Using the Infinispan Command Line Interface

Table of Contents

Chapter 1. Getting Started with the Infinispan CLI

The command line interface (CLI) lets you remotely connect to Infinispan servers to access data and perform administrative functions.

1.1. Adding Infinispan Credentials

Infinispan Server provides a default property realm that restricts access to authenticated users only. Use the Infinispan CLI to add credentials.

Procedure

1. Open a terminal in `$ISPN_HOME`.
2. Define credentials with the `user` command as in the following examples:
   - Create a new user named "myuser" and specify a password:

     Linux

     ```
     $ bin/cli.sh user create myuser -p "qwer1234!"
     ```

     Microsoft Windows

     ```
     $ bin\cli.bat user create myuser -p "qwer1234!"
     ```

   - Create a new user that belongs to the "supervisor", "reader", and "writer" groups if you use security authorization:

     Linux

     ```
     $ bin/cli.sh user create myuser -p "qwer1234!" -g supervisor,reader,writer
     ```

     Microsoft Windows

     ```
     $ bin\cli.bat user create myuser -p "qwer1234!" -g supervisor,reader,writer
     ```

1.2. Connecting to Infinispan Servers

Establish CLI connections to Infinispan.

Prerequisites

Add user credentials and have at least one running Infinispan server instance.

Procedure

1. Open a terminal in `$ISPN_HOME`.

2. Start the CLI.
   - **Linux:**

```
$ bin/cli.sh
```

   - **Microsoft Windows:**

```
$ bin\cli.bat
```

3. Run the `connect` command and enter your username and password when prompted.
   - Infinispan Server on the default port of `11222`:

```
[disconnected]> connect
```

   - Infinispan Server with a port offset of `100`:

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

1.3. Navigating CLI Resources

The Infinispan CLI exposes a navigable tree that allows you to list, describe, and manipulate Infinispan cluster resources.

Press the tab key to display available commands and options. Use the `-h` option to display help text.

When you connect to a Infinispan cluster, it opens in the context of the default cache container.

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

- Use `ls` to list resources.

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

- Use `cd` to navigate the resource tree.

```
[//containers/default]> cd caches
```

- Use `describe` to view information about resources.

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

The Infinispan CLI exposes different resources to:

- create, modify, and manage local or clustered caches.
- perform administrative operations for Infinispan clusters.

Cache Resources

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

`caches`
: Infinispan cache instances. The default cache container is empty. Use the CLI to create caches from templates or `infinispan.xml` files.

`counters`
: `Strong` or `Weak` counters that record the count of objects.

`configurations`
: Infinispan configurations.

`schemas`
: Protocol Buffers (Protobuf) schemas that structure data in the cache.

`tasks`
: Remote tasks creating and managing Infinispan cache definitions.

Cluster Resources

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

`containers`
: Cache containers on the Infinispan cluster.

`cluster`
: Lists Infinispan servers joined to the cluster.

`server`
: Resources for managing and monitoring Infinispan servers.

1.4. Shutting Down Infinispan Servers

Gracefully shut down running Infinispan servers to passivate all entries to disk and persist state.

Procedure

1. Create a CLI connection to Infinispan.
2. Do one of the following:
   - Stop individual servers with the `shutdown server` command:

     ```
     [//containers/default]> shutdown server $hostname
     ```

   - Stop all nodes in the cluster with the `shutdown cluster` command:

     ```
     [//containers/default]> shutdown cluster
     ```

Verification

Check the server logs for the following messages:

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

Chapter 2. Performing Cache Operations with the Infinispan CLI

The command line interface (CLI) lets you remotely connect to Infinispan servers to access data and perform administrative functions.

2.1. Creating Caches with the Infinispan Command Line Interface (CLI)

Use the Infinispan CLI to add caches from templates or configuration files in XML or JSON format.

Prerequisites

Add Infinispan credentials and start at least one Infinispan server instance.

