

# Forcepoint DLP Mobile Agent Clustering

Mobile Agent Clustering | Forcepoint DLP | v8.4.x, v8.5.x, v8.6.x, v8.7.x

The Forcepoint DLP mobile agent enables organizations to secure content (such as email messages, calendar events, or tasks) synced to a mobile device.

This document describes two clustering solutions that can be used to improve the stability and performance on machines using the mobile agent.

- 2-node high-availability solution
- 4-node high-availability and load-distribution solution

When the mobile agent is configured for high-availability, it has the capacity to operate seamlessly and continuously, even in the event of a system outage (such as a hardware or software failure). This enables mobile devices to be continuously protected against data loss without system downtime and without causing any disruption to the end user.

This document describes how to set up a mobile agent for high-availability using a 2-node proxy cluster, or for load-distribution by using a 4-node proxy cluster. Neither method requires installation of additional hardware.

# High level architecture and basic flow

The mobile agent acts as a reverse proxy agent located in front of an Exchange server. All ActiveSync traffic is routed through the agent, allowing it to monitor and potentially manipulate requests and responses to the mobile device.

Typically, the mobile agent is deployed on a single node reverse proxy. In this configuration, the agent has no fault-tolerance capabilities and may, therefore, become a single point of failure in the case of hardware or software failure. A single-node agent may also lack processing power to meet an organization's ActiveSync demand.

Clustering solutions guard against these problems.

A high-availability (HA) cluster solution consists of an active and passive 2-node model based on RedHat's cluster suite, which creates an HA cluster. It is an integral part of the mobile agent operating system (CentOS), and offers a wider set of capabilities than those described in this document (e.g., asymmetric nodes topology, hardware dependent, mechanisms, and so on).

The RedHat cluster suite is installed on top of a regular mobile agent installation, and is completely orthogonal to the Forcepoint DLP system. Administrators, however, are free to use the additional capabilities offered by RedHat based to their actual needs. Refer to the [RedHat Cluster Administration document](#) for further details.

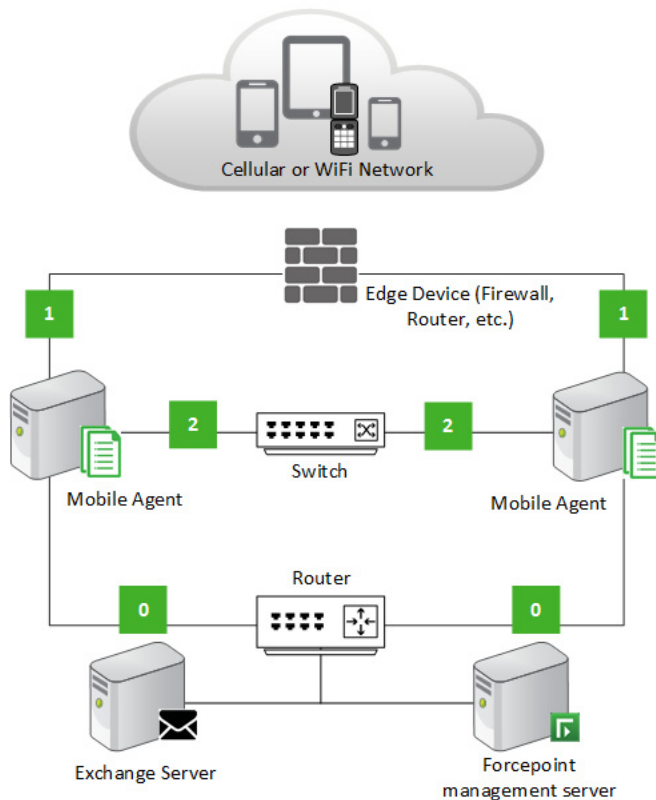
A 4-node high-availability and load-distribution (HA/LD) solution consists of two (or more) 2-node HA clusters with a round robin DNS setup to share the load traversing between these two clusters. The more high-availability clusters deployed, the larger the load distribution.

