-r--. 1 centos centos 19M Dec 11 11:05 hadoop-client-api-3.4.0-SNAPSHOT.jar

$ ls -lh hadoop-client-runtime-3.4.0-SNAPSHOT.jar
-rw-rw-r--. 1 centos centos 30M Dec 11 11:07 hadoop-client-runtime-3.4.0-SNAPSHOT.jar

$ jar tvf hadoop-client-runtime-3.4.0-SNAPSHOT.jar | grep '/shaded/' | awk '{print $8}'
...
org/apache/hadoop/shaded/com/google/common/annotations/VisibleForTesting.class
org/apache/hadoop/shaded/com/google/common/base/Absent.class
...
```

© 2026 NTT DATA Japan Corporation

Java 8からの脱却

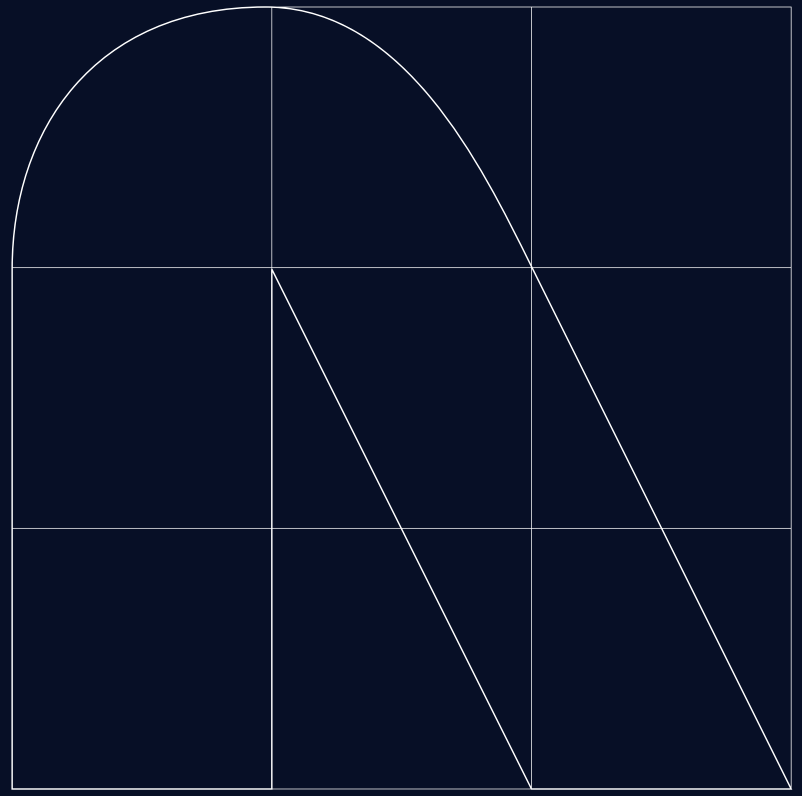
- リリースビルド(に伴うドキュメントのビルド)にはJava 8が必要だった
  - Javadocのdocletとして実行されるJdiffがcom.sun.javadocに依存
  - Java 9で無くなったtools.jarに入っていた部分
  - HADOOP-19402 で後継のjdk.javadoc.docletを使う形に修正された
- それ以外の部分をビルドしたり、動かす上ではJava 11でも問題なかった
  - (ランタイムとしてはHadoop 3.3でjava 11に、Hadoop 3.4でJava 17に対応)

Javadocの生成過程で発生するエラー

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-site-plugin:3.9.1:site (default-site) on project hadoop-main: Error
generating maven-javadoc-plugin:3.0.1:aggregate report:
[ERROR] Exit code: 1 - javadoc: warning - The old Doclet and Taglet APIs in the packages
[ERROR] com.sun.javadoc, com.sun.tools.doclets and their implementations
[ERROR] are planned to be removed in a future JDK release. These
[ERROR] components have been superseded by the new APIs in jdk.javadoc.doclet.
[ERROR] Users are strongly recommended to migrate to the new APIs.
```

© 2026 NTT DATA Japan Corporation

BigtopとJava



© 2026 NTT DATA Japan Corporation

ビルドツールのすり合わせ

- JavaのビルドツールとしてMavenがよく利用される
- プロダクトによっては利用するプラグインとの兼ね合いなどで、バージョンが合っていないとビルドは通らない
  - 原因が分かりにくいので、Mavenの仕組みでビルド時にチェックする仕込みもある

maven-enforcer-pluginによるバージョン指定の例:

```
<plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-enforcer-plugin</artifactId>
          <version>3.3.0</version>
          <executions>
            <execution>
              <id>enforce-versions</id>
              <goals>
                <goal>enforce</goal>
              </goals>
              <configuration>
                <rules>
                  <requireMavenVersion>
                    <version>3.9.6</version>
                  </requireMavenVersion>
                  <requireJavaVersion>
                    <version>1.8</version>
                  </requireJavaVersion>
```

