Infinispan Guide to Cross-Site Replication

Table of Contents

Find out how Infinispan performs cross-site replication so you can get optimal performance and avoid issues. Learn how to configure Infinispan to back up data to remote clusters. Follow procedures to transfer state from one cluster to another, take sites offline, and so on.

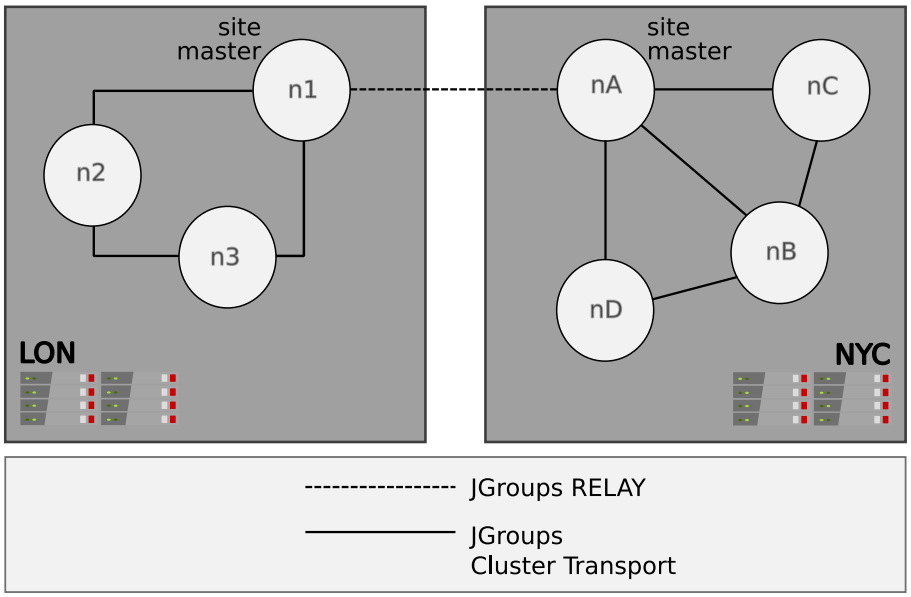
Chapter 1. Infinispan Cross-Site Replication

Cross-site replication allows you to back up data from one Infinispan cluster to another. Learn the concepts to understand how Infinispan cross-site replication works before you configure your clusters.

1.1. Cross-Site Replication

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

Sites are typically data centers in various geographic locations. Cross-site replication bridges Infinispan clusters in sites to form global clusters, as in the following diagram:



LON is a datacenter in London, England.
NYC is a datacenter in New York City, USA.

Infinispan can form global clusters across two or more sites.

For example, configure a third Infinispan cluster running in San Francisco, **SFO**, as backup location for **LON** and **NYC**.

1.1.1. Site Masters

Site masters are the nodes in Infinispan clusters that are responsible for sending and receiving requests from backup locations.

If a node is not a site master, it must forward backup requests to a local site master. Only site masters can send requests to backup locations.

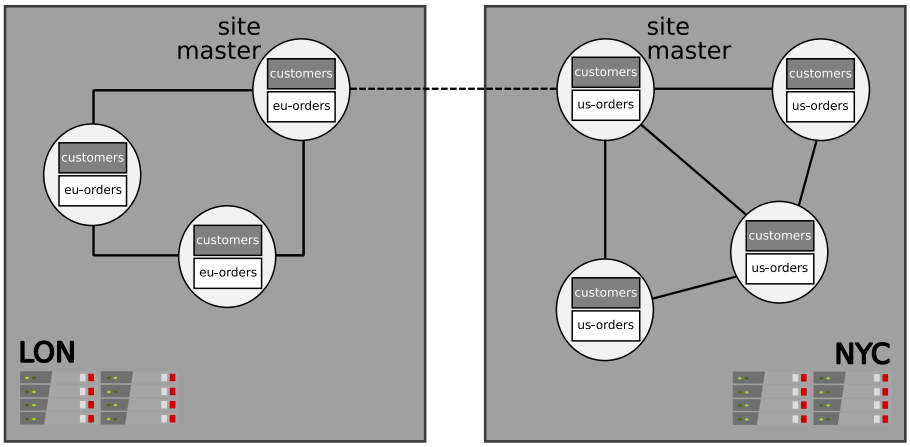
For optimal performance, you should configure all nodes as site masters. This increases the speed of backup requests because each node in the cluster can backup to remote sites directly without having to forward backup requests to site masters.

Diagrams in this document illustrate Infinispan clusters with one site master because this is the default for the JGroups RELAY2 protocol. Likewise, a single site master is easier to illustrate because each site master in a cluster communicates with each site master in the remote cluster.

1.2. Adding Backups to Caches

Name remote sites as backup locations in your cache definitions.

For example, the following diagram shows three caches, "customers", "eu-orders", and "us-orders":



- In **LON**, "customers" names **NYC** as a backup location.
- In **NYC**, "customers" names **LON** as a backup location.
- "eu-orders" and "us-orders" do not have backups and are local to the respective cluster.

1.3. Backup strategies

Infinispan replicates data between clusters at the same time that writes to caches occur. For example, if a client writes "k1" to **LON**, Infinispan backs up "k1" to **NYC** at the same time.

To back up data to a different cluster, Infinispan can use either a synchronous or asynchronous strategy.