# Cluster environment

---

## High-availability 2-node cluster

The following diagram illustrates the components of a 2-node cluster deployment and shows the logical channels between them.



### Channel description

**0** – Interface between each mobile agent node and the LAN that includes the Forcepoint management server and the Microsoft Exchange server

- This abstract channel might be implemented using two NICs per node, depending on network topology.
- There are no special requirements for IP or network addresses beside those required for a non-cluster deployment (routing, etc.).

**1** – Per-node interface used for mobile devices access

- These interfaces are usually connected to some edge device (firewall, NAT, and so on).
- Each interface has its own unique IP address, but the addresses must be part of the same subnet or VLAN.

- The subnet or VLAN must have room for the “floating” IP address that the cluster is actually defining on the active node (which is also the one accessed by mobile devices).

2 – Used for internal-cluster communication to monitor the health of each node

- This channel must enable IGMP and multicast packets.
- The logical interfaces must share the same subnet or VLAN.

In a 2-node HA cluster solution, the traffic received from mobile devices is automatically routed to the active node, which handles it as it would for a single node deployment. Once the cluster, using the heartbeat services on logical channel 2, detects that the current active node is broken, it immediately disables everything it can in the current active node and defines a floating IP address in the passive node. This essentially switches roles between the two nodes, providing the HA capability. Once the other node is revived, it is ready to serve as the passive node in this topology.

The cluster continuously maintains an active node where it defines the floating IP address on logical channel 1. This floating IP address is the one resolved by mobile devices, and is a cluster entity that does not appear in the Forcepoint Security Manager. Under normal conditions, both agents are running and ready to serve requests and the only difference between these two nodes is the floating IP address definition.



#### Note

The Security Manager regards each of the two mobile agent nodes as active and independent from each other; that is, it has no notion of clustering.

---

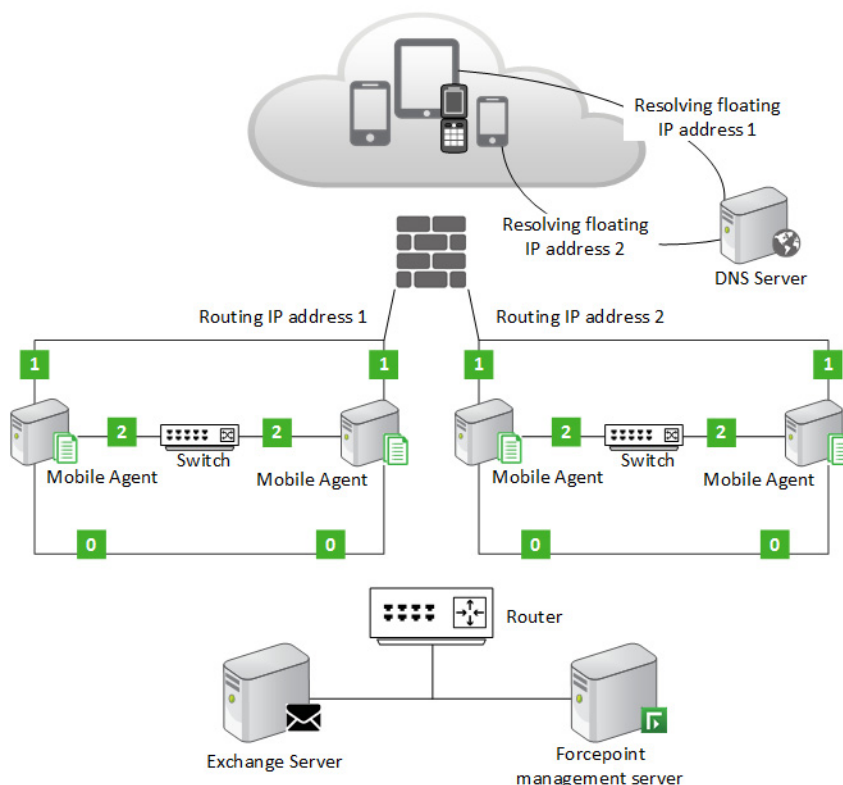
## Load-distribution 4-node cluster

The 4-node HA/LD solution is an extension of the 2-node HA cluster solution. To extend to a 4-node HA/LD cluster solution, deploy a second 2-node HA cluster (using the same deployment guidelines for both clusters) and configure the relevant DNS entry.

