Red Hat Data Grid 8.4

Data Grid コマンドラインインターフェイスの使用

Data Grid CLI を使用してリモートキャッシュにアクセスして管理する

Last Updated: 2024-06-06

Red Hat Data Grid 8.4 Data Grid コマンドラインインターフェイスの使用

Data Grid CLI を使用してリモートキャッシュにアクセスして管理する

法律上の通知

概要

コマンドラインインターフェイス (CLI) を使用して Data Grid Server クラスターに接続し、データにアクセスして、リモートキャッシュを使用して管理操作を実行します。

目次

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。

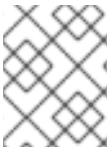
DATA GRID のドキュメント

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 8.4 ドキュメント
- Data Grid 8.4 コンポーネントの詳細
- Data Grid 8.4 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 プロパティー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、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 DATA GRID CLI のスタートガイド

コマンドラインインターフェイス (CLI) を使用すると、Data Grid Server にリモートで接続して、データにアクセスし、管理機能を実行できます。次の手順を実行して、ユーザーの作成、Data Grid への接続、リソースのナビゲートなど、CLI の基本的な使用法を学習します。

1.1. DATA GRID ユーザーの作成

Hot Rod および REST エンドポイントを介して Data Grid Server のデプロイメントで認証するための認証情報を追加します。Data Grid Console へのアクセスやキャッシュ操作を行う前に、Data Grid のコマンドラインインタフェース (CLI) で最低 1 名のユーザーを作成する必要があります。

ヒント

Data Grid は、ロールベースアクセス制御 (RBAC) を使用してセキュリティー認証を実施します。認証情報を初めて追加するときに **admin** ユーザーを作成して、Data Grid デプロイメントに対する完全な **ADMIN** 権限を取得します。

前提条件

- Data Grid Server をダウンロードし、インストールします。

手順

1. **$RHDG_HOME** でターミナルを開きます。
2. **user create** コマンドで **admin** ユーザーを作成します。

```
bin/cli.sh user create admin -p changeme
```

ヒント

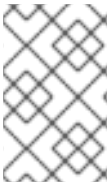
CLI セッションから **help user** を実行し、コマンドの詳細を取得します。

検証

user.properties を開き、ユーザーが存在することを確認します。

```
cat server/conf/users.properties

admin=scram-sha-1\:BYGclAwvf6b...
```



注記

CLI を使用してプロパティーレルムに認証情報を追加すると、接続しているサーバーインスタンスにのみユーザーが作成されます。プロパティーレルムの認証情報をクラスター内の各ノードに手動で同期する必要があります。

1.1.1. ユーザーへのロール付与

ユーザーにロールを割り当て、キャッシュ操作や Data Grid のリソースとのやり取りを行う権限を付与します。

ヒント

同じロールを複数のユーザーに割り当て、それらの権限を一元管理する場合は、ユーザーではなくグループにロールを付与します。

前提条件

- Data Grid に **ADMIN** 権限があること。
- Data Grid ユーザーを作成すること。

手順

1. Data Grid への CLI 接続を作成します。
2. **user roles grant** コマンドでユーザーにロールを割り当てます。以下に例を示します。

```
user roles grant --roles=deployer katie
```

検証

user roles ls コマンドでユーザーに付与したロールを一覧表示します。

```
user roles ls katie
["deployer"]
```

1.1.2. グループへのユーザーの追加

グループを使用すると、複数のユーザーのパーミッションを変更できます。グループにロールを割り当ててから、そのグループにユーザーを追加します。ユーザーは、グループロールからパーミッションを継承します。

注記

グループは、Data Grid Server 設定のプロパティーレームの一部として使用します。各グループは、ユーザー名とパスワードも必要な特別なタイプのユーザーです。

前提条件

- Data Grid に **ADMIN** 権限があること。
- Data Grid ユーザーを作成すること。

手順

1. Data Grid への CLI 接続を作成します。
2. **user create** コマンドを使用してグループを作成します。
   a. **--groups** 引数でグループ名を指定します。
   b. グループのユーザー名とパスワードを設定します。

```
user create --groups=developers developers -p changeme
```

3. グループをリスト表示します。

```
user ls --groups
```

4. グループにロールを付与します。

```
user roles grant --roles=application developers
```

5. グループのロールを一覧表示します。

```
user roles ls developers
```

6. 一度に 1 つずつグループにユーザーを追加します。

```
user groups john --groups=developers
```

検証

groups.properties を開き、グループが存在することを確認します。

```
cat server/conf/groups.properties
```

1.1.3. Data Grid のユーザーロールと権限

Data Grid には、キャッシュや Data Grid のリソースにアクセスするための権限をユーザーに提供するロールがいくつかあります。

Role	権限	説明
admin	ALL	Cache Manager ライフサイクルの制御など、すべてのパーミッションを持つスーパーユーザー。
deployer	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR、CREATE	**application** パーミッションに加えて、Data Grid リソースを作成および削除できます。
application	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR	**observer** パーミッションに加え、Data Grid リソースへの読み取りおよび書き込みアクセスがあります。また、イベントをリッスンし、サーバータスクおよびスクリプトを実行することもできます。
observer	ALL_READ、MONITOR	**monitor** パーミッションに加え、Data Grid リソースへの読み取りアクセスがあります。
monitor	MONITOR	JMX および **metrics** エンドポイント経由で統計を表示できます。

関連情報

- org.infinispan.security.AuthorizationPermission Enum
- Data Grid configuration schema reference

1.2. DATA GRID SERVER への接続

Data Grid への CLI 接続を確立します。

前提条件

ユーザーの認証情報を追加し、稼働中の Data Grid Server インスタンスが 1 つ以上ある。

手順

1. **$RHDG_HOME** でターミナルを開きます。
2. CLI を起動します。
   - **Linux:**

     ```
     bin/cli.sh
     ```

   - **Microsoft Windows:**

     ```
     bin\cli.bat
     ```

3. **connect** コマンドを実行し、プロンプトが表示されたらユーザー名とパスワードを入力します。
   - **11222** のデフォルトポート上の Data Grid Server:

     ```
     [disconnected]> connect
     ```

   - ポートオフセットが **100** の Data Grid Server:

     ```
     [disconnected]> connect 127.0.0.1:11322
     ```

1.3. CLI リソースのナビゲート

Data Grid CLI は、Data Grid クラスターリソースのリスト表示、説明、および操作を可能にするナビゲート可能なツリーを公開します。

ヒント

Tab キーを押して、使用可能なコマンドとオプションを表示します。**-h** オプションを使用して、ヘルプテキストを表示します。

Data Grid クラスターに接続すると、デフォルトのキャッシュコンテナーのコンテキストで開きます。

```
[//containers/default]>
```

- **ls** を使用してリソースをリストします。

```
[//containers/default]> ls
caches
counters
configurations
schemas
tasks
```

- **cd** を使用してリソースツリーをナビゲートします。

```
cd caches
```

- **describe** を使用して、リソースに関する情報を表示します。

```
describe
```

```
{
  "name" : "default",
  "version" : "xx.x.x-FINAL",
  "cluster_name" : "cluster",
  "coordinator" : true,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "___protobuf_metadata",
"org.infinispan.DIST_SYNC", "org.infinispan.LOCAL",
"org.infinispan.INVALIDATION_SYNC", "org.infinispan.REPL_SYNC",
"org.infinispan.SCATTERED_SYNC", "org.infinispan.INVALIDATION_ASYNC",
"org.infinispan.DIST_ASYNC" ],
  "physical_addresses" : "[192.0.2.0:7800]",
  "coordinator_address" : "<hostname>",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "1",
  "running_cache_count" : "1",
  "node_address" : "<hostname>",
  "cluster_members" : [ "<hostname1>", "<hostname2>" ],
  "cluster_members_physical_addresses" : [ "192.0.2.0:7800", "192.0.2.0:7801" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "mycache",
    "started" : true
  }, {
    "name" : "___protobuf_metadata",
    "started" : true
  } ]
}
```

1.3.1. CLI リソース

Data Grid CLI は、以下の目的でさまざまなリソースを公開します。

- ローカルキャッシュまたはクラスター化キャッシュを作成、変更、および管理します。
- Data Grid クラスターの管理操作を実行します。

キャッシュリソース

```
[//containers/default]> ls
caches
counters
configurations
schemas
tasks
```

caches

Data Grid キャッシュインスタンス。デフォルトのキャッシュコンテナーは空です。CLI を使用して、テンプレートまたは **infinispan.xml** ファイルからキャッシュを作成します。

counters

オブジェクトの数を記録する **Strong** カウンターまたは **Weak** カウンター。

configurations

Data Grid 設定。

schemas

キャッシュ内のデータを構造化する Protocol Buffers (Protobuf) スキーマ。

tasks

Data Grid キャッシュ定義を作成および管理するリモートタスク。

クラスターリソース

```
[hostname@cluster/]> ls
containers
cluster
server
```

containers

Data Grid クラスター上のキャッシュコンテナー。

cluster

クラスターに参加している Data Grid サーバーを一覧表示します。

server

Data Grid サーバーを管理および監視するためのリソース。

1.4. DATA GRID SERVER のシャットダウン

個別に実行中のサーバーを停止するか、クラスターを正常に停止します。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかの方法で Data Grid Server をシャットダウンします。
   - **shutdown cluster** コマンドを使用して、クラスターのすべてのノードを停止します。以下に例を示します。

     ```
     shutdown cluster
     ```

このコマンドは、クラスターの各ノードの **data** フォルダーにクラスターの状態を保存します。キャッシュストアを使用する場合、**shutdown cluster** コマンドはキャッシュのすべてのデータも永続化します。

- **shutdown server** コマンドおよびサーバーのホスト名を使用して、個々のサーバーインスタンスを停止します。以下に例を示します。

```
shutdown server <my_server01>
```

重要

shutdown server コマンドは、リバランス操作が完了するまで待機しません。これにより、同時に複数のホスト名を指定すると、データが失われる可能性があります。

ヒント

このコマンドの使用方法の詳細については、**help shutdown** を実行してください。

検証

Data Grid は、サーバーをシャットダウンしたときに以下のメッセージをログに記録します。

```
ISPN080002: Data Grid Server stopping
ISPN000080: Disconnecting JGroups channel cluster
ISPN000390: Persisted state, version=<$version> timestamp=YYYY-MM-DDTHH:MM:SS
ISPN080003: Data Grid Server stopped
```

1.4.1. Data Grid クラスターのシャットダウンおよび再起動

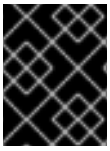
ノードを適切にシャットダウンして再起動することで、データの損失を回避してクラスターの一貫性を確保します。

クラスターのシャットダウン
Data Grid では、クラスターの状態を保存し、キャッシュ内のすべてのデータを永続化する時には、**shutdown cluster** コマンドを使用してクラスター内のすべてのノードを停止することを推奨します。**shutdown cluster** コマンドは、ノードが 1 つ含まれるクラスターに対しても使用できます。

Data Grid クラスターをオンラインに戻すと、すべてのノードが再度参加するまで、クラスター内のすべてのノードおよびキャッシュが利用できなくなります。Data Grid は、不整合やデータ損失を防ぐために、クラスターに保存されているデータへのアクセスと、クラスター状態の変更を制限します。さらに、Data Grid はクラスターのリバランスを無効にし、起動時にローカルキャッシュストアがパージされないようにします。

クラスターの回復プロセス中に、コーディネーターノードは、新しいノードが参加するたびにメッセージをログに記録し、どのノードが使用可能でどのノードがまだ欠落しているかを示します。Data Grid クラスター内の他のノードには、参加時のビューが表示されます。Data Grid Console または REST API を使用して、キャッシュの可用性を監視できます。

