

**Pentaho Data Integration (DI)  
Server in High Availability (HA)  
End to End**

# HITACHI

## Inspire the Next

Change log (if you want to use it):

Date	Version	Author	Changes

# Contents

- Overview..... 1
  - Before You Begin ..... 1
    - Use Cases ..... 1
- Pentaho DI Server High Availability ..... 2
  - Point Each Server to the Same Database Instance ..... 2
  - Cluster the Server Nodes..... 3
    - Configure Jackrabbit Journal ..... 3
    - Configure Quartz ..... 3
    - Start and Test the Cluster ..... 3
  - Configure and Test Load Balancer for HA..... 3
    - Tomcat Configuration ..... 4
    - Apache Configuration (Windows Environment) ..... 4
    - Apache Configuration (Linux Environment)..... 5
    - About Load Balancing and DI Server..... 6
    - Test the DI Server Load Balancer..... 7
  - High Availability for the Backend Repository..... 9
- Related Information ..... 10
- Finalization Checklist..... 10

This page intentionally left blank.

## Overview

We have collected a set of best practice recommendations and information for you to use when you want to set up your Pentaho Data Integration (DI) servers with a clustered high availability (HA) solution. You will learn information about Pentaho DI servers and high availability, clustering the DI server nodes, configuring a load balancer, and testing your load balancer.

Our intended audience is Pentaho and database administrators, or anyone with a background in data source configuration who is interested in setting up data integration in a high availability environment.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	7.x, 8.0

The [Components Reference](#) in Pentaho Documentation has a complete list of supported software and hardware.

## Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

To use this guide:

- A load balancer should already be installed and running.
- All configurations will be shown in an Apache web server context (Apache HTTPD).
- Make sure that each server in your cluster is already configured to use the repository database of your choice, and that each server is [pointing to the same database instance](#).

## Use Cases

Use cases employed in this document include the following:

### *Use Case 1: Preparing a Server for High Traffic Flow*

---

*Fabiola is a database administrator who wants to set up her Pentaho DI server with a clustered HA solution to address increasing data processing and concurrent user connections. Pairing her server with this solution will place a load balancer in front of a cluster, sending traffic to only one DI server. She must first make sure that her servers are configured to use the repository database, and make sure that they are aimed at the same database instance.*

---

# Pentaho DI Server High Availability

Most installations of Pentaho are single-server installations. This solution works well in small-sized and medium-sized organizations where users and developers are limited to a handful of people. However, in large scale deployments, you need a clustered high availability solution to address the increase in data processing and concurrent user connections.



*High availability solutions for Pentaho DI servers need to be set as **active/passive** and not as **active/active**.*

In the Pentaho HA setup, the load balancer sits in front of a cluster and only one DI server gets traffic. The secondary server only gets traffic if the primary server goes down.

This model allows the servers to deal with failover: if one Pentaho server goes down, service is not interrupted but is instead taken over by a live server.

A single point of failure with this model would be with the load balancer; therefore, further consideration to cluster the load balancer would need to be taken. In this best practice guide, we will only be covering the configuration of a single load balancer, so this document should not be used as the final point of your configuration.

You can find information on the following topics in these sections:

- [Point Each Server to the Same Database Instance](#)
- [Cluster the Server Nodes](#)
- [Configure and Test Load Balancer for HA](#)
- [HA for Backend Repository](#)

## Point Each Server to the Same Database Instance

Initialize your database using the steps in the appropriate article for your system. [Cluster the Application Server](#) has sections for PostgreSQL, MySQL, MS SQL Server, and Oracle databases. After you have initialized and configured your repository, you should clean up these files by following these steps:

- Locate the `...pentaho-server/tomcat` directory and remove all files and folders from the `temp` and `work` folders.
- Locate the `...pentaho-server/pentaho-solutions/system/jackrabbit/repository` directory and remove all files and folders from the `workspaces` and `final repository` folders.

You now have a configured repository and are ready to move to the next step for clustering.

## Cluster the Server Nodes

Next, you need to configure the cluster nodes to use the same backend repository. The following instructions will walk you through this process.



Figure 1: Configure and Test Cluster Nodes

### Configure Jackrabbit Journal

Pentaho uses Apache Jackrabbit as its content repository. You will need to configure the Jackrabbit Journal for clustering by following the official instructions found in Pentaho Documentation: [Configure Jackrabbit Journal](#).