The process is simple:

1. Install 2 (or more) separate clusters.
2. Give each cluster a dedicated floating IP address (to be resolved by the same DNS hostname entry).
3. Make sure the DNS service is configured to resolve this DNS entry according to a cyclic mechanism.

The following diagram shows the HA/LD topology and illustrates the components of a 4-node, load-distribution deployment.



Each HA cluster here is simply a replica of the 2-node HA cluster (as described in the [High-availability 2-node cluster](#) section).

Each of the 2-node HA cluster should be configured with an appropriate dedicated floating IP address. The method required to use these two floating IP addresses in a round robin fashion by the DNS system depends on the DNS provider, and is beyond the scope of this document.

The DNS entry resolved by mobile devices for this site returns the two floating IP addresses owned by the two HA clusters in a cyclic-varied order (i.e., the first querying mobile device receives the list in a different order than the second device, and so forth). Most clients use the first IP address in the list. Since the DNS is configured in a way that produces cyclic ordering for a sequence of query requests, the effective resolving by different devices complies with a round robin schema.

Once a mobile device picks up an IP address to use, it generates an ActiveSync flow that is quite identical to the flow described above. One thing to take note of is that the mobile device will be served by one of the two clusters based on the resolved floating IP address.

The cluster can be extended as much as required by adding additional 2-node HA clusters and floating IP addresses to this setup.

**Tip**

As a best practice, use multiple mobile agent boxes in a load-balancing environment where there are multiple simultaneous users (rather than having a single agent connected to multiple policy engines).

---

**Note**

The system regards each of the mobile agent nodes (two for a 2-node cluster, or four for a 4-node cluster) as active and independent from each other (with no notion of clustering).

---

# Configure two mobile agent nodes

For mobile agent installation instructions, see the [Forcepoint DLP Installation Guide](#).

The following procedure uses a single-NIC deployment where all logical network interfaces are served by a single physical interface. If preferred, multiple NICs can be used. Changing to a multi-NIC setup is quite simple for those who understand the single-NIC deployment.

Before configuring the mobile agent:

- Install the mobile agent image on 2 physical nodes. Use identical or similar hardware for the nodes.
- Make sure the network device or devices connecting the nodes allows IGMP and multicast packets. This is essential for the cluster-internal heartbeat traffic (logical channel 2 above).

To configure mobile agent:

1. On each node, log on as user **root**, then enter **wizard** to run the configuration wizard:
  - a. Define the mobile mode, accept the license, and define the admin password.
  - b. Define a unique IP address for each node's eth0. The addresses must be in the same subnet (required by logical channels 1 and 2 above).

Note that although interface naming conventions changed in CentOS 7, this document uses the older convention (eth0) for the sake of simplicity.
  - c. Use a single NIC as the default gateway.
  - d. Define eth0 as the "Management NIC" (logical channel 0 above).
  - e. Define a unique hostname per node.
    - Define a Subject field for the default certificate. This field should be identical in the two nodes in order for the cluster to identify itself as a single abstract object to mobile devices.
    - Define a DNS server. Typically, this is the same for both nodes.
    - Define the same date for both nodes (ntpd could help keep them in sync).
    - Register both nodes to the same Forcepoint management server.
2. Log on to the Data Security module of the Forcepoint Security Manager.
3. Go to the **Settings > Deployment > System Modules** page.
4. Configure both nodes using identical parameters. Two important points to consider here:
  - Set the Mobile Devices Connection IP address **ALL IP Address**.
  - If a custom certificate is required, deploy the same certificate in both nodes.
5. Deploy the settings to complete the process.