ただし、すべてのノードを待機することが不要または望ましくない場合は、現在のトポロジーで使用可能なキャッシュを設定できます。この方法は、CLI (下記参照) または REST API を通じて可能です。



重要

トポロジーを手動でインストールすると、データが失われる可能性があります。この操作は、初期トポロジーを再作成できない場合にのみ実行してください。

サーバーのシャットダウン

shutdown server コマンドを使用してノードを停止した後、オンラインに戻った最初のノードは、他のメンバーを待たずにすぐに使用できるようになります。残りのノードはすぐにクラスターに参加し、状態繊維がトリガーされますが、最初にローカル永続性が読み込まれるため、エントリーが失効する可能性があります。起動時にパージするように設定されたローカルキャッシュストアは、サーバーの起動時に空になります。**purge=false** としてマークされたローカルキャッシュストアは、サーバーの再起動後に使用可能になりますが、古いエントリーが含まれている可能性があります。

shutdown server コマンドを使用してクラスター化されたノードをシャットダウンする場合は、データ損失やキャッシュ内の古いエントリーに関連する問題が発生しないようにするために、各サーバーを逆の順序で再起動する必要があります。
たとえば、**server1** をシャットダウンしてから、**server2** をシャットダウンする場合は、最初に **server2** を起動してから **server1** を起動する必要があります。ただし、クラスター化されたノードを逆の順序で再起動しても、データ損失や古いエントリーを完全に防ぐことはできません。

第2章 PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

コマンドラインインターフェイス (CLI) を使用して、キャッシュの作成、データの操作、リバランスなどのリモートキャッシュの操作を実行します。

2.1. DATA GRID CLI を使用したリモートキャッシュの作成

Data Grid コマンドラインインターフェイス (CLI) を使用して、Data Grid Server にリモートキャッシュを追加します。

前提条件

- **admin** パーミッションを持つ Data Grid ユーザーを作成します。
- 1 つ以上の Data Grid Server インスタンスを起動します。
- Data Grid キャッシュ設定があります。

手順

1. CLI を起動します。

```
bin/cli.sh
```

2. **connect** コマンドを実行し、プロンプトが表示されたらユーザー名とパスワードを入力します。
3. **create cache** コマンドを使用してリモートキャッシュを作成します。
たとえば、以下のように **mycache.xml** という名前のファイルから"mycache"という名前のキャッシュを作成します。

```
create cache --file=mycache.xml mycache
```

検証

1. **ls** コマンドを使用して、すべてのリモートキャッシュをリスト表示します。

```
ls caches
mycache
```

2. **describe** コマンドでキャッシュ設定を表示します。

```
describe caches/mycache
```

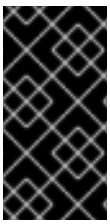
2.1.1. キャッシュ設定

XML、JSON、および YAML 形式で宣言型キャッシュ設定を作成できます。

すべての宣言型キャッシュは Data Grid スキーマに準拠する必要があります。JSON 形式の設定は XML 設定の構造に従う必要があります。要素がオブジェクトに対応し、属性はフィールドに対応します。

重要

Data Grid では、キャッシュ名またはキャッシュテンプレート名の文字数を最大 **255** 文字に制限しています。この文字制限を超えると、Data Grid は例外を出力します。簡潔なキャッシュ名とキャッシュテンプレート名を記述します。



重要

ファイルシステムによってファイル名の長さに制限が設定される場合があるため、キャッシュの名前がこの制限を超えないようにしてください。キャッシュ名がファイルシステムの命名制限を超えると、そのキャッシュに対する一般的な操作または初期化操作が失敗する可能性があります。簡潔なファイル名を書きます。

分散キャッシュ

XML

```
<distributed-cache owners="2"
                   segments="256"
                   capacity-factor="1.0"
                   l1-lifespan="5000"
                   mode="SYNC"
                   statistics="true">
  <encoding media-type="application/x-protostream"/>
  <locking isolation="REPEATABLE_READ"/>
  <transaction mode="FULL_XA"
               locking="OPTIMISTIC"/>
  <expiration lifespan="5000"
              max-idle="1000" />
  <memory max-count="1000000"
          when-full="REMOVE"/>
  <indexing enabled="true"
            storage="local-heap">
    <index-reader refresh-interval="1000"/>
    <indexed-entities>
      <indexed-entity>org.infinispan.Person</indexed-entity>
    </indexed-entities>
  </indexing>
  <partition-handling when-split="ALLOW_READ_WRITES"
                      merge-policy="PREFERRED_NON_NULL"/>
  <persistence passivation="false">
    <!-- Persistent storage configuration. -->
  </persistence>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "mode": "SYNC",
    "owners": "2",
    "segments": "256",
    "capacity-factor": "1.0",
    "l1-lifespan": "5000",
```

```
    "statistics": "true",
    "encoding": {
      "media-type": "application/x-protostream"
    },
    "locking": {
      "isolation": "REPEATABLE_READ"
    },
    "transaction": {
      "mode": "FULL_XA",
      "locking": "OPTIMISTIC"
    },
    "expiration" : {
      "lifespan" : "5000",
      "max-idle" : "1000"
    },
    "memory": {
      "max-count": "1000000",
      "when-full": "REMOVE"
    },
    "indexing" : {
          "enabled" : true,
          "storage" : "local-heap",
          "index-reader" : {
            "refresh-interval" : "1000"
          },
          "indexed-entities": [
            "org.infinispan.Person"
          ]
    },
    "partition-handling" : {
      "when-split" : "ALLOW_READ_WRITES",
      "merge-policy" : "PREFERRED_NON_NULL"
    },
    "persistence" : {
      "passivation" : false
    }
  }
}
```

YAML

```
distributedCache:
  mode: "SYNC"
  owners: "2"
  segments: "256"
  capacityFactor: "1.0"
  l1Lifespan: "5000"
  statistics: "true"
  encoding:
    mediaType: "application/x-protostream"
  locking:
    isolation: "REPEATABLE_READ"
  transaction:
    mode: "FULL_XA"
    locking: "OPTIMISTIC"
```

```
  expiration:
    lifespan: "5000"
    maxIdle: "1000"
  memory:
    maxCount: "1000000"
    whenFull: "REMOVE"
  indexing:
    enabled: "true"
    storage: "local-heap"
    indexReader:
      refreshInterval: "1000"
    indexedEntities:
      - "org.infinispan.Person"
  partitionHandling:
    whenSplit: "ALLOW_READ_WRITES"
    mergePolicy: "PREFERRED_NON_NULL"
  persistence:
    passivation: "false"
    # Persistent storage configuration.
```

レプリケートされたキャッシュ

XML

```
<replicated-cache segments="256"
                  mode="SYNC"
                  statistics="true">
  <encoding media-type="application/x-protostream"/>
  <locking isolation="REPEATABLE_READ"/>
  <transaction mode="FULL_XA"
               locking="OPTIMISTIC"/>
  <expiration lifespan="5000"
              max-idle="1000" />
  <memory max-count="1000000"
          when-full="REMOVE"/>
  <indexing enabled="true"
            storage="local-heap">
    <index-reader refresh-interval="1000"/>
    <indexed-entities>
      <indexed-entity>org.infinispan.Person</indexed-entity>
    </indexed-entities>
  </indexing>
  <partition-handling when-split="ALLOW_READ_WRITES"
                      merge-policy="PREFERRED_NON_NULL"/>
  <persistence passivation="false">
    <!-- Persistent storage configuration. -->
  </persistence>
</replicated-cache>
```

JSON

```
{
  "replicated-cache": {
    "mode": "SYNC",
```

```
    "segments": "256",
    "statistics": "true",
    "encoding": {
      "media-type": "application/x-protostream"
    },
    "locking": {
      "isolation": "REPEATABLE_READ"
    },
    "transaction": {
      "mode": "FULL_XA",
      "locking": "OPTIMISTIC"
    },
    "expiration" : {
      "lifespan" : "5000",
      "max-idle" : "1000"
    },
    "memory": {
      "max-count": "1000000",
      "when-full": "REMOVE"
    },
    "indexing" : {
      "enabled" : true,
      "storage" : "local-heap",
      "index-reader" : {
        "refresh-interval" : "1000"
        },
      "indexed-entities": [
        "org.infinispan.Person"
      ]
    },
    "partition-handling" : {
      "when-split" : "ALLOW_READ_WRITES",
      "merge-policy" : "PREFERRED_NON_NULL"
    },
    "persistence" : {
      "passivation" : false
    }
  }
}
```

YAML

```
replicatedCache:
  mode: "SYNC"
  segments: "256"
  statistics: "true"
  encoding:
    mediaType: "application/x-protostream"
  locking:
    isolation: "REPEATABLE_READ"
  transaction:
    mode: "FULL_XA"
    locking: "OPTIMISTIC"
  expiration:
    lifespan: "5000"
```

```
    maxIdle: "1000"
  memory:
    maxCount: "1000000"
    whenFull: "REMOVE"
  indexing:
    enabled: "true"
    storage: "local-heap"
    indexReader:
      refreshInterval: "1000"
    indexedEntities:
      - "org.infinispan.Person"
  partitionHandling:
    whenSplit: "ALLOW_READ_WRITES"
    mergePolicy: "PREFERRED_NON_NULL"
  persistence:
    passivation: "false"
    # Persistent storage configuration.
```

複数のキャッシュ

XML

```
<infinispan
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:infinispan:config:14.0 https://infinispan.org/schemas/infinispan-config-
14.0.xsd
                          urn:infinispan:server:14.0 https://infinispan.org/schemas/infinispan-server-14.0.xsd"
      xmlns="urn:infinispan:config:14.0"
      xmlns:server="urn:infinispan:server:14.0">
  <cache-container name="default"
                   statistics="true">
    <distributed-cache name="mycacheone"
                       mode="ASYNC"
                       statistics="true">
      <encoding media-type="application/x-protostream"/>
      <expiration lifespan="300000"/>
      <memory max-size="400MB"
              when-full="REMOVE"/>
    </distributed-cache>
    <distributed-cache name="mycachetwo"
                       mode="SYNC"
                       statistics="true">
      <encoding media-type="application/x-protostream"/>
      <expiration lifespan="300000"/>
      <memory max-size="400MB"
              when-full="REMOVE"/>
    </distributed-cache>
  </cache-container>
</infinispan>
```

JSON

```
{
  "infinispan" : {
```

```
    "cache-container" : {
      "name" : "default",
      "statistics" : "true",
      "caches" : {
        "mycacheone" : {
          "distributed-cache" : {
            "mode": "ASYNC",
            "statistics": "true",
            "encoding": {
              "media-type": "application/x-protostream"
            },
            "expiration" : {
              "lifespan" : "300000"
            },
            "memory": {
              "max-size": "400MB",
              "when-full": "REMOVE"
            }
          }
        },
        "mycachetwo" : {
          "distributed-cache" : {
            "mode": "SYNC",
            "statistics": "true",
            "encoding": {
              "media-type": "application/x-protostream"
            },
            "expiration" : {
              "lifespan" : "300000"
            },
            "memory": {
              "max-size": "400MB",
              "when-full": "REMOVE"
            }
          }
        }
      }
    }
  }
}
```

YAML

```
infinispan:
  cacheContainer:
    name: "default"
    statistics: "true"
    caches:
      mycacheone:
       distributedCache:
          mode: "ASYNC"
          statistics: "true"
          encoding:
            mediaType: "application/x-protostream"
          expiration:
```

```
      lifespan: "300000"
    memory:
      maxSize: "400MB"
      whenFull: "REMOVE"
mycachetwo:
  distributedCache:
    mode: "SYNC"
    statistics: "true"
    encoding:
      mediaType: "application/x-protostream"
    expiration:
      lifespan: "300000"
    memory:
      maxSize: "400MB"
      whenFull: "REMOVE"
```

