Guide to Creating Load-Balancing and Failover Clusters for PAS

Issued on September 14, 2007

This Guide is intended for customers, Opsware Systems Engineers (SEs), and Customer Engineers (CEs) who have installed or are deploying Opsware PAS 2.2.

Clustering for PAS

If you run large volumes of PAS flows and if high availability is a requirement for your PAS installation, you can create a cluster that provides your PAS configuration with load-balancing and failover clustering support in any combination of the following:

- A load-balancing cluster for PAS Central servers and any standalone JRAS and NRAS installations

  You can use PAS Load Balancer to create, configure, and run such a load-balancing cluster.

- Failover support for the PAS Load Balancer

  You can use the clustering software of your choice to provide this failover support.

- Failover support of Central servers and automatic run recovery (which means if one of the cluster members fails, one of the other nodes will take over and resume the failed node's runs).

  You can configure PAS Central to create the failover/run recovery support for the Central servers.

- A load-balancing and/or failover cluster for the Central database servers, which contains the Central nodes' run histories, schedules (if you install Scheduler), and system configuration, such as enabling of user authentication providers.

  You can use any clustering software to cluster the Central database servers.

Each PAS Central node must be its own dedicated machine, and use its own repository (a hierarchical folder structure of XML files). PAS provides flows or operations with which you can synchronize repositories and iActions between all the nodes of the application cluster.

To combine multiple-authoring with a PAS Central cluster, install PAS Central in a staging environment to which the authors will publish and from which they will update. This staging instance of PAS Central should be the only source for publishing flows, iActions, and other PAS objects to the production environment. You can do this and maintain consistency between the nodes' repositories by one of the following:

- Publish from the staging Central server to each of the nodes in the production PAS Central cluster.

- Run a flow that publishes from the staging Central repository to the repositories on the nodes.

Each Central node, by default, can run 100 concurrent connections either through browsers or automatically initiated runs. If your cluster is handling more requests than that, any remaining boxes should be able to handle your peak load during a flood.

Following is a schematic of what a clustered PAS installation might look like. Note that this schematic assumes that one of the Remote Action Services (JRAS or NRAS) is installed on a standalone host.

