



**NTT DATA**

**Open Source Conference 2025 Osaka**

**Apache Bigtop 3.4 (仮)**

2025/01/25

田中 正浩、岩崎 正剛

日本Hadoopユーザ会 / 株式会社NTTデータグループ

© 2025 NTT DATA Japan Corporation

本資料に記載されている会社名、商品名、又はサービス名は、各社の登録商標または商標です。

# Apache Bigtopの概要

# 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。』
- 一言で言うと「Hadoop・Spark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OSS」です。



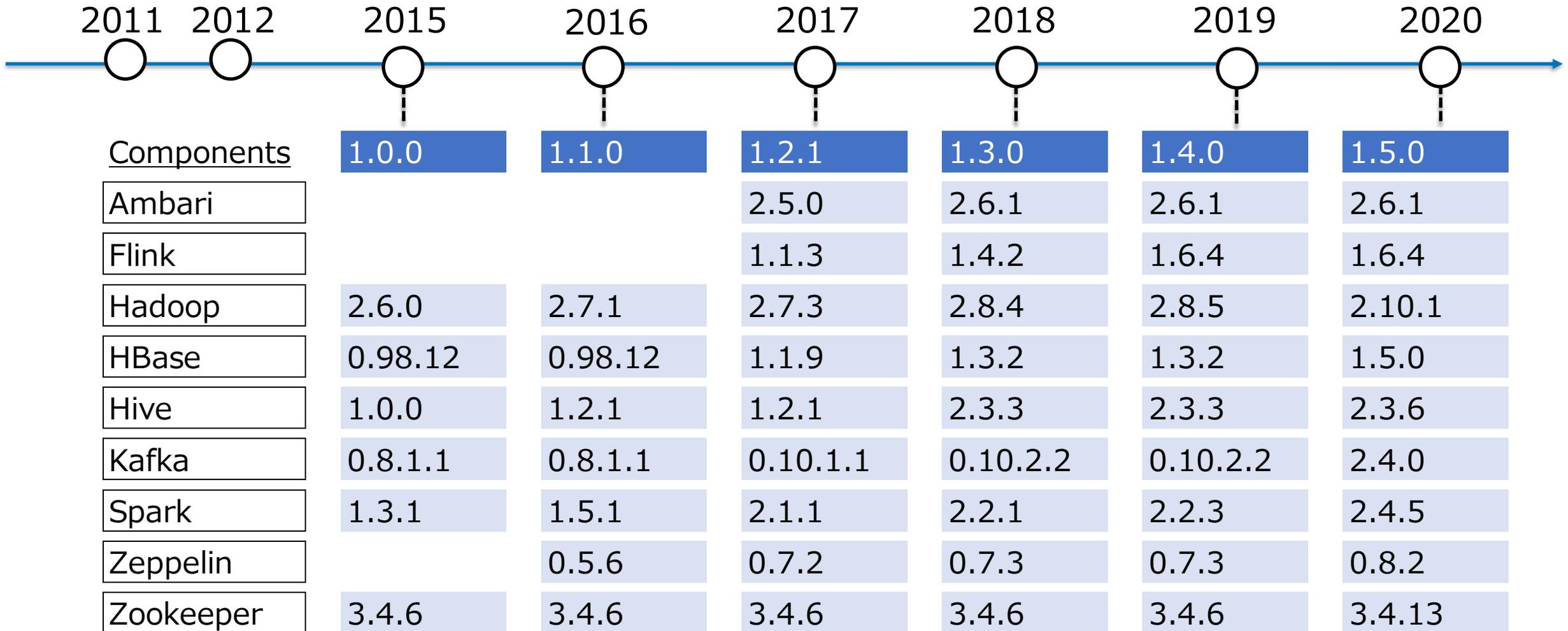
# データ基盤構築のためにBigtopが提供する機能

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

# Apache Bigtop の歴史

Cloudera社からApacheソフトウェア財団（以下ASF）に、incubatorプロジェクトとして寄贈される

Apache incubator プロジェクトからトップレベルプロジェクトに昇格

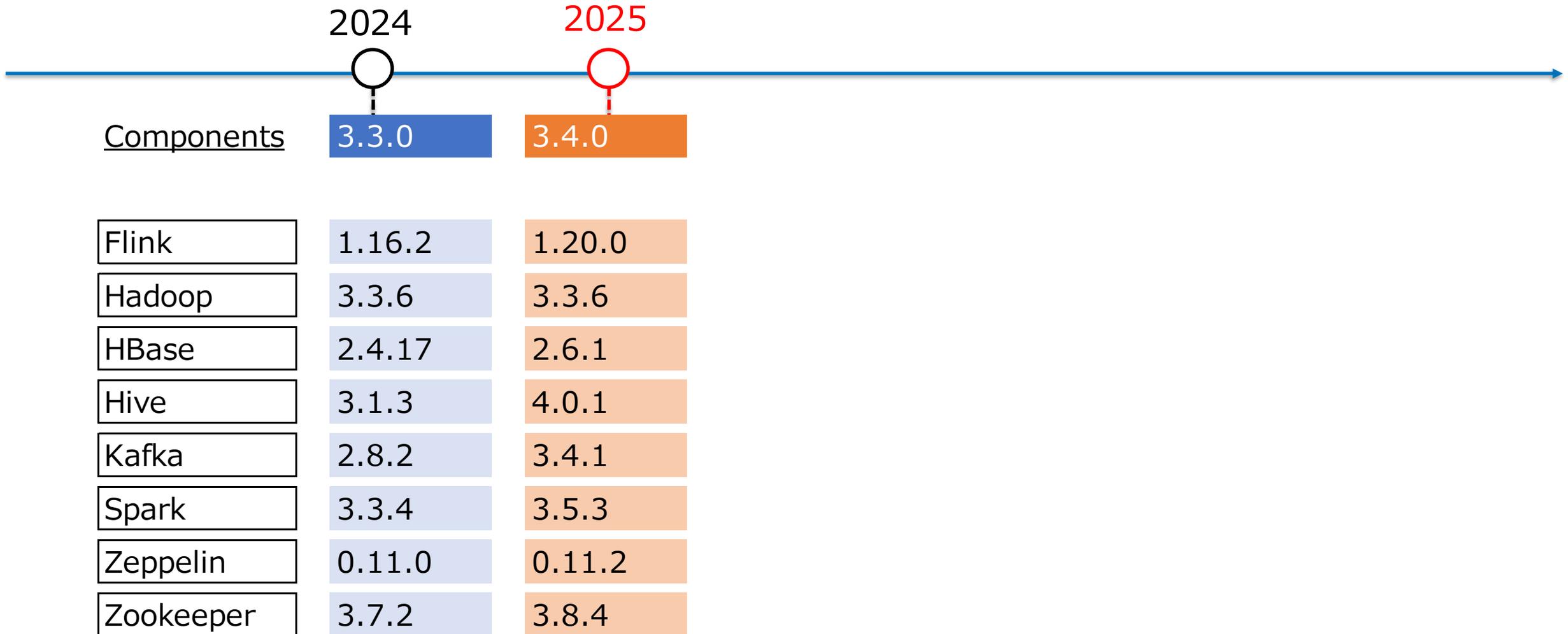


# Apache Bigtop の歴史

A horizontal timeline arrow points from left to right, with circular markers for the years 2021, 2022, 2023, and 2024. Below the timeline, a table lists the versions of various components for each year. The 'Components' row is highlighted in dark blue, while the other rows are light blue.