Synchronous strategy

When Infinispan replicates data to backup locations, it writes to the cache on the local cluster and the cache on the remote cluster concurrently. With the synchronous strategy, Infinispan waits for both write operations to complete before returning.

You can control how Infinispan handles writes to the cache on the local cluster if backup operations fail. Infinispan can do the following:

- Ignore the failed backup and silently continue the write to the local cluster.
- Log a warning message or throw an exception and continue the write to the local cluster.
- Handle failed backup operations with custom logic.

Synchronous backups also support two-phase commits with caches that participate in optimistic transactions. The first phase of the backup acquires a lock. The second phase commits the modification.

Two-phase commit with cross-site replication has a significant performance impact because it requires two round-trips across the network.

Asynchronous strategy

When Infinispan replicates data to backup locations, it does not wait until the operation completes before writing to the local cache.

Asynchronous backup operations and writes to the local cache are independent of each other. If backup operations fail, write operations to the local cache continue and no exceptions occur. When this happens Infinispan also retries the write operation until the remote cluster disconnects from the cross-site view.

Synchronous vs asynchronous backups

Synchronous backups offer the strongest guarantee of data consistency across sites. If `strategy=sync`, when `cache.put()` calls return you know the value is up to date in the local cache and in the backup locations.

The trade-off for this consistency is performance. Synchronous backups have much greater latency in comparison to asynchronous backups.

Asynchronous backups, on the other hand, do not add latency to client requests so they have no performance impact. However, if `strategy=async`, when `cache.put()` calls return you cannot be sure of that the value in the backup location is the same as in the local cache.

1.4. Automatically Taking Backups Offline

You can configure backup locations to go offline automatically when the remote sites become unavailable. This prevents Infinispan nodes from continuously attempting to replicate data to offline backup locations, which results in error messages and consumes resources.

Timeout for backup operations

Backup configurations include timeout values for operations to replicate data. If operations do not complete before the timeout occurs, Infinispan records them as failures.

```
<!-- Operations to replicate data to NYC are recorded as failures if they do not
complete after 10 seconds. -->
<backup site="NYC" strategy="ASYNC" timeout="10000">
    <!-- Configuration for backup locations goes here. -->
</backup>
```

Number of failures

You can specify the number of **consecutive** failures that can occur before backup locations go offline.

For example, the following configuration for **NYC** sets five as the number of failed operations before it goes offline:

```
<!-- If a cluster attempts to replicate data to NYC and five consecutive operations
fail, NYC automatically goes offline. -->
<backup site="NYC" strategy="ASYNC" timeout="10000">
  <take-offline after-failures="5"/>
</backup>
```

Time to wait

You can also specify how long to wait before taking sites offline when backup operations fail. If a backup request succeeds before the wait time runs out, Infinispan does not take the site offline.

```
<!-- If a cluster attempts to replicate data to NYC and there are more than five
consecutive failures and 15 seconds elapse after the first failed operation, NYC
automatically goes offline. -->
<backup site="NYC" strategy="ASYNC" timeout="10000">
  <take-offline after-failures="5"
                min-wait="15000"/>
</backup>
```

Set a negative or zero value for the `after-failures` attribute if you want to use only a minimum time to wait to take sites offline.

```
<take-offline after-failures="-1" min-wait="10000"/>
```

1.5. State Transfer

State transfer is an administrative operation that synchronizes data between sites.

For example, **LON** goes offline and **NYC** starts handling client requests. When you bring **LON** back online, the Infinispan cluster in **LON** does not have the same data as the cluster in **NYC**.

To ensure the data is consistent between **LON** and **NYC**, you can push state from **NYC** to **LON**.

- State transfer is bidirectional. For example, you can push state from **NYC** to **LON** or from **LON** to **NYC**.
- Pushing state to offline sites brings them back online.
- State transfer overwrites only data that exists on both sites, the originating site and the receiving site. Infinispan does not delete data.

  For example, "k2" exists on **LON** and **NYC**. "k2" is removed from **NYC** while **LON** is offline.

When you bring **LON** back online, "k2" still exists at that location. If you push state from **NYC** to **LON**, the transfer does not affect "k2" on **LON**.

To ensure contents of the cache are identical after state transfer, remove all data from the cache on the receiving site before pushing state.

Use the `clear()` method or the `clearcache` command from the CLI.

- State transfer does not overwrite updates to data that occur after you initiate the push.

  For example, "k1,v1" exists on **LON** and **NYC**. **LON** goes offline so you push state transfer to **LON** from **NYC**, which brings **LON** back online. Before state transfer completes, a client puts "k1,v2" on **LON**.

  In this case the state transfer from **NYC** does not overwrite "k1,v2" because that modification happened after you initiated the push.

1.5.1. Automatic State Transfer

By default you must manually perform cross-site state transfer operations with the CLI or via JMX or REST.

However, when using the asynchronous backup strategy, Infinispan can automatically perform cross-site state transfer operations. When it detects that a backup location has come back online, and the network connection is stable, Infinispan initiates bi-directional state transfer between backup locations. For example, Infinispan simultaneously transfers state from **LON** to **NYC** and **NYC** to **LON**.

To avoid temporary network disconnects triggering state transfer operations, there are two conditions that backup locations must meet to go offline. The status of a backup location must be offline and it must not be included in the cross-site view with JGroups RELAY2.