関連情報

- Data Grid configuration schema reference
- infinispan-config-14.0.xsd

2.2. DATA GRID キャッシュ設定の変更

Data Grid CLI を使用してリモートキャッシュ設定を変更します。キャッシュ設定の属性を一度に 1 つずつ変更することも、XML、JSON、または YAML 形式でキャッシュ設定を提供して複数の属性を一度に変更することもできます。

前提条件

- Data Grid クラスターに少なくとも 1 つのリモートキャッシュを作成している。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかの方法で、**alter** コマンドを使用してキャッシュ設定を変更します。
   - **--file** オプションを使用して、1 つ以上の属性が変更された設定ファイルを指定します。
   - **--attribute** および **--value** オプションを使用して、特定の設定属性を変更します。

     ヒント

     詳細と例については、 **help alter** コマンドを実行してください。
3. **describe** コマンドで変更を確認します。以下に例を示します。

```
describe caches/mycache
```

2.3. キャッシュエントリーの追加

データコンテナーに **key:value** ペアのエントリーを作成します。

前提条件

データを保存できる Data Grid キャッシュを作成している。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のように、エントリーをキャッシュに追加します。
   - **put** コマンドで **--cache =** を使用します。

     ```
     put --cache=mycache hello world
     ```

   - キャッシュのコンテキストで **put** コマンドを使用します。

     ```
     [//containers/default/caches/mycache]> put hello world
     ```

3. **get** コマンドを使用して、エントリーを確認します。

   ```
   [//containers/default/caches/mycache]> get hello
   world
   ```

2.4. キャッシュのクリアとエントリーの削除

Data Grid CLI を使用してキャッシュからデータを削除します。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかを行います。
   - **clearcache** コマンドを使用してすべてのエントリーを削除します。

     ```
     clearcache mycache
     ```

   - **remove** コマンドを使用して特定のエントリーを削除します。

     ```
     remove --cache=mycache hello
     ```

2.5. キャッシュの削除

キャッシュをドロップしてキャッシュを削除し、キャッシュに含まれるすべてのデータを削除します。

手順

1. Data Grid への CLI 接続を作成します。
2. **drop** コマンドでキャッシュを削除します。

   ```
   drop cache mycache
   ```

2.6. 自動キャッシュリバランスの設定

デフォルトでは、ノードがクラスターに参加およびクラスターから離脱すると、Data Grid は自動的にキャッシュのバランスを取り直します。キャッシュマネージャーレベルで、またはキャッシュごとに無効または有効にすることで、自動キャッシュリバランスを設定できます。

手順

1. Data Grid への CLI 接続を作成します。
2. **rebalance disable** コマンドを使用して、すべてのキャッシュの自動リバランスを無効にします。

```
rebalance disable
```

3. **rebalance enable** コマンドを使用して、特定のキャッシュの自動リバランスを有効にします。次の例では、mycache という名前のキャッシュのリバランスのみを有効にします。

```
rebalance enable caches/mycache
```

4. すべてのキャッシュの自動リバランスを再度有効にします。

```
rebalance enable
```

rebalance コマンドの詳細については、**help rebalance** を実行してください。

2.7. STABLE トポロジーの設定

デフォルトでは、クラスターのシャットダウン後、Data Grid はすべてのノードがクラスターに参加してトポロジーを復元するまで待機します。ただし、特定キャッシュの現在のトポロジーを stable とマークするための CLI コマンドがあります。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかを行います。
   - 特定キャッシュの現在のトポロジーを stable として設定します。

```
topology set-stable cacheName
```

   - 現在のトポロジーに所有者の数よりも多くのノードが不足している場合は、操作を確定するために強制フラグが必要です。

```
topology set-stable cacheName -f
```

topology set-stable コマンドの詳細は、**topology set-stable -h** を実行してください。

重要

トポロジーを手動でインストールすると、データが失われる可能性があります。この操作は、初期トポロジーを再作成できない場合にのみ実行してください。

第3章 バッチ操作の実行

インタラクティブに、またはバッチファイルを使用して、グループで操作を処理します。

前提条件

- Data Grid クラスターを実行中である。

3.1. ファイルを使用したバッチ操作の実行

一連の操作を含むファイルを作成し、それらを Data Grid CLI に渡します。

手順

1. 一連の操作を含むファイルを作成している。
   たとえば、**mybatch** という名前のキャッシュを作成する **batch** という名前のファイルを作成し、キャッシュに 2 つのエントリーを追加して、CLI から切断します。

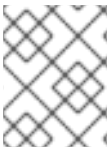
```
connect --username=<username> --password=<password> <hostname>:11222
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
ls caches/mybatch
disconnect
```

   ヒント

   バッチファイルで直接 **connect** コマンドを使用する代わりに、**autoconnect-url** プロパティーを使用して CLI を設定します。

2. CLI を実行し、ファイルを入力として指定します。

```
bin/cli.sh -f batch
```



注記

CLI バッチファイルは、システムプロパティーの拡張をサポートします。**${property}** 形式を使用する文字列は、**property** システムプロパティーの値に置き換えられます。

3.2. インタラクティブなバッチ操作の実行

標準の入力ストリーム**stdin**を使用して、バッチ操作をインタラクティブに実行します。

手順

1. インタラクティブモードで Data Grid CLI を起動します。

```
bin/cli.sh -c localhost:11222 -f -
```

ヒント

-c 引数を使用する代わりに、**autoconnect-url** プロパティーを使用して CLI 接続を設定できます。

2. バッチ操作を実行します。以下に例を示します。

```
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
disconnect
quit
```

第4章 DATA GRID CLI の設定

Data Grid CLI の設定プロパティーを定義します。

4.1. DATA GRID CLI プロパティーと永続ストレージの設定

Data Grid CLI の起動操作を設定し、永続ストレージの場所をカスタマイズします。

前提条件

少なくとも 1 人の Data Grid ユーザーを作成している。

手順

1. オプションで、次のいずれかの方法で Data Grid CLI ストレージディレクトリーへのカスタムパスを設定します。
   - **cli.dir** システムプロパティーの使用:

     ```
     bin/cli.sh -Dcli.dir=/path/to/cli/storage ...
     ```

   - **ISPN_CLI_DIR** 環境変数の使用:

     ```
     export ISPN_CLI_DIR=/path/to/cli/storage
     bin/cli.sh ...
     ```

2. **config set** コマンドを使用して、設定プロパティーの値を設定します。
   たとえば、CLI が自動的にその URL に接続するように、**autoconnect-url** プロパティーを設定します。

   注記

   リモート接続の場合は、URL を指定し、認証情報を入力します。

   - **http[s]://<username>:<password>@<hostname>:<port>**(Basic 認証用)
   - **http[s]://<token>@<hostname>:<port>**(OAuth 認証用)

   ```
   bin/cli.sh config set autoconnect-url http://<username>:<password>@<hostname>:11222
   ```

3. **config get** コマンドで設定プロパティーを確認します。

   ヒント

   help config を実行して、使用可能な設定プロパティーを確認し、使用例を取得します。

4.2. コマンドエイリアスの作成

Data Grid CLI コマンドのエイリアスを作成して、カスタムショートカットを定義します。

手順

1. **alias <alias>=<command>** コマンドを使用してエイリアスを作成します。
たとえば、**quit** コマンドのエイリアスとして **q** を設定します。

```
alias q=quit
```

2. **alias** コマンドを実行して、定義されたエイリアスを確認します。

```
alias
alias q='quit'
```

3. **unalias** コマンドを使用してエイリアスを削除します。以下に例を示します。

```
unalias q
```

4.3. DATA GRID SERVER 接続の信頼

SSL/TLS 証明書を使用して Data Grid Server への Data Grid CLI 接続を保護します。Data Grid Server の SSL ID としてキーストアを作成する場合、CLI はサーバー証明書を検証して ID を検証できます。

前提条件

- Data Grid Server の SSL ID を設定している。
- 少なくとも 1 人の Data Grid ユーザーを作成している。

手順

1. 次の例のように、サーバーキーストアの場所を指定します。

```
bin/cli.sh config set truststore /home/user/my-trust-store.jks
```

2. **オプション:** トラストストアのパスワードを定義します。次の例では、**secret** をトラストストアのパスワードとして設定します。

```
bin/cli.sh config set truststore-password secret
```

3. **オプション:** サーバーのクライアント証明書認証が必要な場合は、クライアント鍵ストアの場所を指定します。次の例を考慮して、**<path>** をキーストアファイルへの絶対ディレクトリーパスに置き換え、**<key_store_file>** をキーストアファイルの名前に置き換えます。

```
bin/cli.sh config set keystore /<emphasis><path></emphasis>/<emphasis><key_store_file>
</emphasis>
```

4. **オプション:** 鍵ストアのパスワードを定義します。次の例では、**secret** をキーストアのパスワードとして設定します。

```
bin/cli.sh config set keystore-password secret
```

5. CLI 設定を確認します。

```
bin/cli.sh config get truststore
```

```
bin/cli.sh config get truststore-password
```

関連情報

- Setting Up SSL Identities for Data Grid Server

4.4. DATA GRID CLI ストレージディレクトリー

Data Grid CLI は、設定を次のデフォルトディレクトリーに保存します。

オペレーティングシステム	デフォルトパス
Linux/Unix	**$HOME/.config/red_hat_data_grid**
Microsoft Windows	**%APPDATA%/Sun/Java/red_hat_data_grid**
Mac OS	**$HOME/Library/Java/red_hat_data_grid**

このディレクトリーには以下のファイルが格納されています。

cli.properties
: CLI 設定プロパティーの値を格納します。

aliases
: コマンドエイリアスを格納します。

history
: CLI 履歴を保存します。

第5章 カウンターの操作

カウンターは、オブジェクトの数を記録するアトミック増減分操作を提供します。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

5.1. カウンターの作成

Data Grid CLI を使用して強力なカウンターと弱いカウンターを作成します。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切な引数を指定して **create counter** コマンドを実行します。
   a. **my-weak-counter** を作成します。

```
create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --type=weak my-weak-counter
```

   b. **my-strong-counter** を作成します。

```
create counter --initial-value=3 --storage=PERSISTENT --type=strong my-strong-counter
```

3. 使用可能なカウンターをリスト表示します。

```
ls counters
```

4. カウンター設定を確認します。
   a. **my-weak-counter** について説明します。

```
describe counters/my-weak-counter
```

```
{
   "weak-counter":{
      "initial-value":5,
      "storage":"PERSISTENT",
      "concurrency-level":1
   }
}
```

   b. **my-strong-counter** について説明します。

```
describe counters/my-strong-counter
```

```
{
   "strong-counter":{
      "initial-value":3,
      "storage":"PERSISTENT",
      "upper-bound":5
   }
}
```

5.2. カウンターへのデルタの追加

任意の値でカウンターに増分または減分を適用します。

手順

1. カウンターを選択します。

```
counter my-weak-counter
```

2. 現在のカウントをリスト表示します。

```
[//containers/default/counters/my-weak-counter]> ls
5
```

3. カウンター値を **2** 増やします。

```
[//containers/default/counters/my-weak-counter]> add --delta=2
```

4. カウンター値を **-4** 減らします。

```
[//containers/default/counters/my-weak-counter]> add --delta=-4
```