Continue with [Configure the node agent](#), page 8.

# Configure the node agent

1. To enable incoming traffic for the Red Hat High Availability Add-On, add the following iptables rules (in addition to the default mobile agent firewall hardening rules):

```
iptables -I INPUT -m state --state NEW -p udp -m multiport --dports 5404,5405 -j ACCEPT

iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 2224 -j ACCEPT

iptables -I INPUT -p igmp -j ACCEPT

iptables -I INPUT -m addrtype --dst-type MULTICAST -j ACCEPT
```

This setup might be further secured by having additional source or destination restriction firewall rules.

2. After updating the iptables rules, enter the following commands to save and enable the changes:

```
service iptables save
service iptables reload
```

3. Add the following lines to the **/opt/websense/rproxy/conf/iptables** file:

```
-I INPUT -m state --state NEW -p udp -m multiport --dports 5404,5405 -j ACCEPT

-I INPUT -p tcp -m state --state NEW -m tcp --dport 2224 -j ACCEPT

-I INPUT -p igmp -j ACCEPT

-I INPUT -m addrtype --dst-type MULTICAST -j ACCEPT
```

4. Add the following lines to the **/etc/hosts** file on each mobile agent node:

```
...
<IPv4_address_node1> node1
<IPv4_address_node2> node2
```

- Use the IP address defined for eth0 as the IP address of each node. The names will be used by the cluster manager facilities (described later in this document).
- The names used here for clustering purposes (node1 and node2) may or may not be identical to those defined as the hostname in the installation wizard. If the same name is used in both instances, and for both purposes, its first appearance in the file (resolving the loopback address [127.0.0.1]) should be removed.

The loopback entry is generated again whenever an administrator changes the hostname using the wizard tool.

5. To delete the localhost alias, change this:

```
127.0.0.1 node1 node1.domain localhost.domain localhost
```

To this (removing the alias “node1”):

```
127.0.0.1 node1.domain localhost.domain localhost
```



6. Use the following commands to restart the httpd\_filter and httpd\_broker services:

```
systemctl httpd_filter restart
```

```
systemctl httpd_broker restart
```

Continue with [\*Install pacemaker configuration tools on the node\*](#), page 10.

# Install pacemaker configuration tools on the node

Pacemaker is a cluster resource manager intended to achieve maximum availability for cluster services (in this case, mobile agent nodes).

Pacemaker key features include:

- Detection and recovery of node and service-level failures
- Storage agnostic, no requirement for shared storage
- Resource agnostic, anything that can be scripted can be clustered
- Supports fencing (also referred to by the acronym STONITH) for ensuring data integrity
- Supports large and small clusters
- Supports both quorate and resource-driven clusters
- Supports practically any redundancy configuration
- Automatically replicated configuration that can be updated from any node
- Ability to specify cluster-wide service ordering, colocation and anti-colocation
- Support for advanced service types
- Clones for services that need to be active on multiple nodes
- Multi-state for services with multiple modes (primary/secondary)
- Unified, scriptable cluster management tools

A cluster configured with Pacemaker includes component daemons that monitor cluster membership, scripts that manage the services, and resource management subsystems that monitor the disparate resources.

The following components form the Pacemaker architecture:

- Cluster Information Base (CIB)  
This Pacemaker information daemon uses XML internally to distribute and synchronize current configuration and status information from the Designated Coordinator (DC) to all other cluster nodes.
- Cluster Resource Management Daemon (CRMD)  
Pacemaker cluster resource actions are routed through this daemon. Resources managed by CRMD can be queried by client systems, moved, instantiated, and changed when needed.
- Shoot the Other Node in the Head (STONITH)  
Often deployed in conjunction with a power switch, STONITH acts as a cluster resource that processes fence requests, forcefully powering down nodes and removing them from the cluster to ensure data integrity. STONITH is configured in CIB and can be monitored as a normal cluster resource.
- PCS

PCS is a command-line based program that can control all aspects of Pacemaker and the Corosync heartbeat daemon:

- Create and configure a Pacemaker/Corosync cluster
- Modify configuration of the cluster while it is running
- Remotely configure both Pacemaker and Corosync, as well as start, stop, and display status information of the cluster
- pcsd web UI  
A graphical user interface to create and configure Pacemaker/Corosync clusters, with the same features and abilities as the command-line based pcs utility.