Reference

- org.infinispan.Cache.clear()
- Clearing Caches with the CLI
- Clearing Caches with the REST API
- Configure Cross-Site State Transfer

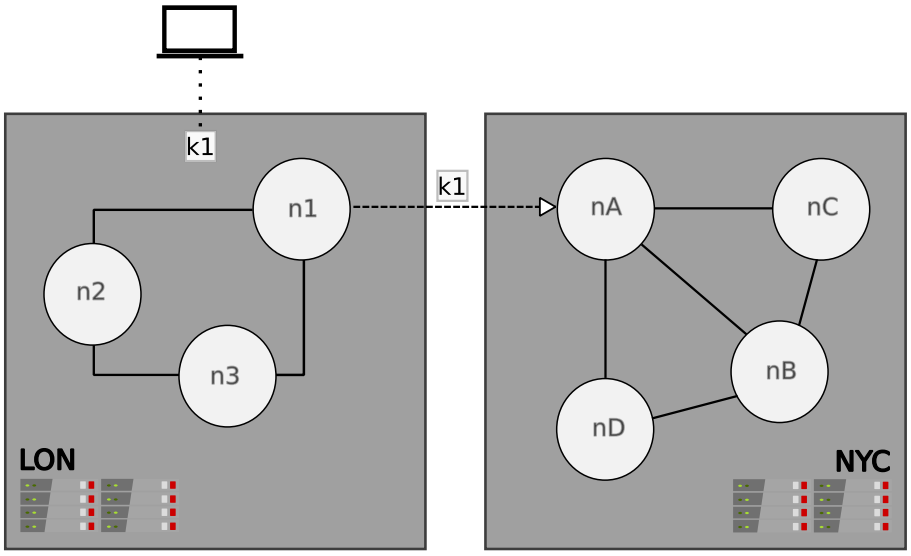
1.6. Client Connections Across Sites

Clients can write to Infinispan clusters in either an Active/Passive or Active/Active configuration.

Active/Passive

The following diagram illustrates Active/Passive where Infinispan handles client requests from one

site only:



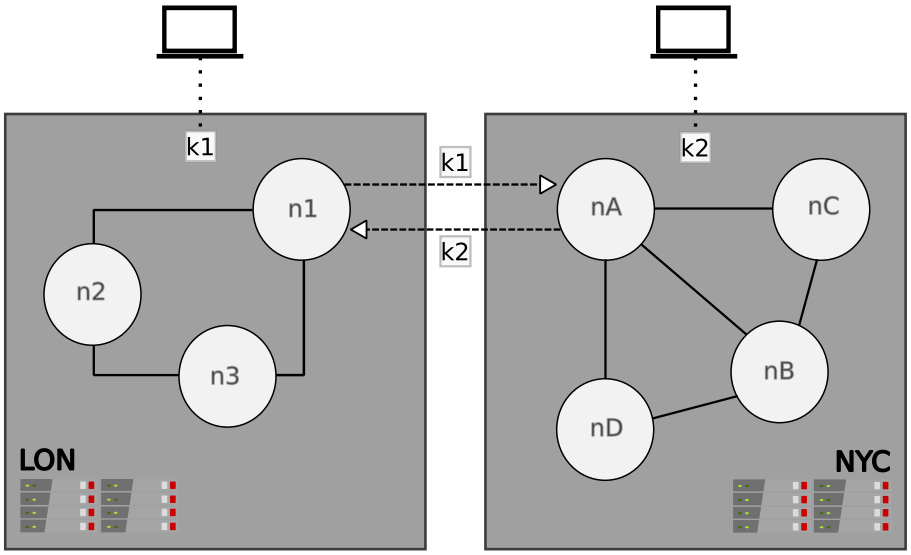
In the preceding image:

1. Client connects to the Infinispan cluster at **LON**.
2. Client writes "k1" to the cache.
3. The site master at **LON**, "n1", sends the request to replicate "k1" to the site master at **NYC**, "nA".

With Active/Passive, **NYC** provides data redundancy. If the Infinispan cluster at **LON** goes offline for any reason, clients can start sending requests to **NYC**. When you bring **LON** back online you can synchronize data with **NYC** and then switch clients back to **LON**.

Active/Active

The following diagram illustrates Active/Active where Infinispan handles client requests at two sites:



In the preceding image:

1. Client A connects to the Infinispan cluster at **LON**.
2. Client A writes "k1" to the cache.
3. Client B connects to the Infinispan cluster at **NYC**.
4. Client B writes "k2" to the cache.
5. Site masters at **LON** and **NYC** send requests so that "k1" is replicated to **NYC** and "k2" is replicated to **LON**.

With Active/Active both **NYC** and **LON** replicate data to remote caches while handling client requests. If either **NYC** or **LON** go offline, clients can start sending requests to the online site. You can then bring offline sites back online, push state to synchronize data, and switch clients as required.

Backup strategies

With an Active/Active configuration, you should always use an asynchronous backup strategy (`strategy=async`).

If multiple clients attempt to write to the same entry concurrently, and the backup strategy is synchronous (`strategy=sync`), then deadlocks occur. However you can use the synchronous backup strategy with an Active/Passive configuration if both sites access different data sets, in which case there is no risk of deadlocks from concurrent writes.

1.6.1. Concurrent Writes and Conflicting Entries

Conflicting entries can occur with Active/Active site configurations if clients write to the same entries at the same time but at different sites.