注記

強力なカウンターは、演算が適用された後に値を返します。**--quiet = true** を使用して、戻り値を非表示にします。

たとえば、**add --delta=3 --quiet=true** を追加します。

弱いカウンターは空の応答を返します。

第6章 クロスサイトレプリケーション操作の実行

異なる場所で実行されている Data Grid クラスターは、データをバックアップするために相互に検出および通信できます。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

6.1. バックアップ場所のオフラインおよびオンライン化

バックアップ場所を手動でオフラインにし、オンラインに戻します。

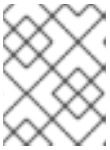
前提条件

- Data Grid への CLI 接続を作成します。

手順

1. **site status** コマンドを使用して、バックアップの場所がオンラインかオフラインかを確認します。

```
site status --cache=cacheName --site=NYC
```



注記

--site はオプションの引数です。設定されていない場合、CLI はすべてのバックアップ場所を返します。

ヒント

--all-caches オプションを使用して、すべてのキャッシュのバックアップの場所のステータスを取得します。

2. 次のようにバックアップ場所を管理します。
   - **bring-online** コマンドを使用して、バックアップの場所をオンラインにします。

```
site bring-online --cache=customers --site=NYC
```

   - **take-offline** コマンドを使用して、バックアップの場所をオフラインにします。

```
site take-offline --cache=customers --site=NYC
```

ヒント

--all-caches オプションを使用して、バックアップの場所をオンラインにするか、すべてのキャッシュでバックアップの場所をオフラインにします。

詳細と例については、**help site** コマンドを実行してください。

6.2. クロスサイト状態遷移モードの設定

バックアップの場所がオンラインになったことを DataGrid が検出したときに自動的に発生するように、サイト間の状態遷移操作を設定できます。または、状態遷移を手動で実行するデフォルトのモードを使用できます。

前提条件

- Data Grid への CLI 接続を作成します。

手順

1. 次の例のように、**site** コマンドを使用して状態遷移モードを設定します。
   - 現在の状態遷移モードを取得します。

     ```
     site state-transfer-mode get --cache=cacheName --site=NYC
     ```

   - キャッシュとバックアップの場所の自動状態遷移操作を設定します。

     ```
     site state-transfer-mode set --cache=cacheName --site=NYC --mode=AUTO
     ```

ヒント

詳細と例については、**help site** コマンドを実行してください。

6.3. バックアップ場所への状態のプッシュ

キャッシュの状態をバックアップの場所に転送します。

前提条件

- Data Grid への CLI 接続を作成します。

手順

- 以下の例のように、**site push-site-state** コマンドを使用して、状態遷移をプッシュします。

  ```
  site push-site-state --cache=cacheName --site=NYC
  ```

ヒント

--all-caches オプションを使用して、すべてのキャッシュの状態遷移をプッシュします。

詳細と例については、**help site** コマンドを実行してください。

第7章 DATA GRID クラスターのバックアップおよび復元

キャッシュされたエントリー、キャッシュ設定、Protobuf スキーマ、およびサーバースクリプトを含む Data Grid リソースのアーカイブを作成します。その後、バックアップアーカイブを使用して、再起動または移行後に Data Grid Server クラスターを復元できます。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

7.1. DATA GRID クラスターのバックアップ

ダウンロードまたは Data Grid Server に保存できる **.zip** 形式のバックアップアーカイブを作成します。

前提条件

バックアップアーカイブは、最新のクラスター状態を反映している必要があります。このため、バックアップアーカイブを作成する前に、クラスターが書き込み要求を受け付けていないことを確認する必要があります。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切なオプションを指定して **backup create** コマンドを実行します。以下に例を示します。
   - 自動生成された名前ですべてのリソースをバックアップします。

     ```
     backup create
     ```

   - **example-backup** という名前のバックアップアーカイブにすべてのリソースをバックアップします。

     ```
     backup create -n example-backup
     ```

   - サーバー上の **/some/server/dir** パスにすべてのリソースをバックアップします。

     ```
     backup create -d /some/server/dir
     ```

   - キャッシュとキャッシュテンプレートのみをバックアップします。

     ```
     backup create --caches=* --templates=*
     ```

   - 指定した Protobuf スキーマのみをバックアップします。

     ```
     backup create --proto-schemas=schema1,schema2
     ```

3. サーバー上で利用可能なバックアップアーカイブをリスト表示します。

   ```
   backup ls
   ```

4. サーバーから **example-backup** アーカイブをダウンロードします。
バックアップ操作がまだ進行中の場合、コマンドはバックアップ操作が完了するのを待ちます。

```
backup get example-backup
```

5. オプションで、 **example-backup** アーカイブをサーバーから削除します。

```
backup delete example-backup
```

7.2. バックアップアーカイブからの DATA GRID クラスターの復元

バックアップアーカイブのコンテンツを Data Grid クラスターに適用して、バックアップされた状態に復元します。

前提条件

- Data Grid CLI のローカルか、Data Grid Server に保存されているバックアップアーカイブを作成している。
- ターゲットコンテナーがバックアップアーカイブのコンテナー名と一致していることを確認してください。コンテナー名が一致しない場合、バックアップを復元することはできません。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切なオプションを指定して **backup restore** コマンドを実行します。
   - サーバーでアクセス可能なバックアップアーカイブからすべてのコンテンツを復元します。

     ```
     backup restore /some/path/on/the/server
     ```

   - ローカルバックアップアーカイブからすべてのコンテンツを復元します。

     ```
     backup restore -u /some/local/path
     ```

   - サーバー上のバックアップアーカイブからキャッシュコンテンツのみを復元します。

     ```
     backup restore /some/path/on/the/server --caches=*
     ```

第8章 コマンドリファレンス

Data Grid CLI コマンドのマニュアルページを確認してください。

ヒント

help コマンドを使用して、CLI セッションから直接マニュアルページにアクセスします。

たとえば、**get** コマンドのマニュアルページを表示するには、次の手順を実行します。

```
$ help get
```

8.1. ADD(1)

8.1.1. 名前

add - 任意の値でカウンターに増分または減分を適用します。

8.1.2. 概要

add ['OPTIONS'] ['COUNTER_NAME']

8.1.3. オプション

--delta='nnn'
　カウンター値を増加または減少させるデルタを設定します。デフォルトは **1** です。

-q, --quiet='[true|false]'
　強力なカウンターの戻り値を非表示にします。デフォルトは **false** です。

8.1.4. 例

add --delta=10 cnt_a
cnt_a の値を **10** 増やします。

add --delta=-5 cnt_a
cnt_a の値を **5** 減らします。

8.1.5. 関連項目

cas(1)、reset(1)

8.2. ALIAS(1)

8.2.1. 名前

alias - エイリアスを作成または表示します。

8.2.2. 概要

alias ['ALIAS-NAME'='COMMAND']

8.2.3. 例

alias q=quit
quit コマンドのエイリアスとして **q** を作成します。

alias
定義されているすべてのエイリアスをリスト表示します。

8.2.4. 関連項目

config(1)、unalias(1)

8.3. ALTER(1)

8.3.1. 名前

alter - Data Grid Server のキャッシュの設定を変更します。

8.3.2. 概要

alter cache ['OPTIONS'] **CACHE_NAME**

変更が既存の設定と互換性がある場合にのみ 、**alter** コマンドを使用してキャッシュを変更できます。

たとえば、レプリケートされたキャッシュ設定を使用して分散キャッシュを変更することはできません。同様に、特定の属性を使用してキャッシュ設定を作成する場合、代わりに別の属性を使用するように設定を変更することはできません。たとえば、**max-count** 属性の値を指定してキャッシュ設定を変更しようとすると、**max-size** がすでに設定されている場合、無効な設定になります。

8.3.3. キャッシュ変更のオプション

-f, --file='FILE'

既存の設定を変更する XML、JSON、または YAML 形式の設定ファイルを指定します。**--attribute** オプションと相互に排他的です。

--attribute='ATTRIBUTE'

既存の設定で変更する属性を指定します。Tab キーを押して、属性のリストを表示します。**--value** オプションと組み合わせて使用する必要があります。**--file** オプションと相互に排他的です。

--value='VALUE'

設定属性の新しい値を指定します。**--attribute** オプションと組み合わせて使用する必要があります。

8.3.4. 例

alter cache mycache --file=/path/to/mycache.json
mycache.json ファイルを使用して、**mycache** という名前のキャッシュの設置を変更します。

alter cache mycache --attribute=clustering.remote-timeout --value=5000
clustering.remote-timeout 属性の値が'5000'となるように、**mycache** という名前のキャッシュの設置を変更します。

8.3.5. 関連項目

create(1)、drop(1)

8.4. AVAILABILITY(1)

8.4.1. 名前

availability - ネットワークパーティション内のクラスター化されたキャッシュの可用性を管理します。

8.4.2. 概要

availability ['OPTIONS'] ['CACHE_NAME']

8.4.3. オプション

--mode='[AVAILABLE|DEGRADED_MODE]'

DENY_READ_WRITES または ALLOW_READS パーティション処理戦略のいずれかを使用する場合、キャッシュの可用性を AVAILABLE または DEGRADED_MODE に設定します。

AVAILABLE は、ネットワークパーティション内のすべてのノードでキャッシュを利用できるようにします。DEGRADED_MODE は、ネットワークパーティションが発生したときにキャッシュの読み取りおよび書き込み操作を防止します。

8.4.4. 例

availability cache1
キャッシュ 'cache1'の現在の可用性を取得します。

availability --mode=AVAILABLE cache1
キャッシュ'cache1'の可用性を AVAILABLE に設定します。

8.5. BACKUP(1)

8.5.1. 名前

backup - コンテナーのバックアップ作成と復元を管理します。

8.5.2. 概要

backup create ['OPTIONS']

backup delete ['OPTIONS'] **BACKUP_NAME**

backup get ['OPTIONS'] **BACKUP_NAME**

backup ls

backup restore ['OPTIONS'] **BACKUP_PATH**

8.5.3. バックアップ作成のオプション

-d, --dir='PATH'

バックアップアーカイブを作成および保存するサーバー上のディレクトリーを指定します。

-n, --name='NAME'

バックアップアーカイブの名前を定義します。

--caches='cache1,cache2,...'

バックアップするキャッシュをリスト表示します。'*'を使用して、すべてのキャッシュをバックアップします。

--templates='template1,template2,...'

バックアップするキャッシュテンプレートをリスト表示します。'*'を使用して、すべてのテンプレートをバックアップします。

--counters='counter1,counter2,...'

バックアップするカウンターをリスト表示します。'*'を使用して、すべてのカウンターをバックアップします。

--proto-schemas='schema1,schema2,...'

バックアップする Protobuf スキーマをリスト表示します。'*'を使用して、すべてのスキーマをバックアップします。

--tasks='task1,task2,...'

バックアップするサーバータスクをリスト表示します。'*'を使用して、すべてのタスクをバックアップします。

8.5.4. バックアップ取得のオプション

--no-content

コンテンツをダウンロードしません。このコマンドは、バックアップ操作が完了したときにのみ返します。

8.5.5. バックアップ復元のオプション

-u, --upload

サーバーにアップロードされるローカルバックアップアーカイブへのパスを定義します。

-n, --name='NAME'

復元要求の名前を定義します。

--caches='cache1,cache2,...'

復元するキャッシュをリスト表示します。'*'を使用して、バックアップアーカイブからすべてのキャッシュを復元します。

--templates='template1,template2,...'

復元するキャッシュテンプレートをリスト表示します。'*'を使用して、バックアップアーカイブからすべてのテンプレートを復元します。

--counters='counter1,counter2,...'