Components	2021	2022	2023	2024
Ambari	2.7.5	2.7.5	2.7.5	2.8.0
Flink	1.11.3	1.11.6	1.11.6	1.15.3
Hadoop	3.2.2	3.2.2	3.2.3	3.3.5
HBase	2.2.6	2.2.6	2.4.8	2.4.13
Hive	3.1.2	3.1.2	3.1.2	3.1.3
Kafka	2.4.1	2.4.1	2.8.1	2.8.1
Spark	3.0.1	3.0.1	3.1.2	3.2.1
Zeppelin	0.9.0	0.10.0	0.10.0	0.10.1
Zookeeper	3.4.14	3.4.14	3.5.9	3.5.9

# Apache Bigtop の歴史



# Bigtop 3.3のソフトウェアスタック

コンポーネント	バージョン	説明
Alluxio	2.9.3	ストレージ仮想化
Ambari	2.7.5	クラスタ構築・管理
Flink	1.16.2	ストリーム処理系
GPDB	6.23.1	MPP
Hadoop	3.3.6	分散ファイルシステム・ 並列分散処理基盤
HBase	2.4.17	分散KVS
Hive	3.1.3	Hadoop用クエリ処理系
Kafka	2.8.2	分散メッセージブローカ
Livy	0.8.0	Spark用RESTゲートウェイ

コンポーネント	バージョン	説明
Oozie	5.2.1	ジョブスケジューラ
Phoenix	5.1.3	HBase用クエリ処理系
<b>Ranger</b>	<b>2.4.0</b>	<b>認可</b>
Solr	8.11.4	全文検索エンジン
Spark	3.3.4	並列分散処理エンジン・ ストリーム処理系
Tez	0.10.2	並列分散処理エンジン
YCSB	0.17.0	ベンチマークツール
Zeppelin	0.11.0	ノートブック
Zookeeper	3.7.2	分散コーディネーション

## ※3.3 で削除されたコンポーネント

Ambari, Oozie, YCSB

# Bigtop 3.4のソフトウェアスタック(仮)

コンポーネント	バージョン	説明
Alluxio	2.9.6	ストレージ仮想化
Flink	1.20.0	ストリーム処理系
GPDB	6.23.1	MPP
Hadoop	3.3.6	分散ファイルシステム・ 並列分散処理基盤
HBase	2.6.1	分散KVS
Hive	4.0.1	Hadoop用クエリ処理系
Kafka	3.4.1	分散メッセージブローカ
Livy	0.8.0	Spark用RESTゲートウェイ

コンポーネント	バージョン	説明
Phoenix	5.2.1	HBase用クエリ処理系
<b>Ranger</b>	<b>2.5.0</b>	<b>認可</b>
Solr	8.11.4	全文検索エンジン
Spark	3.5.3	並列分散処理エンジン・ ストリーム処理系
Tez	0.10.4	並列分散処理エンジン
Zeppelin	0.11.2	ノートブック
Zookeeper	3.8.4	分散コーディネーション

## ※3.4 で削除されるコンポーネント

GPDB

# Bigtop 3.4 のソフトウェアスタックにおける主な変更点

- GPDB削除
  - GPDBの開発が終了した(ように見える)ため。
- Hiveのメジャーバージョンアップ
  - Hadoop 3.3対応や、メモリリーク対応のパッチを多数バックポートしてたのが不要に
- Sparkのメジャーバージョンアップ(仮)
  - ZeppelinやLivyが対応できていなければ、スキップするかも。

# Bigtop 3.4のサポートするOSディストリビューション

- 2025/1時点ではまだ未確定
- 維持コストの削減のため同系列で複数LTSをサポートしない方向性
- CentOS 7は2024/06/30でEOL
  - CentOS Streamへの路線変更やAmbariとの兼ね合いで長く残っていた

Linux distro	Bigtop 3.3.0	Bigtop 3.4.0
CentOS	7	-
Rocky Linux	8	<b>9</b>
Fedora	38	<b>40</b>
<b>openEuler</b>	22.03	22.03
Debian	11	<b>12</b>
Ubuntu	22.04	<b>24.04</b>

# Bigtop 3.3のサポートするCPUアーキテクチャ

- x86\_64とaarch64はAWS等を利用して開発、テストできる。
  - 中国系(企業)のcontributorはaarch64利用が多い印象。
- ppc64le環境でのビルドはベストエフォートの。
  - CI環境も乏しい(1ノード)ため、対応を落としがち。
- ネイティブコードを含むJavaライブラリで、x86\_64だと、aarch64やppc64leの個別対応が必要になる。

CPU アーキテクチャ	Bigtop 3.2.1	Bigtop 3.3.0
x86_64	○	○
aarch64	○	○
ppc64le	○(Rocky8を除く)	○(Rocky8、???を除く)

## プラットフォームと互換性

# なぜLinuxなのか

- BigtopがサポートするのはLinuxの主要ディストリビューション
  - 主要プロダクトの多くがJavaベースではあるものの
- HadoopのサーバプラットフォームとしてはLinuxが前提
  - 主要部分はJava: 多くのプラットフォームで動く
  - CLIはbash (>=3.0): 多くのプラットフォームで動く
    - Windows用のhadoop.cmdも存在するがもうメンテされていない
  - Cで実装されたネイティブコード(libhadoop.so)はLinux依存
    - オプションな機能や性能向上を提供: 特にHDFSのサーバ側で重要
    - Linuxのシステムコール前提
- (Hadoopの)開発者はUbuntu on x86\_64が多数派という印象
- HadoopのCI環境もUbuntu on x86\_64
- それ以外のプラットフォーム(RHEL系、aarch、ppc64le)の問題が見落とされがち

# nativeライブラリのポータビリティ

- Hadoopプロジェクト配布版binary tarballにもlibhadoop.soは入っているが、どのLinux環境でも動くとは限らない

# binary tarballは特定の環境でビルドしたものをoptionalに提供

# 例えばHadoop 3.2.2の場合はUbuntu 16.04でビルドしたもの

glibcのバージョンミスマッチでnativeライブラリが使えない状態:

```
$ hadoop checknative
Native library checking:
hadoop:  false
zlib:    false
...

$ ldd libhadoop.so
./libhadoop.so: /lib64/libc.so.6: version `GLIBC_2.14' not found (required by ./libhadoop.so)
        linux-vdso.so.1 => (0x00007fffc0959f000)
        ...
```

# Javaのバージョン

- パッケージング対象プロダクトの大半はJava/JVM言語で実装
- Bigtopは各種Linuxディストリビューション提供のOpenJDK **8**を(ビルドに)利用
- 主にHadoop開発環境のJava 11移行が難航しているため...
  - Jerseyのupgradeがblocker([HADOOP-15984](#))
    - 影響を受けるコードが非常に多い
    - Jersey 1とJersey 2(以降)がJAX-RSの異なるバージョンに基づき、両方を共存させてサブモジュールごとに段階的にアップデートできない。
    - [upgradeのPR](#)がつい最近merge。Hadoop 3.5.0で解決済と期待。
- Hadoop 3.3(以降)は、実行環境としてはJava 11をサポート
- 他のJavaプロダクトもJava 11上で実行可能(なはず)

# JavaのAPI

- 公開APIと内部APIの分離が不完全
  - サブモジュール間で参照するためにpublicにされたクラス/メソッドが多い
  - 下流のプロダクトがテストのために内部APIを利用できてしまう
  - 各プロダクトの開発者は、下流のプロダクトを意識しきれない
- プロダクトXの内部APIを変更すると、依存するプロダクトYのビルドが壊れたりしがち
  
- 公開APIを示すためのアノテーションを利用する例もあるが、強制力はない
  - @InterfaceAudience.PublicなものだけJavadocが出力される
  - @InterfaceAudience.Privateだとマイナーバージョンアップでも変更されうる

