



Red Hat Data Grid 8.2

Data Grid のアップグレード

Data Grid を 8.2 にアップグレード

Red Hat Data Grid 8.2 Data Grid のアップグレード

Data Grid を 8.2 にアップグレード

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Upgrading_Data_Grid.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Data Grid クラスターをある 8.x バージョンから次のバージョンにアップグレードローリングアップグレードを実行して、Data Grid が互換性のためにデータを変換するダウンタイムまたはオフラインアップグレードを回避することができます。

目次

RED HAT DATA GRID	3
DATA GRID のドキュメント	4
DATA GRID のダウンロード	5
多様性を受け入れるオープンソースの強化	6
RED HAT ドキュメントへのフィードバック (英語のみ)	7
第1章 DATA GRID 8 のアップグレードに関する注意点	8
1.1. DATA GRID 8.2 へのアップグレード	8
単一ファイルキャッシュストアを使用したデプロイメントのアップグレード	8
最低でも 8.1 からのアップグレード	8
ProtoStream マーシャラー設定の移行	8
第2章 DATA GRID サーバーのローリングアップグレードの実行	10
2.1. ターゲットクラスターの設定	10
2.1.1. ローリングアップグレードのリモートキャッシュストア	10
2.2. ターゲットクラスターへのデータの同期	11
第3章 キャッシュストア間のデータの移行	13
3.1. キャッシュストアマイグレーション	13
3.2. STORE MIGRATOR の取得	13
3.3. ストア移行の設定	14
3.3.1. 移行プロパティの保存	15
3.4. キャッシュストアの移行	19

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。

DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- [Data Grid 8.2 ドキュメント](#)
- [Data Grid 8.2 コンポーネントの詳細](#)
- [Data Grid 8.2 でサポートされる設定](#)
- [Data Grid 8 機能のサポート](#)
- [Data Grid で非推奨の機能](#)

DATA GRID のダウンロード

Red Hat カスタマーポータルで [Data Grid Software Downloads](#) にアクセスします。



注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[弊社の CTO である Chris Wright のメッセージ](#) を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社の技術的な内容についてのフィードバックに感謝します。ご意見をお聞かせください。コメントの追加、Insights の提供、誤字の修正、および質問を行う必要がある場合は、ドキュメントで直接行うこともできます。



注記

Red Hat アカウントがあり、カスタマーポータルにログインしている必要があります。

カスタマーポータルからドキュメントのフィードバックを送信するには、以下の手順を実施します。

1. **Multi-page HTML** 形式を選択します。
2. ドキュメントの右上にある **Feedback** ボタンをクリックします。
3. フィードバックを提供するテキストのセクションを強調表示します。
4. ハイライトされたテキストの横にある **Add Feedback** ダイアログをクリックします。
5. ページの右側のテキストボックスにフィードバックを入力し、**送信** をクリックします。

フィードバックを送信すると、自動的に問題の追跡が作成されます。**Submit** をクリックすると表示されるリンクを開き、問題の監視を開始するか、さらにコメントを追加します。

貴重なフィードバックにご協力いただきありがとうございます。

第1章 DATA GRID 8 のアップグレードに関する注意点

本セクションの詳細を確認してから1つの Data Grid 8 バージョンから別のバージョンにアップグレードしてください。

1.1. DATA GRID 8.2 へのアップグレード

以下の情報を参照して、以前のバージョンの Data Grid 8 から 8.2 へのアップグレードが成功するようにします。

単一ファイルキャッシュストアを使用したデプロイメントのアップグレード

Data Grid を 8.2.0 にアップグレードすると、**SingleFileStore** 永続性設定を含むキャッシュで、データの破損につながる問題が発生する可能性があります。

この問題は、Data Grid 8.2.0 へのアップグレードのみに影響します。Data Grid 8.2.1 以降、この問題はアップグレード中に発生しなくなりました。

以前のバージョンから 8.2.0 にすでにアップグレードしている場合は、できるだけ早く次のことを行う必要があります。

1. **\$RHDG_HOME/server/data/*.dat** ファイルをバックアップします。
2. Data Grid 8.2.1 以降にアップグレードします。

