HITACHI
Inspire the Next

Pentaho 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

This page intentionally left blank.

Overview

This document covers some best practices on setting up your Pentaho servers with a clustered High Availability (HA) solution. Due to the clusters, the solutions presented are for non-ETL use.

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.

Software	Version(s)
Pentaho	7.x, 8.x

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.

Terms You Should Know

Here are some terms you should be familiar with:

- **Cluster**: A collection of [application] servers (for example, Pentaho Servers or Carte Servers) that communicate with each other and that share and update a common data repository (usually a database) to make a set of services highly available to clients.
- **Quartz Scheduler**: The scheduling engine used by Pentaho for managing and executing schedules for jobs and reports used on data integration (DI) and business analytics (BA) applications.
- **Jackrabbit**: Pentaho's repository that holds all user, server, and content information.
- **Load Balancing**: The distribution of workloads across multiple computing resources.
- **HTTP traffic***:* A text-based client that can send requests and responses.

Table 1: Reasons to Use Load Balancing

Feature	Reason
High Availability (HA)	Client requests and application loads are distributed across many servers in the same data center, or many data centers, in either an active-active or active-passive mode. Requests and application loads automatically fail over, when using routing policies, in case of failure of primary servers.
Scalability	Load balancing can spread Pentaho application footprints across many servers that are either co-located or in different data centers. This serves growing application loads and reduces the CPU, memory, and I/O strains on a single server.

Feature	Reason
Application Latency	Load balancing can solve application latency by routing client requests or traffic to servers that are geographically close to application clients, effectively solving old long network round trips.
Geolocation	Load balancing can provide a convenient way to achieve geolocation in circumstances where organizations require that client requests be restricted to data stored in certain geographic radii, based on point of origin.
Application Maintenance	Load balancing prevents the need for single server deployment of Pentaho or any other application to occasionally go offline for planned or emergency maintenance.
Disaster Recovery	Load balancing can be used to route traffic from a primary to a backup set of servers to prepare business continuity plans for medium to large customers running Pentaho, in the event of an emergency.

Other Prerequisites

In this guide, we will be using Apache HTTPD. All configurations will be shown in an Apache web server context.

1. This guide assumes that a load balancer is already installed and running.
2. You will need to make sure that each Pentaho server in your cluster is already configured to use the repository database of your choice.
3. Initialize your database using the appropriate steps for your system. Pentaho documentation has instructions for PostgreSQL, MySQL, MS SQL Server, and Oracle databases.
4. After you have initialized and configured your repository, clean up temporary files by locating the `../pentaho-server/tomcat` directory and removing all files and folders from the `temp` and `work` folders.
5. 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 Case: Managing Increased Server Traffic

Janice needs to address and manage increasing data processing and concurrent user connections within her Pentaho server. We recommend setting up a cluster of Pentaho servers with a clustered High Availability solution, which places a load balancer in front of the Pentaho cluster and directs traffic accordingly. Before doing so, Janice will need to make sure that her servers are configured to use the repository database, and make sure that they are prepared to reach the same database instance.

Pentaho Server High Availability

Most installations of Pentaho are single-server installations. This solution works well in small- and medium-sized organizations where users and developers are limited to a handful of people. However, in large scale deployments, a clustered High Availability (HA) solution is needed to address the increase in data processing and concurrent user connections.

HA solutions for Pentaho servers follow the typical load balancing models, where a load balancer (such as Apache HTTPD) sits in front of a cluster of Pentaho servers and forwards traffic using either a round robin fashion or other methods, such as worker server quotas.