For example, client A writes to "k1" in **LON** at the same time that client B writes to "k1" in **NYC**. In this case, "k1" has a different value in **LON** than in **NYC**. After replication occurs, there is no guarantee which value for "k1" exists at which site.

To ensure data consistency, Infinispan uses a vector clock algorithm to detect conflicting entries during backup operations, as in the following illustration:

```
            LON       NYC

k1=(n/a)    0,0       0,0

k1=2        1,0  -->  1,0   k1=2

k1=3        1,1  <--  1,1   k1=3

k1=5        2,1       1,2   k1=8

                 -->  2,1 (conflict)
(conflict)  1,2  <--
```

Vector clocks are timestamp metadata that increment with each write to an entry. In the preceding example, `0,0` represents the initial value for the vector clock on "k1".

A client puts "k1=2" in **LON** and the vector clock is `1,0`, which Infinispan replicates to **NYC**. A client then puts "k1=3" in **NYC** and the vector clock updates to `1,1`, which Infinispan replicates to **LON**.

However if a client puts "k1=5" in **LON** at the same time that a client puts "k1=8" in **NYC**, Infinispan detects a conflicting entry because the vector value for "k1" is not strictly greater or less between **LON** and **NYC**.

When it finds conflicting entries, Infinispan uses the Java `compareTo(String anotherString)` method to compare site names. To determine which key takes priority, Infinispan selects the site name that is lexicographically less than the other. Keys from a site named **AAA** take priority over keys from a site named **AAB** and so on.

Following the same example, to resolve the conflict for "k1", Infinispan uses the value for "k1" that originates from **LON**. This results in "k1=5" in both **LON** and **NYC** after Infinispan resolves the conflict and replicates the value.

Prepend site names with numbers as a simple way to represent the order of priority for resolving conflicting entries; for example, **1LON** and **2NYC**.

Infinispan performs conflict resolution with the asynchronous backup strategy (`strategy=async`) only. However, you should not use the synchronous backup strategy with an Active/Active configuration because concurrent writes result in deadlocks.

Conflict resolution algorithms

Infinispan provides different algorithms for resolving conflicts in addition to an `XSiteEntryMergePolicy` SPI that lets you implement custom conflict resolution strategies.

Apart from the default conflict resolution strategy, which uses lexicographical comparison, you can use Infinispan conflict resolution algorithms to:

- Always remove conflicting entries.
- Keep write operations when write/remove conflicts occur.
- Remove entries when write/remove conflicts occur.

Find all available algorithms and their descriptions in the `org.infinispan.xsite.spi.XSiteMergePolicy` enum.

Additional resources

- java.lang.String#compareTo()
- org.infinispan.xsite.spi.XSiteMergePolicy
- org.infinispan.xsite.spi.XSiteEntryMergePolicy
- Customizing the Conflict Resolution Strategy

1.7. Expiration with Cross-Site Replication

Expiration removes cache entries based on time. Infinispan provides two ways to configure expiration for entries:

`lifespan`
: Sets the maximum amount of time that entries can exist.
When you set `lifespan` with cross-site replication, Infinispan clusters expire entries independently of remote sites.

`max-idle`
: Specifies how long entries can exist based on read or write operations in a given time period.
When you set a `max-idle` with cross-site replication, Infinispan clusters send touch commands to coordinate idle timeout values with remote sites.

Using maximum idle expiration in cross-site deployments can impact performance because the additional processing to keep `max-idle` values synchronized means some operations take longer to complete.

Chapter 2. Configuring Infinispan for Cross-Site Replication

Configuring Infinispan to replicate data across sites, you first set up cluster transport so Infinispan clusters can discover each other and site masters can communicate. You then add backup locations to cache definitions in your Infinispan configuration.

2.1. Configuring Cluster Transport for Cross-Site Replication

Add JGroups RELAY2 to your transport layer so that Infinispan clusters can communicate with backup locations.

Procedure

1. Open `infinispan.xml` for editing.
2. Add the RELAY2 protocol to a JGroups stack.
3. Configure Infinispan cluster transport to use the stack.

```
<infinispan>
  <jgroups>
    <!-- Extends the default UDP stack. -->
    <stack name="xsite" extends="udp">
      <!-- Adds RELAY2 for cross-site replication. -->
      <!-- Names the local site as LON. -->
      <!-- Specifies 1000 nodes as the maximum number of site masters. -->
      <relay.RELAY2 site="LON" xmlns="urn:org:jgroups" max_site_masters="1000"/>
      <!-- Uses the default TCP stack for inter-cluster communication. -->
      <!-- Names all sites that act as backup locations. -->
      <remote-sites default-stack="tcp">
         <remote-site name="LON"/>
         <remote-site name="NYC"/>
      </remote-sites>
    </stack>
  </jgroups>
  <cache-container name="default" statistics="true">
    <!-- Use the "xsite" stack for cluster transport. -->
    <transport cluster="${cluster.name}" stack="xsite"/>
  </cache-container>
</infinispan>
```

4. Save and close `infinispan.xml`.

Reference

- JGroups RELAY2 Stacks
- Infinispan Configuration Schema

2.1.1. JGroups RELAY2 Stacks

Infinispan clusters use JGroups RELAY2 for inter-cluster discovery and communication.

```
<jgroups>
   <!-- Uses the default JGroups UDP stack for intra-cluster traffic. -->
   <stack name="xsite" extends="udp">
      <!-- Adds RELAY2 to the stack for inter-cluster transport. -->
      <!-- Names the local site. Data in caches from the local site is replicated to
backup locations. -->
      <!-- Configures a maximum of 1000 site masters for the local cluster. -->
      <relay.RELAY2 xmlns="urn:org:jgroups"
                    site="LON"
                    max_site_masters="1000"/>
      <!-- Specifies all site names and uses the default JGroups TCP stack for inter-
cluster transport. -->
      <remote-sites default-stack="tcp">
         <!-- Names all sites that participate in cross-site replication. -->
         <remote-site name="LON"/>
         <remote-site name="NYC"/>
      </remote-sites>
   </stack>
</jgroups>
```

Set `max_site_masters` >= the number of nodes in the Infinispan cluster for optimal performance with backup requests.