Procedure

1. Create a CLI connection to Infinispan.
2. Add cache definitions with the `create cache` command.
   - Add a cache definition from an XML or JSON file with the `--file` option.

     ```
     [//containers/default]> create cache --file=configuration.xml mycache
     ```

   - Add a cache definition from a template with the `--template` option.

     ```
     [//containers/default]> create cache --template=org.infinispan.DIST_SYNC mycache
     ```

     

     Press the tab key after the `--template=` argument to list available cache templates.

3. Verify the cache exists with the `ls` command.

   ```
   [//containers/default]> ls caches
   mycache
   ```

4. Retrieve the cache configuration with the `describe` command.

   ```
   [//containers/default]> describe caches/mycache
   ```

Reference

- Creating Infinispan CLI Connections
- Performing Cache Operations with the Infinispan CLI

2.1.1. XML Configuration

Infinispan configuration in XML format must conform to the schema and include:

- `<infinispan>` root element.
- `<cache-container>` definition.

Example XML Configuration

```
<infinispan>
    <cache-container>
        <distributed-cache name="myCache" mode="SYNC">
          <encoding media-type="application/x-protostream"/>
          <memory max-count="1000000" when-full="REMOVE"/>
        </distributed-cache>
    </cache-container>
</infinispan>
```

2.1.2. JSON Configuration

Infinispan configuration in JSON format:

- Requires the cache definition only.
- Must follow the structure of an XML configuration.
  - XML elements become JSON objects.
  - XML attributes become JSON fields.

Example JSON Configuration

```
{
  "distributed-cache": {
    "name": "myCache",
    "mode": "SYNC",
    "encoding": {
      "media-type": "application/x-protostream"
      },
    "memory": {
      "max-count": 1000000,
      "when-full": "REMOVE"
    }
  }
}
```

2.2. Adding Cache Entries

Create `key:value` pair entries in the data container.

Prerequisites

Create a Infinispan cache that can store your data.

Procedure

1. Create a CLI connection to Infinispan.
2. Add entries into your cache as follows:
   - Use the `put` command from the context of a cache:

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

   - Use the `--cache=` with the `put` command:

     ```
     [//containers/default]> put --cache=mycache hello world
     ```

3. Use the `get` command to verify entries.

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

2.3. Clearing Caches and Deleting Entries

Remove data from caches with the Infinispan CLI.

Procedure

1. Create a CLI connection to Infinispan.
2. Do one of the following:
   - Delete all entries with the `clearcache` command.

     ```
     [//containers/default]> clearcache mycache
     ```

   - Remove specific entries with the `remove` command.

     ```
     [//containers/default]> remove --cache=mycache hello
     ```

2.4. Deleting Caches

Drop caches to remove them and delete all data they contain.

Procedure

1. Create a CLI connection to Infinispan.

2. Remove caches with the `drop` command.

```
[//containers/default]> drop cache mycache
```

Chapter 3. Performing Batch Operations

Process operations in groups, either interactively or using batch files.

Prerequisites

- A running Infinispan cluster.

3.1. Performing Batch Operations with Files

Create files that contain a set of operations and then pass them to the Infinispan CLI.

Procedure

1. Create a file that contains a set of operations.

   For example, create a file named `batch` that creates a cache named `mybatch`, adds two entries to the cache, and disconnects from the CLI.

   ```
   $ cat > batch<<EOF
   create cache --template=org.infinispan.DIST_SYNC mybatch
   put --cache=mybatch hello world
   put --cache=mybatch hola mundo
   disconnect
   EOF
   ```

2. Run the CLI and specify the file as input.

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

3. Create a new Infinispan CLI connection and verify `mybatch`.

   ```
   [//containers/default]> ls caches
   ___protobuf_metadata
   mybatch
   [//containers/default]> ls caches/mybatch
   hola
   hello
   [//containers/default]> disconnect
   [disconnected]>
   ```

CLI batch files support system property expansion. Strings that use the `${property}` format are replaced with the value of the `property` system property.