アップグレードが正常に完了すると、Data Grid は破損したデータを回復し、最初の起動時に単一ファイルストアを復元します。

クロスサイトレプリケーションの状態転送

クロスサイトレプリケーションを介して他のクラスターにバックアップするキャッシュの場合は、8.2 にアップグレードした後に状態転送を実行する必要があります。

Infinispan CLI から、次のように **site push-site-state** コマンドを使用します。

```
[//containers/default]> site push-site-state --cache=cacheName --site=NYC
```

最低でも 8.1 からのアップグレード

8.0 からアップグレードする場合は、最初に 8.1 にアップグレードする必要があります。Data Grid 8.0 の永続データは、Data Grid 8.2 とのバイナリー互換性がありません。これは、8.2 ではユーザーのシリアルライゼーションコンテキストが Data Grid のシリアルライゼーションコンテキストから分離されているためです。この非互換性の問題に対処するために、Data Grid 8.2 では、クラスター起動時に Data Grid 8.1 から既存の永続キャッシュストアを自動的に変換します。ただし、Data Grid は、Data Grid 8.0 からキャッシュストアを変換しません。

ProtoStream マーシャラー設定の移行

Data Grid 8.2 は、マーシャリング機能を提供する ProtoStream ライブラリーをアップグレードします。Data Grid 8.1 からのアップグレードプロセスの一部として、ProtoStreams エントリーを Protobuf としてエンコードする方法の違いにより、データ互換性の問題が発生するのを防ぐために、ProtoStream 移行の詳細も確認する必要があります。

さらに、**MessageMarshaller** API および **ProtoSchemaBuilder** アノテーションは ProtoStream API で非推奨になりました。Data Grid 8.1 のシリアルライズコンテキストイニシャライザーを、Data Grid 8.2 へのアップグレードの一環として **AutoProtoSchemaBuilder** アノテーションに移行する必要があります。

関連情報

- [ProtoStream アノテーション](#)
- [シリアル化のコンテキストイニシャライザーの作成](#)
- [AutoProtoSchemaBuilder アノテーションへのアプリケーションの移行](#)

第2章 DATA GRID サーバーのローリングアップグレードの実行

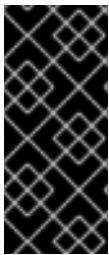
Data Grid クラスターのローリングアップグレードを実行して、ダウンタイムやデータの損失なしにバージョン間で変更します。ローリングアップグレードは、Hot Rod 経由で Data Grid サーバーおよびデータの両方をターゲットバージョンに移行します。

2.1. ターゲットクラスターの設定

ターゲット Data Grid バージョンを実行し、リモートキャッシュストアを使用してソースクラスターからデータを読み込むクラスターを作成します。

前提条件

- ターゲットアップグレードバージョンとともに Data Grid クラスターをインストールします。



重要

ターゲットクラスターのネットワークプロパティはソースクラスターのネットワークプロパティが重複していないことを確認します。JGroups トランスポート設定でターゲットおよびソースクラスターの一意の名前を指定する必要があります。環境に応じて、異なるネットワークインターフェイスを使用し、ターゲットクラスターとソースクラスターを分離するためにポートオフセットを指定することもできます。

手順

- ソースクラスターから移行する各キャッシュについて、ターゲットクラスターに **aRemoteCacheStore** を追加します。
リモートキャッシュストアは Hot Rod プロトコルを使用して、リモート Data Grid クラスターからデータを取得します。リモートキャッシュストアをターゲットクラスターに追加する場合は、ソースクラスターからデータをレイジーに読み込み、クライアント要求を処理します。
- すべての要求の処理を開始するために、クライアントをターゲットクラスターに切り替えます。
 - クライアント設定をターゲットクラスターの場所で更新します。
 - クライアントを再起動します。

2.1.1. ローリングアップグレードのリモートキャッシュストア

以下のようにローリングアップグレードを実行するには、特定のリモートキャッシュストア設定を使用する必要があります。

```
<!-- Remote cache stores for rolling upgrades must disable passivation. -->
<persistence passivation="false">
  <!-- The value of the cache attribute matches the name of a cache in the source cluster. Target
clusters load data from this cache using the remote cache store. -->
  <!-- The "protocol-version" attribute matches the Hot Rod protocol version of the source cluster. 2.5
is the minimum version and is suitable for any upgrade path. -->
  <!-- You should enable segmentation for remote cache stores only if the number of segments in the
target cluster matches the number of segments for the cache in the source cluster. -->
  <remote-store xmlns="urn:infinispan:config:store:remote:12.1"
    cache="myDistCache"
    protocol-version="2.5"
```

```

        hotrod-wrapping="true"
        raw-values="true"
        segmented="false">
<!-- Configures authentication and encryption according to the security realm of the source
cluster. -->
<security>
  <authentication server-name="infinispan">
    <digest username="admin"
      password="changeme"
      realm="default"/>
  </authentication>
</security>
<!-- Points to the location of the source cluster. -->
<remote-server host="127.0.0.1" port="11222"/>
</remote-store>
</persistence>