### Configure Quartz

Pentaho uses Quartz for scheduling. As with the Jackrabbit Journal, Quartz also needs to be configured for a cluster. The official instructions can be found in Pentaho Documentation: [Configure Quartz](#).



*Make sure to set all Scheduled Jobs on your passive node(s) to **PAUSE**. This will direct all Scheduled Jobs to your **active node** only and prevent round-robin scheduling.*

[Pentaho's Developer Center](#) has information on how to [Pause](#), [Stop](#), or [Start](#) the scheduler.

### Start and Test the Cluster

Follow the instructions below to start the cluster and verify that it is working properly.

1. Start the solution database.
2. Start the application server on each node.
3. Make sure that the load balancer is able to ping each node.
4. Access the login screen for each server in your cluster.
5. Browse directly to each node instead of going through the load balancer.

## Configure and Test Load Balancer for HA

Now that each server in the Pentaho cluster can be accessed individually, it is time to configure the load balancer to manage traffic to each node.

The first section will cover configuring Apache as a load balancer with the DI server cluster nodes. We provide instructions for configuring Apache for both [Windows](#) and [Linux](#).

## Tomcat Configuration

1. Shut down the Tomcat server on each node in the cluster.
2. Open the `tomcat/conf/server.xml` file in a text editor.
3. Locate the following line: `<Engine name="Catalina" defaultHost="localhost">`.
4. Add the `jvmRoute` attribute like this:

---

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="server1">
```

---



The `jvmRoute` value will need to be unique for each node. This value will map to the `BalancerMember` setup in the following section when configuring Apache.

5. Edit the `tomcat/webapps/pentaho/WEB-INF/web.xml` and change the `fully-qualified-server-url` `param-value` to the load balancer's URL.
6. Start the Tomcat service backup after configuring this file.
7. Repeat this process throughout all nodes within the cluster.

## Apache Configuration (Windows Environment)

1. Open the `C:\Program Files (x86)\Apache24\conf\httpd.conf` file in a text editor.
2. The following modules will need to be uncommented if they aren't already:
  - o `LoadModule proxy_module modules/mod_proxy.so`
  - o `LoadModule proxy_ajp_module modules/mod_proxy_ajp.so`
  - o `LoadModule proxy_http_module modules/mod_proxy_http.so`
  - o `LoadModule proxy_balancer_module modules/mod_proxy_balancer.so`
3. At the bottom of the file, add the following code, making sure to reflect the `BalancerMember` `hostname` with your Pentaho cluster nodes:

---

```
ProxyPass /pentaho balancer://dicluster
```

```
<Proxy balancer://dicluster>
    BalancerMember http://hostname1:8080/pentaho retry=30
    BalancerMember http://hostname2:8080/pentaho status=+H retry=0
</Proxy>
```

---



In the above code snippet, notice that **sticky sessions** are not used or needed. That is because this configuration is used for HA failover only, [not for true load balancing](#).

4. After adding the above code, save the file and restart the Apache service.
5. You should now be able to browse to your load balancer and see the Pentaho User Console, for example:

---

```
http://loadbalancerhostname/pentahodi
```

---



## Apache Configuration (Linux Environment)

1. The following modules will need to be loaded:

- o proxy\_ajp
- o proxy\_balancer
- o proxy\_http
- o lbmethod\_byrequests

You will need to use the proper OS specific commands to load these modules. For example, with Debian-based distributions, run the command to load the required modules:

---

```
sudo a2enmod proxy proxy_ajp proxy_balancer proxy_http lbmethod_byrequests
```

---



*For RPM-based distributions, you will need to enable these modules in the `httpd.conf` file by making sure they are uncommented, similar to the [Windows configuration](#) above. These may already be uncommented by default.*

2. Locate and edit the Apache configuration file.
  - o For Debian-based distributions, this is located by default in `/etc/apache2/sites-available/000-default.conf`.
  - o For RPM-based distributions, this is located by default in `/etc/httpd/conf/httpd.conf`.
3. Within this configuration file, you will need to edit the existing (or add a new) `VirtualHost` directive. It will need to be set up like this. Make sure to reflect the `BalancerMember` hostname with your Pentaho cluster nodes:

---

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Location "/pentaho">
        ProxyPass balancer://dicluster
        ProxyPassReverseCookiePath / /pentaho
    </Location>
</VirtualHost>

<Proxy balancer://dicluster>
    BalancerMember http://linuxvm1:8080/pentaho retry=30
    BalancerMember http://linuxvm2:8080/pentaho status=+H retry=0
</Proxy>
```

---



In the above code snippet, notice that **sticky sessions** are not used or needed. That is because this configuration is used for HA failover only, [not for true load balancing](#).