3.2. Performing Batch Operations Interactively

Use the standard input stream, **stdin**, to perform batch operations interactively.

Procedure

1. Start the Infinispan CLI in interactive mode.

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

If you do not use the `-c` flag, you must run the `connect` command.

```
$ bin/cli.sh -f -
connect
```

2. Run batch operations, for example:

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

Use `echo` to add commands in interactive mode.

The following example shows how to use `echo describe` to get cluster information:

```
$ echo describe|bin/cli.sh -c localhost:11222 -f -
{
  "name" : "default",
  "version" : "10.0.0-SNAPSHOT",
  "coordinator" : false,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "___protobuf_metadata",
"org.infinispan.DIST_SYNC", "qcache", "org.infinispan.LOCAL", "dist_cache_01",
"org.infinispan.INVALIDATION_SYNC", "org.infinispan.REPL_SYNC",
"org.infinispan.SCATTERED_SYNC", "mycache", "org.infinispan.INVALIDATION_ASYNC",
"mybatch", "org.infinispan.DIST_ASYNC" ],
  "cluster_name" : "cluster",
  "physical_addresses" : "[192.168.1.7:7800]",
  "coordinator_address" : "thundercat-34689",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "4",
  "running_cache_count" : "4",
  "node_address" : "thundercat-47082",
  "cluster_members" : [ "thundercat-34689", "thundercat-47082" ],
  "cluster_members_physical_addresses" : [ "10.36.118.25:7801", "192.168.1.7:7800" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "___protobuf_metadata",
    "started" : true
  }, {
    "name" : "mybatch",
    "started" : true
  } ]
}
```

Chapter 4. Configuring the Infinispan CLI

Define configuration properties for the Infinispan CLI.

4.1. Configuring Startup Properties

Configure Infinispan CLI to automatically connect to specific URLs on startup and run batch files.

Prerequisites

- Add Infinispan credentials.
- Optionally create CLI batch files to run at startup.

Procedure

1. Specify a hostname and port to which the CLI automatically connects.

   For example, connect to the default server location when you start the CLI:

   ```
   $ bin/cli.sh config set autoconnect-url http://127.0.0.1:11222
   ```

2. If you want to run a batch file when you start the CLI, specify the path to it as follows:

   ```
   $ bin/cli.sh config set autoexec /path/to/batch/file
   ```

3. Verify your CLI configuration.

   ```
   $ bin/cli.sh config get autoconnect-url
   autoconnect-url=http://127.0.0.1:11222

   $ bin/cli.sh config get autoexec
   autoexec=/path/to/batch/file
   ```

4. Start the Infinispan CLI and enter your credentials when prompted.

   ```
   $ bin/cli.sh

   [ //containers/default]>
   ```

Reference

Performing Batch Operations

4.2. Trusting Infinispan Server Connections

Secure Infinispan CLI connections to Infinispan Server with SSL/TLS certificates. If you create a key

store as an SSL identity for Infinispan Server, the CLI can validate server certificates to verify the identity.

Prerequisites

- Set up an SSL identity for Infinispan Server.
- Add Infinispan credentials.

Procedure

1. Specify the location of the server key store, as in the following example:

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

2. Define the key store password, if necessary, as follows:

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

3. Verify your CLI configuration.

```
$ bin/cli.sh config get truststore
truststore=/home/user/my-trust-store.jks

$ bin/cli.sh config get truststore-password
truststore-password=secret
```

Reference

Setting Up SSL Identities for Infinispan Server

4.3. Infinispan CLI Configuration Properties

Property	Description
`autoconnect-url`	Specifies the URL of a Infinispan server to connect to on startup.
`autoexec`	Defines the path to a CLI batch file that runs on startup. If an `autoconnect-url` property is specified, a connection to the server is established before the batch file runs.
`trustall`	Configures the CLI to trust all server certificates. Boolean value. Set to `true` to trust all certificates and bypass certificate validation.
`truststore`	Defines the path to the server key store. The CLI uses this key store to validate Infinispan Server identity.