© 2026 NTT DATA Japan Corporation

bigtop_toolchain

- OS標準のmavenパッケージでは、バージョンが古くて合わないことが多い
- 新しめのバイナリディストリビューションをダウンロードして配置するPuppetマニフェストを用意して対応
- 1つのバージョンで対応できなければ、ビルド対象のプロダクトに応じて切り替えることもできる
  - (今は1つのバージョンでカバーできている)

bigtop_toolchainのMavenインストール用Puppetマニフェスト(bigtop_toolchain/manifests/maven.pp)から抜粋:

```
$mvnversion = latest_maven_binary("3.9.[0-9]*")
 $mvn = "apache-maven-$mvnversion"

 $apache_prefix = nearest_apache_mirror()

 exec { 'Download Maven binaries':
   command => "/usr/bin/wget $apache_prefix/maven/maven-3/$mvnversion/binaries/$mvn-bin.tar.gz",
   cwd     => "/usr/src",
   unless  => "/usr/bin/test -f /usr/src/$mvn-bin.tar.gz",
 } ~>

 exec { 'Download Maven binaries signature':
   command => "/usr/bin/wget $apache_prefix/maven/maven-3/$mvnversion/binaries/$mvn-bin.tar.gz.asc",
   cwd     => "/usr/src",
   unless  => "/usr/bin/test -f /usr/src/$mvn-bin.tar.gz.asc",
 } ~>

 ...
```

© 2026 NTT DATA Japan Corporation

Hadoopのネイティブコード

- (古い)Javaから直接利用できないLinuxの機能を利用するためなど、意外とネイティブコードが利用されている
  - JNIが関係する部分は、Javaのアップグレードにも影響する可能性あり
  - libhadoop.so
    - compression codec(lz4, snappy, zlib, zstd)
    - 高速化(CRC32 by pipelined SSE42, user/group mapping, OpenSSL cipher)
    - HDFSの諸機能
      - Short Circuit Local Read: sendmsg/recvmsgでのオープン済ファイルディスクリプタ受け渡し
      - read ahead/drop behind: posix_fadviseによるキャッシュ戦略指定
      - cacheadmin: mmap/mlockによるデータブロックのメモリ保持
  - libhdfs (HDFSのCクライアント)
  - libhdfspp (HDFSのC++クライアント)
  - Linux container-executor(YARNのセキュリティ機能)
- コンパイラやビルドツール、必要なライブラリ一式はbigtop_toolchainのPuppetマニフェストでセットアップできる。

- 最近ではJavaから直接使えたり、Javaのwrapperが外部提供されたりしている部分もある
- (Hadoop以外の周辺ミドルウェアで、ネイティブコードの利用箇所は少ない)

© 2026 NTT DATA Japan Corporation

CPUアーキテクチャ固有のコードへの対応

- Javaは基本的にはマルチプラットフォーム対応
- だがJNIなどネイティブコードを使うライブラリで、公式配布バイナリが特定プラットフォームしかサポートしていないことがある
- Bigtopでは、ビルド環境で作ったバイナリを、artifactとしてローカルリポジトリ(~/.m2/repository)にインストールして使う

Bigtopのパッケージング用スクリプト(do-component-build)から抜粋:

```
if [ $HOSTTYPE = "powerpc64le" ] ; then
  mvn install:install-file -DgroupId=com.google.protobuf -DartifactId=protoc -Dversion=2.5.0 ¥
            -Dclassifier=linux-ppcle_64 -Dpackaging=exe -Dfile=/usr/local/protobuf-2.5.0/bin/protoc
  MVN_ARGS+="-Dprotobuf.group=com.google.protobuf -Dprotoc.version=2.5.0 "
elif [ $HOSTTYPE = "aarch64" ] ; then
  mvn install:install-file -DgroupId=com.google.protobuf -DartifactId=protoc -Dversion=2.5.0 ¥
            -Dclassifier=linux-aarch_64 -Dpackaging=exe -Dfile=/usr/local/protobuf-2.5.0/bin/protoc
  MVN_ARGS+="-Dprotobuf.group=com.google.protobuf -Dprotoc.version=2.5.0 "
fi
```

© 2026 NTT DATA Japan Corporation

OpenJDKパッケージへの依存関係

- OpenJDKのパッケージはLinuxの各OSディストリビュータや、サードパーティが提供している
  - 異なるメジャーバージョンは共存可能で、alternativesで切り替え可能になっていることが多い
- Bigtopのミドルウェアのパッケージには、JDKパッケージへの依存関係を付けず、ユーザが選べるようにしている
  - (同様の理由でPythonにも依存関係を付けていない)