4. After adding the above code, save the file and restart the Apache service.
5. You should now be able to browse to your load balancer and see the Pentaho User Console, for example:

---

```
http://loadbalancerhostname/pentahodi
```

---

## About Load Balancing and DI Server

The DI Servers use an **Active/Passive** approach for load balancing, which means that all connections will connect to the same node unless it goes down, in which case failover to the secondary node will happen.



Make sure to set all Scheduled Jobs on your passive node(s) to **PAUSE**. This will direct all Scheduled Jobs to your **active node** only and prevent round-robin scheduling. [Pentaho's Developer Center](#) has information on how to [Pause](#), [Stop](#), or [Start](#) the scheduler.

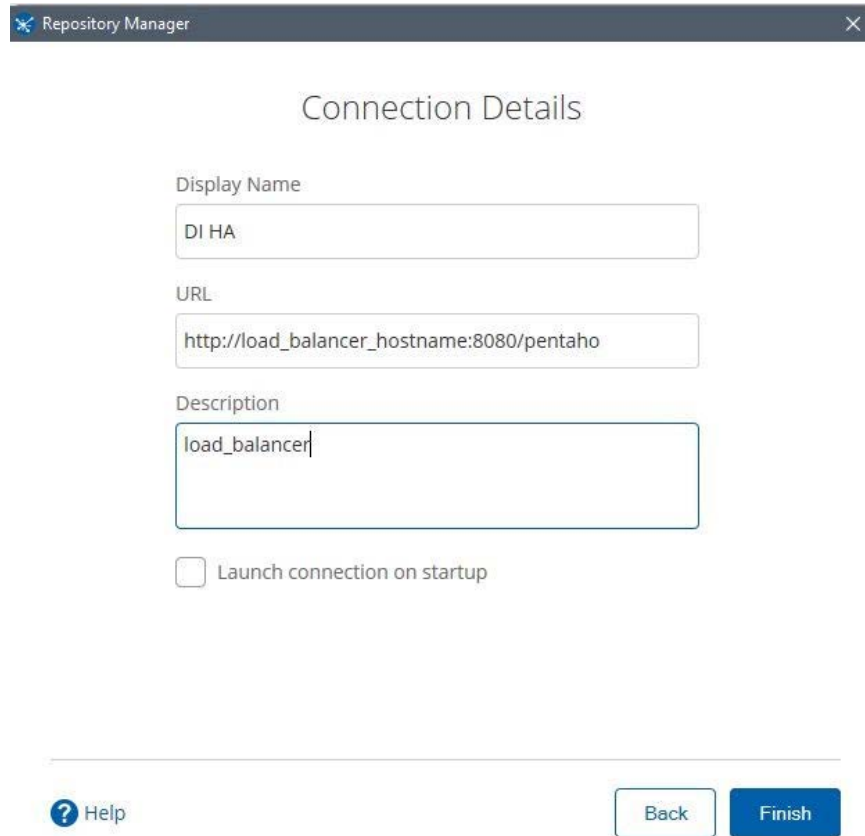
In the [code snippet](#) in [Apache Configuration \(Linux Environment\)](#) above, you will see that `linuxvm1` is designated as primary node, while `linuxvm2` is designated as secondary, or Hot Standby. This is indicated with `linuxvm2` using the parameter `status=+H`. The primary node, `linuxvm1`, has the parameter of `retry=30`, which means if it goes down, Apache will try every 30 seconds to see if the node is back up. Once it is, traffic and scheduled jobs will then be routed back to the primary node again.

If you are looking for a true load balancing solution where you can configure multiple DI Servers or multiple Carte servers for parallel processing, then you will want to consider creating Slave Servers and Cluster Schemas. Pentaho documentation has information on [how to set up Carte clusters](#).