2.1.2. Custom JGroups RELAY2 Stacks

The following configuration adds a custom RELAY2 stack that extends the default TCP stack:

```
<jgroups>
  <!-- Uses TCPPING instead of MPING for discovery. -->
   <stack name="relay-global" extends="tcp">
     <MPING stack.combine="REMOVE"/>
     <TCPPING initial_hosts="192.0.2.0[7800]"
              stack.combine="INSERT_AFTER"
              stack.position="TCP"/>
   </stack>
   <!-- Extends the default UDP stack with RELAY2. -->
   <!-- Specifies RELAY2 properties. -->
   <stack name="xsite" extends="udp">
      <relay.RELAY2 site="LON" xmlns="urn:org:jgroups"
                    max_site_masters="10"
                    can_become_site_master="true"/>
      <remote-sites default-stack="relay-global">
         <remote-site name="LON"/>
         <remote-site name="NYC"/>
      </remote-sites>
   </stack>
</jgroups>
```

You can also reference externally defined JGroups stack files, for example:

```
<infinispan>
  <jgroups>
     <stack-file name="relay-global" path="jgroups-relay.xml"/>
  </jgroups>
  <cache-container default-cache="replicatedCache">
    <transport stack="relay-global" />
    <replicated-cache name="replicatedCache"/>
  </cache-container>
</infinispan>
```

Where `jgroups-relay.xml` references a JGroups stack file such as:

```
<config xmlns="urn:org:jgroups"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:org:jgroups http://www.jgroups.org/schema/jgroups-
4.1.xsd">

    <!-- Use TCP for inter-cluster transport. -->
    <TCP bind_addr="127.0.0.1"
         bind_port="7200"
         port_range="30"

         thread_pool.min_threads="0"
         thread_pool.max_threads="8"
         thread_pool.keep_alive_time="5000"
    />

    <!-- Use TCPPING for inter-cluster discovery. -->
    <TCPPING timeout="3000"
             initial_hosts="127.0.0.1[7200]"
             port_range="3"
             ergonomics="false"/>

    <!-- Provide other JGroups stack configuration as required. -->
</config>
```

Reference

- Setting Up Infinispan Clusters
- JGroups RELAY2
- Relaying between multiple sites (RELAY2)

2.2. Adding Backup Locations to Caches

Specify the names of remote sites so Infinispan can back up data to those locations.

Procedure

1. Add the `backups` element to your cache definition.
2. Specify the name of each remote site with the `backup` element.

   As an example, in the **LON** configuration, specify **NYC** as the remote site.
3. Repeat the preceding steps so that each site is a backup for all other sites. For example, you cannot add **LON** as a backup for **NYC** without adding **NYC** as a backup for **LON**.

Cache configurations can be different across sites and use different backup strategies. Infinispan replicates data based on cache names.

Example "customers" configuration in **LON**

```
<replicated-cache name="customers">
  <backups>
    <backup site="NYC" strategy="ASYNC" />
  </backups>
</replicated-cache>
```

Example "customers" configuration in **NYC**

```
<distributed-cache name="customers">
  <backups>
    <backup site="LON" strategy="SYNC" />
  </backups>
</distributed-cache>
```

Reference

- Infinispan Configuration Schema

2.3. Backing Up to Caches with Different Names

By default, Infinispan replicates data between caches that have the same name.

Procedure

- Use `backup-for` to replicate data from a remote site into a cache with a different name on the local site.

For example, the following configuration backs up the "customers" cache on **LON** to the "eu-customers" cache on **NYC**.

```
<distributed-cache name="eu-customers">
  <backups>
    <backup site="LON" strategy="SYNC" />
  </backups>
  <backup-for remote-cache="customers" remote-site="LON" />
</distributed-cache>
```

2.4. Configuring Cross-Site State Transfer

Change cross-site state transfer settings to optimize performance and specify whether operations happen manually or automatically.

Procedure

- Use `<state-transfer>` to configure state transfer operations.

For example, the following configuration automatically performs cross-site state transfer with

custom settings:

```
<distributed-cache name="eu-customers">
   <backups>
      <backup site="LON" strategy="ASYNC">
         <state-transfer chunk-size="64"
                         timeout="30000"
                         max-retries="30"
                         wait-time="2000"
                         mode="AUTO"/>
      </backup>
   </backups>
</distributed-cache>
```

2.5. Customizing the Conflict Resolution Algorithm

Configure Infinispan to use a different algorithm to resolve conflicting entries between backup locations.

Procedure

1. Open your Infinispan configuration for editing.
2. Specify one of the Infinispan algorithms or a custom implementation with the `merge-policy` attribute for the `backups` element.