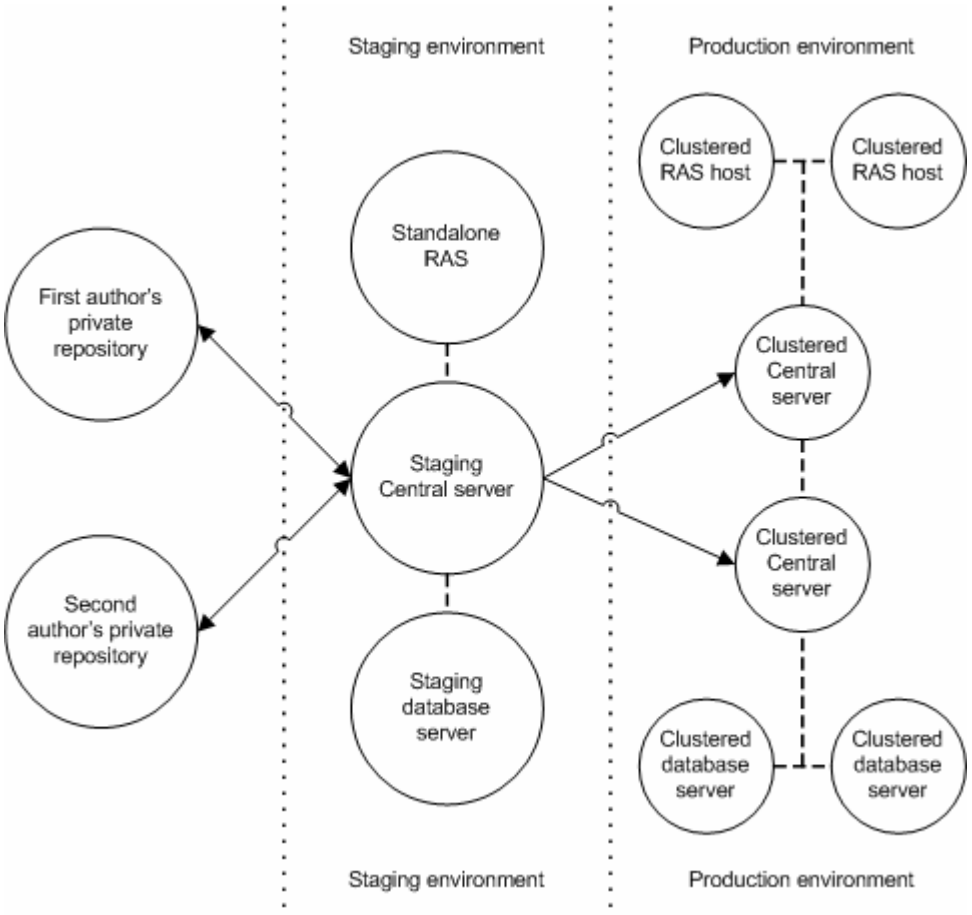


Figure 1 - PAS Configuration with Staging Environment and Clustered Production Environment

Clustering concepts

Clustering is the grouping of computers in order to accomplish either load-balancing (evenly distributing the request load for an application between several servers), failover (if a server fails, having a second server pick up the failed server's tasks while the failed server is down), or both. You can cluster PAS in the following ways:

- **Load balancing for Central, JRAS, and/or NRAS**

  PAS requests can be routed to a cluster of servers to distribute the load across multiple machines, allowing PAS to scale to as many concurrent runs as you need. PAS includes the PAS Load Balancer, with which you can create multiple

load-balancing clusters. You may also use Load Balancers that are in your existing infrastructure.

- **Active/passive failover support and automatic run recovery for Central and PAS Load Balancer**

  If a server in the cluster (a *node*) fails, to enable the other nodes to pick up the failed node's load and resume the flow runs that were abandoned by its failure.

- **Load balancing and/or failover support for the Central database server**

  If the Central cluster is configured for failover and automatic run recovery, the nodes in the Central cluster all store the following data in the same database:

  - Run data
  - Scheduling data
  - Configurations made on the Administration tab of Central

  Whether you cluster Central or not, Central can point to a clustered database environment. Load-balancing and failover support for database servers depends on the clustering technology that you use with your database servers.

The following diagram shows how you can provide load-balancing and failover support for a Central server cluster.