```

参照資料

- [リモートキャッシュストア設定スキーマ](#)
- [RemoteStore](#)
- [RemoteStoreConfigurationBuilder](#)

2.2. ターゲットクラスターへのデータの同期

ターゲットクラスターがリモートキャッシュストアを使用してクライアント要求を実行し、オンデマンドでデータを読み込む場合は、ソースクラスターからターゲットクラスターにデータを同期できます。

この操作はソースクラスターからデータを読み取り、ターゲットクラスターに書き込みます。データは、ターゲットクラスターのすべてのノードに並行して移行され、各ノードはデータのサブセットを受け取ります。Data Grid 設定で、各キャッシュの同期を実行する必要があります。

手順

1. ターゲットクラスターに移行する Data Grid 設定の各キャッシュの同期操作を開始します。Data Grid REST API を使用し、**?action=sync-data** パラメーターで **POST** 要求を呼び出します。たとえば、myCache という名前のキャッシュ内のデータをソースクラスターからターゲットクラスターに同期するには、以下を実行します。

```
POST /v2/caches/myCache?action=sync-data
```

操作が完了すると、Data Grid はターゲットクラスターにコピーされたエントリーの合計数で応答します。

または、**RollingUpgradeManager** MBean で **synchronizeData(migratorName=hotrod)** を呼び出すことで JMX を使用できます。

2. ターゲットクラスター内の各ノードをソースクラスターから切断します。たとえば、ソースクラスターから myCache キャッシュを切断するには、以下の **POST** 要求を呼び出します。

```
POST /v2/caches/myCache?action=disconnect-source
```

JMX を使用するには、**RollingUpgradeManager** MBean で **disconnectSource(migratorName=hotrod)** を呼び出します。

次のステップ

ソースクラスターからすべてのデータを同期した後、ローリングアップグレードプロセスが完了しました。ソースクラスターの使用を停止できるようになりました。

第3章 キャッシュストア間のデータの移行

Data Grid は、キャッシュストア間で永続化されたデータを移行するための Java ユーティリティを提供します。

Data Grid をアップグレードする場合、メジャーバージョン間の機能相違点は、キャッシュストア間の後方互換性を許可しません。**StoreMigrator** を使用してデータを変換し、ターゲットバージョンとの互換性を持つことができます。

たとえば、Data Grid 8.0 にアップグレードすると、デフォルトのマーシャラーが Protostream に変更されます。以前の Data Grid バージョンでは、キャッシュストアはバイナリー形式を使用し、マーシャリングする変更との互換性がありません。つまり、Data Grid 8.0 は、以前の Data Grid バージョンでキャッシュストアから読み込むことができません。

他の場合は、Data Grid のバージョンが、JDBC Mixed および Binary ストアなどのキャッシュストア実装を非推奨または削除します。このような場合は、**StoreMigrator** を使用して異なるキャッシュストア実装に変換できます。

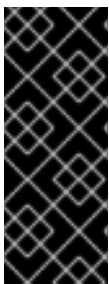
3.1. キャッシュストアマイグレーション

Data Grid は、最新の Data Grid キャッシュストア実装のデータを再作成する **StoreMigrator.java** ユーティリティを提供します。

StoreMigrator は以前のバージョンの Data Grid のキャッシュストアを取得し、キャッシュストア実装をターゲットとして使用します。

StoreMigrator を実行すると、**EmbeddedCacheManager** インターフェイスを使用して定義したキャッシュストアタイプでターゲットキャッシュが作成されます。**StoreMigrator** は、ソースストアからメモリーにエントリーを読み込み、それらをターゲットキャッシュに配置します。