復元するカウンターをリスト表示します。'*'を使用して、バックアップアーカイブからすべてのカウンターを復元します。

--proto-schemas='schema1,schema2,...'

復元する Protobuf スキーマをリスト表示します。'*'を使用して、バックアップアーカイブからすべてのスキーマを復元します。

--tasks='task1,task2,...'

復元するサーバータスクをリスト表示します。'*'を使用して、バックアップアーカイブからすべてのタスクを復元します。

8.5.6. 例

backup create -n example-backup
example-backup という名前ですべてのコンテナーコンテンツのバックアップを開始します。

backup create -d /some/server/dir
すべてのコンテナーコンテンツのバックアップを開始し、サーバーのパス **/some/server/dir** に保存します。

backup create --caches=* --templates=*
キャッシュとキャッシュ設定リソースのみを含むバックアップを開始します。

backup create --proto-schemas=schema1,schema2
指定されたスキーマリソースのみを含むバックアップを開始します。

backup ls
サーバーで使用可能なすべてのバックアップをリスト表示します。

backup get example-backup
example-backup アーカイブをサーバーからダウンロードします。バックアップ操作が進行中の場合、コマンドはバックアップ操作が完了するのを待ちます。

backup restore /some/path/on/the/server
サーバー上のバックアップアーカイブからすべてのコンテンツを復元します。

backup restore -u /some/local/path
サーバーにアップロードされたローカルバックアップアーカイブからすべてのコンテンツを復元します。

backup restore /some/path/on/the/server --caches=*
サーバー上のバックアップアーカイブからキャッシュコンテンツのみを復元します。

backup restore /some/path/on/the/server --proto-schemas=schema1,schema2
サーバー上のバックアップアーカイブから、指定されたスキーマリソースのみを復元します。

backup delete example-backup
example-backup アーカイブをサーバーから削除します。

8.5.7. 関連項目

drop(1)

8.6. BENCHMARK(1)

8.6.1. 名前

benchmark - キャッシュに対してパフォーマンスベンチマークを実行します。

HTTP および HotRod プロトコル **http**、**https**、**hotrod**、および **hotrods** のパフォーマンスベンチマークを実行できます。URI を使用してベンチマークのプロトコルを指定します。プロトコルを指定しない場合、ベンチマークは現在の CLI 接続の URI を使用します。

Hot Rod URI のベンチマークは、クラスター全体に接続します。HTTP URI の場合、ベンチマークは単一のノードにのみ接続します。

ベンチマークは、既存のキャッシュに対してパフォーマンスをテストします。ベンチマークを実行する

前に、測定する機能を備えたキャッシュを作成する必要があります。たとえば、クロスサイトレプリケーションのパフォーマンスを評価する場合は、バックアップの場所を持つキャッシュを作成する必要があります。永続性のパフォーマンスをテストする場合は、適切なキャッシュストアを使用するキャッシュを作成します。

8.6.2. 概要

benchmark ['OPTIONS'] [**uri**]

8.6.3. ベンチマークのオプション

-t, --threads='num'

作成するスレッドの数を指定します。デフォルトは **10** です。

--cache='cache'

ベンチマークが実行されるキャッシュの名前を指定します。デフォルトは **benchmark** です。キャッシュがまだ存在しない場合は、ベンチマークを実行する前にキャッシュを作成する必要があります。

***--key-size='num'**

キーのサイズをバイト単位で設定します。デフォルトは 16 バイトです。

***--value-size='num'**

値のサイズをバイト単位で設定します。デフォルトは 1000 バイトです。

***--keyset-size='num'**

テストキーセットのサイズをバイト単位で定義します。デフォルトは **1000** です。

--verbosity=['SILENT', 'NORMAL', 'EXTRA']

出力の詳細レベルを指定します。可能な値は、最も簡素なものから最も詳細なものまで、**SILENT**、**NORMAL**、および **EXTRA** です。デフォルトは **NORMAL** です。

-c, --count='num'

実行する測定の反復回数を指定します。デフォルトは **5** です。

--time='time'

各反復にかかる時間を秒単位で設定します。デフォルトは **10** です。

--warmup-count='num'

実行するウォームアップの反復回数を指定します。デフォルトは **5** です。

--warmup-time='time'

各ウォームアップの反復にかかる時間を秒単位で設定します。デフォルトは **1** です。

--mode='mode'

ベンチマークモードを指定します。可能な値は、**Throughput**、**AverageTime**、**SampleTime**、**SingleShotTime**、および **All** です。デフォルトは **Throughput** です。

--time-unit='unit'

ベンチマークレポートの結果の時間単位を指定します。可能な値は、**NANOSECONDS**、**MICROSECONDS**、**MILLISECONDS**、および **SECONDS** です。デフォルトは **MICROSECONDS** です。

8.6.4. 例

benchmark hotrod://localhost:11222
HotRod プロトコルを使用してベンチマークテストを実行します。

benchmark --value-size=10000 --cache=largecache hotrod://localhost:11222
サイズが 10000 バイトのテスト値を使用して、**largecache** キャッシュに対して HotRod プロトコルでベンチマークテストを実行します。

benchmark --mode=All --threads=20 https://user:password@server:11222
20 スレッドを使用して HTTPS プロトコルでベンチマークテストを実行し、レポートにすべてのモードを含めます。

8.7. CACHE(1)

8.7.1. 名前

cache - 後続のコマンドのデフォルトキャッシュを選択します。

8.7.2. 概要

cache ['CACHE_NAME']

8.7.3. 例

cache mycache
mycache を選択ます。**cd caches/mycache** を使用してリソースツリーをナビゲートするのと同じです。

8.7.4. 関連項目

cd(1)、clear(1)、container(1)、get(1)、put(1)、remove(1)

8.8. CAS(1)

8.8.1. 名前

cas - 強力なカウンターで'compare-and-swap'操作を実行します。

8.8.2. 概要

cas ['OPTIONS'] ['COUNTER_NAME']

8.8.3. オプション

--expect='nnn'
カウンターの期待値を指定します。

--value='nnn'
カウンターに新しい値を設定します。

-q, --quiet='[true|false]'
戻り値を非表示にします。デフォルトは false です。

8.8.4. 例

cas --expect=10 --value=20 cnt_a
現在の値が **10** の場合にのみ、**cnt_a** の値を **20** に設定します。

8.8.5. 関連項目

add(1)、cas(1)、reset(1)

8.9. CD(1)

8.9.1. 名前

cd - サーバーリソースツリーをナビゲートします。

8.9.2. 説明

PATH は、絶対パスまたは現在のリソースに対する相対パスです。**../** は親リソースを指定します。

8.9.3. 概要

cd ['PATH']

8.9.4. 例

cd caches
リソースツリーの **caches** パスに変更します。

8.9.5. 関連項目

cache(1)、ls(1)、container(1)

8.10. CLEARCACHE(1)

8.10.1. 名前

clearcache - キャッシュからすべてのエントリーを削除します。

8.10.2. 概要

clearcache ['CACHE_NAME']

8.10.3. 例

clearcache mycache
mycache からすべてのエントリーを削除します。

8.10.4. 関連項目

cache(1)、drop(1)、remove(1)

8.11. CONFIG(1)

8.11.1. 名前

config - CLI 設定プロパティーを管理します。

8.11.2. 概要

config

config set 'name' 'value'

config get 'name'

config convert --outputFormat=[xml|json|yaml] [-o outputFile] [inputFile]

8.11.3. 説明

CLI 設定プロパティーを管理 (一覧表示、設定、取得) し、さまざまな形式 (XML、JSON、YAML) 間の設定変換を提供します。

8.11.4. コマンドの概要

config
　設定されているすべての設定プロパティーをリスト表示します。

config set 'name' ['value']
　特定のプロパティーの値を設定します。値を指定しない場合、プロパティーは設定されません。

config get 'name'
　特定のプロパティーの値を取得します。

config reset
　すべてのプロパティーをデフォルト値にリセットします。

config convert --format=[xml|json|yaml] [-o outputFile] [inputFile]
　設定ファイルを別の形式に変換します。

8.11.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help
　コマンドまたはサブコマンドのヘルプページを表示します。

8.11.6. 変換オプション

次のオプションが **convert** コマンドに適用されます。

-f, --format='xml|json|yaml'
　変換の形式を指定します。

-o, --output='path'
　出力ファイルへのパスを指定します。パスを指定しない場合は、標準出力 (**stdout**) を使用します。

8.11.7. プロパティー

autoconnect-url

起動時に CLI が自動的に接続する URL を指定します。

autoexec

起動時に実行する CLI バッチファイルのパスを指定します。

trustall

すべてのサーバー証明書を信頼するかどうかを指定します。値は **false**(デフォルト) および **true** です。

truststore

サーバー ID を検証する証明書チェーンを含むキーストアへのパスを定義します。

truststore-password

トラストストアにアクセスするためのパスワードを指定します。

keystore

証明書を含むキーストアへのパスを定義します。証明書はクライアントを識別します。サーバーがクライアント証明書認証を必要とする場合は、**keystore** プロパティーを使用します。

keystore-password

キーストアにアクセスするためのパスワードを指定します。

8.11.8. 例

config set autoconnect-url http://192.0.2.0:11222
CLI の起動時に、カスタム IP アドレスでサーバーに接続します。

config get autoconnect-url
autoconnect-url 設定プロパティーの値を返します。

config set autoexec /path/to/mybatchfile
CLI の起動時に、"mybatchfile"という名前のバッチファイルを実行します。

config set trustall true
すべてのサーバー証明書を信頼します。

config set truststore /home/user/my-trust-store.jks
"my-trust-store.jks"という名前のキーストアのパスを指定します。

config set truststore-password secret
必要に応じて、キーストアのパスワードを設定します。

config convert -f yaml -o infinispan.yaml infinispan.xml
infinispan.xml ファイルを YAML に変換し、出力を **infinispan.yaml** ファイルに書き込みます。

config convert -f json
設定を標準入力から JSON に変換し、出力を標準出力に書き込みます。

8.11.9. 関連項目

alias(1)、unalias(1)

8.12. CONNECT(1)

8.12.1. 名前

connect - 実行中の Data Grid サーバーに接続します。

8.12.2. 説明

デフォルトは **http://localhost:11222** で、認証が必要な場合は認証情報の入力を求められます。

8.12.3. 概要

connect ['OPTIONS'] ['SERVER_LOCATION']

8.12.4. オプション

-u, --username='USERNAME'
Data Grid サーバーで認証するユーザー名を指定します。

-p, --password='PASSWORD'
パスワードを指定します。

-t, --truststore='PATH'
トラストストアを指定します。

-s, --truststore-password='PASSWORD'
トラストストアのパスワードを指定します。

-k, --keystore='PATH'
クライアント証明書を含むキーストアを指定します。

-w, --keystore-password='PASSWORD'
キーストアのパスワードを指定します。

--hostname-verifier='REGEX'
SSL/TLS 対応サーバーへの接続中にホスト名にマッチする正規表現。

--trustall
すべての証明書を信頼します。

--context-path='PATH'
サーバー REST コネクターのコンテキストパス。指定されていない場合は、デフォルトで **/rest** になります。

8.12.5. 例

connect 127.0.0.1:11322 -u test -p changeme
100 のポートオフセットとサンプルの認証情報を使用して、ローカルで実行されているサーバーに接続します。

8.12.6. 関連項目

disconnect(1)

8.13. CONTAINER(1)

8.13.1. 名前

container - 後続のコマンドを実行するためのコンテナーを選択します。

8.13.2. 概要

container ['CONTAINER_NAME']

8.13.3. 例

container default
デフォルトのコンテナーを選択します。これは、**cd containers/default** を使用してリソースツリーをナビゲートするのと同じです。