To administer Red Hat Clusters with Pacemaker, install and run the following Pacemaker configuration tools on each node:

- `yum install -y pacemaker pcs resource-agents`
- `systemctl start pcsd`
- `systemctl enable pcsd`
- `systemctl enable corosync`
- `systemctl enable pacemaker`

To install the supplemental CentOS packages on each mobile agent node, use one of the following methods:

- If the mobile agent nodes have access to the Web, use the **yum** installation command. (The yum utility is a software management system for installing, updating, and removing software.)  

```
yum install -y package.name
```

This installs the packages considering all dependencies.
- If the mobile agent nodes do not have access to the Web, manually download the required packages to each node's disk, then use the **rpm** command to install them. The set of packages is described in [Complete set of required packages](#), page 20.

To install the packages manually:

```
cd /downloaded/packages/folder  
rpm -ivh package1.rpm package2.rpm <...>
```

Continue with [Configure the cluster](#), page 12.

# Configure the cluster

After the Pacemaker tools are installed, a new user, **hacluster**, is added to the system. After installation, remote login is disabled for this user.

For tasks like synchronizing configuration or starting services on other nodes, the password must be the same for this user on all nodes.

To configure communication between the nodes:

1. Use the following command to change the hacluster user's password:

```
passwd hacluster
```

2. Use the following commands to set up authentication between the two nodes:

- On node 1:

```
pcs cluster auth Node1 Node2 -u hacluster -p *****
```

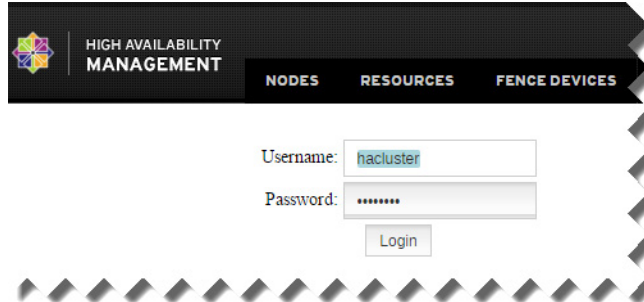
- On node 2:

```
pcs cluster auth Node2 Node1 -u hacluster -p *****
```

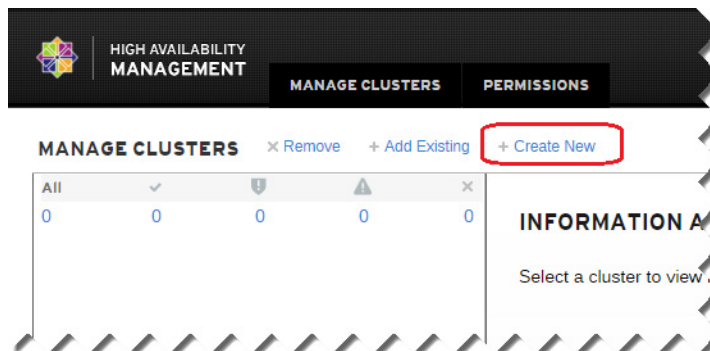
Once communication is configured, generate and synchronize the Corosync configuration on the each host. The examples below use the cluster name “mdlp,” but it can have any name.