Using Infinispan algorithms

Find all Infinispan algorithms and their descriptions in the `org.infinispan.xsite.spi.XSiteMergePolicy` enum.

The following example configuration uses the `ALWAYS_REMOVE` algorithm that deletes conflicting entries from both sites:

```
<distributed-cache name="eu-customers">
   <backups merge-policy="ALWAYS_REMOVE">
      <backup site="LON" strategy="ASYNC"/>
   </backups>
</distributed-cache>
```

Using custom implementations

1. Create a custom `XSiteEntryMergePolicy` implementation.

```
public interface XSiteEntryMergePolicy<K, V> {
   CompletionStage<SiteEntry<V>> merge(K key, SiteEntry<V> localEntry, SiteEntry<V>
remoteEntry);
}
```

2. Specify the fully qualified class name as the value of the `merge-policy` attribute, as in the following example:

```
<distributed-cache name="eu-customers">
   <backups merge-policy="org.mycompany.MyCustomXSiteEntryMergePolicy">
      <backup site="LON" strategy="ASYNC"/>
   </backups>
</distributed-cache>
```

Additional resources

- org.infinispan.xsite.spi.XSiteEntryMergePolicy
- org.infinispan.xsite.spi.XSiteMergePolicy
- org.infinispan.xsite.spi.SiteEntry
- Infinispan Configuration Schema

2.6. Verifying Cross-Site Views

After you configure Infinispan for cross-site replication, you should verify that Infinispan clusters successfully form cross-site views.

Procedure

- Check log messages for `ISPN000439: Received new x-site view` messages.

For example, if the Infinispan cluster in **LON** has formed a cross-site view with the Infinispan cluster in **NYC**, it provides the following messages:

```
INFO  [org.infinispan.XSITE] (jgroups-5,${server.hostname}) ISPN000439: Received new
x-site view: [NYC]
INFO  [org.infinispan.XSITE] (jgroups-7,${server.hostname}) ISPN000439: Received new
x-site view: [NYC, LON]
```

2.7. Configuring Hot Rod Clients for Cross-Site Replication

Configure Hot Rod clients to use Infinispan clusters at different sites.

`hotrod-client.properties`

```
# Servers at the active site
infinispan.client.hotrod.server_list = LON_host1:11222,LON_host2:11222,LON_host3:11222

# Servers at the backup site
infinispan.client.hotrod.cluster.NYC =
NYC_hostA:11222,NYC_hostB:11222,NYC_hostC:11222,NYC_hostD:11222
```

`ConfigurationBuilder`

```
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.addServers("LON_host1:11222;LON_host2:11222;LON_host3:11222")
       .addCluster("NYC")
       .addClusterNodes(
"NYC_hostA:11222;NYC_hostB:11222;NYC_hostC:11222;NYC_hostD:11222")
```

Use the following methods to switch Hot Rod clients to the default cluster or to a cluster at a different site:

- `RemoteCacheManager.switchToDefaultCluster()`
- `RemoteCacheManager.switchToCluster(${site.name})`

Reference

- org.infinispan.client.hotrod.configuration package description
- org.infinispan.client.hotrod.configuration.ConfigurationBuilder
- org.infinispan.client.hotrod.RemoteCacheManager

Chapter 3. Performing Cross-Site Replication Operations

Bring sites online and offline. Transfer cache state to remote sites.

3.1. Performing Cross-Site Operations with the CLI

The Infinispan command line interface lets you remotely connect to Infinispan servers, manage sites, and push state transfer to backup locations.

Prerequisites

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

3.1.1. 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.

3.1.2. Configuring Cross-Site State Transfer Modes

You can configure cross-site state transfer operations to happen automatically when Infinispan detects that backup locations come online. Alternatively you can use the default mode, which is to manually perform state transfer through the CLI or via JMX or REST.

Procedure

1. Create a CLI connection to Infinispan.
2. Use the `site` command to configure state transfer modes, as in the following examples:
   - Retrieve the current state transfer mode.

```
[//containers/default]> site state-transfer-mode get --cache=cacheName
--site=NYC
"MANUAL"
```

   - Configure automatic state transfer operations for a cache and backup location.

```
[//containers/default]> site state-transfer-mode set --cache=cacheName --site=NYC
--mode=AUTO
```

Run the `help site` command for more information and examples.

3.1.3. 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.

Reference

Infinispan Command Line Interface

3.2. Performing Cross-Site Operations with the REST API

Infinispan servers provide a REST API that allows you to perform cross-site operations.

3.2.1. Getting Status of All Backup Locations

Retrieve the status of all backup locations with `GET` requests.

```
GET /v2/caches/{cacheName}/x-site/backups/
```

Infinispan responds with the status of each backup location in JSON format, as in the following example:

```
{
  "NYC": "online",
  "LON": "offline"
}
```

Table 1. Returned Status

Value	Description
`online`	All nodes in the local cluster have a cross-site view with the backup location.
`offline`	No nodes in the local cluster have a cross-site view with the backup location.
`mixed`	Some nodes in the local cluster have a cross-site view with the backup location, other nodes in the local cluster do not have a cross-site view. The response indicates status for each node.

3.2.2. Getting Status of Specific Backup Locations

Retrieve the status of a backup location with `GET` requests.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}
```

Infinispan responds with the status of each node in the site in JSON format, as in the following example:

```
{
  "NodeA":"offline",
  "NodeB":"online"
}
```

Table 2. Returned Status

Value	Description
`online`	The node is online.
`offline`	The node is offline.
`failed`	Not possible to retrieve status. The remote cache could be shutting down or a network error occurred during the request.

3.2.3. Taking Backup Locations Offline

Take backup locations offline with `POST` requests and the `?action=take-offline` parameter.