InterfaceAudienceアノテーションの例:

```
@InterfaceAudience.LimitedPrivate({ "MapReduce", "HBase" })  
@InterfaceStability.Unstable  
public class DistributedFileSystem extends FileSystem  
    implements KeyProviderTokenIssuer, BatchListingOperations {
```

# Javaライブラリの依存関係(とdependency hell)

- 同一パッケージ名、同一クラス名のライブラリを複数バージョン混在させることは不可能
- ダメなパターンの例:
  - プロダクトXがプロダクトYに依存
  - プロダクトXがhoge-1.0に依存し、hoge-1.5に無いAPIを利用
  - プロダクトYがhoge-1.5に依存し、hoge-1.0に無いAPIを利用
- 汎用的なライブラリは特にプロダクト間で競合しがち
  - SLF4J, Log4j
  - commons-logging, commons-cli, commons-httpclient
  - Jackson
  - Guava
  - Netty, Jetty, Jersey
  - protobuf-java
  - ZooKeeper, Curator
- 脆弱性対応のために依存ライブラリをアップグレードすると、非互換な変更が入ることも

# Bigtopの役割

- ちゃんと機能するプロダクトのバージョンの組み合わせは自明ではない
  - Bigtopは「機能するバージョンの組み合わせ」を選定
  - 場合によってはパッチをあてて辻褄を合わせる
- CPUアーキテクチャ、OSディストリビューションごとの調整も必要
  - Bigtopはそれぞれの環境でビルドしてパッケージングしてテストする仕組みを提供

## Bigtopの基本素材

# Bigtopのソースコード

- ソースコードはGitHubにある
  - # 2019年4月からASF(Apache Software Foundation)全体として  
# <https://blogs.apache.org/foundation/entry/the-apache-software-foundation-expands>
- 開発の管理はJIRA
  - <https://issues.apache.org/jira/projects/BIGTOP>
  - issueを作ってパッチを添付するかpull requestのリンクを貼る

Bigtopのソースコードの取得:

```
$ git clone https://github.com/apache/bigtop
$ cd bigtop
```

# Bigtopのタスク

- Gradleを利用して各種処理を実行する

タスク一覧の表示:

```
$ ./gradlew tasks

> Task :tasks

-----
Tasks runnable from root project - Bigtop
-----

Apache Creadur tasks
-----
rat - Runs Apache Rat checks

Build tasks
-----
assemble - Assembles the outputs of this project.
...
```

# Bigtop BOM

- 部品表: 対象プロダクトとそのバージョンなどが記述されている
- GroovyのConfigSlurperの書式

```
$ cat bigtop.bom
bigtop { // *the name should be change: the parsing code depends on it*
  version = "STACK-VERSION" // *required*
  stack { // *required* Fundamental properties of the Stack: JDK, SDK, GDK, etc
    'jdk' { version = '1.8'; version_base = version }
    'scala' { version = '2.10.4'; version_base = version }
  }
  ...
  components {
    'hadoop' {
      name      = 'hadoop'
      relNotes = 'Apache Hadoop'
      version { base = '3.2.2'; pkg = base; release = 1 }
      tarball { destination = "${name}-${version.base}.tar.gz"
                source      = "${name}-${version.base}-src.tar.gz" }
      url      { download_path = "/$name/common/$name-${version.base}"
                site         = "${apache.APACHE_MIRROR}/${download_path}"
                archive      = "${apache.APACHE_ARCHIVE}/${download_path}" }
    }
  }
}
```

bigtop\_toolchain

# bigtop\_toolchain

- プロダクト群をビルドするための環境をセットアップする資材
  - Puppetをインストールするスクリプト
  - ビルドツール群をインストールPuppetマニフェスト
    - JDK, Maven, GCC, CMake, Protocol Buffers, lib\*-devel, ...
- OS提供パッケージだとバージョンが合わないものは個別にセットアップ
- プラットフォーム差分に対応するためのワークアラウンドも提供
  - 例) aarch64対応のバックポートパッチを当ててprotobuf-2.5.0をビルド

bigtop\_toolchainによるビルド環境のセットアップ:

```
$ sudo bigtop_toolchain/bin/puppetize.sh  
$ ./gradlew toolchain
```

# Protocol Buffers

- 最近のHadoopやHBaseはprotobuf-maven-pluginを利用
  - protocのバイナリをビルド時に自動的にダウンロードして利用
- バイナリが未提供のプラットフォーム/バージョンではうまくいかない
  - com.google.protobuf:protoc < 3.5.0 ではaarch64用バイナリが未提供
  - com.google.protobuf:protoc < 3.7.0 ではppc64le用バイナリが未提供
- bigtop-toochainはワークアラウンドとしてProtocol Buffersにパッチを当ててビルド
  - ビルド時にmvn install:install-fileでローカルアーティファクトをインストールして利用
- (FYI)Hadoop 3.3.0でprotobuf-2.5.0からprotobuf-3.7.1に移行
  - だがしかしTezはまだprotobuf-2.5.0を利用

do-component-build内でローカルビルドしたprotocからアーティファクトをインストール:

```
if [ $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/bin/protoc
fi
```

bigtop-packages

# bigtop-packages

- プロダクトをビルドしてパッケージングする
- 「プロダクト名-pkg」というGradleタスクで実行
- .rpmと.debの2種類に対応
  - ビルド環境に応じたパッケージが作られる
    - Red Hat系なら.rpm
    - Debian系なら.deb

パッケージングタスクの実行例:

```
$ ./gradlew hadoop-pkg
$ ./gradlew zookeeper-pkg
$ ./gradlew bigtop-jsvc-pkg bigtop-utils-pkg bigtop-groovy-pkg
```

# パッケージング資材

- 共通のビルド手順とインストール手順を.debと.rpm用の資材から呼び出して使う
  - ビルド手順: do-component-build
  - インストール手順: install\_プロダクト名.sh

パッケージング資材の配置:

```
bigtop-packages
├── src
│   ├── common
│   │   ├── hadoop
│   │   │   ├── do-component-build
│   │   │   └── install_hadoop.sh
│   ├── deb
│   │   ├── hadoop
│   │   │   ├── control
│   │   │   └── rules
│   └── rpm
│       ├── hadoop
│       │   └── SPECS
│       │       └── hadoop.spec
```

# Bigtopパッケージングにおけるパッチ適用

- パッケージング時にpatch.\*diffという名前のパッチファイルを適用
- 各プロダクトのリリース版そのままでは解決できない問題に対応
  - 特定プラットフォームでビルドが失敗する問題への対処
  - パッケージング対象バージョンに含まれない修正のバックポート
  - 互換性を壊す変更のrevert
  - プロダクト間のdependenciesのつじつま合わせ
- patchを書いたらなるべく本家にfeedback

Bigtopが適用するpatchの例:

```
$ find bigtop-packages/src/common/hadoop -name 'patch*.diff' | sort
bigtop-packages/src/common/hadoop/patch0-revert-HADOOP-16598.diff
bigtop-packages/src/common/hadoop/patch1-HADOOP-15939.diff
bigtop-packages/src/common/hadoop/patch2-exclude-spotbugs-annotations.diff
bigtop-packages/src/common/hadoop/patch3-fix-broken-dir-detection.diff
```

# パッケージリポジトリの作成

- \$BIGTOP\_HOME/output 下に出力されるパッケージファイルでリポジトリを作成する
- repoタスクで実行
- その環境に応じたリポジトリが作られる (Red Hat系ならYumの、Debian系ならAptの)