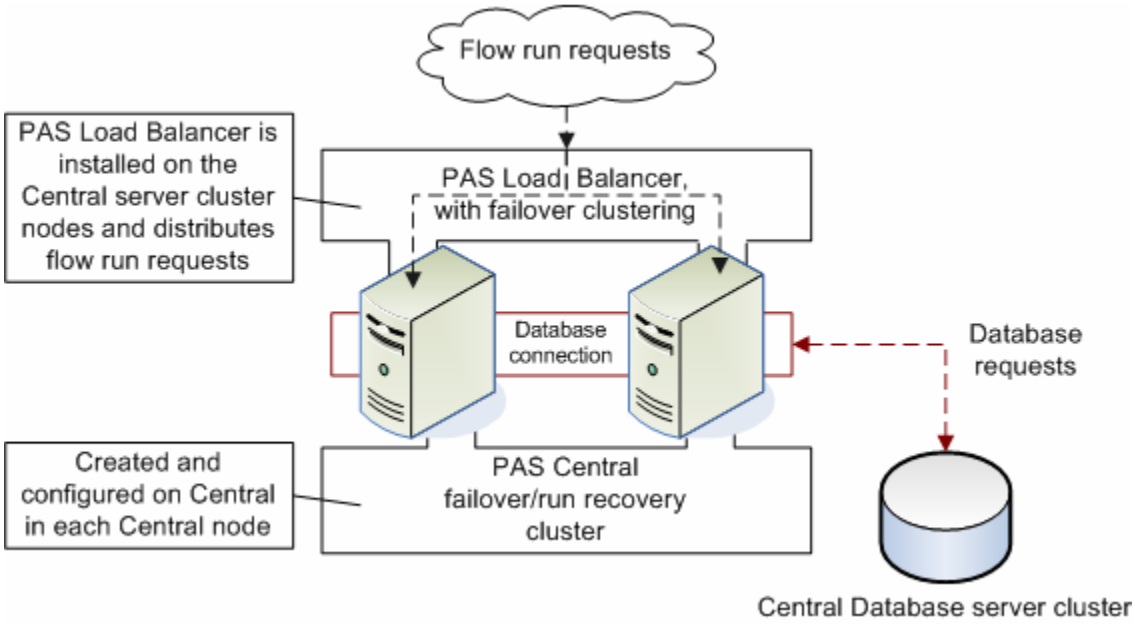


Figure 2 - Load balancing and failover for a Central server cluster

The repository in each node of a Central cluster must be identical to the repository in each of the other nodes, so one of the tasks of maintaining the cluster is to replicate the repository across all the cluster nodes whenever the repository changes.

Creating a load-balancing cluster for PAS Central

Creating a load-balancing and failover cluster for PAS Central involves the following steps:

1. Installing the PAS Load Balancer

2. Creating and configuring a Central cluster for load-balancing
3. Creating a failover cluster for the PAS Load Balancer
4. Creating and configuring a Central cluster for failover/run recovery
5. Replicating the repository across the nodes of the Central cluster

If you were to install and configure load-balancing and failover clustering in a minimal fashion on two servers, the installed and configured clusters and PAS programs might look like the following:

- Server 1
  - Windows/Linux Cluster1 in active/passive mode
  - Load Balancer 1
  - Central 1
  - JRAS 1
  - NRAS 1
- Server 2
  - Windows/Linux Cluster1 in active/passive mode
  - Load Balancer 2
  - Central 2
  - JRAS 2
  - NRAS 2

Server 1 Load Balancer would handle all HTTP requests and route them to Central 1 or Central 2, JRAS 1 or JRAS 2, NRAS 1 or NRAS 2. Server 2 Load Balancer would be inactive.

If Server 1 goes down, then Server 2 Load Balancer would become active and handle all HTTP requests, sending them to Central 2, JRAS 2, and NRAS 2, as long as Server 1 is down.

Installing the PAS Load Balancer

The PAS Load Balancer is a Java command-line utility whose installation varies according to whether you install it in a Windows or a Linux environment. We'll look at a Windows installation first.

To install the PAS Load Balancer for PAS Central on a Windows operating system

1. To start the PAS Studio Setup Wizard, navigate to and double-click PASClusterInstaller.exe.
2. On the **Welcome** page, click **Next.**
3. On the **License Agreement** page, accept the terms of the license agreement, and then click **Next.**
4. On the **Select Destination Location** page, do one of the following, and then click **Next**:
   - Accept the default location.
   - Type the location where you want the PAS Cluster to be installed.
   - Click **Browse,** specify a different location.
5. In the **Select Start Menu Folder** page, do one of the following, and then click **Next**:

- Accept the default folder.
- Type the folder where you want the Studio files installed.
- Click **Browse** and specify a different location.

The **Ready to Install** page appears, displaying the information that you specified.

6. To proceed, click **Install**.

   The Setup Wizard tracks progress on the **Installing** page.

7. When the installation completes, click **Finish**.

To install the PAS Load Balancer for PAS Central on a Linux operating system

1. On the installation CD, in the Clustering subdirectory, locate and copy the compressed tar file PASCluster-7.0.tar.gz and uncompress it (tar –xzvf PASCluster-7.0.tar.gz) to the location where you're going to perform the installation.

   Note: The following instructions reproduce the information in INSTALL.txt (contained in PASCluster-7.0.tar.gz).

2. Run configure.sh script.
3. Use the config-clustering.jar tool to create the cluster.
4. Run the PASLB.sh start command to start the service.