## Test the DI Server Load Balancer

Follow these steps to test your DI server load balancer:

1. When connecting to the DI server repository via the PDI Client, you will want to use the load balancer hostname followed by the web application name you configured in the Apache configuration file like this:



Repository Manager

### Connection Details

Display Name  
DI HA

URL  
http://load\_balancer\_hostname:8080/pentaho

Description  
load\_balancer

Launch connection on startup

[? Help](#) [Back](#) [Finish](#)

Figure 2: Connect to DI Server Repository

- Once connected, create a slave connection using the same load balancer hostname and web application name like this:

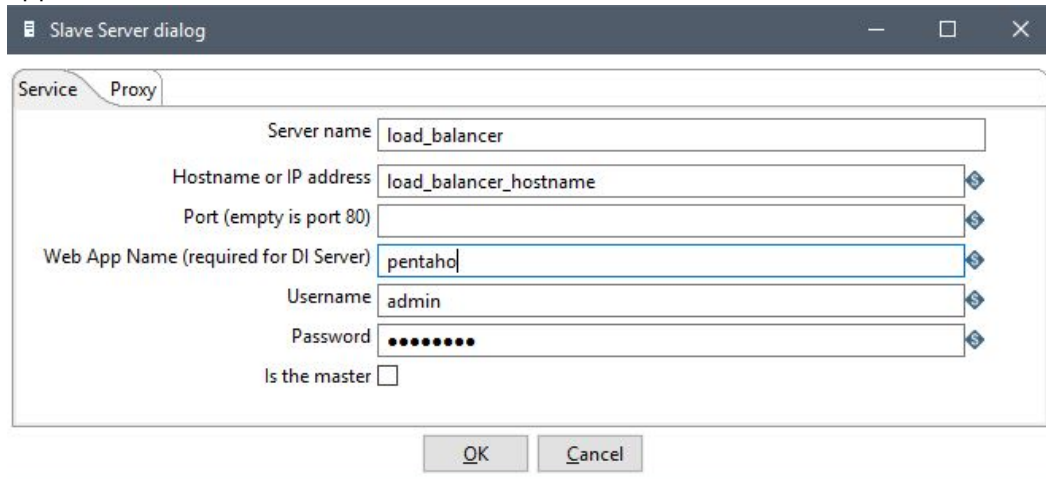


Figure 3: Create a Slave Connection Using Same Hostname

- Create a simple transformation; for example, one that just writes some text to the log and wraps it in a parent job. Configure the transformation to execute on a remote slave server like this:

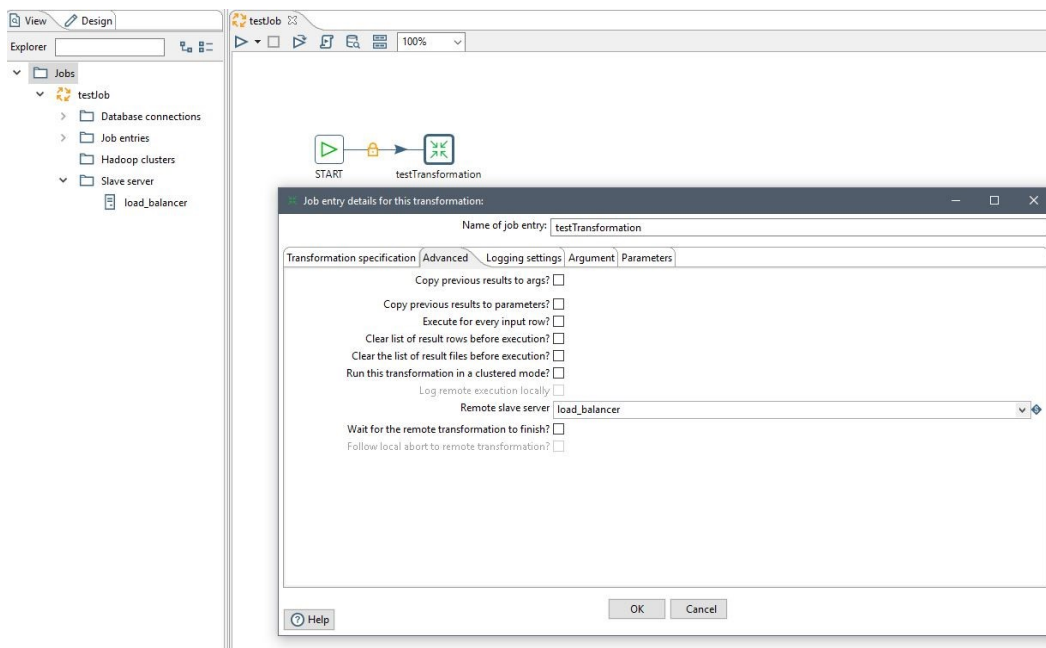


Figure 4: Create a Transformation to Execute on Slave Server

- Schedule the job to run every minute and end within 10 minutes. This should give you enough time to see the job execute on the primary node, stop the primary node, and watch the schedule resume on the secondary node. The schedule can be created by going to **Action → Schedule**.

- In a web browser, go directly to the node you have configured as `primary`. You should see both the job and transformation here, with a status of **Finished** and **Waiting**.

## Status

Transformation name	Carte Object ID	Status	Last log date	Remove from list
<a href="#">Row generator test</a>	06e0fc77-ae9f-4291-be88-1a5a1d202ad9	Waiting	-	<a href="#">Remove</a>
<a href="#">testTransformation</a>	c76cc803-1983-46eb-85f9-5cdfd7a9ef76	Waiting	2015/08/27 14:19:46.790	<a href="#">Remove</a>

Job name	Carte Object ID	Status	Last log date	Remove from list
<a href="#">testJob</a>	34c0283f-f211-4927-ac03-4493074ae7f1	Finished	2015/08/27 14:19:46.509	<a href="#">Remove</a>

## Configuration details:

Parameter	Value
The maximum size of the central log buffer	10000 lines
The maximum age of a log line	2880 minutes
The maximum age of a stale object	240 minutes
Repository name	

*These parameters can be set in the slave server configuration XML file: `/home/chris/pentaho/pdi-ee/data-integration-server/pe`*

*Figure 5: Browse to Node in a Web Browser*

- Shut down the primary node, and browse to the secondary node in the web browser. You should see the scheduled execution resume here now.
- Start the primary node backup, and you should see the schedule resume on the primary node again.

## High Availability for the Backend Repository

At this point, we have covered high availability for the DI server. Even if you have clustered the load balancers, you may still notice that a single point of failure still exists: the backend repository database.

Cluster the database used to hold the repository. Doing this requires no special configuration on the Pentaho-side, since each of the servers will see a clustered database as a single entity. You will just need to follow the documentation from the proper vendor to cluster the database according to your environment and needs. Once you have done this, the same instructions for initializing your database in the prerequisites of this guide can be followed with no additional changes.

## Related Information

Here are some links to information that you may find helpful while using this best practices document:

- [Components Reference](#)
- [Configure Jackrabbit Journal](#)
- [Configure Quartz](#)
- [Configure Your Repository Database](#)
- [Pause the Scheduler](#)
- [Pentaho Developer Center](#)
- [Shut Down the Scheduler](#)
- [Start the Scheduler](#)
- [Use Carte Clusters](#)

The following vendor links can help you get started on clustering the supported databases:

- [Oracle: Clustering](#)
- [MySQL: Cluster Reference Guides](#)
- [PostgreSQL: Creating a Database Cluster](#)

## Finalization Checklist

This checklist is designed to be added to any implemented project that uses this collection of best practices, to verify that all items have been considered and reviews have been performed.

Name of the Project: \_\_\_\_\_

Date of the Review: \_\_\_\_\_

Name of the Reviewer: \_\_\_\_\_

Item	Response	Comments
<b>Did you point each server to the same database instance?</b>	YES_____ NO_____	
<b>Did you configure and test the load balancer for HA?</b>	YES_____ NO_____	
<b>Did you cluster your DI Server nodes?</b>	YES_____ NO_____	
.....configure Jackrabbit?	YES_____ NO_____	
.....configure Quartz?	YES_____ NO_____	
.....start and test the cluster?	YES_____ NO_____	
<b>Did you pause the scheduler on passive nodes?</b>	YES_____ NO_____	