リポジトリの作成(CentOS 8で実行した場合の例):

```
$ ./gradlew repo
$ tree output
...
├── hadoop
│   ├── hadoop-3.2.2-1.el8.src.rpm
│   └── x86_64
│       ├── hadoop-3.2.2-1.el8.x86_64.rpm
│       ├── hadoop-client-3.2.2-1.el8.x86_64.rpm
│       └── hadoop-conf-pseudo-3.2.2-1.el8.x86_64.rpm
...
├── hadoop-hdfs-3.2.2-1.el8.x86_64.rpm
...
├── repodata
...
│   ├── aa0ff69b812187315d6825fcf7d862c326ac1ba42bf9625e57b826cd6462a68c-filelists.xml.gz
│   ├── cb8d35ada5c22a89e1f65a62229acde45b4175d5281a513fc6e1190d0ce4544c-filelists.sqlite.bz2
│   ├── f2077b7afd203a11428c21ca3c41f02a786fc2f514888274f12f514673d4f88f-primary.xml.gz
│   └── repomd.xml
```

# Dockerコンテナを利用したパッケージング

- 「プロダクト名-pkg-ind」というGradleタスクで実行(in Docker container)
- ひとつの環境で全OSディストリビューションのパッケージをビルドできる
- Docker Hubのbigtop/slavesのイメージを利用
  - -POS=centos-7 -Pprefix=trunk だとtrunk-centos-7タグになる(x86\_64の場合)
  - fedora-38やubuntu-22.04は-Pdocker-run-option="--privileged"が追加が必要
- CPUアーキテクチャはホスト環境に応じて決まる
  - aarch64のマシン上でやるとtrunk-centos-7-aarch64タグ
- 毎回ライブラリをすべてダウンロードすると遅い(BIGTOP-3801)
  - -Pmvn-cache-volume=true でローカルリポジトリをvolumeに保存して再利用
- repo-indタスクは、repoタスクをDockerコンテナで実行
  - Docker provisioner(後述)と組み合わせてテストに利用

パッケージングタスクの実行例:

```
$ ./gradlew hadoop-pkg-ind -POS=rockylinux-8 -Pprefix=trunk -Pmvn-cache-volume=true
$ ./gradlew repo-ind -POS=rockylinux-8 -Pprefix=trunk
```

# ビルド済パッケージのリポジトリ定義ファイル

[Bigtop の公式サイト](#) 以下のリンクを辿って取得

Apache Bigtop | Releases | Project Information | Downloads | ASF

[Latest release](#)

## Apache Bigtop

Bigtop is an Apache Foundation project for Infrastructure Engineers and Data Scientists to build and manage source big data components. Bigtop supports a wide range of components/projects, including Hadoop, Hive, Pig, and Tez.

Packaging	Smoke testing
Bigtop packages Hadoop RPMs and DEBs, so that you can manage and maintain your Hadoop cluster.	Bigtop provides an integrated framework, alongside a suite of tools for testing and validation.

Bigtop support many Operating Systems, including Debian, Ubuntu, CentOS, Fedora, openSUSE, and SLES.

Bigtop includes tools and a framework for testing at various levels (packaging, platform, runtime) not just the individual components.

### Download latest release of Bigtop

[Latest release](#)

Apache Bigtop | Releases | Project Information

You can download the latest most stable release of Bigtop for processing of large amounts of data.

- The latest release of Apache Bigtop software for processing of large amounts of data is [Bigtop 3.1.1](#) (pgp sha256 sha512)
- Repositories of installable binary packages built from the latest release of Bigtop software for processing of large amounts of data are available at [Installable binary artifacts](#)

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">centos-7/</a>	2022-08-07 04:07	-	
<a href="#">debian-10/</a>	2022-08-07 04:07	-	
<a href="#">debian-11/</a>	2022-08-07 04:07	-	
<a href="#">fedora-35/</a>	2022-08-07 04:07	-	
<a href="#">rockylinux-8/</a>	2022-08-07 04:07	-	
<a href="#">ubuntu-18.04/</a>	2022-08-07 04:07	-	
<a href="#">ubuntu-20.04/</a>	2022-08-07 04:07	-	
<a href="#">GPG-KEY-bigtop</a>	2022-08-02 23:55	3.8K	
<a href="#">GPG-KEY-bigtop.asc</a>	2022-08-02 23:55	833	

Linux ディストリビューション別に、rpm 用の .repo ファイルや deb 用の .list ファイルがダウンロード可能

# ビルド済パッケージのインストール手順(RPM)

- RPMパッケージを用いたコンポーネントのインストールは以下の3ステップ

## 1. リポジトリ設定ファイルをダウンロード・配置

```
$ sudo wget -P /etc/yum.repos.d -q \  
https://dlcdn.apache.org/bigtop/bigtop-3.1.1/repos/centos-7/bigtop.repo
```

## 2. GPG キーをインポート

```
$ sudo rpm --import \  
https://dlcdn.apache.org/bigtop/bigtop-3.1.1/repos/GPG-KEY-bigtop
```

```
$ sudo yum install hadoop
```

## 3. リポジトリ情報のアップデート・コンポーネントのインストール

```
$ hadoop version  
Hadoop 3.2.4
```

# ビルド済パッケージのインストール手順(DEB)

- debパッケージを用いたコンポーネントのインストール手順も同様

```
$ sudo wget -P /etc/apt/sources.list.d -q \  
https://dlcdn.apache.org/bigtop/bigtop-3.1.1/repos/ubuntu-20.04/bigtop.list
```

1. リポジトリ設定ファイルをダウンロード・配置

```
$ curl -s https://dlcdn.apache.org/bigtop/bigtop-3.1.1/repos/GPG-KEY-bigtop | \  
sudo apt-key add -
```

2. GPG キーをインポート

```
$ sudo apt-get update
```

```
$ sudo apt-get install hadoop
```

3. リポジトリアップデート・コンポーネントインストール

```
$ hadoop version  
Hadoop 3.2.4
```

bigtop-deploy

# bigtop-deploy

- パッケージの導入でインストール作業は省力化できたが、以下の問題は解決されていない
  - サーバの台数が多い場合、1台1台インストールのコマンドを投入するのは大変
  - パッケージのインストール後も、各ソフトウェアに適切な設定を施したり、サービスを起動したりといった作業が必要
- Bigtop はこれらの作業を自動化するためにPuppet マニフェストを提供
  - [Puppet](#) はRubyで実装された構成管理ツール
  - Puppet manifest はシステムの「あるべき姿」を宣言的に記述したDSL
    - 記述に従って Puppet がパッケージのインストールや設定、サービス起動などを行う
  - [Hiera](#)はキーバリュ型の設定管理システム
    - ユーザが環境に応じて変更したいパラメータをmanifestと独立に指定できる