1. Open a pcs web UI instance log in:

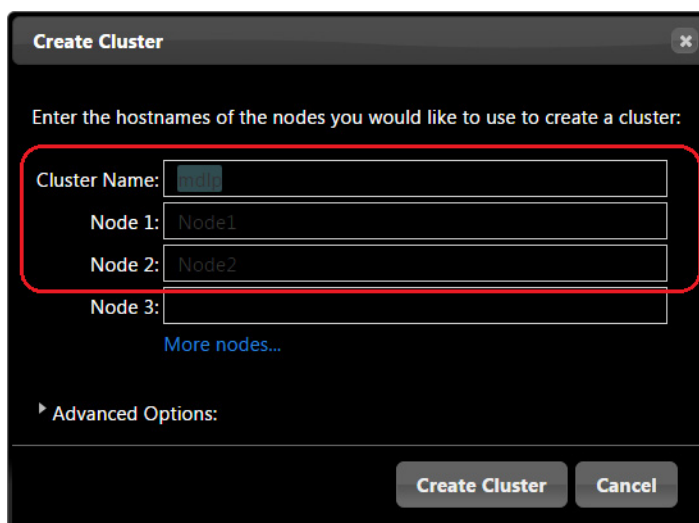
```
https://Node1:2224
```



2. Select **Create New** to add a new cluster.

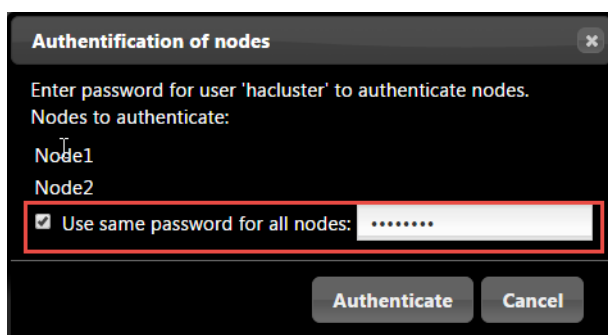


3. In the Create Cluster dialog box, enter the Cluster Name and the name of Node 1 and Node 2, then click **Create Cluster**.



The 'Create Cluster' dialog box has a title bar with a close button. Below the title bar, it says 'Enter the hostnames of the nodes you would like to use to create a cluster:'. There are four input fields: 'Cluster Name:' with the value 'hdp', 'Node 1:' with the value 'Node1', 'Node 2:' with the value 'Node2', and 'Node 3:' which is empty. A red rectangle highlights the first three fields. Below 'Node 3:' is a blue link that says 'More nodes...'. At the bottom, there is a section titled 'Advanced Options:' with a right-pointing triangle. At the very bottom are two buttons: 'Create Cluster' and 'Cancel'.

4. If prompted for node authentication data, mark the **Use same password for all nodes** check box and enter the password for the “hacluster” user, then click **Authenticate**.



The 'Authentication of nodes' dialog box has a title bar with a close button. Below the title bar, it says 'Enter password for user 'hacluster' to authenticate nodes.' and 'Nodes to authenticate:'. There are two labels: 'Node1' and 'Node2'. Below these is a checkbox labeled 'Use same password for all nodes:' which is checked. To the right of the checkbox is a password input field containing seven dots. A red rectangle highlights the checkbox and the password field. At the bottom are two buttons: 'Authenticate' and 'Cancel'.

5. Go to the Cluster Properties page and do the following:
  - a. In the No Quorum Policy drop-down list, select **Ignore**.

- b. Clear the **Stonith Enabled** check box.

Cluster: mdip

**CLUSTER PROPERTIES**

Batch Limit: 0 jobs

No Quorum Policy: Ignore

Symmetric: ☒

Stonith Enabled: ☐

Stonith Action: Reboot

Cluster Delay: 60s

6. Go to the Resources page and click **Add**, then add the following lsb resources:

Class/Provider	Type	Resource ID	Clone
lsb	httpd_broker	httpd_broker	yes
lsb	httpd_filter	httpd_filter	yes
lsb	PE	PE	yes
lsb	RProxyWD	RProxyWD	yes