StoreMigrator を使用すると、あるタイプのキャッシュストアから別のストアにデータを移行することもできます。たとえば、JDBC String ベースのキャッシュストアから Single File キャッシュストアに移行することができます。



重要

StoreMigrator は、セグメント化されたキャッシュストアから以下にデータを移行できません。

- 非セグメント化されたキャッシュストア。
- セグメント数が異なるセグメント化されたキャッシュストア。

3.2. STORE MIGRATOR の取得

StoreMigrator は、Data Grid ツールライブラリー **infinispan-tools** の一部として利用でき、Maven リポジトリに含まれます。

手順

- **StoreMigrator** の **pom.xml** を以下のように設定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.infinispan.example</groupId>
    <artifactId>jdbc-migrator-example</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
    <dependency>
    <groupId>org.infinispan</groupId>
    <artifactId>infinispan-tools</artifactId>
    </dependency>
    <!-- Additional dependencies -->
    </dependencies>

    <build>
    <plugins>
    <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>1.2.1</version>
    <executions>
    <execution>
    <goals>
    <goal>java</goal>
    </goals>
    </execution>
    </executions>
    <configuration>
    <mainClass>org.infinispan.tools.store.migrator.StoreMigrator</mainClass>
    <arguments>
    <argument>path/to/migrator.properties</argument>
    </arguments>
    </configuration>
    </plugin>
    </plugins>
    </build>
</project>

```

3.3. ストア移行の設定

ソースおよびターゲットのキャッシュストアのプロパティを **migrator.properties** ファイルに設定します。

手順

1. **migrator.properties** ファイルを作成します。
2. ソースキャッシュストアを **migrator.properties** に設定します。
 - a. 以下の例にあるように、すべての設定プロパティの先頭に **source.** を追加します。

```
source.type=SOFT_INDEX_FILE_STORE
source.cache_name=myCache
source.location=/path/to/source/sifs
```

3. **migrator.properties** でターゲットキャッシュストアを設定します。

a. 以下の例のように、すべての設定プロパティの先頭に **target.** を付けます。

```
target.type=SINGLE_FILE_STORE
target.cache_name=myCache
target.location=/path/to/target/sfs.dat
```

3.3.1. 移行プロパティの保存

ソースおよびターゲットのキャッシュストアを **StoreMigrator** プロパティで設定します。

表3.1 キャッシュストアタイププロパティ

プロパティ	説明	必須/オプション
type	<p>ソースまたはターゲットのキャッシュストアタイプのタイプを指定します。</p> <p>.type=JDBC_STRING</p> <p>.type=JDBC_BINARY</p> <p>.type=JDBC_MIXED</p> <p>.type=LEVELDB</p> <p>.type=ROCKSDB</p> <p>.type=SINGLE_FILE_STORE</p> <p>.type=SOFT_INDEX_FILE_STORE</p> <p>.type=JDBC_MIXED</p>	必須

表3.2 一般的なプロパティ

プロパティ	説明	値の例	必須/オプション
cache_name	ストアがバックアップするキャッシュに名前を付けます。	.cache_name=myCache	必須

プロパティ	説明	値の例	必須/オプション
segment_count	<p>セグメンテーションを使用できるターゲットキャッシュストアのセグメント数を指定します。</p> <p>セグメント数は、Data Grid 設定の clustering.hash.num Segments と一致する必要があります。</p> <p>つまり、キャッシュストアのセグメント数は、対応するキャッシュのセグメント数と一致する必要があります。セグメントの数が同一でない場合、Data Grid はキャッシュストアからデータを読み込めません。</p>	.segment_count=256	Optional

表3.3 JDBC プロパティ

プロパティ	説明	必須/オプション
dialect	基礎となるデータベースのダイアレクトを指定します。	必須
version	<p>ソースキャッシュストアのマーシャラーバージョンを指定します。以下のいずれかの値を設定します。</p> <p>* Data Grid 7.2.x の場合は 8</p> <p>* Data Grid 7.3.x の場合は 9</p> <p>* Data Grid 8.x の場合は 10</p>	<p>ソースストアにのみ必要です。</p> <p>例: source.version=9</p>
marshaller.class	カスタムマーシャラークラスを指定します。	カスタムマーシャラーを使用する場合に必要です。
marshaller.externalizers	[id]:<Externalizer class> 形式で読み込むカスタム AdvancedExternalizer 実装のコンマ区切りリストを指定します。	Optional

プロパティ	説明	必須/オプション
connection_pool.connection_url	JDBC 接続 URL を指定します。	必須
connection_pool.driver_classes	JDBC ドライバーのクラスを指定します。	必須
connection_pool.username	データベースユーザー名を指定します。	必須
connection_pool.password	データベースユーザー名のパスワードを指定します。	必須
db.major_version	データベースのメジャーバージョンを設定します。	Optional
db.minor_version	データベースのマイナーバージョンを設定します。	Optional
db.disable_upsert	データベース upsert を無効にします。	Optional
db.disable_indexing	テーブルインデックスが作成されるかどうかを指定します。	Optional
table.string.table_name_prefix	テーブル名の追加接頭辞を指定します。	Optional
table.string.<id data timestamp>.name	列名を指定します。	必須
table.string.<id data timestamp>.type	列タイプを指定します。	必須
key_to_string_mapper	TwoWayKey2StringMapper クラスを指定します。	Optional

注記

Binary キャッシュストアから古い Data Grid バージョンの移行には、以下のプロパティで **table.string.*** を **table.binary.*** に変更します。

- **source.table.binary.table_name_prefix**
- **source.table.binary.<id|data|timestamp>.name**
- **source.table.binary.<id|data|timestamp>.type**

Example configuration for migrating to a JDBC String-Based cache store