Property	Description
`truststore_password`	Specifies a password for the key store, if required.

Persistent configuration

Infinispan CLI stores configuration in the following OS-specific directories:

- Linux/Unix: `$HOME/.config/{brandshortname}`
- Windows: `$APPDATA/Sun/Java/{brandshortname}`
- OS X: `$HOME/Library/Java/{brandshortname}`

This directory contains the following files:

- `cli.properties`: CLI configuration properties.
- `aliases`: custom CLI aliases.
- `history`: CLI history.

Chapter 5. Working with Counters

Counters provide atomic increment and decrement operations that record the count of objects.

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

5.1. Creating Counters

Create strong and weak counters with the Infinispan CLI.

Procedure

1. Create a CLI connection to Infinispan.
2. Run the `create counter` command with the appropriate arguments.
   a. Create `my-weak-counter`.

```
[//containers/default]> create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --type=weak my-weak-counter
```

   b. Create `my-strong-counter`.

```
[//containers/default]> create counter --initial-value=3 --storage=PERSISTENT --type=strong my-strong-counter
```

3. List available counters.

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

4. Verify counter configurations.
   a. Describe `my-weak-counter`.

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

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

b. Describe `my-strong-counter`.

```
[//containers/default]> describe counters/my-strong-counter

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

5.2. Adding Deltas to Counters

Increment or decrement counters with arbitrary values.

Procedure

1. Select a counter.

```
[//containers/default]> counter my-weak-counter
```

2. List the current count.

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

3. Increment the counter value by `2`.

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

4. Decrement the counter value by `-4`.

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

Strong counters return values after the operation is applied. Use `--quiet=true` to hide the return value.

For example, `my-strong-counter]> add --delta=3 --quiet=true`.

Weak counters return empty responses.

Chapter 6. Querying Caches with Protobuf Metadata

Infinispan supports using Protocol Buffers (Protobuf) to structure data in the cache so that you can query it.

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

6.1. Configuring Media Types

Encode cache entries with different media types to store data in a format that best suits your requirements.

For example, the following procedure shows you how to configure the `application/x-protostream` media type.

Procedure

1. Create a Infinispan configuration file that adds a distributed cache named `qcache` and configures the media type, for example:

```
<infinispan>
   <cache-container>
      <distributed-cache name="qcache">
         <encoding>
            <key media-type="application/x-protostream"/>
            <value media-type="application/x-protostream"/>
         </encoding>
      </distributed-cache>
   </cache-container>
</infinispan>
```

2. Create `qcache` from `pcache.xml` with the `--file=` option.

```
[//containers/default]> create cache --file=pcache.xml pcache
```

3. Verify `pcache`.

```
[//containers/default]> ls caches
pcache
___protobuf_metadata
[//containers/default]> describe caches/pcache
{
  "distributed-cache" : {
    "mode" : "SYNC",
    "encoding" : {
      "key" : {
        "media-type" : "application/x-protostream"
      },
      "value" : {
        "media-type" : "application/x-protostream"
      }
    },
    "transaction" : {
      "mode" : "NONE"
    }
  }
}
```

4. Add an entry to `pcache` and check the encoding.

```
[//containers/default]> put --cache=pcache good morning
[//containers/default]> cd caches/pcache
[//containers/default/caches/pcache]> get good
{
  "_type" : "string",
  "_value" : "morning"
}
```

6.2. Registering Protobuf Schemas

Protobuf schemas contain data structures known as messages in `.proto` definition files.

Procedure

1. Create a schema file named `person.proto` with the following messages:

```
package org.infinispan.rest.search.entity;

message Address {
    required string street = 1;
    required string postCode = 2;
}

message PhoneNumber {
    required string number = 1;
}

message Person {
    optional int32 id = 1;
    required string name = 2;
    required string surname = 3;
    optional Address address = 4;
    repeated PhoneNumber phoneNumbers = 5;
    optional uint32 age = 6;
    enum Gender {
        MALE = 0;
        FEMALE = 1;
    }

    optional Gender gender = 7;
}
```

2. Register `person.proto`.