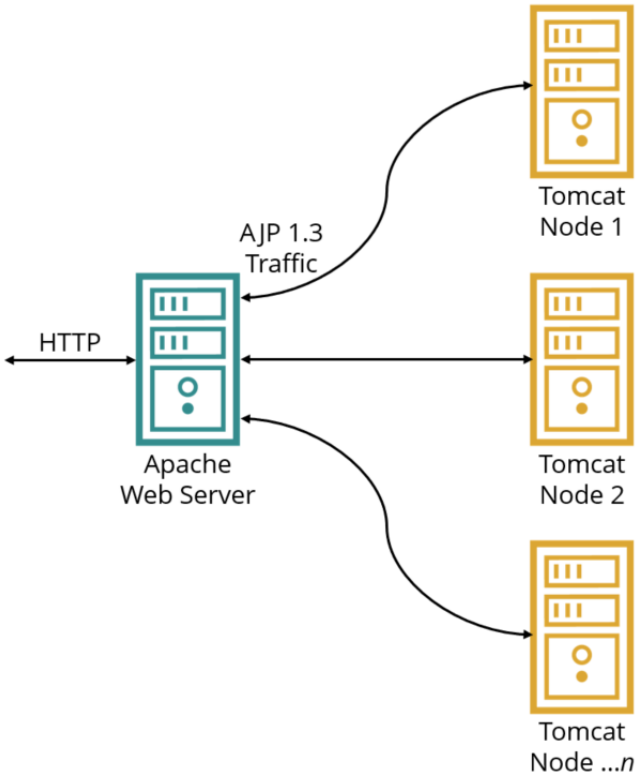

Figure 1: Pentaho Server in High Availability with Apache Web Server

This model allows not only for distributed processing, but also for failover. This way, 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, so 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 consider all the factors in your situation to determine if you should use only one load balancer. Methods for load balancing your web application tier include:

- A combination of an Apache web server and multiple Tomcat containers/application servers spread across one or more compute engines (servers).
- Enterprise or cloud load balancers such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or others.

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

- Pointing Each Server to the Same Database Instance
- Clustering the Pentaho Server Nodes
- Configuring a Load Balancer for Pentaho Server High Availability

Pointing Each Server to the Same Database Instance

Initialize your database using the steps in the appropriate article for your system. Set Up a Cluster 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 `...pentaho-server/tomcat` and remove all files and folders from the `temp` and `work` folders.
- Locate `...pentaho-server/pentaho-solutions/system/jackrabbit/repository` 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.

Clustering the Pentaho Server Nodes

The first step in developing a Pentaho server with HA solution would be to configure the cluster nodes to use the same backend repository.

Figure 2: Cluster Node Configuration Workflow

Configuring 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.

Configuring 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.*

More information is available at the Quartz Configuration Reference site.

Starting and Testing the Cluster

Follow these instructions 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 can 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.

Configuring a Load Balancer for Pentaho Server High Availability

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.

It is common for medium to large-sized enterprises to standardize on an enterprise load balancer, such as F5 or Citrix. If your Pentaho installation is in a cloud environment, you can use a cloud load balancer to route traffic directly to individual Tomcat containers running the Pentaho in dedicated or shared compute engines or virtual machines.