```

target.type=STRING
target.cache_name=myCache
target.dialect=POSTGRES
target.marshaller.class=org.example.CustomMarshaller
target.marshaller.externalizers=25:Externalizer1,org.example.Externalizer2
target.connection_pool.connection_url=jdbc:postgresql:postgres
target.connection_pool.driver_class=org.postgresql.Driver
target.connection_pool.username=postgres
target.connection_pool.password=redhat
target.db.major_version=9
target.db.minor_version=5
target.db.disable_upsert=false
target.db.disable_indexing=false
target.table.string.table_name_prefix=tablePrefix
target.table.string.id.name=id_column
target.table.string.data.name=datum_column
target.table.string.timestamp.name=timestamp_column
target.table.string.id.type=VARCHAR
target.table.string.data.type=bytea
target.table.string.timestamp.type=BIGINT
target.key_to_string_mapper=org.infinispan.persistence.keymappers.
DefaultTwoWayKey2StringMapper

```

表3.4 RocksDB プロパティ

プロパティ	説明	必須/オプション
location	データベースディレクトリを設定します。	必須
圧縮	使用する圧縮タイプを指定します。	Optional

```

# Example configuration for migrating from a RocksDB cache store.
source.type=ROCKSDB
source.cache_name=myCache
source.location=/path/to/rocksdb/database
source.compression=SNAPPY

```

表3.5 SingleFileStore プロパティ

プロパティ	説明	必須/オプション
location	キャッシュストア .dat ファイルが含まれるディレクトリを設定します。	必須

```

# Example configuration for migrating to a Single File cache store.
target.type=SINGLE_FILE_STORE
target.cache_name=myCache
target.location=/path/to/sfs.dat

```

表3.6 SoftIndexFileStore プロパティ

プロパティ	説明	値
必須/オプション	location	データベースディレクトリーを設定します。
必須	index_location	データベースインデックスディレクトリーを設定します。

```
# Example configuration for migrating to a Soft-Index File cache store.
target.type=SOFT_INDEX_FILE_STORE
target.cache_name=myCache
target.location=path/to/sifs/database
target.index_location=path/to/sifs/index
```

3.4. キャッシュストアの移行

StoreMigrator を実行して、あるキャッシュストアから別のキャッシュストアにデータを移行します。

前提条件

- **infinispan-tools.jar** を取得します。
- ソースおよびターゲットのキャッシュストアを設定する **migrator.properties** ファイルを作成します。

手順

- ソースから **infinispan-tools.jar** をビルドする場合は、以下を実行します。
 1. JDBC ドライバーなどのソースおよびターゲットのデータベースの **infinispan-tools.jar** および依存関係をクラスパスに追加します。
 2. **migrator.properties** ファイルを **StoreMigrator** の引数として指定します。
- Maven リポジトリから **infinispan-tools.jar** をプルする場合は、以下のコマンドを実行します。

```
mvn exec:java
```