Creating and configuring a Central cluster for load-balancing

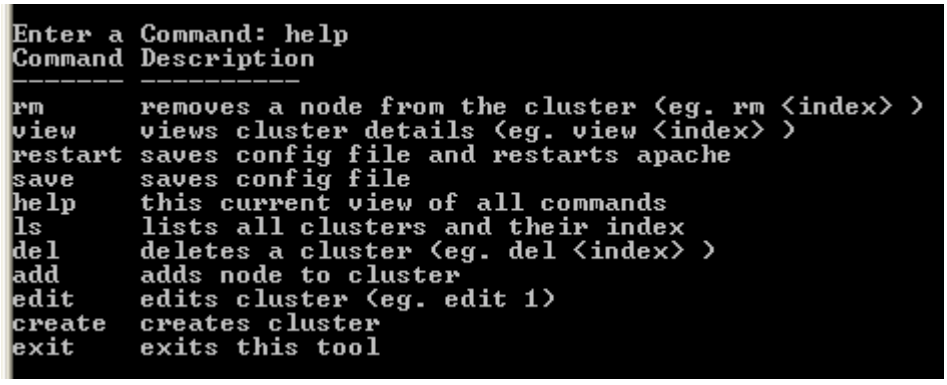
You both create and configure the Central load-balancing cluster with the PAS Load-Balancer.

To configure a cluster

1. Open a command window.
2. Navigate to the location where you installed the PAS Load Balancer.
3. Open cluster-config.jar, using the following:

   ```
   java –jar cluster-config.jar
   ```

   Note: For the command help for cluster-config.jar, type **help**.



4. To create a cluster, type **create** and then press ENTER.
5. At the prompt **Enter the cluster name**, type the cluster name, and then press ENTER.

6. At the prompt **Enter the cluster type**, type the cluster type (PAS [Central], JRAS, or NRAS, and then press ENTER.
7. At the prompt **Enter the cluster port**, type the port for the cluster (by default, this is 8443) , and then press ENTER.

   A message appears, reading “Cluster <clustername> successfully created!”

```
Enter a Command: create
Enter the cluster name: mycluster
Enter the cluster type (PAS, JRAS, NRAS): PAS
Enter the cluster port: 8443
Cluster mycluster successfully added!
```

8. To add a node to the cluster, type **add** and then press ENTER.
9. At the prompt **Enter the node host**, type the machine name or IP address of one of the nodes.
10. At the prompt **Enter the node port**, type the port that the node will use to communicate with the other nodes.

    This can be the same port number as the port that you specified for the cluster.

    Each time you successfully add a node, the current state of the cluster, including its nodes, is displayed.

```
Enter a Command: add
Enter the node host: hamlet.battleground.ad
Enter the node port: 8443
Node hamlet.battleground.ad successfully added!
Name:    mycluster
Host:    PAS
Port:    8443
Nodes:
         Port     Host
         ------   ----------------
1        8443     127.0.0.1
2        8443     hamlet.battleground.ad
```

11. To save the config file, at the prompt, type **save** and then press ENTER.

Creating a failover cluster for PAS Load Balancer

Creating an active-passive failover cluster for the PAS Load Balancer provides high availability for the Load Balancer.

To create a failover cluster for PAS Load Balancer

1. Install PAS Load Balancer on each node of the cluster.

   For instructions on installing PAS Load Balancer, see Installing the PAS Load Balancer, above.

2. Create the load-balancing cluster in the PAS Load Balancer installation on one node, as described in Creating and configuring a Central cluster for load-balancing.
3. Copy the PAS.conf file from the node on which you created the load-balancing cluster to the other nodes.
4. Create the failover cluster for PAS Load Balancer, using the clustering technology of your choice.

   See the documentation for your clustering technology for instructions on creating the cluster.

Creating a Central cluster for failover and run recovery

In the PAS Central Central.properties file, you can define one or more clusters of Central servers, each cluster providing failover capability and automatic recovery of runs that were abandoned as a result of the failure of one of the nodes. When one of the cluster nodes fails,

Enabling automatic run-recovery involves some manual configuration of the Central.properties file on each member of the Central cluster, including:

- Enabling clustering support.
- Enabling run recovery.
- Giving the failover cluster a unique name.