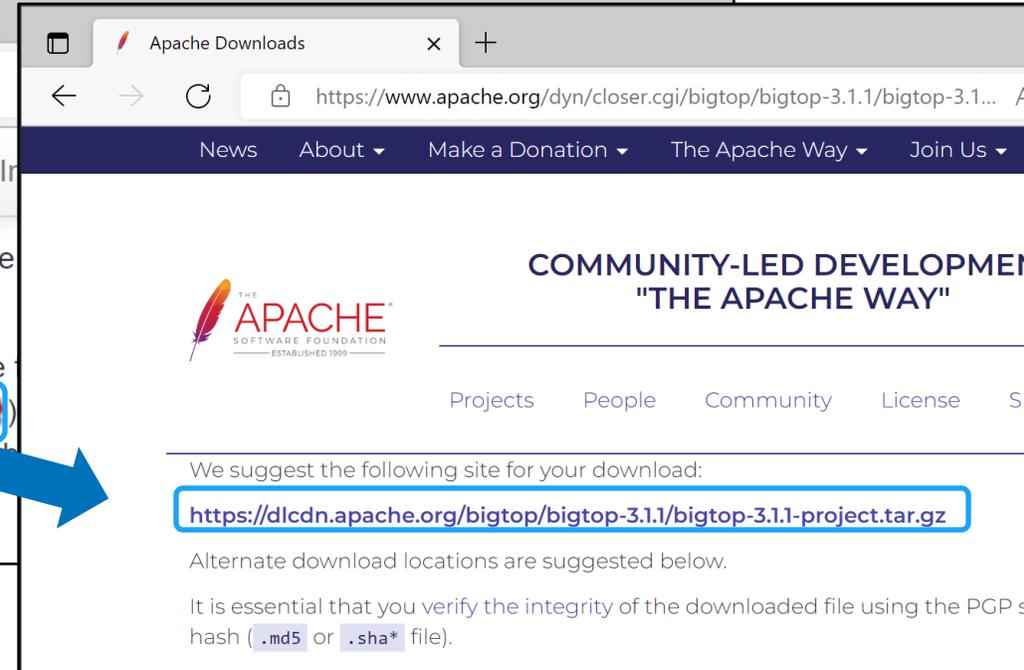
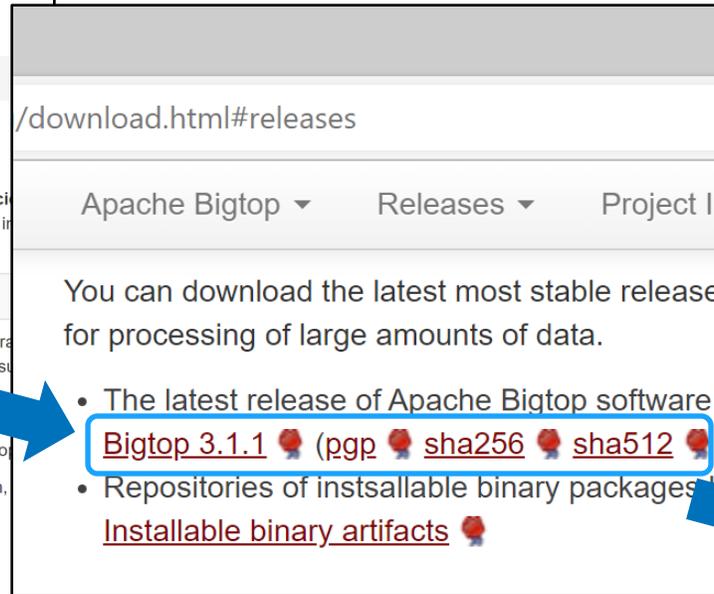
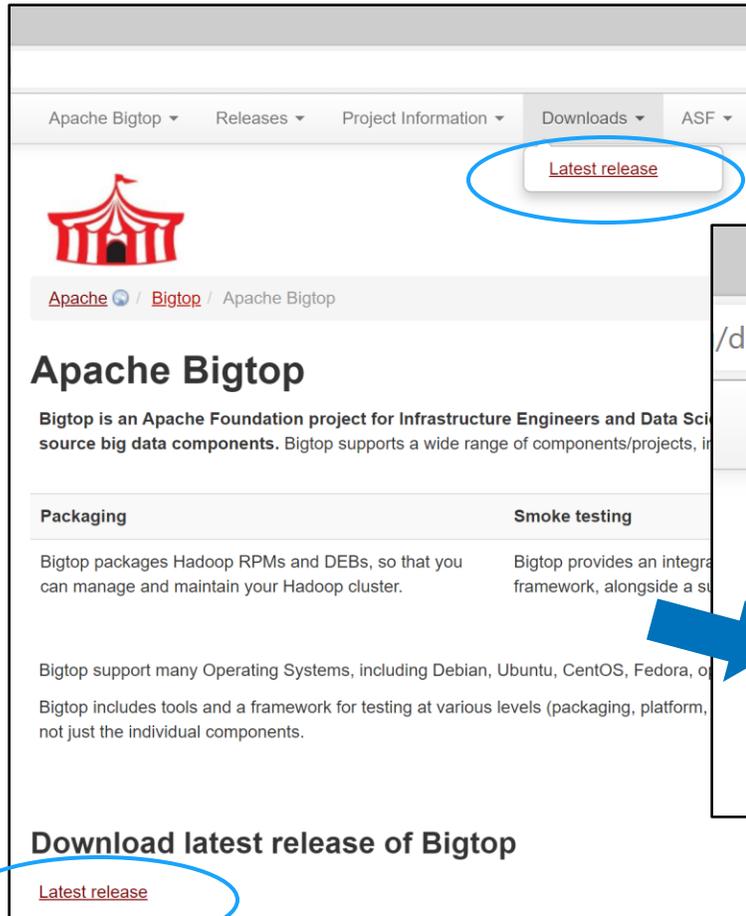
# bigtop-deploy/puppet

- Hadoopクラスタを構築するためのPuppetマニフェスト:

```
$ tree bigtop-deploy/puppet
bigtop-deploy/puppet
├── hieradata
│   ├── bigtop
│   │   └── cluster.yaml
│   └── ...
├── site.yaml
├── hiera.yaml
├── manifests
│   ├── bigtop_repo.pp
│   └── cluster.pp
├── ...
├── hadoop
├── ...
├── manifests
│   ├── init.pp
│   └── templates
│       ├── container-executor.cfg
│       └── core-site.xml
├── ...
```

# Puppet マニフェストのダウンロード

- Bigtop のサイトからsource tarballをダウンロード
- または<https://github.com/apache/bigtop>からソースコードを取得:



# Puppet マニフェストを用いたコンポーネントのデプロイ手順

- Puppet マニフェストを用いたコンポーネントのデプロイは、以下の4ステップ (次ページに続く)

```
$ curl -sLO https://dlcdn.apache.org/bigtop/bigtop-3.1.1/bigtop-3.1.1-project.tar.gz
$ tar xf bigtop-3.1.1-project.tar.gz
$ cd bigtop-3.1.1
$ sudo bigtop_toolchain/bin/puppetize.sh
$ vi bigtop-deploy/puppet/hieradata/site.yaml
$ cat bigtop-deploy/puppet/hieradata/site.yaml
---
bigtop::hadoop_head_node: "fqdn.of.head.node"

hadoop::hadoop_storage_dirs:
  - /data

hadoop_cluster_node::cluster_components:
  - hdfs
  - yarn
  - mapreduce
```

1. Bigtop の tarball をダウンロードし展開

2. 中に含まれるスクリプトを実行して Puppet をインストール

3. Hiera ファイルを編集。最低限必要な設定は左記の3つだが、インストールするコンポーネントに応じて必要な設定を記述する。

- bigtop::hadoop\_head\_node: Hadoop のマスターノードのFQDN
- hadoop::hadoop\_storage\_dirs: Hadoop のデータ格納先ファイルパス
- hadoop\_cluster\_node::cluster\_components: インストールするコンポーネントのリスト。例では HDFS, YARN, MapReduceをインストールしている。

# Puppet マニフェストを用いたコンポーネントのデプロイ手順 (続き)

```
$ sudo cp -r bigtop-deploy/puppet/hiera* /etc/puppet
$ sudo puppet apply --parser future \
  --modulepath=bigtop-deploy/puppet/modules:/etc/puppet/modules \
  bigtop-deploy/puppet/manifests
```