```
POST /v2/caches/{cacheName}/x-site/backups/{siteName}?action=take-offline
```

3.2.4. Bringing Backup Locations Online

Bring backup locations online with the `?action=bring-online` parameter.

```
POST /v2/caches/{cacheName}/x-site/backups/{siteName}?action=bring-online
```

3.2.5. Pushing State to Backup Locations

Push cache state to a backup location with the `?action=start-push-state` parameter.

```
POST /v2/caches/{cacheName}/x-site/backups/{siteName}?action=start-push-state
```

3.2.6. Canceling State Transfer

Cancel state transfer operations with the `?action=cancel-push-state` parameter.

```
POST /v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-push-state
```

3.2.7. Getting State Transfer Status

Retrieve status of state transfer operations with the `?action=push-state-status` parameter.

```
GET /v2/caches/{cacheName}/x-site/backups?action=push-state-status
```

Infinispan responds with the status of state transfer for each backup location in JSON format, as in the following example:

```
{
   "NYC":"CANCELED",
   "LON":"OK"
}
```

Table 3. Returned Status

Value	Description
SENDING	State transfer to the backup location is in progress.
OK	State transfer completed successfully.
ERROR	An error occurred with state transfer. Check log files.
CANCELLING	State transfer cancellation is in progress.

3.2.8. Clearing State Transfer Status

Clear state transfer status for sending sites with the `?action=clear-push-state-status` parameter.

```
POST /v2/caches/{cacheName}/x-site/local?action=clear-push-state-status
```

3.2.9. Modifying Take Offline Conditions

Sites go offline if certain conditions are met. Modify the take offline parameters to control when backup locations automatically go offline.

Procedure

1. Check configured take offline parameters with `GET` requests and the `take-offline-config` parameter.

   ```
   GET /v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
   ```

   The Infinispan response includes `after_failures` and `min_wait` fields as follows:

   ```
   {
     "after_failures": 2,
     "min_wait": 1000
   }
   ```

2. Modify take offline parameters in the body of `PUT` requests.

   ```
   PUT /v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
   ```

   If the operation successfully completes, the service returns `204 (No Content)`.

3.2.10. Canceling State Transfer from Receiving Sites

If the connection between two backup locations breaks, you can cancel state transfer on the site that is receiving the push.

Cancel state transfer from a remote site and keep the current state of the local cache with the `?action=cancel-receive-state` parameter.

```
POST /v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-receive-state
```

3.2.11. Getting Status of Backup Locations

Retrieve the status of all backup locations from Cache Managers with `GET` requests.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/
```

Infinispan responds with status in JSON format, as in the following example:

```
{
   "SFO-3":{
      "status":"online"
   },
   "NYC-2":{
      "status":"mixed",
      "online":[
         "CACHE_1"
      ],
      "offline":[
         "CACHE_2"
      ]
   }
}
```

Table 4. Returned Status

Value	Description
`online`	All nodes in the local cluster have a cross-site view with the backup location.
`offline`	No nodes in the local cluster have a cross-site view with the backup location.
`mixed`	Some nodes in the local cluster have a cross-site view with the backup location, other nodes in the local cluster do not have a cross-site view. The response indicates status for each node.

3.2.12. Taking Backup Locations Offline

Take backup locations offline with the `?action=take-offline` parameter.

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=take-offline
```

3.2.13. Bringing Backup Locations Online

Bring backup locations online with the `?action=bring-online` parameter.

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=bring-online
```

3.2.14. Retrieving the State Transfer Mode

Check the state transfer mode with `GET` requests.

```
GET /rest/v2/caches/{cacheName}/x-site/backups/{site}/state-transfer-mode
```

3.2.15. Setting the State Transfer Mode

Configure the state transfer mode with the `?action=set` parameter.

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{site}/state-transfer-mode?action=set&mode={mode}
```

3.2.16. Starting State Transfer

Push state of all caches to remote sites with the `?action=start-push-state` parameter.

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=start-push-state
```

3.2.17. Canceling State Transfer

Cancel ongoing state transfer operations with the `?action=cancel-push-state` parameter.

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=cancel-push-state
```

3.3. Performing Cross-Site Operations with JMX

Infinispan provides JMX tooling to perform cross-site operations such as pushing state transfer and bringing sites online.

3.3.1. Registering JMX MBeans

Infinispan can register JMX MBeans that you can use to collect statistics and perform administrative operations. You must also enable statistics otherwise Infinispan provides `0` values for all statistic attributes in JMX MBeans.

Procedure

1. Open `infinispan.xml` for editing.
2. Add the `<jmx enabled="true" />` element to the cache container.