Important: JGroups is the mechanism that the Central cluster manages its nodes and with which the nodes recognize each other as members of the cluster. By default, Central uses IP multicasting over UDP for communication between nodes in a cluster. In the step in which you configure the cluster using the JGroups framework, you configure the cluster nodes to communicate with each other using either IP multicasting or TCP ping. If you don't know whether your Central servers can use IP multicasting, you can use TCP ping.

To configure Central cluster nodes for failover and automatic run recovery

1. Locate the Central.properties file in the PAS\Central home directory and open it in a text editor.
2. In Central.properties, locate the line `dharma.cluster.support=` and set it to true, as follows:
   `dharma.cluster.support=true`
3. To enable automatic run recovery support, locate the line `dharma.cluster.run_recovery=` and set it to true, as follows:
   `dharma.cluster.run_recovery=true`
   This line allows runs to be recovered by other Central nodes in the same cluster.

   Notes:

   - Automatic run recovery does not require that the Central server is a member of a failover cluster. That is, you can set this line to true even if dharma.cluster.support=false.
   - Only headless runs are recovered if the Central server they were running on crashes while the run was in a Running state.

   Next, you'll use JGroups configuration to define the multicast address that the cluster nodes use to communicate with each other.

   You configure JGroups with the dharma.jgroups.prop settings in Central.properties. Each Central cluster requires a unique pair of settings for the cluster's multicast address (mcast_addr) and the multicast port (mcast_port). The values used for the mcast_addr and mcast_port must not overlap with the IP address/port combination used for other applications that run in the environment.

   In Central.properties, there is a commented-out example of dharma.jgroups.prop which shows how to use TCP pinging. The TCP(start_port=45566) identifies which port is locally used and should be connected by from other Central nodes in the cluster. This configuration requires that each Central server in the cluster know

the hostname and port used by all the other Central nodes in the cluster. TCPPING(initial_hosts=…….) is a comma delimited list of host[port] values which identify all the other Central nodes in the cluster and what their start_port is set to.

If the Central cluster nodes cannot use IP multicasting, the JGroups framework can use TCP pinging for communication between nodes. The advantage of IP multicasting is that it requires less maintenance than TCP ping: When you add a node to a failover Central cluster that uses TCP ping, you must add the node's entry (see `TCPPING(initial_hosts=` in the following step) to every other node's Central.properties file.

4. To configure the JGroups framework for UDP multicasting
   a. In Central.properties, in the line that starts `dharma.jgroups.prop=`, locate `mcast_port=45566`.
   b. Either retain the default (45566) or change it to a port number that 1) is not blocked on any of the cluster nodes and 2) is not used for other purposes iin your production environment.
   c. Locate mcast_addr=228.10.10.10 and either retain the default (228.10.10.10) or change it to an address that the cluster should use in your production environment in order not to conflict with other applications.

   OR to configure the JGroups framework for TCP ping:

   a. Comment out the line that starts `dharma.jgroups.prop=`
   b. Locate the commented-out group of lines that starts `dharma.jgroups.prop=TCP` and remove the comment-out characters.

      The lines should look as follows:

```
dharma.jgroups.prop=TCP(start_port=45566):\

TCPPING(initial_hosts=host1.mycompany.com[45566],host2.mycompany.
com[45566];port_range=5;timeout=3000;num_initial_members=3;up_thr
ead=true;down_thread=true):\

MERGE2(max_interval=10000;down_thread=false;min_interval=5000;up_
thread=false):\
 FD(timeout=2000;max_tries=3;down_thread=false;up_thread=false):\
 VERIFY_SUSPECT(timeout=1500;down_thread=false;up_thread=false):\

pbcast.NAKACK(max_xmit_size=60000;down_thread=false;use_mcast_xmi
t=false;gc_lag=0;\

discard_delivered_msgs=true;up_thread=false;retransmit_timeout=10
0,200,300,600,1200,2400,4800):\

pbcast.STABLE(stability_delay=1000;desired_avg_gossip=50000;max_b
ytes=400000;down_thread=false;up_thread=false):\

VIEW_SYNC(down_thread=false;avg_send_interval=60000;up_thread=fal
se):\

pbcast.GMS(print_local_addr=true;join_timeout=3000;down_thread=fa
lse;join_retry_timeout=2000;up_thread=false;shun=true):\
```

```
FC(max_credits=2000000;down_thread=false;up_thread=false;min_threshold=0.10):\
 FRAG2(frag_size=60000;down_thread=false;up_thread=false):\
 pbcast.STATE_TRANSFER(down_thread=false;up_thread=false)
```