© 2026 NTT DATA Japan Corporation

ビルド用のOpenJDKセットアップ

- bigtop_toolchainのPuppetマニフェストでセットアップ可能
- 基本はOSバンドルを使うが、古いバージョンが提供終了しているケースで、別のOpenJDKディストリビューションで代用
- 最近まで全部OpenJDK 8でやっていたが、今後アップデートが必要な部分

bigtop_toolchainのPuppetマニフェスト(jdk.pp)から抜粋:

```
case $::operatingsystem {
   /Debian/: {
     include apt

     package { 'jdk8' :
       name => 'temurin-8-jdk',
       ensure => present,
     }
   }
   /Ubuntu/: {
     include apt

     package { 'jdk8' :
       name => 'openjdk-8-jdk',
       ensure  => present,
     }
   }
   /(CentOS|Amazon|Fedora|RedHat|Rocky|openEuler)/: {
     package { 'jdk8' :
       name => 'java-1.8.0-openjdk-devel',
       ensure => present
     }
```

© 2026 NTT DATA Japan Corporation

ランタイム用のOpenJDKセットアップ

- ビルド用のbigtop_toolchainと同様の資材でセットアップ可能
- 実行時にインストールされているJDKを探してJAVA_HOMEをセットする、補助スクリプトも提供
- 最近まで全部OpenJDK 8でやっていたが、今後アップデートが必要な部分

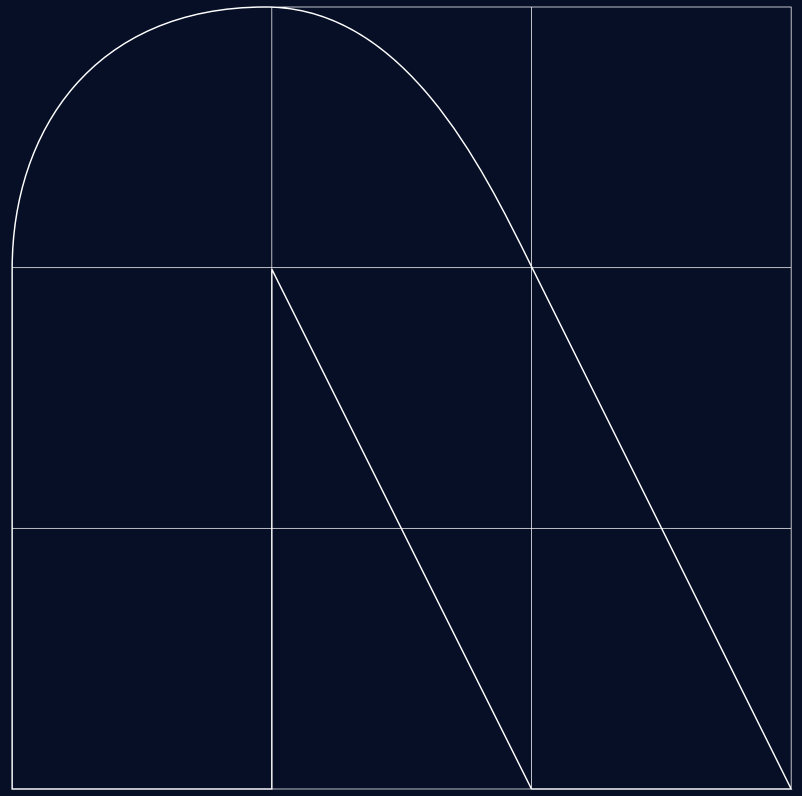
bigtop-detect-javahomeのスクリプトから抜粋:

```
OPENJAVA8_HOME_CANDIDATES=(
    '/usr/lib/jvm/java-1.8.0-openjdk-amd64'
    '/usr/lib/jvm/java-1.8.0-openjdk-ppc64el'
    '/usr/lib/jvm/java-1.8.0-openjdk'
    '/usr/lib64/jvm/java-1.8.0-openjdk-1.8.0'
    '/usr/lib/jvm/temurin-8-jdk'
)

MISCJAVA_HOME_CANDIDATES=(
    '/Library/Java/Home'
    '/usr/java/default'
    '/usr/lib/jvm/java'
    '/usr/lib/jvm/jre'
    '/usr/lib/jvm/default-java'
    '/usr/lib/jvm/java-openjdk'
    '/usr/lib/jvm/jre-openjdk'
    '/usr/lib/jvm/java-oracle'
)
```

© 2026 NTT DATA Japan Corporation

まとめ



© 2026 NTT DATA Japan Corporation

まとめ

- 言語処理系のアップグレード対応は、規模の大きなミドルウェアでは一苦労
- 特にHadoopはJava 8からの脱却にとても時間を要した
- Javaのdependency hellは、アップグレードも難しくする
- LLMの登場と進化で、巨大なコード修正や、車輪の再発明も、いまでは容易になった?
- 書き換えたコードを確認して、テストするのは、それでも大変
- 地味だけどオープンソースを長く安定して使い続けるために、重要な取り組み

© 2026 NTT DATA Japan Corporation


NTT DATA