8.13.4. 関連項目

cd(1)、clear(1)、container(1)、get(1)、put(1)、remove(1)

8.14. COUNTER(1)

8.14.1. 名前

counter - 後続のコマンドのデフォルトカウンターを選択します。

8.14.2. 概要

counter ['COUNTER_NAME']

8.14.3. 例

counter cnt_a
cnt_a を選択します。**cd counters/cnt_a** を使用してリソースツリーをナビゲートするのと同じです。

8.14.4. 関連項目

add(1)、cas(1)

8.15. CREATE(1)

8.15.1. 名前

create - Data Grid サーバーにキャッシュとカウンターを作成します。

8.15.2. 概要

create cache ['OPTIONS'] **CACHE_NAME**

create counter ['OPTIONS'] **COUNTER_NAME**

8.15.3. キャッシュ作成のオプション

-f, --file='FILE'
XML、JSON、または YAML 形式の設定ファイルを指定します。

-t, --template='TEMPLATE'

設定テンプレートを指定します。タブのオートコンプリートを使用して、使用可能なテンプレートを表示します。

-v, --volatile='[true|false]'

キャッシュが永続的であるか揮発性であるかを指定します。デフォルトは false です。

8.15.4. カウンター作成のオプション

-t, --type='[weak|strong]'

カウンターが弱いか強いかを指定します。

-s, --storage='[PERSISTENT|VOLATILE]'

カウンターが永続的であるか揮発性であるかを指定します。

-c, --concurrency-level='nnn'

カウンターの同時並行性レベルを設定します。

-i, --initial-value='nnn'

カウンターの初期値を設定します。

-l, --lower-bound='nnn'

強力なカウンターの下限を設定します。

-u, --upper-bound='nnn'

強力なカウンターの上限を設定します。

8.15.5. 例

create cache --template=org.infinispan.DIST_SYNC mycache
DIST_SYNC テンプレートから **MyCache** という名前のキャッシュを作成します。

create counter --initial-value=3 --storage=PERSISTENT --type=strong cnt_a
cnt_a という名前の強力なカウンターを作成します。

8.15.6. 関連項目

drop(1)

8.16. CREDENTIALS(1)

8.16.1. 名前

credentials - Data Grid Server のクレデンシャルを含むキーストアを管理します

8.16.2. 概要

credentials ls

credentials add 'alias'

credentials remove 'alias'

credentials mask -i iterations -s salt secret

8.16.3. 説明

キーストア内の認証情報を一覧表示、作成、および削除し、キーストアのパスワードをマスクします。デフォルトでは、コマンドはサーバー設定ディレクトリーの **credentials.pfx** キーストアを管理します。

8.16.4. 概要

credentials ls

キーストアに保存されている認証情報のエイリアスをリスト表示します。

クレデンシャルの追加

credentials add 'alias'

エイリアスと対応するクレデンシャルをキーストアに追加します。

クレデンシャルの削除

credentials remove 'alias'

エイリアスと対応するクレデンシャルをキーストアから削除します。

credentials mask -i iterations -s salt 'secret'

セキュリティーを強化するために、キーストアのパスワードをマスクで隠します。

8.16.5. オプション

-h, --help

コマンドのヘルプを出力します。

-s, --server-root='path-to-server-root'

サーバーのルートディレクトリーへのパスを指定します。デフォルトは **server** です。

--path='credentials.pfx'

クレデンシャルキーストアへのパスを指定します。デフォルトはサーバー設定ディレクトリー **server/conf** です。

-p, --password='password'

クレデンシャルキーストアのパスワードを指定します。

-t, --type='PKCS12'

認証情報を含むキーストアのタイプを指定します。サポートされているタイプは **PKCS12** または **JCEKS** です。デフォルトは **PKCS12** です。

8.16.6. クレデンシャル追加のオプション

-c, --credential='credential'

保存する認証情報を指定します。

8.16.7. 認証情報マスクオプション

-i, --iteration='n'

反復回数を設定します。

-s, --salt='salt'

ソルトを設定し、長さは 8 でなければなりません。

8.16.8. 例

credentials add dbpassword -c changeme -p "secret1234!"
認証情報キーストアがまだ存在しない場合は、新しいデフォルトの認証情報キーストアを作成し、"changeme"のパスワードに"dbpassword"のエイリアスを追加します。このコマンドはクレデンシャルキーストアのパスワードとして"secret1234!"も設定します。このパスワードは、サーバー設定のパスワード **<clear-text-credential clear-text="secret1234!"/>** と一致する必要があります。

credentials ls -p "secret1234!"
デフォルトのクレデンシャルキーストア内のすべてのエイリアスをリスト表示します。

credentials add ldappassword -t JCEKS -p "secret1234!"
JCEKS 形式で認証情報キーストアを作成し、エイリアス"ldappassword"を追加します。このコマンドは、エイリアスに対応するパスワードを指定するように要求します。

credentials mask "secret1234!" -i 100 -s pepper99 認証情報 secret1234! のマスクされた表現を作成します。文字列 **pepper99** をソルトとして使用して 100 回の反復を使用します。

8.17. DESCRIBE(1)

8.17.1. 名前

describe - リソースに関する情報を表示します。

8.17.2. 概要

describe ['PATH']

8.17.3. 例

describe //containers/default
デフォルトのコンテナーに関する情報を表示します。

describe //containers/default/caches/mycache
mycache キャッシュに関する情報を表示します。

describe //containers/default/caches/mycache/k1
k1 キーに関する情報を表示します。

describe //containers/default/counters/cnt1
cnt1 カウンターに関する情報を表示します。

8.17.4. 関連項目

cd(1)、ls(1)

8.18. DISCONNECT(1)

8.18.1. 名前

disconnect - Data Grid サーバーとの CLI セッションを終了します。

8.18.2. 概要

disconnect

8.18.3. 例

disconnect
現在の CLI セッションを終了します。

8.18.4. 関連項目

connect(1)

8.19. DROP(1)

8.19.1. 名前

drop - キャッシュとカウンターを削除します。

8.19.2. 概要

drop cache CACHE_NAME

drop counter COUNTER_NAME

8.19.3. 例

drop cache mycache
mycache キャッシュを削除します。

drop counter cnt_a
cnt_a カウンターを削除します。

8.19.4. 関連項目

create(1)、clearcache(1)

8.20. ENCODING(1)

8.20.1. 名前

encoding - キャッシュエントリーのエンコーディングを表示および設定します。

8.20.2. 説明

キャッシュに対する**put**および**get**操作のデフォルトのエンコーディングを設定します。引数が指定されていない場合、**encoding**コマンドは現在のエンコーディングを表示します。

有効なエンコーディングでは、次のような標準の MIME タイプ (IANA メディアタイプ) の命名規則が使用されます。

- **text/plain**
- **application/json**

- **application/xml**
- **application/octet-stream**

8.20.3. 概要

encoding ['ENCODING']

8.20.4. 例

encoding application/json
エントリーを **application/json** としてエンコードするように、現在選択されているキャッシュを設定します。

8.20.5. 関連項目

get(1)、put(1)

8.21. GET(1)

8.21.1. 名前

get - キャッシュからエントリーを取得します。

8.21.2. 概要

get ['OPTIONS'] **KEY**

8.21.3. オプション

-c, --cache='NAME'
エントリーを取得するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

8.21.4. 例

get hello -c mycache
mycache から **hello** という名前のキーの値を取得します。

8.21.5. 関連項目

query(1)、put(1)

8.22. HELP(1)

8.22.1. 名前

help - コマンドのマニュアルページを出力します。

8.22.2. 概要

help ['COMMAND']

8.22.3. 例

help get
getコマンドのマニュアルページを出力します。

8.22.4. 関連項目

version(1)

8.23. INDEX(1)

8.23.1. 名前

index - キャッシュインデックスを管理します。

8.23.2. 概要

index reindex 'cache-name'

index clear 'cache-name'

index update-schema 'cache-name'

index stats 'cache-name'

index clear-stats 'cache-name'

8.23.3. 例

index reindex mycache
キャッシュを再インデックス化します。

index clear mycache
キャッシュインデックスをクリアします。

index update-schema mycache
キャッシュのインデックススキーマを更新します。

index stats mycache
キャッシュのインデックス作成と検索の統計を表示します。

index clear-stats mycache
キャッシュのインデックス作成と検索の統計をクリアします。

8.23.4. 関連項目

query(1)

8.24. INSTALL(1)

8.24.1. 名前

install - Data Grid サーバーのアーティファクトをダウンロードしてインストールします。

8.24.2. 説明

アーティファクトを **server/lib** ディレクトリーにダウンロードしてインストールします。アーティファクトのダウンロード場所は、Maven アーティファクト座標、URL、またはローカルファイルパスとして指定できます。

Maven アーティファクトをダウンロードする場合は、オプションの Maven **settings.xml** ファイルによって、リモートおよびローカルリポジトリーの場所と、認証情報およびプロキシー設定が決定されます。

アーティファクトを **zip**、**tar.gz**、または **tgz** アーカイブとしてダウンロードすると、コンテンツが抽出されます。

8.24.3. 概要

install 'artifact-1[[|algorithm]|checksum]' ['artifact-2[[|algorithm]|checksum]'...]

8.24.4. アーティファクト名

アーティファクト名は、次のいずれかになります。

- **org.postgresql:postgresql:42.3.1** などの **groupId:artifactId:version** 形式を使用した Maven 座標。
- HTTP、HTTPS、または FTP URL
- ローカルパス

8.24.5. チェックサム検証

ダウンロード後にアーティファクトのチェックサムを検証できます。アルゴリズムのデフォルトは **SHA-256** ですが、**MD-5**、**SHA-1**、**SHA-256**、**SHA-384**、または `SHA-512' にすることもできます。

8.24.6. パッチリストのオプション

--server-home='path/to/server'

サーバーインストールのパスを設定します。

--server-root='server'

サーバーのホームに相対的なサーバーのルートディレクトリーを設定します。

***--maven-settings='$HOME/.m2/maven-settings.xml'**

Maven **settings.xml** ファイルのパスを設定し、指定されていない場合はデフォルトの場所を使用します。

-o, --overwrite

server/lib ディレクトリー内のアーティファクトを強制的に上書きします。デフォルトでは、アーティファクトは上書きされないため、アーティファクトがすでに存在する場合、インストールは失敗します。

-v、--verbose

アーティファクトのダウンロードに関する詳細情報を表示します。

-f、--force

すでにローカルに存在する場合でも、リモートアーティファクトを強制的にダウンロードします。

-r, --retries=num

ダウンロードされたアーティファクトが提供されたチェックサムと一致しない場合の再試行回数。

--clean

アーティファクトをダウンロードする前に、**server/lib** ディレクトリーの内容をすべて削除します。

8.24.7. 例

install -o org.postgresql:postgresql:42.3.1
PostgreSQL JDBC ドライバー JAR をインストールし、すでに存在する場合は上書きします。

install https://example.org/artifact.zip
artifact.zip をダウンロードし、内容を抽出します。

install https://example.org/artifact.zip|52d73f9b3611610ebbb4dca7c2ac1171218eb09891c1faba10f5f54c1d2acc13
artifact.zip をダウンロードし、その SHA-256 チェックサムを検証して、コンテンツを抽出します。

install https://example.org/artifact.zip|MD5|2b48d1871ee26f969d8481db94e103c2
artifact.zip をダウンロードし、その MD-5 チェックサムを検証して、コンテンツを抽出します。