c. Within the lines, locate `TCPPING(initial_hosts=host1.mycompany.com[45566], host2.mycompany.com[45566]`… and substitute your cluster nodes' names or IP addresses and, within the square brackets, the port that they will use, separating the node entries with a comma and with the semicolon following the last entry.

For instance, suppose your nodes are:

- edgar.mydomain.ad (IP address of 224.0.10.159), using port 888
- richardii. mydomain.ad (IP address of 224.0.10.260), using port 777
- rosalind. mydomain.ad (IP address of 224.0.10.361), using port 555
- fool. mydomain.ad (IP address of 224.0.10.462), using port 444

then if you added the nodes using their domain names, the line would look like the following:

```
TCPPING(initial_hosts=edgar.mydomain.ad[888],richardii.mydomain.ad[777],rosalind.mydomain.ad[555],fool.mydomain.ad[444];
```

If you added the nodes using their IP addresses, the line would look like the following:

```
TCPPING(initial_hosts=224.0.10.159[888],224.0.10.260[777],224.0.10.361[555],224.0.10.462[444];
```

Important: Make sure that the port number that you specify in these entries is 1) not blocked on any of the cluster nodes and 2) not used for other purposes in your production environment.

The node addresses can be any Class D address (in the range 224.0.0.0 to 239.255.255.255) that is unique within your network environment.

5. To give the Central cluster a unique name, in Central.properties, find the following line:

   `dharma.central_cluster_name=`

   and add **=**<your_Central_cluster_name>

   If, for example, you name the cluster **STAGING**, you would change the line to read,

   `dharma.central_cluster_name=STAGING`

6. Save and close Central.properties.

To verify that the Central failover cluster setup is correct

- Review the Central\logs\wrapper.log for messages such as the following, ,which indicate the successful formation and configuration of the cluster:

```
INFO   | jvm 1    | 2007/08/09 19:26:57 |  INFO
[WrapperSimpleAppMain] (19:26:57,309)
com.iconclude.dharma.commons.cluster.JGroupsClusterService - Now
coordinator of cluster group MYCLUSTER_CENTRAL
INFO   | jvm 1   | 2007/08/09 19:26:57 |  INFO
[WrapperSimpleAppMain] (19:26:57,309)
```

```
com.iconclude.dharma.services.cluster.CentralClusterService -
Elected cluster master - starting master services
INFO   | jvm 1   | 2007/08/09 19:26:57 |  INFO
[WrapperSimpleAppMain] (19:26:57,309)
com.iconclude.dharma.commons.cluster.JGroupsClusterService - Node
10.255.147.110:45566 added to cluster group MYCLUSTER_CENTRAL
INFO   | jvm 1   | 2007/08/09 19:28:50 |  INFO [UpHandler (TCPPING)]
(19:28:50,373)
com.iconclude.dharma.commons.cluster.JGroupsClusterService - Node
10.255.147.111:45566 added to cluster group MYCLUSTER_CENTRAL
```

OR

To verify the cluster status, use the Get Cluster Servers flow . This flow queries a node in the cluster to retrieve the entire state of the cluster, the master node and all the slave nodes that have been discovered. (To locate the Get Cluster Servers flow, use **Search** in PAS Central.)

Replicating the repository across the nodes of the cluster

Each of the nodes in a cluster has its own repository, and all of their repositories must be identical to each other. Thus, after you have created the cluster, you must replicate the repository from the master node to the slave nodes.