```
<cache-container>
  <jmx enabled="true" />
</cache-container>
```

3.3.2. Performing Cross-Site Operations

Perform cross-site operations via JMX clients.

Prerequisites

- Configure Infinispan to register JMX MBeans

Procedure

1. Connect to Infinispan with any JMX client.
2. Invoke operations from the following MBeans:
   - `XSiteAdmin` provides cross-site operations for caches.
   - `GlobalXSiteAdminOperations` provides cross-site operations for Cache Managers.

     For example, to bring sites back online, invoke `bringSiteOnline(siteName)`.

See the *Infinispan JMX Components* documentation for details about available cross-site operations.

Reference

- XSiteAdmin MBean
- GlobalXSiteAdminOperations MBean

Chapter 4. Monitoring and Troubleshooting Global Infinispan Clusters

Infinispan provides cache-level statistics for cross-site replication operations via JMX or the `/metrics` endpoint for Infinispan Server.

Infinispan also includes an `org.infinispan.XSITE` logging category so you can monitor and troubleshoot common issues with networking and state transfer operations.

4.1. JMX MBeans for Cross-Site Replication

Infinispan provides JMX MBeans for cross-site replication that let you gather statistics and perform remote operations.

The `org.infinispan:type=Cache` component provides the following JMX MBeans:

- `XSiteAdmin` exposes cross-site operations that apply to specific cache instances.
- `StateTransferManager` provides statistics for state transfer operations.
- `RpcManager` provides statistics about network requests for cross-site replication.
- `AsyncXSiteStatistics` provides statistics for asynchronous cross-site replication, including queue size and number of conflicts.

The `org.infinispan:type=CacheManager` component includes the following JMX MBean:

- `GlobalXSiteAdminOperations` exposes cross-site operations that apply to all caches in a cache container.

For details about JMX MBeans along with descriptions of available operations and statistics, see the *Infinispan JMX Components* documentation.

Reference

Infinispan JMX Components

4.2. Collecting Logs and Troubleshooting Cross-Site Replication

Diagnose and resolve issues related to Infinispan cross-site replication. Use the Infinispan Command Line Interface (CLI) to adjust log levels at run-time and perform cross-site troubleshooting.

Procedure

1. Open a terminal in `$ISPN_HOME`.
2. Create a Infinispan CLI connection.
3. Adjust run-time logging levels to capture DEBUG messages if necessary.

For example, the following command enables DEBUG log messages for the org.infinispan.XSITE category:

```
[//containers/default]> logging set --level=DEBUG org.infinispan.XSITE
```

You can then check the Infinispan log files for cross-site messages in the `${infinispan.server.root}/log` directory.

4. Use the `site` command to view status for backup locations and perform troubleshooting.

For example, check the status of the "customers" cache that uses "LON" as a backup location:

```
[//containers/default]> site status --cache=customers
{
  "LON" : "online"
}
```

Another scenario for using the Infinispan CLI to troubleshoot is when the network connection between backup locations is broken during a state transfer operation.

If this occurs, Infinispan clusters that receive state transfer continually wait for the operation to complete. In this case you should cancel the state transfer to the receiving site to return it to a normal operational state.

For example, cancel state transfer for "NYC" as follows:

```
[//containers/default]> site cancel-receive-state --cache=mycache --site=NYC`
```

Reference

- Infinispan Server Troubleshooting
- Working with Infinispan Server Logs

4.2.1. Cross-Site Log Messages

Find user actions for log messages related to cross-site replication.

Log level	Identifier	Message	Description
DEBUG	ISPN000400	Node null was suspected	Infinispan prints this message when it cannot reach backup locations. Ensure that sites are online and check network status.
INFO	ISPN000439	Received new x-site view: ${site.name}	Infinispan prints this message when sites join and leave the global cluster.

Log level	Identifier	Message	Description
INFO	ISPN100005	Site ${site.name} is online.	Infinispan prints this message when a site comes online.
INFO	ISPN100006	Site ${site.name} is offline.	Infinispan prints this message when a site goes offline. If you did not take the site offline manually, this message could indicate a failure has occurred. Check network status and try to bring the site back online.
WARN	ISPN000202	Problems backing up data for cache ${cache.name} to site ${site.name}:	Infinispan prints this message when issues occur with state transfer operations along with the exception. If necessary adjust Infinispan logging to get more fine-grained logging messages.
WARN	ISPN000289	Unable to send X-Site state chunk to ${site.name}.	Indicates that Infinispan cannot transfer a batch of cache entries during a state transfer operation. Ensure that sites are online and check network status.
WARN	ISPN000291	Unable to apply X-Site state chunk.	Indicates that Infinispan cannot apply a batch of cache entries during a state transfer operation. Ensure that sites are online and check network status.
WARN	ISPN000322	Unable to re-start x-site state transfer to site ${site.name}	Indicates that Infinispan cannot resume a state transfer operation to a backup location. Ensure that sites are online and check network status.
ERROR	ISPN000477	Unable to perform operation ${operation.name} for site ${site.name}	Indicates that Infinispan cannot successfully complete an operation on a backup location. If necessary adjust Infinispan logging to get more fine-grained logging messages.

Log level	Identifier	Message	Description
FATAL	ISPN000449	XSite state transfer timeout must be higher or equals than 1 (one).	Results when the value of the `timeout` attribute is `0` or a negative number. Specify a value of at least `1` for the `timeout` attribute in the state transfer configuration for your cache definition.
FATAL	ISPN000450	XSite state transfer waiting time between retries must be higher or equals than 1 (one).	Results when the value of the `wait-time` attribute is `0` or a negative number. Specify a value of at least `1` for the `wait-time` attribute in the state transfer configuration for your cache definition.
FATAL	ISPN000576	Cross-site Replication not available for local cache.	Cross-site replication does not work with the local cache mode. Either remove the backup configuration from the local cache definition or use a distributed or replicated cache mode.