(略)

```
Notice: Finished catalog run in 65.99 seconds
```

```
$ echo $?
```

```
0
```

```
$ sudo jps
```

```
7873 ResourceManager
```

```
8722 DataNode
```

```
8486 NodeManager
```

```
8922 Jps
```

```
8251 NameNode
```

```
7775 WebAppProxyServer
```

```
8159 JobHistoryServer
```

4. Hiera ファイル群を所定のパスにコピーし、Puppet を実行  
※"--parser future"オプションはCentOS 7にのみ付与  
※--modulepathオプションに指定している"/etc/puppet/modules"の部分は  
ディストリビューションによって異なる(puppet moduleのインストール先を指定する)

※一連の手順を parallel-ssh (<http://code.google.com/p/parallel-ssh/>)  
などを用いて実行することで、クラスタ内の全サーバでデプロイを行うことが可能

provisioner

# Docker provisioner

- ローカルやステージング環境で確認したいケースは多々ある
  - 業務上・システム上の要件を満たせるか
  - 構築手順が正しいこと
  - ソフトウェアをバージョンアップで問題が生じないか
- Bigtop は コンテナ上でコンポーネントをデプロイできる Provisioner を提供
- 事前にDockerとdocker-composeのセットアップが必要

# Docker provisioner の使用方法

```
$ ./gradlew docker-provisioner -Pnum_instances=3 -POS=centos-7 -Pprefix=3.1.1 -Pstack=hdfs,yarn
```

...

3台の Docker コンテナ上 (OS は CentOS 7) に Bigtop 3.1.1 の HDFS, YARN をインストールするコマンドを実行

```
Pulling bigtop (bigtop/puppet:3.1.1-centos-7)...
```

```
3.1.1-centos-7: Pulling from bigtop/puppet
```

```
Digest: sha256:42f20509d1b9460b753b70f6e21882699de3a488f93af28375c466181bc8727a
```

```
Status: Downloaded newer image for bigtop/puppet:3.1.1-centos-7
```

```
Creating 20220929_115113_r20838_bigtop_1 ... done
```

```
Creating 20220929_115113_r20838_bigtop_2 ... done
```

```
Creating 20220929_115113_r20838_bigtop_3 ... done
```

(ローカルに存在しなければ) Bigtop 3.1.1 の CentOS 7 イメージが pull され、そのイメージからコンテナが作成される

...

```
Notice: Roles to deploy: [resourcemanager, nodemanager, mapred-app, hadoop-client, namenode, datanode]
```

```
Notice: Roles to deploy: [nodemanager, mapred-app, datanode]
```

```
Notice: Roles to deploy: [nodemanager, mapred-app, datanode]
```

指定したコンポーネントの Puppet マニフェストを使って、コンテナ上へのデプロイが実行される

...

```
Notice: Finished catalog run in 231.67 seconds
```

```
BUILD SUCCESSFUL in 4m 39s
```

# Docker provisioner の使用方法 (続き)

プロビジョニングしたコンテナは、通常通り  
`docker ps` コマンドで確認できる

```
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS          NAMES
6df7279c442e   bigtop/puppet:3.1.1-centos-7       "/sbin/init"           4 minutes ago   Up 4 minutes
20220929_115113_r20838_bigtop_3
a37ba6399273   bigtop/puppet:3.1.1-centos-7       "/sbin/init"           4 minutes ago   Up 4 minutes
20220929_115113_r20838_bigtop_2
850c8ce4785a   bigtop/puppet:3.1.1-centos-7       "/sbin/init"           4 minutes ago   Up 4 minutes
20220929_115113_r20838_bigtop_1
```

```
$ docker exec -it 20220929_115113_r20838_bigtop_1 bash
[root@850c8ce4785a /]# jps
2416 Jps
1528 JobHistoryServer
1676 NodeManager
1212 ResourceManager
2156 DataNode
1086 WebAppProxyServer
1951 NameNode
[root@850c8ce4785a /]# exit
```

`docker exec` で任意のコマンドをコンテナ上で実行可能

```
$ ./gradlew docker-provision-destroy
...
BUILD SUCCESSFUL in 14s
1 actionable task: 1 executed
```

別のクラスタを起動する前には、既存のクラスタを破棄する必要がある

# Docker provisioner の使用方法 (続き)

```
$ ./gradlew tasks
```

```
...
```

```
Deployment tasks
```

```
-----
```

```
docker-provisioner -
```

```
Provision a Bigtop stack cluster on Docker container(s). Default to CentOS and 1 node.
```

```
Properties:
```

- Pconfig=[CONFIG\_FILE] (located under provisioner/docker/)
- Penable\_local\_repo
- Pimage=[DOCKER\_IMAGE] (overwrites -POS and -Pprefix)
- Pmemory=[4g|8g|...]
- Pnum\_instances=[NUM\_INSTANCES]
- Pnexus=[NEXUS\_URL] (NEXUS\_URL is optional)
- POS=[centos-7|fedora-35|debian-10|ubuntu-18.04|opensuse-42.3]
- Pprefix=[trunk|1.2.1|1.2.0|1.1.0|...]
- Prepository=[REPO\_URL]
- Prun\_smoke\_tests (run test components defined in config file)
- Psmoke\_tests=[COMPONENTS]
- Pstack=[COMPONENTS]

Docker provisioner はオプションが多いため  
`./gradlew tasks` コマンドで一覧を確認

# Docker provisionerの実装

- 中身はDockerコンテナを起動してPuppetでクラスタを構築するスクリプト
- スクリプト(provisioner/docker/docker-hadoop.sh)を直接使ってもよい
- cgroup v2のホストでは--docker-compose-yml docker-compose-cgroupv2.ymlを指定
  - cgroup v2未対応のsystemdはコンテナ内で動かない(例: centos-7 on Ubuntu 22.04)

Dockerコンテナを利用したデプロイの例:

```
$ cd provisioner/docker
$ ./docker-hadoop.sh \
  --create 1 \
  --image bigtop/puppet:trunk-ubuntu-22.04 \
  --docker-compose-yml docker-compose-cgroupv2.yml \
  --docker-compose-plugin \
  --memory 16g \
  --repo file:///bigtop-home/output/apt \
  --disable-gpg-check \
  --stack hdfs,yarn,mapreduce,spark

$ ./docker-hadoop.sh --list
      Name                Command      State  Ports
-----
20210124_122554_r12819_bigtop_1  /sbin/init  Up

$ ./docker-hadoop.sh --exec 1 /bin/bash
```

# Docker provisionerの仕組み

- `docker-hadoop.sh --create`で出力された`docker-compose.yml`を見ると分かる
- Bigtopのソースツリーを`/bigtop-home`にマウント
- Hieraの設定を生成して`puppet apply`

Docker provisionerの設定ファイル(の一部):

```
$ cat docker-compose.yml
bigtop:
  ...
  volumes:
  - ../../:/bigtop-home
  - ./config/hiera.yaml:/etc/puppet/hiera.yaml
  - ./config/hieradata:/etc/puppet/hieradata
  - ./config/hosts:/etc/hosts
  - /sys/fs/cgroup:/sys/fs/cgroup:ro

$ cat config/hieradata/site.yaml
bigtop::hadoop_head_node: b2d614be6039.bigtop.apache.org
hadoop::hadoop_storage_dirs: [/data/1, /data/2]
bigtop::bigtop_repo_uri: file:///bigtop-home/output
bigtop::bigtop_repo_gpg_check: false
hadoop_cluster_node::cluster_components: [hdfs,yarn,mapreduce]
hadoop_cluster_node::cluster_nodes: [172.17.0.3, 172.17.0.4, 172.17.0.2]
...
```