The following procedure assumes that, per best practices, you have a Central server that is not in the production environment (and so is not a member of the production Central cluster), but rather is designated as a staging Central server. You can, however, run the Publish Staging to Production Cluster flow from one of the (target) cluster members or even from a server that is neither the staging server nor a member of the cluster.

Note: The user account that runs the Publish Staging to Production Cluster flow must have sufficient capabilities and permissions on the relevant Central servers (the source server, the target servers, and the server from which the flow is run).

To copy the Central repository across the nodes of the cluster

1. When you are ready to distribute new work to the Central cluster in your production environment, publish the repository to the staging Central server.
2. Run the flow Publish Staging to Production Cluster, providing the URL of the staging server and the URL of one of the Central servers in the cluster.

   Tip: To find the Publish Staging to Production Cluster flow, use **Search** in PAS Central.

   Here is an example of server URLs that you might supply to the flow inputs:

   - If the Staging Central server is named “staging.domain1.ad”, its URL would be https://staging.domain1.ad:8443
   - If Production Central server cluster members are prod1.domain2.ad, prod2.domain2.ad, and prod3.domain2.ad, you could use any of the following URLs for the serverURL input: https://prod1.domain2.ad:8443, https://prod2.domain2.ad:8443, or https://prod3.domain2.ad:8443

Configuring Linux Virtual Server load balancer to use PAS

To provide load balancing for PAS Central on a Linux Virtual Server

1. Install a Linux Virtual Server and configure it for load balancing.

   For more information on setting up and configuring a Linux Virtual Server for load balancing, see http://www.linuxvirtualserver.org/.

2. When configuring the load balancer, be sure to configure two ports in the following fashion:

   - **With IP affinity for browser connections**

     Set the NLB timeout to 31 minutes. This sets the limit before timing out to 31 minutes for any browser, which is a minute longer than the PAS Central session timeout. This prevents a Central user's session from timing out and ensures that the user is directed to same node throughout his or her session (unless of course, the node fails). The affinity is handled through IP. After the timeout, on subsequent requests, the user may or may not be directed to the same node as their last session.

   - **Without IP affinity for automatic connections**

     This is a useful setting for invoking flows automatically through the use of another tool. To ensure proper load-balancing for a flood of requests from a single machine, requests for automatic connections should not have IP affinity on, and should be requested on a different port than for browser connections.

   For example, if IP affinity is turned on for all requests coming from https://virtualmachine:8443/OpsForce and IP affinity is turned off for all requests coming from https://virtualmachine:443/OpsForce, both requests would be mapped, via the LVS to hit https://nodeX:8443/PAS.

3. When all your configurations are complete for the Linux Virtual Server, add each PAS Central node behind the PAS Load-Balancing cluster.
4. To test the cluster by:
   a. Make requests from different machines via a browser.

      The requests should be routed correctly to each node, and automatic requests should also be routed correctly.

   b. Take down each node in succession, then bring them back up to make sure they rejoin the cluster.

Administering a Central server cluster

Administering a PAS Load Balancer cluster involves:

- Adding nodes to and removing them from the cluster. For information on how to start the Load Balancer for configuriing, see the *PAS Installation Guide.*

Administering a Central failover/run recovery cluster involves:

- Adding nodes to and removing them from the cluster.

  For information on adding a node to a Central failover cluster (whether the cluster uses IP multicasting or TCP ping for internal communication), see the *PAS Installation Guide.*

Important: When you add a node to a cluster whose nodes use TCP ping to communicate, you must add the node in the JGroups list in each node's Central.properties file (see the PAS Installation Guide [PAS_InstallGuide.pdf]).

- Maintaining the consistency of the repository across the cluster.

  You can use the Publish Staging to Production Clusters flow to replicate a repository across a cluster. You must provide the flow with the URL for the staging server and the URL for one of the cluster nodes. With just one cluster node URl supplied, the flow discovers the rest of the nodes in the cluster and iterates through them, publishing the repository to each one.

  Although best practice is to have a staging Central server and publish the repository from there to the Central cluster in the production environment, you can run the flow from one of the nodes in the cluster.