8.25. LOGGING(1)

8.25.1. 名前

logging - Data Grid サーバーのランタイムロギング設定を検査および操作します。

8.25.2. 概要

logging list-loggers

logging list-appenders

logging set ['OPTIONS'] [**LOGGER_NAME**]

logging remove LOGGER_NAME

8.25.3. ロギング設定のオプション

-l, --level='OFF|TRACE|DEBUG|INFO|WARN|ERROR|ALL'

特定のロガーのログレベルを指定します。

-a, --appender='APPENDER'

特定のロガーに設定するアペンダーを指定します。このオプションは、複数のアペンダーに対して繰り返すことができます。

注記

ロガー名なしで**logging set**を呼び出すと、ルートロガーが変更されます。

8.25.4. 例

logging list-loggers
利用可能なすべてのロガーをリスト表示します。

logging set --level=DEBUG --appenders=FILE org.infinispan
org.infinispan ロガーのログレベルを **DEBUG** に設定し、**FILE** アペンダーを使用するように設定します。

8.26. LS(1)

8.26.1. 名前

ls - 現在のパスまたは特定のパスのリソースをリスト表示します。

8.26.2. 概要

ls ['PATH']

8.26.3. オプション

-f, --format='[NAMES|VALUES|FULL]'
このオプションは現在、キャッシュを一覧表示する場合にのみ適用されます。

- **NAMES**: キーのみを表示
- **VALUES**: キーと値を表示
- **FULL**: キー、値、およびメタデータを表示

-l
このオプションは、キャッシュを一覧表示する場合にのみ適用されます。**-f FULL** のショートカット。

-p, --pretty-print='[TABLE|CSV|JSON]'
次のレイアウトのいずれかを使用して出力を印刷します。

- **TABLE**: 表形式。列のサイズは、端末の幅によって決まります。これはデフォルトになります。
- **CSV**: コンマ区切り値。
- **JSON**: JSON 形式。

-m, --max-items='num'
このオプションは、キャッシュを一覧表示する場合にのみ適用されます。表示するアイテムの最大数。デフォルトは -1 (無制限) です。

8.26.4. 例

ls caches
使用可能なキャッシュをリスト表示します。

ls ../
親リソースをリスト表示します。

ls -l --pretty-print=CSV /containers/default/caches/mycache > mycache.csv
キー、値、メタデータを含むキャッシュのコンテンツを一覧表示し、コンテンツをファイルにリダイレクトします。

8.26.5. 関連項目

cd(1)

8.27. MIGRATE(1)

8.27.1. 名前

migrate - Data Grid のあるバージョンから別のバージョンにデータを移行します。

8.27.2. 概要

migrate cluster connect

migrate cluster synchronize

migrate cluster disconnect

migrate cluster source-connection

8.27.3. 説明

あるバージョンの Data Grid から別のバージョンにデータを移行するには、 **migrate** コマンドを使用します。

8.27.4. コマンドの概要

クラスターの移行

migrate cluster connect
ターゲットクラスターをソースクラスターに接続します。

migrate cluster synchronize
ソースクラスターとターゲットクラスターの間でデータを同期します。

migrate cluster disconnect
ターゲットクラスターをソースクラスターから切断します。

migrate cluster source-connection
ターゲットクラスターの接続設定を取得します。接続が確立されていない場合、コマンドは"Not Found"を出力します。

8.27.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help

コマンドまたはサブコマンドのヘルプページを表示します。

クラスター接続のオプション

> *-c, --cache*='name'::
> The name of the cache to connect to the source.
>
> *-f, --file*='FILE'::
> Specifies a configuration file in JSON format, containing a single 'remote-store' element.
>
> CLUSTER SYNCHRONIZE OPTIONS
> ---
>
> *-c, --cache*='name'::
> The name of the cache to synchronize.
>
> *-b, --read-batch*='num'::
> The amount of entries to process in a batch. Defaults to 10000.
>
> *-t, --threads*='num'::
> The number of threads to use. Defaults to the number of cores on the server.
>
> CLUSTER DISCONNECT OPTIONS

-c, --cache='name'

ソースから切断するキャッシュの名前。

8.27.6. クラスター接続のオプション

-c, --cache='name'

接続設定を取得するキャッシュの名前。

8.28. PATCH(1)

8.28.1. 名前

patch - サーバーパッチを管理します。

8.28.2. 説明

サーバーパッチをリスト表示、説明、インストール、ロールバック、および作成します。

パッチは、サーバーをアップグレードして問題を解決したり、新しい機能を追加したりするためのアーティファクトを含む zip アーカイブファイルです。パッチは、異なるバージョンの複数のサーバーインストールにターゲットバージョンを適用できます。

8.28.3. 概要

patch ls

patch install 'patch-file'

patch describe 'patch-file'

patch rollback

patch create 'patch-file' 'target-server' 'source-server-1' ['source-server-2'...]

8.28.4. パッチリストのオプション

--server='path/to/server'
現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

-v、--verbose
個々のファイルに関する情報を含む、インストールされている各パッチの内容を表示します。

8.28.5. パッチインストールのオプション

--dry-run
パッチが変更を適用せずに実行する操作を示します。

--server='path/to/server'
現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

8.28.6. パッチ説明のオプション

-v、--verbose
個々のファイルに関する情報を含む、パッチの内容を表示します。

8.28.7. パッチロールバックのオプション

--dry-run
パッチが変更を適用せずに実行する操作を示します。

--server='path/to/server'
現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

8.28.8. パッチ作成のオプション

-q, --qualifier='name'
パッチの説明的な修飾子文字列を指定します (例:'one-off for issue nnnn')。

8.28.9. 例

patch ls
サーバーに現在インストールされているパッチをインストール順にリスト表示します。

patch install mypatch.zip
サーバーの現在のディレクトリーに"mypatch.zip"をインストールします。

patch install mypatch.zip --server=/path/to/server/home
サーバーの別のディレクトリーに"mypatch.zip"をインストールします。

patch describe mypatch.zip
"mypatch.zip"のターゲットバージョンとソースバージョンのリストを表示します。

patch create mypatch.zip 'target-server' 'source-server-1' ['source-server-2'...]
ターゲットサーバーのバージョンを使用し、ソースサーバーのバージョンに適用する"mypatch.zip"という名前のパッチファイルを作成します。

patch rollback
サーバーに適用された最後のパッチをロールバックし、以前のバージョンを復元します。

8.29. PUT(1)

8.29.1. 名前

put - キャッシュエントリーを追加または更新します。

8.29.2. 説明

新しいキーのエントリーを作成します。既存のキーの値を置き換えます。

8.29.3. 概要

put ['OPTIONS'] **KEY** [**VALUE**]

8.29.4. オプション

-c, --cache='NAME'

キャッシュの名前を指定します。デフォルトは現在選択されているキャッシュです。

-e, --encoding='ENCODING'

値のメディアタイプを設定します。

-f, --file='FILE'

エントリーの値を含むファイルを指定します。

-l, --ttl='TTL'

エントリーが自動的に削除されるまでの秒数 (存続時間) を設定します。**0** の場合または指定されていない場合、デフォルトはキャッシュ設定の **lifespan** の値になります。負の値を設定すると、エントリーが削除されることはありません。

-i, --max-idle='MAXIDLE'

エントリーをアイドル状態にできる秒数を設定します。最大アイドル時間が経過してもエントリーの読み取りまたは書き込み操作が発生しない場合、エントリーは自動的に削除されます。**0** の場合または指定されていない場合、デフォルトはキャッシュ設定の **maxIdle** の値になります。負の値を設定すると、エントリーが削除されることはありません。

-a, --if-absent=[true|false]

エントリーが存在しない場合にのみエントリーを配置します。

8.29.5. 例

put -c mycache hello world
値が **world** の **hello** キーを **mycache** キャッシュに追加します。

put -c mycache -f myfile -i 500 hola
値が **myfile** の内容の **hola** キーを追加します。また、最大アイドル時間を **500** 秒に設定します。

8.29.6. 関連項目

get(1)、remove(1)

8.30. QUERY(1)

8.30.1. 名前

query - Ickle クエリーを実行して、リモートキャッシュのエントリーを照合します。

8.30.2. 概要

query ['OPTIONS'] **QUERY_STRING**

8.30.3. オプション

-c, --cache='NAME'
照会するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

--max-results='MAX_RESULTS'
返す結果の最大数を設定します。デフォルトは **10** です。

-o, --offset='OFFSET'
返される最初の結果のインデックスを指定します。デフォルトは **0** です。

8.30.4. 例

query "from org.infinispan.example.Person p where p.gender = 'MALE'"
リモートキャッシュの値にクエリーを行い、性別データ型が **MALE** であるいる Protobuf **Person** エンティティーからのエントリーを検索します。

8.30.5. 関連項目

index(1) schema(1)

8.31. QUIT(1)

8.31.1. 名前

quit - コマンドラインインターフェイスを終了します。

8.31.2. 概要

quit

exitと**bye**はコマンドエイリアスです。

8.31.3. 例

quit
CLI セッションを終了します。

exit
CLI セッションを終了します。

bye
CLI セッションを終了します。

8.31.4. 関連項目

disconnect(1)、shutdown(1)

8.32. REBALANCE(1)

8.32.1. 名前

rebalance - キャッシュの自動リバランスを管理します。

8.32.2. 概要

rebalance enable ['PATH']

rebalance disable ['PATH']

8.32.3. 例

rebalance enable
現在のコンテキストで自動リバランスを有効にします。このコマンドをルートコンテキストで実行すると、すべてのキャッシュのリバランスが有効になります。

rebalance enable caches/mycache
mycache という名前のキャッシュの自動リバランスを有効にします。

rebalance disable
現在のコンテキストで自動リバランスを無効にします。このコマンドをルートコンテキストで実行すると、すべてのキャッシュのリバランスが無効になります。

rebalance disable caches/mycache
mycache という名前のキャッシュの自動リバランスを無効にします。

8.33. REMOVE(1)

8.33.1. 名前

remove - キャッシュからエントリーを削除します。

8.33.2. 概要

remove KEY ['OPTIONS']

8.33.3. オプション

--cache='NAME'

エントリーを削除するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

8.33.4. 例

remove --cache=mycache hola
mycache キャッシュから **hola** エントリーを削除します。

8.33.5. 関連項目

cache(1)、drop(1)、clearcache(1)

8.34. RESET(1)

8.34.1. 名前

reset - カウンターの初期値を復元します。

8.34.2. 概要

reset ['COUNTER_NAME']

8.34.3. 例

reset cnt_a
cnt_a カウンターをリセットします。

8.34.4. 関連項目

add(1)、cas(1)、drop(1)

8.35. SCHEMA(1)

8.35.1. 名前

schema - Protobuf スキーマを操作します。

8.35.2. 概要

schema ls

schema upload --file=/path/to/schema.proto schema.proto

schema remove schema.proto

schema get schema.proto

8.35.3. 説明

ls、**upload**、**get**、**remove** サブコマンドを使用してスキーマを管理します。

8.35.4. コマンドの概要

schema ls

サーバーにインストールされているスキーマを一覧表示します。

schema upload --file='/path/to/schema.proto' 'schema.proto'

ProtoBuf スキーマファイルをサーバーにアップロードします。

schema get 'schema.proto'

指定したスキーマの内容を表示します。

schema remove 'schema.proto'

指定されたスキーマをサーバーから削除します。

8.35.5. アップロードオプション

-f, --file='FILE'

指定された名前の protobuf スキーマとしてファイルをアップロードします。

8.35.6. 例

schema upload --file=person.proto person.proto
Protobuf スキーマ **person.proto** を登録します。

8.35.7. 関連項目

query(1)

8.36. SERVER(1)

8.36.1. 名前

server - サーバー設定と状態管理。

8.36.2. 説明

serverコマンドは、サーバーエンドポイントコネクターとデータソースを記述および管理し、サーバーとホストの両方に関する集約された診断レポートを取得します。

レポートは、設定ファイルとログファイルに加えて、CPU、メモリー、開いているファイル、ネットワークソケットとルーティング、スレッドに関する詳細を提供します。

8.36.3. 概要

server report

server heap-dump [--live]

server connector ls

server connector describe 'connector-name'

server connector start 'connector-name'

server connector stop 'connector-name'

server connector ipfilter ls 'connector-name'

server connector ipfilter set 'connector-name' --rules='[ACCEPT|REJECT]/cidr',...

server connector ipfilter clear 'connector-name'

server datasource ls

server datasource test 'datasource-name'

8.36.4. サーバーコネクター IP フィルターのオプション

--rules='[ACCEPT|REJECT]/cidr',...