# Dockerクラスタの終了

- スクリプト経由でコンテナ停止と、一時ファイル等の削除
- クラスタ起動中の問題などでやり直す場合も、まず一度--destroyしてから

Dockerコンテナクラスタの停止:

```
$ ./docker-hadoop.sh --destroy
```

bigtop-tests

# smoke-tests

- パッケージをデプロイした状態で動くかざっくり確認するためのテスト
- おもにGroovyで実装されている
- CLIを実行して期待する出力が得られるかを確認というパターンが多い

smoke-testsのテストケースの例:

```
$ cat bigtop-tests/test-artifacts/hadoop/src/main/groovy/.../hdfs/TestTextSnappy.groovy
...
@Test
void testTextSnappy() {
    String cmd = "hadoop fs -text ${testDir}/${snappyFileName}"
    System.out.println(cmd)
    sh.exec(cmd)
    String output = sh.getOut().join("\n")
    logError(sh)
    String expected = "1\ttrafferty\t31\n2\tjones\t33\n3\tsteinberg\t33"
    System.out.println("Expected output:\n${expected}")
    System.out.println("Actual output:\n${output}")
    assertEquals("Incorrect output", expected, output)
}
```

# スモークテストの実行方法

- スモークテストの実行は以下の3ステップ。

```
$ curl -sLO https://downloads.apache.org/bigtop/bigtop-3.1.1/bigtop-3.1.1-project.tar.gz
$ tar xf bigtop-3.1.1-project.tar.gz
$ cd bigtop-3.1.1
```

1. Bigtop の tarball をダウンロードし展開

```
$ . /usr/lib/bigtop-utils/bigtop-detect-javahome
$ export HADOOP_CONF_DIR=/etc/hadoop/conf
```

2. JAVA\_HOME, およびテストの実行に必要な環境変数の定義. (コンポーネントごとに必要な環境変数は異なる. 確認するには展開したファイルの bigtop-tests/smoke-tests/コンポーネント名/build.gradle の doFirst メソッドを参照)

```
$ ./gradlew bigtop-tests:smoke-tests:yarn:test -Psmoke.tests --info
```

3. この例ではYARNのテストを実行

```
org.apache.bigtop.itest.hadoop.yarn.TestNode > testNodeBasic STANDARD_OUT
-list
-status
```

```
org.apache.bigtop.itest.hadoop.yarn.TestRmAdmin > testRmAdminBasic STANDARD_OUT
-help
-getGroups
```

...

# スモークテストの実行方法 (続き)

テストの詳細な結果を知りたい場合は  
これらのファイルを参照

```
Gradle Test Executor 2 finished executing tests.
```

```
> Task :bigtop-tests:smoke-tests:yarn:test
```

```
Finished generating test XML results (0.009 secs) into: /home/vagrant/bigtop-3.1.1/bigtop-tests/smoke-tests/yarn/build/test-results/test
```

```
Generating HTML test report...
```

```
Finished generating test html results (0.02 secs) into: /home/vagrant/bigtop-3.1.1/bigtop-tests/smoke-tests/yarn/build/reports/tests/test
```

```
Now testing...
```

```
:bigtop-tests:smoke-tests:yarn:test (Thread[Execution worker for ':',5,main]) completed. Took 7.453 secs.
```

```
BUILD SUCCESSFUL in 53s
```

# Docker provisionerでtestを実行

- Docker provisionerで起動したクラスタ上でテストを実行することもできる

起動済みのDockerコンテナを利用してsmoke testsを実行:

```
$ cd provisioner/docker  
$ ./docker-hadoop.sh --smoke-tests hdfs,yarn,mapreduce
```

## 開発コミュニティの動向

# Bigtop の最近の開発動向 (v3.4でリリースされる・・・かも?)

ここまでは、Apache Bigtopに関する既存の機能やそのアップデートについて話したが、最後にそれ以外で最近の開発動向について以下をピックアップ。

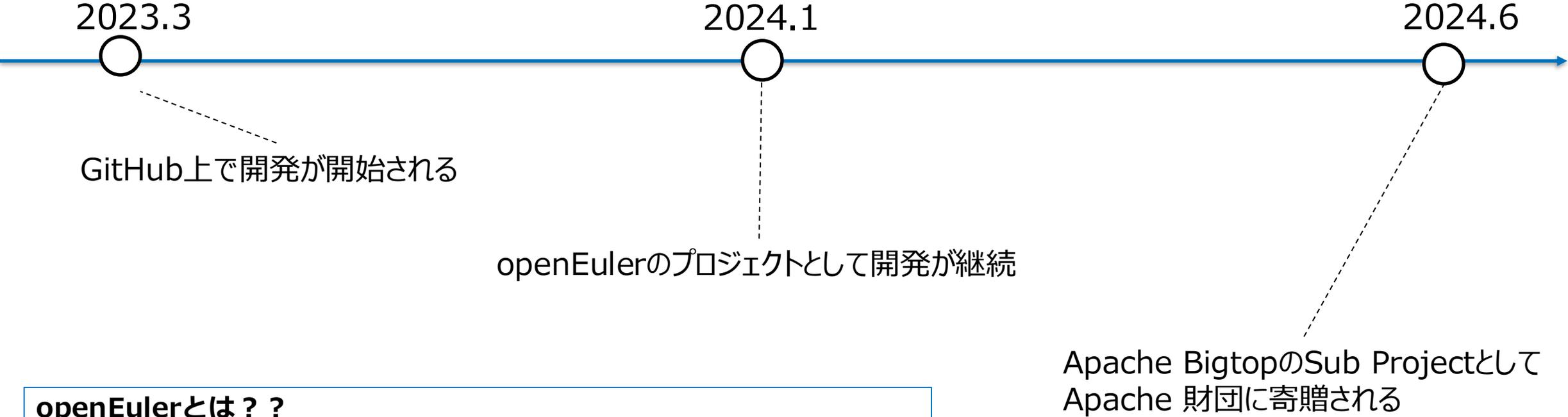
- Bigtop Manager
- レガシーからの脱却

# Bigtop Managerとは

- Apache Hadoopなどのビッグデータ処理関連OSSは、複数のサーバーに分散したコンポーネントを管理する必要がある
  - コンポーネントによっては起動の順番を守ることが推奨されたりと、オペレーションが難しい
  - Apache Bigtopにはデプロイ用のPuppet等も付属するが、Puppetの使い方を覚える必要がある
- Apache Bigtopコミュニティとしてコントローラブルで使いやすい管理ツールがあると便利