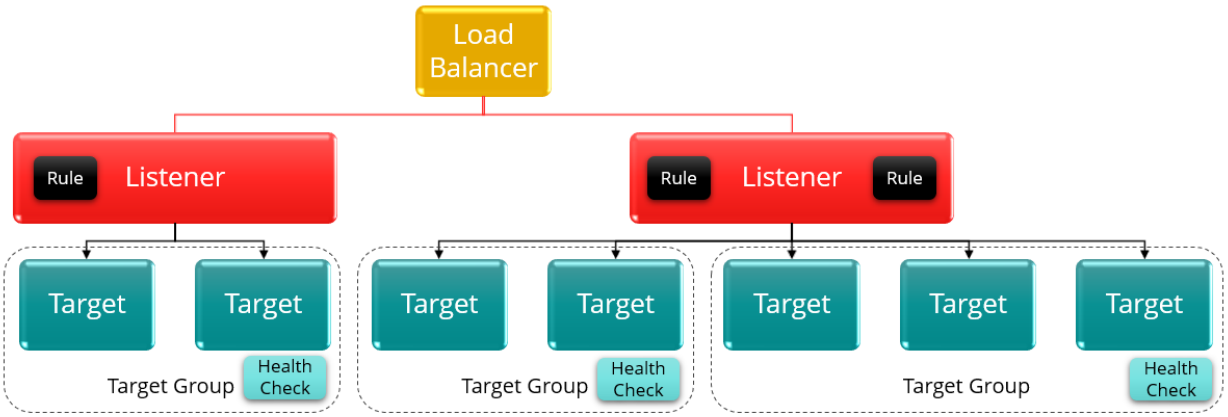

Figure 3: Using a Cloud Load Balancer

We recommend the following for load balancing setups:

Table 2: Load Balancing Recommendations

Recommendation	Details
Choose round robin	Round robin is best for indicating load balancing algorithms. It should be used in a Pentaho Server setup, but does not work for a DI-only environment.
Enable persistence using cookies	Persistence, session affinity, or "stickiness" is important, and the best way to enforce it is to use client browser-based cookies, because source IP-based affinity has some restrictions depending on your network configuration.
Select load balancer for cloud provider	Select one of the following for your Pentaho web applications, according to your cloud provider: • HTTP(S) load balancer (GCP) • Application load balancer (AWS) • Application gateway (Microsoft Azure)
Set up automatic monitoring	Set up automatic monitoring and make sure health checks are reported or logged to your enterprise monitoring platform (Stackdriver for Google, Splunk, etc.), whether the load balancer is on-premises or in the cloud. See the figure following this table for an illustration.
Switch ports where the load balancer is exposed to internet traffic	In places your load balancer is exposed to internet traffic, switch ports from the defaults of `80` (`HTTP`) or `443` (`HTTPS`) to some other port, such as `8443`, both on the load balancer backend and on Tomcat. This provides an extra security measure.
Install Secure Sockets Layer (SSL) certificates directly on the load balancer	Installing SSL certificates directly on the load balancer, instead of on individual servers running Pentaho within Tomcat containers, makes it easier to add Tomcat nodes behind the cluster without having to pay for additional SSL certificates.
Mask the identity of servers	Translate IP addresses on outbound traffic, where possible to mask the identity of servers behind the load balancer.
Set up computing resources in different availability zones	Take advantage of cross-zone load balancing by setting up computing resources in different availability zones or regions, increasing application availability and resiliency, and simplifying your machine cycles.
Use routing policies for global load balancing	Using routing policies for global load balancing helps overcome the limitations of routing traffic across regions or zones. For example, in AWS, Route 53 policies allow global traffic to be routed across data centers around the globe.

Next, we will cover configuring Apache as a load balancer with the Pentaho Server cluster nodes. We provide instructions for configuring Apache for both Windows and Linux, and additional considerations to keep in mind while you are working through these procedures.

Configuring Tomcat

To configure Tomcat:

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. To close connections so they do not become orphaned and cause errors, locate the following line:

```
<Connector URIEncoding="UTF-8" port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

Change it to:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
           connectionTimeout="10000" keepAliveTimeout="10000" />
```

6. Edit the `pentaho-solutions/system/server.properties` and change the `fully-qualified-server-url param-value` to the load balancer's URL.
7. Start the Tomcat service back up after configuring this file.
8. Repeat this process throughout all nodes within the cluster.

`connectionTimeout` is the number of milliseconds this connector will wait, after accepting a connection, for the request URI line to be presented. The default value for AJP protocol connectors is `-1` (infinite).

`keepAliveTimeout` is the number of milliseconds this connector will wait for another AJP request before closing the connection. The default value is to use the value that has been set for the `connectionTimeout` attribute.