```
[//containers/default]> schema --upload=person.proto person.proto
```

3. Verify `person.proto`.

```
[//containers/default]> cd caches/___protobuf_metadata
[//containers/default/caches/___protobuf_metadata]> ls
person.proto
[//containers/default/caches/___protobuf_metadata]> get person.proto
```

6.3. Querying Caches with Protobuf Schemas

Infinispan automatically converts JSON to Protobuf so that you can read and write cache entries in JSON format and use Protobuf schemas to query them.

For example, consider the following JSON documents:

lukecage.json

```
{
  "_type":"org.infinispan.rest.search.entity.Person",
  "id":2,
  "name":"Luke",
  "surname":"Cage",
  "gender":"MALE",
  "address":{"street":"38th St","postCode":"NY 11221"},
  "phoneNumbers":[{"number":4444},{"number":5555}]
}
```

jessicajones.json

```
{
  "_type":"org.infinispan.rest.search.entity.Person",
  "id":1,
  "name":"Jessica",
  "surname":"Jones",
  "gender":"FEMALE",
  "address":{"street":"46th St","postCode":"NY 10036"},
  "phoneNumbers":[{"number":1111},{"number":2222},{"number":3333}]
}
```

matthewmurdock.json

```
{
  "_type":"org.infinispan.rest.search.entity.Person",
  "id":3,
  "name":"Matthew",
  "surname":"Murdock",
  "gender":"MALE",
  "address":{"street":"57th St","postCode":"NY 10019"},
  "phoneNumbers":[]
}
```

Each of the preceding JSON documents contains:

- A `_type` field that identifies the Protobuf message to which the JSON document corresponds.
- Several fields that correspond to datatypes in the `person.proto` schema.

Procedure

1. Navigate to the `pcache` cache.

```
[//containers/default/caches]> cd pcache
```

2. Add each JSON document as an entry to the cache, for example:

```
[//containers/default/caches/pcache]> put --encoding=application/json
--file=jessicajones.json jessicajones
[//containers/default/caches/pcache]> put --encoding=application/json
--file=matthewmurdock.json matthewmurdock
[//containers/default/caches/pcache]> put --encoding=application/json
--file=lukecage.json lukecage
```

3. Verify that the entries exist.

```
[//containers/default/caches/pcache]> ls
lukecage
matthewmurdock
jessicajones
```

4. Query the cache to return entries from the Protobuf `Person` entity where the gender datatype is `MALE`.

```
[//containers/default/caches/pcache]> query "from
org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
{
  "total_results" : 2,
  "hits" : [ {
    "hit" : {
      "_type" : "org.infinispan.rest.search.entity.Person",
      "id" : 2,
      "name" : "Luke",
      "surname" : "Cage",
      "gender" : "MALE",
      "address" : {
        "street" : "38th St",
        "postCode" : "NY 11221"
      },
      "phoneNumbers" : [ {
        "number" : "4444"
      }, {
        "number" : "5555"
      } ]
    }
  }, {
    "hit" : {
      "_type" : "org.infinispan.rest.search.entity.Person",
      "id" : 3,
      "name" : "Matthew",
      "surname" : "Murdock",
      "gender" : "MALE",
      "address" : {
        "street" : "57th St",
        "postCode" : "NY 10019"
      }
    }
  } ]
}
```

Chapter 7. Performing Cross-Site Replication Operations

Infinispan clusters running in different locations can discover and communicate with each other to backup data.

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

7.1. Bringing Backup Locations Offline and Online

Take backup locations offline manually and bring them back online.

Procedure

1. Create a CLI connection to Infinispan.
2. Check if backup locations are online or offline with the `site status` command:

```
//containers/default]> site status --cache=cacheName --site=NYC
```

`--site` is an optional argument. If not set, the CLI returns all backup locations.

3. Manage backup locations as follows:
   - Bring backup locations online with the `bring-online` command:

```
//containers/default]> site bring-online --cache=customers --site=NYC
```

   - Take backup locations offline with the `take-offline` command:

```
//containers/default]> site take-offline --cache=customers --site=NYC
```

For more information and examples, run the `help site` command.

7.2. Pushing State to Backup Locations

Transfer cache state to remote backup locations.

Procedure

1. Create a CLI connection to Infinispan.
2. Use the `site` command to push state transfer, as in the following example:

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

For more information and examples, run the `help site` command.

Chapter 8. Backing Up and Restoring Infinispan Clusters

Create archives of Infinispan resources that include cached entries, cache configurations, Protobuf schemas, and server scripts. You can then use the backup archives to restore Infinispan Server clusters after a restart or migration.

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

8.1. Backing Up Infinispan Clusters

Create backup archives in `.zip` format that you can download or store on Infinispan Server.

Prerequisites

Backup archives should reflect the most recent cluster state. For this reason you should ensure the cluster is no longer accepting write requests before you create backup archives.

Procedure

1. Create a CLI connection to Infinispan.
2. Run the `backup create` command with the appropriate options, for example:
   - Back up all resources with an automatically generated name.

     ```
     [//containers/default]> backup create
     ```

   - Back up all resources in a backup archive named `example-backup`.

     ```
     [//containers/default]> backup create -n example-backup
     ```

   - Back up all resources to the `/some/server/dir` path on the server.

     ```
     [//containers/default]> backup create -d /some/server/dir
     ```

   - Back up only caches and cache configurations.

     ```
     [//containers/default]> backup create --caches=* --cache-configs=*
     ```

   - Back up named Protobuf schemas only.

```
[//containers/default]> backup create --proto-schemas=schema1,schema2
```

3. List available backup archives on the server.

```
[//containers/default]> backup ls
```

4. Download the `example-backup` archive from the server.

   If the backup operation is still in progress, the command waits for it to complete.

```
[//containers/default]> backup get example-backup
```

5. Optionally delete the `example-backup` archive from the server.

```
[//containers/default]> backup delete example-backup
```

8.2. Restoring Infinispan Clusters from Backup Archives

Apply the content of backup archives to Infinispan clusters to restore them to the backed up state.

Prerequisites

- Create a backup archive that is either local to the Infinispan CLI or stored on Infinispan Server.
- Ensure that the target container matches the container name in the backup archive. You cannot restore backups if the container names do not match.

Procedure

1. Create a CLI connection to Infinispan.
2. Run the `backup restore` command with the appropriate options.
   - Restore all content from a backup archive accessible on the server.

```
[//containers/default]> backup restore /some/path/on/the/server
```

   - Restore all content from a local backup archive.

```
[//containers/default]> backup restore -u /some/local/path
```

   - Restore only cache content from a backup archive on the server.

```
[//containers/default]> backup restore /some/path/on/the/server --caches=*
```

Chapter 9. Command Reference

Review manual pages for Infinispan CLI commands.

Use `help` command to access manual pages directly from your CLI session.

For example, to view the manual page for the `get` command do the following:

```
$ help get
```

Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//add.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//backup.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//cache.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//cas.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//cd.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//clearcache.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//config.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//connect.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//container.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//counter.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//create.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//describe.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//disconnect.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//drop.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//encoding.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//get.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//help.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../../cli/cli-client/src/main/resources/help//logging.adoc[leveloffset=+1] Unresolved directive in

../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//ls.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//migrate.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//patch.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//put.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//query.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//quit.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//remove.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//reset.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//schema.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//shutdown.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//site.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//stats.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//task.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//user.adoc[leveloffset=+1] Unresolved directive in ../../stories/assembly_cli_command_reference.adoc - include::../../../../cli/cli-client/src/main/resources/help//version.adoc[leveloffset=+1]