# Bigtop Managerの歴史

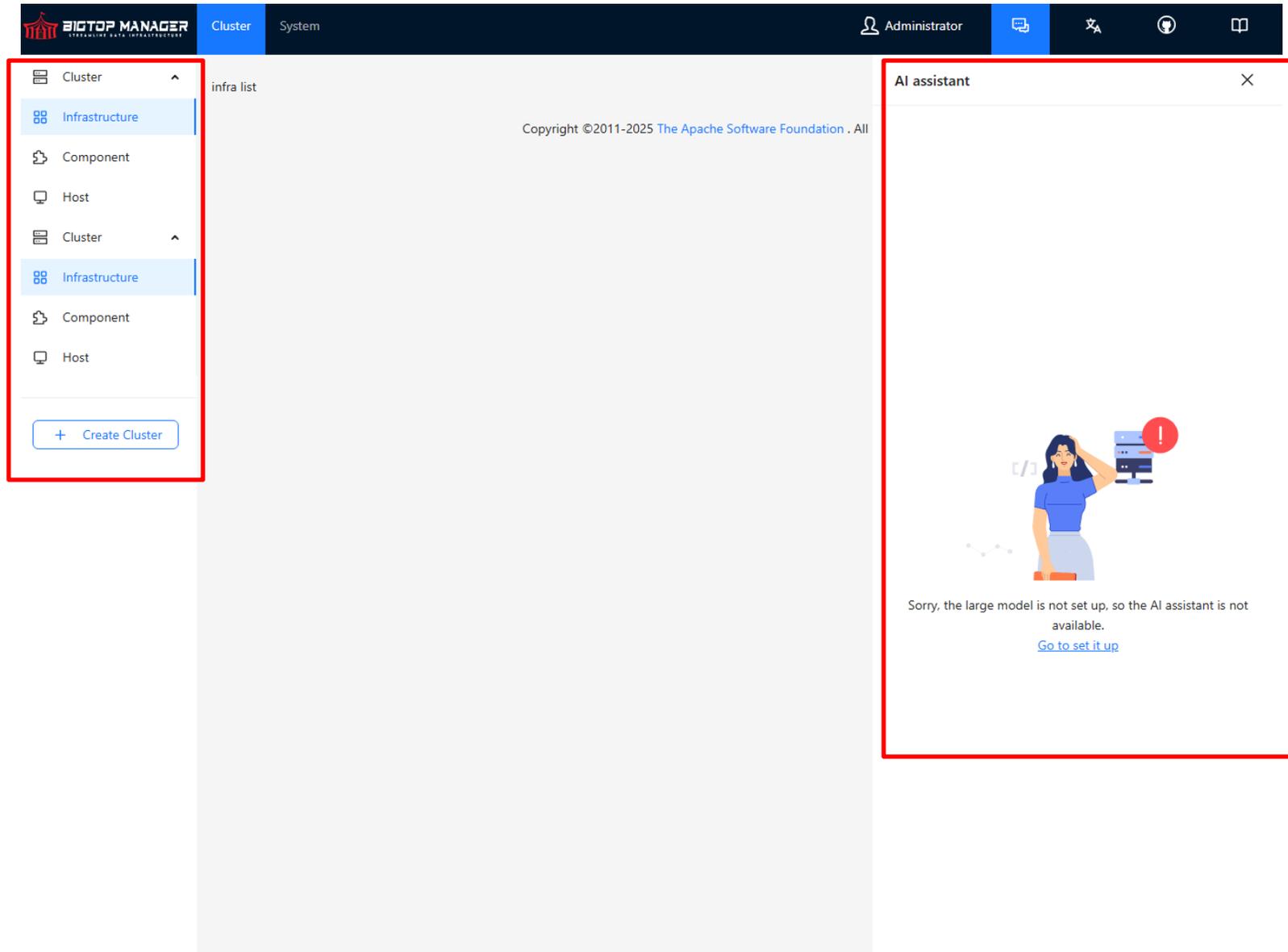


**openEulerとは??**  
Huaweiが主導するオープンソースのコミュニティ。CentOSベースのOSや、その上で動くソフトウェアの開発などが行われている。

# Bigtop Managerの特徴

- GUIの管理ツール
  - Apache Ambari, Cloudera Managerと類似
- モダンな監視ツールとの連携
  - Prometheus/Grafanaなど
- K8sサポート
  - 今のところは物理・仮想マシンへの対応を実施している
  - 将来的にはK8s環境へのデプロイ・管理も予定
- AIOpsへの対応
  - いわゆるLLM系のサービスと連携してクラスタの管理等ができるように

# Bigtop Manager外観



Cluster等の管理を行えるようになっている

外部のAIサービスと連携することができる

# Apache Ambariは怎么样了?

- ビッグデータ処理関連OSSを管理するツールとして、Apache AmbariというOSSがある
- Ambariは一時期Bigtopにも組み込まれていたが、色々な不都合がありDropされることに ([BIGTOP-3909](#))
  - Ambari側とBigtop側、それぞれの資材を更新する必要がある
  - AmbariとBigtopの開発サイクルがあまり合わず、開発の進行に影響が出る可能性がある
  - Ambariはかなり昔に作られたOSSでもあり、現代的なモニタリングツールの組み入れや新規機能の追加などが難しいという事情もある。
- AmbariにInspireされた形でBigtop Managerという独自の管理ツールがコミュニティに提案される ([メーリングリスト](#))

# Bigtop の最近の開発動向: レガシーからの脱却

Bigtopも歴史のあるプロダクトであり、過去の遺産もそれなりに存在する。

継続的にメンテナンスを実施しているが、その中でも最近取り組んでいる以下について紹介

- Python依存の削除
- Systemdへの移行

# Python依存の削除

- Bigtopが管理するコンポーネントの中には、Pythonを内部で利用するものが多い
  - 例: Apache Spark, Apache Ranger, Apache Hive
- Pythonを実行時に利用するだけでなく、ビルドするときにも利用するコンポーネントもある
- 特定のコンポーネントにおいて、Python2 (!! ) を利用するコードが残っていた
  - 歴史的にやむなく。。
- これまではパッケージングの依存関係としてPythonを含めていたが、これからはPythonを利用したいユーザが必要に応じて自身でインストールするように方針を変更

# Systemdへの移行

## 背景

- Hadoopなどのビッグデータ処理OSSでは、自身のプロセスを起動するためのスクリプトなどは提供しているが、ソフトウェアごとにバラバラの起動方式になっていた（例えば、ZooKeeperとHadoopの起動方法は異なるなど）
- BigtopはSysVinitで起動するためのラッパーを用意し、ユーザ向けに統一されたプロセス管理方式を提供している  
\$ service zookeeper-server start のような形で起動可能
- 今時のGNU/Linuxでは、SysVinitではなくsystemdを使うことが一般的だが、systemd-sysv-generatorの機能により、自動でSysVinit向けスクリプトがsystemd向けのunitファイルに変換され、systemdでも起動できていた。

## モチベーション

- Systemdの方でSysVinitのサポートがDeprecatedとなる (systemd [v256](#))

*> Support for System V service scripts is deprecated and will be removed in a future release. Please make sure to update your software *\*now\** to include a native systemd unit file instead of a legacy System V script to retain compatibility with future systemd releases.*

# Systemdへの移行の大変さ

本質的には、SysV スクリプトをSystemd Unitファイルに置き換えていけば良い。  
しかし、Bigtopならではの対応の大変さがある。

- Bigtop 3.3.0 時点で 18 コンポーネント存在
  - 比例して、Systemdで管理しないといけないプロセス数も多い
- 対応OSの多さ
  - Rocky Linux 9, Fedora 40, openEuler 22.03, Debian 12, Ubuntu 24.04 (予定)
  - 各OSで当然Systemdのバージョンも異なり、挙動も微妙に違う

## Unitファイルの一例

```
[Unit]
```

```
Documentation=https://hadoop.apache.org/  
Description=Hadoop MapReduce HistoryServer  
(略)
```

```
[Service]
```

```
User=mapred  
Group=mapred  
(略)  
ExecStart=/usr/bin/mapred --config /etc/hadoop/conf --daemon start historyserver  
ExecStop=/usr/bin/mapred --config /etc/hadoop/conf --daemon stop historyserver
```

おわりに

# おわりに

- BigtopはHadoopエコシステムのプロダクトの「使える組み合わせ」を提供
- Bigtop 3.4でも粛々と新しいプロダクトへの対応が実施されている
- Bigtop Managerなど、これまでになかったプロダクトもリリースされる(かも)

A low-angle photograph of a city skyline with several tall skyscrapers. The sky is a clear, deep blue. In the foreground, there are some trees and a street with a few vehicles. The text 'NTT Data' is overlaid in the center in a white, bold, sans-serif font.

**NTT Data**