Summary of Apache Configuration

When you configure Apache, some of the actions you will take include:

Ensuring Sticky Sessions are Enabled

Use `stickysession=JSESSIONID` to enable sticky sessions so that your session is tied to a single node. By default, Tomcat uses `JSESSIONID` as the name of its session cookie.

Configuring ProxyPass and ProxyPassReverseCookiePath Directives

Both `ProxyPass` and `ProxyPassReverseCookiePath` are directives of the `mod_proxy` module. This allows Apache to map remote servers to the local server URL-space. In this instance, the remove servers are referencing clusters rather than a single server. You will also notice that two instances of these directives are required for proper Pentaho Server mapping.

1. The first set of directives needs to point to a cluster that uses the address `ajp://{hostname}:8009/pentaho`, which is the main Pentaho web application.
2. The second set of directives needs to point to a cluster that uses the address `ajp://{hostname}:8009/pentaho-style`, which is the web application used for images, stylesheets, and the overall look and feel of the User Console.

Configuring Proxy Balancer

This is where you configure the actual cluster nodes. In a previous example, `Proxy balancer` was configured in such a way that traffic is loaded evenly across all nodes within the cluster. This is done with the `loadfactor` and `route` parameters. `Route` was configured with the same `jvmRoute` ID that was configured in the previous section on Tomcat Configuration. This is how Apache knows which node in the `Proxy balancer` maps to which Tomcat instance.

Other parameters can be used to specify certain counting algorithms that will distribute traffic based on server quota, as well as other parameters that can control other aspects of the load balancer. Apache's documentation has the list of parameters that can be used and descriptions of counting algorithms.

Due to better optimization and speed, the AJP protocol and corresponding Tomcat port have been used in the cluster node URLs. HTTP could be used instead; however, the cluster node's URL would not then be masked when traffic is forwarded from Apache. AJP will mask the cluster node URL with the load balancer's URL so that the redirect is hidden from the user. Other differences between the two protocols are documented on Apache's site in their FAQ on connectors.

Configuring Apache (Windows Environment)

There are different ways to configure Apache in a Windows environment. One example is presented here:

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:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
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://reportingcluster
ProxyPassReverseCookiePath / /pentaho
ProxyPass /pentaho-style balancer://reportingcluster-style
ProxyPassReverseCookiePath / /pentaho

Timeout 600
ProxyTimeout 600

<Proxy balancer://reportingcluster>
      ProxySet stickysession=JSESSIONID timeout=360 scolonpathdelim=On
failonstatus=502,503
      BalancerMember ajp://192.168.1.8:8009/pentaho connectiontimeout=10
loadfactor=10 route=server1 max=20 ttl=120 retry=300
      BalancerMember ajp://192.168.1.9:8009/pentaho connectiontimeout=10
loadfactor=10 route=server2 max=20 ttl=120 retry=300
</Proxy>

<Proxy balancer://reportingcluster-style>
      ProxySet stickysession=JSESSIONID timeout=360 scolonpathdelim=On
failonstatus=502,503
      BalancerMember ajp://192.168.1.8:8009/pentaho-style
connectiontimeout=10 loadfactor=10 route=server1 max=20 ttl=120 retry=300
      BalancerMember ajp://192.168.1.9:8009/pentaho-style
connectiontimeout=10 loadfactor=10 route=server2 max=20 ttl=120 retry=300
</Proxy>
```

Each proxy balancer has two servers in the cluster. If you have more than two servers in your cluster, you will need to add each of them.

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/pentaho
```

Configuring Apache (Linux Environment)

There are different ways to configure Apache in a Linux environment. One example is presented here:

1. The following modules will need to be loaded:
   ```
   proxy_ajp
   proxy_balancer
   proxy_http
   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.
   - For Debian-based distributions, this is located by default in `/etc/apache2/sites-available/000-default.conf`.
   - For RPM-based distributions, this is located by default in `/etc/httpd/conf/httpd.conf`.
3. Within this configuration file, edit the existing (or add a new) `VirtualHost` directive.

4. 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://reportingcluster
		ProxyPassReverseCookiePath / /pentaho
	</Location>
	<Location "/pentaho-style">
		ProxyPass balancer://reportingcluster-style
		ProxyPassReverseCookiePath / /pentaho
	</Location>
</VirtualHost>

ProxyPass /pentaho balancer://reportingcluster
ProxyPassReverseCookiePath / /pentaho
ProxyPass /pentaho-style balancer:// reportingcluster-style
ProxyPassReverseCookiePath / /pentaho

Timeout 600
ProxyTimeout 600

<Proxy balancer://reportingcluster>
	ProxySet stickysession=JSESSIONID timeout=360 scolonpathdelim=On
failonstatus=502,503
	BalancerMember ajp://192.168.1.8:8009/pentaho connectiontimeout=10
loadfactor=10 route=server1 max=20 ttl=120 retry=300
	BalancerMember ajp://192.168.1.9:8009/pentaho connectiontimeout=10
loadfactor=10 route=server2 max=20 ttl=120 retry=300
</Proxy>

<Proxy balancer://reportingcluster-style>
	ProxySet stickysession=JSESSIONID timeout=360 scolonpathdelim=On
failonstatus=502,503
	BalancerMember ajp://192.168.1.8:8009/pentaho-style
connectiontimeout=10 loadfactor=10 route=server1 max=20 ttl=120 retry=300
	BalancerMember ajp://192.168.1.9:8009/pentaho-style
connectiontimeout=10 loadfactor=10 route=server2 max=20 ttl=120 retry=300
</Proxy>
```

Each proxy balancer has two servers in the cluster. If you have more than two servers in your cluster, you will need to add each of them.

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

```
http://loadbalancerhostname/pentaho
```

Known Issues

Some known issues and solutions or workarounds are:

User Prompted to Reauthenticate/PUC Image Load Error

A user is redirected by the load balancer to the PUC login screen where they will enter their credentials. After selecting **Log in**, another authentication box will pop up asking for credentials again. Upon reauthenticating, PUC does not load all images for the home screen correctly.

Solution

This could be because the `pentaho-style` web application is not configured properly for the `ProxyPassReverseCookiePath` directive. Rather than use the exact examples above, you may need to change it from this:

```
<Location "/pentaho-style">
ProxyPass balancer://reportingcluster-style ProxyPassReverseCookiePath /
/pentaho
</Location>
```

To this:

```
<Location "/pentaho-style">
ProxyPass balancer://reportingcluster-style ProxyPassReverseCookiePath /
/pentaho-style
</Location>
```

Lack of Session Failover

The Pentaho Server does not support session failover. The net result of this is that while you can cluster Pentaho Servers for load-balancing, if any node in the cluster goes down, all users on that node will end up without a valid server session, and their next request will be sent to another node in the cluster where they would have to re-log-in (as if they had timed out).

For reference, failover is handled by the `failonstatus` parameter. In the example configuration in this document, server failover will happen when a request receives HTTP error codes `502` and `503`.

Related Information

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

Pentaho documentation:

- Cluster the Application Server: Configure Jackrabbit Journal
- Cluster the Application Server: Configure Quartz
- Components Reference
- Set Up a Cluster
- Use MS SQL Server as Your Repository Database (Manual Installation)
- Use MySQL as Your Repository Database (Manual Installation)
- Use Oracle as Your Repository Database (Manual Installation)
- Use PostgreSQL as Your Repository Database (Manual Installation)

External documentation:

- AJP Connector
- Amazon: What Is an Application Load Balancer?
- Apache: Connector FAQ
- Apache: Module `mod_proxy`
- Apache: Module `mod_proxy_balancer`
- Google: Setting Up HTTP(S) Load Balancing
- Microsoft Azure: What is Azure Load Balancer?
- Quartz Configuration Reference: Configure Clustering with JDBC JobStore