1 つ以上の IP フィルタリングルール。

8.36.5. 例

server report
ネットワーク、スレッド、メモリーなどに関する情報を含むサーバーレポートを取得します。

server heap-dump
サーバーデータディレクトリーに JVM ヒープダンプを生成し、生成されたファイルの名前を返します。

server connector ls
サーバーで使用可能なすべてのコネクターをリスト表示します。

server connector describe endpoint-default
ホスト、ポート、ローカルおよびグローバル接続、IP フィルタリングルールなど、指定されたコネクターに関する情報を表示します。

server connector stop my-hotrod-connector
クラスター全体で確立されたすべての接続をドロップするコネクターを停止します。要求を処理しているコネクターを停止しようとすると、このコマンドは拒否されます。

server connector start my-hotrod-connector
クラスター全体の接続を受け入れることができるように、コネクターを開始します。

server connector ipfilter ls my-hotrod-connector
クラスター全体のコネクターでアクティブなすべての IP フィルタリングルールを一覧表示します。

server connector ipfilter set my-hotrod-connector --rules=ACCEPT/192.168.0.0/16,REJECT/10.0.0.0/8 クラスター全体のコネクターに IP フィルタリングルールを設定します。既存のすべてのルールを置き換えます。拒否ルールの 1 つが、呼び出された接続のアドレスと一致する場合、このコマンドは拒否されます。

server connector ipfilter clear my-hotrod-connector
クラスター全体のコネクターのすべての IP フィルタリングルールを削除します。

server datasource ls
サーバー上で使用可能なすべてのデータソースをリスト表示します。

server datasource test my-datasource
データソースでテスト接続を実行します。

8.37. SHUTDOWN(1)

8.37.1. 名前

shutdown - サーバーインスタンスとクラスターを停止します。

8.37.2. 概要

shutdown server ['SERVERS']

shutdown cluster

shutdown container

8.37.3. 例

shutdown server
CLI が接続されているサーバーを停止します。

shutdown server my_server01
ホスト名が **my_server01** のサーバーを停止します。

shutdown cluster
クラスターの状態を保存し、キャッシュストアがある場合はエントリーを永続化した後、クラスター内のすべてのノードを停止します。

shutdown container
サーバープロセスを終了せずにデータコンテナーを停止します。クラスターの状態を保存し、キャッシュストアがある場合はエントリーを保持します。サーバーインスタンスは、アクティブなエンドポイントとクラスタリングで引き続き実行されます。コンテナーリソースへの REST 呼び出しは、503 Service Unavailable 応答になります。**shutdown container** コマンドは、リソースのライフサイクル管理を自動化する Kubernetes などの環境を対象としています。自己管理環境では、**shutdown server** または **shutdown cluster** コマンドを使用してサーバーを停止する必要があります。

8.37.4. 関連項目

connect(1)、disconnect(1)、quit(1)

8.38. SITE(1)

8.38.1. 名前

site - バックアップの場所を管理し、サイト間のレプリケーション操作を実行します。

8.38.2. 概要

site status ['OPTIONS']

site bring-online ['OPTIONS']

site take-offline ['OPTIONS']

site push-site-state ['OPTIONS']

site cancel-push-state ['OPTIONS']

site cancel-receive-state ['OPTIONS']

site push-site-status ['OPTIONS']

site state-transfer-mode get|set ['OPTIONS']

site name

site view

site is-relay-node

site relay-nodes

8.38.3. オプション

-c, --cache='CACHE_NAME'
キャッシュを指定します。

-a, --all-caches
コマンドをすべてのキャッシュに適用します。

-s, --site='SITE_NAME'
バックアップの場所を指定します。

8.38.4. 状態転送モードのオプション

--mode='MODE'
状態転送モードを設定します。値は **MANUAL**(デフォルト) または **AUTO** です。

8.38.5. 例

site status --cache=mycache
mycache のすべてのバックアップ場所のステータスを返します。

site status --all-caches
バックアップのあるすべてのキャッシュの各バックアップロケーションのステータスを返します。

site status --cache=mycache --site=NYC
mycache の **NYC** のステータスを返します。

site bring-online --cache=mycache --site=NYC
mycache のサイト **NYC** をオンラインにします。

site take-offline --cache=mycache --site=NYC
mycache のサイト **NYC** をオフラインにします。

site push-site-state --cache=mycache --site=NYC
キャッシュをリモートバックアップの場所にバックアップします。

site push-site-status --cache=mycache
mycache をバックアップする操作のステータスを表示します。

site cancel-push-state --cache=mycache --site=NYC
mycache を **NYC** にバックアップする操作をキャンセルします。

site cancel-receive-state --cache=mycache --site=NYC
NYC から状態を受信する操作をキャンセルします。

site clear-push-state-status --cache=myCache
mycache の状態をプッシュする操作のステータスをクリアします。

site state-transfer-mode get --cache=myCache --site=NYC
mycache の状態転送モードを **NYC** に取得します。

site state-transfer-mode set --cache=myCache --site=NYC --mode=AUTO
mycache の **NYC** への自動状態転送を設定します。

site name
ローカルサイトの名前を返します。クロスサイトレプリケーションが設定されていない場合、ローカルサイトの名前は常に"local"です。

site view
すべてのサイトの名前のリストを返します。クロスサイトレプリケーションが設定されていない場合は、空のリスト ("[]") を返します。

site is-relay-node
ノードがクラスター間の RELAY メッセージを処理する場合は true を返します。

site relay-nodes
リレーノードのリストを論理名で返します。

8.39. STATS(1)

8.39.1. 名前

stats - リソースに関する統計を表示します。

8.39.2. 概要

stats ['PATH']

8.39.3. 例

stats //containers/default
デフォルトのコンテナーに関する統計を表示します。

stats //containers/default/caches/mycache
mycache キャッシュに関する統計を表示します。

8.39.4. 関連項目

cd(1)、ls(1)、describe(1)

8.40. TASK(1)

8.40.1. 名前

task - サーバー側のタスクとスクリプトを実行してアップロードします

8.40.2. 概要

task upload --file='script' 'TASK_NAME'

task exec ['TASK_NAME']

8.40.3. 例

task upload --file=hello.js hello
hello.js ファイルからスクリプトをアップロードし、**hello** という名前を付けます。

task exec @@cache@names
使用可能なキャッシュ名を返すタスクを実行します。

task exec hello -Pgreetee=world
hello という名前のスクリプトを実行し、**world** の値で **greetee** パラメーターを指定します。

8.40.4. オプション

-P, --parameters='PARAMETERS'
パラメーター値をタスクとスクリプトに渡します。

-f, --file='FILE'
指定された名前のスクリプトファイルをアップロードします。

8.40.5. 関連項目

ls(1)

8.41. UNALIAS(1)

8.41.1. 名前

unalias - エイリアスを削除します。

8.41.2. 概要

unalias 'ALIAS-NAME'

8.41.3. 例

unalias q
q エイリアスを削除します。

8.41.4. 関連項目

config(1)、alias(1)

8.42. USER(1)

8.42.1. 名前

user - プロパティーセキュリティーレルムで Data Grid ユーザーを管理します。

8.42.2. 概要

user ls

user create 'username'

user describe 'username'

user remove 'username'

user password 'username'

user groups 'username'

user encrypt-all

user roles ls 'principal'

user roles grant --roles='role1'[,'role2'...] 'principal'

user roles deny --roles='role1'[,'role2'...] 'principal'

user roles create --permissions='perm1'[,'perm2'...] 'role'

user roles remove 'role'

8.42.3. 説明

ls、**create**、**describe**、**remove**、**password**、**groups**、および **encrypt-all** サブコマンドを使用して、プロパティーレルムのユーザーを管理します。承認にクラスターロールマッパーを使用する場合は、**roles** サブコマンドを使用してプリンシパルからロールへのマッピングをリスト表示および変更します。

8.42.4. コマンドの概要

user ls
　プロパティーファイルに存在するユーザーまたはグループをリスト表示します。

user create 'username'
　パスワードの入力を求めた後、ユーザーを作成します。

user describe 'username'
　ユーザー名、レルム、およびユーザーが属するグループを含め、ユーザーについて説明します。

user remove 'username'
　指定されたユーザーをプロパティーファイルから削除します。

user password 'username'
　ユーザーのパスワードを変更します。

user groups 'username'

ユーザーが属するグループを設定します。

user encrypt-all

プレーンテキストのユーザープロパティーファイル内のすべてのパスワードを暗号化します。

user roles ls 'principal'

指定されたプリンシパル (ユーザーまたはグループ) のすべてのロールをリスト表示します。

user roles grant --roles='role1'[,'role2'...] 'principal'

プリンシパルに 1 つ以上のロールを付与します。

user roles deny --roles='role1'[,'role2'...] 'principal'

プリンシパルに対する 1 つ以上のロールを拒否します。

user roles create --permissions='perm1'[,'perm2'...] 'role'

指定された権限を持つ新しいロールを作成します。

user roles remove 'role'

既存のロールを削除します。

8.42.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help

コマンドまたはサブコマンドのヘルプページを表示します。

-s, --server-root='path-to-server-root'

サーバールートへのパス。デフォルトは **server** です。

-f, --users-file='users.properties'

ユーザーパスワードを含むプロパティーファイルの名前。デフォルトは **users.properties** です。

-w, --groups-file='groups.properties'

ユーザーからグループへのマッピングを含むプロパティーファイルの名前。デフォルトは **groups.properties** です。

8.42.6. ユーザー作成/変更のオプション

-a, --algorithms

パスワードのハッシュに使用されるアルゴリズムを指定します。

-g, --groups='group1,group2,...'

ユーザーが属するグループを指定します。

-p, --password='password'

ユーザーのパスワードを指定します。

-r, --realm='realm'

レルム名を指定します。

--plain-text

パスワードをプレーンテキストで保存するかどうかを定義します (非推奨)。

8.42.7. ユーザーリストのオプション

--groups

ユーザーの代わりにグループのリストを表示します。

8.42.8. ユーザー暗号化 (すべて) のオプション

-a, --algorithms

パスワードのハッシュに使用されるアルゴリズムを指定します。

8.42.9. ユーザーのロールのオプション

-p, --permissions

LIFECYCLE、**READ**、**WRITE**、**EXEC**、**LISTEN**、**BULK_READ**、**BULK_WRITE**、**ADMIN**、**CREATE**、**MONITOR**、**ALL**、**ALL_READ**、**ALL_WRITE** パーミッションから 1 つ以上指定します。

8.43. VERSION(1)

8.43.1. 名前

version - サーバーのバージョンと CLI のバージョンを表示します。

8.43.2. 概要

version

8.43.3. 例

version
サーバーと CLI のバージョンを返します。

8.43.4. 関連項目

help(1)