**Add Resource**

Class/Provider: ocf:heartbeat

Type: IPaddr2

Description: Manages virtual IPv4 and IPv6 addresses (Linux specific version) ⓘ

Resource Group: None

Clone: ☐

Master/Slave: ☐

Disabled: ☐

Resource ID: ip

Optional Arguments: 10.0.163.55

Create Resource

7. Add the following ocf:heartbeat resource:

Class/Provider	Type	Resource ID	IP Address
ocf:heartbeat	IPaddr2	ClusterIP	<Cluster.Floating.IP>

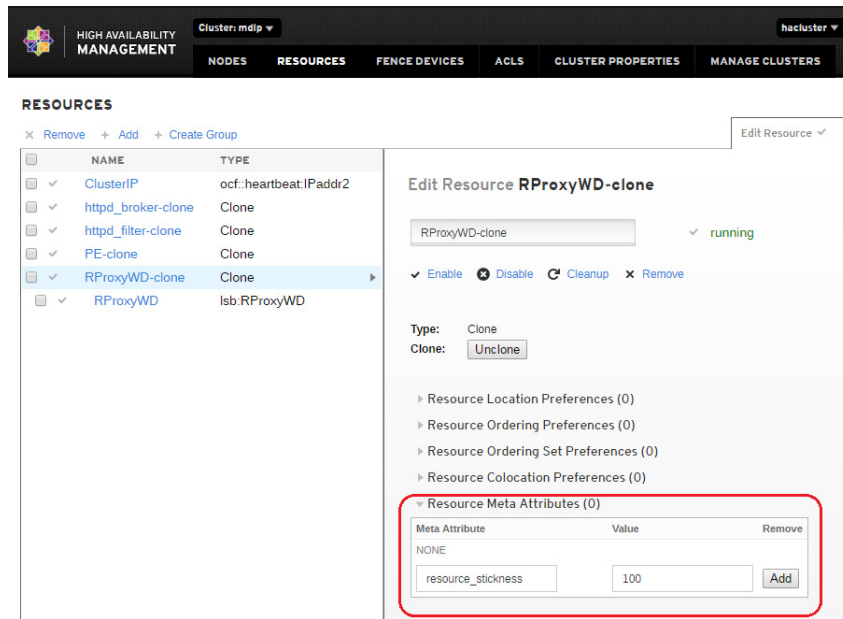
The ClusterIP resource status might be “blocked” at first. It can take some time for the status to change to “running.”

8. For each resource, open the Resource Meta Attributes and add the following information. Note that the entries are case sensitive:
  - a. Enter the following Meta Attribute:
 

```
resource_stickiness
```
  - b. Enter the following value:
 

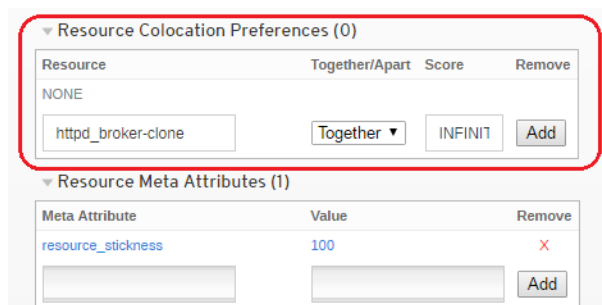
```
100
```

c. Click **Add**.



9. Select the ClusterIP resource, open the Resource Colocation Preferences, and enter the following items. Click **Add** after each entry.

Resource	Together/Apart	Score
httpd_broker-clone	Together	INFINITY
httpd-filter-clone	Together	INFINITY
PE-clone	Together	INFINITY
RProxyWD-clone	Together	INFINITY



After configuring clustering in the graphical interface, update configuration files as follows:

1. Add the following lines to the `/opt/websense/rproxy/conf/httpd/filter/httpd.conf` file:

```
Define ClusterIP <Cluster.Floating.IP>
Listen ${ClusterIP}:5443
```



2. Add the following line to the **/opt/websense/rproxy/conf/httpd/broker/broker.conf** file:

```
Listen <Cluster.Floating.IP>:443
```

**Note**

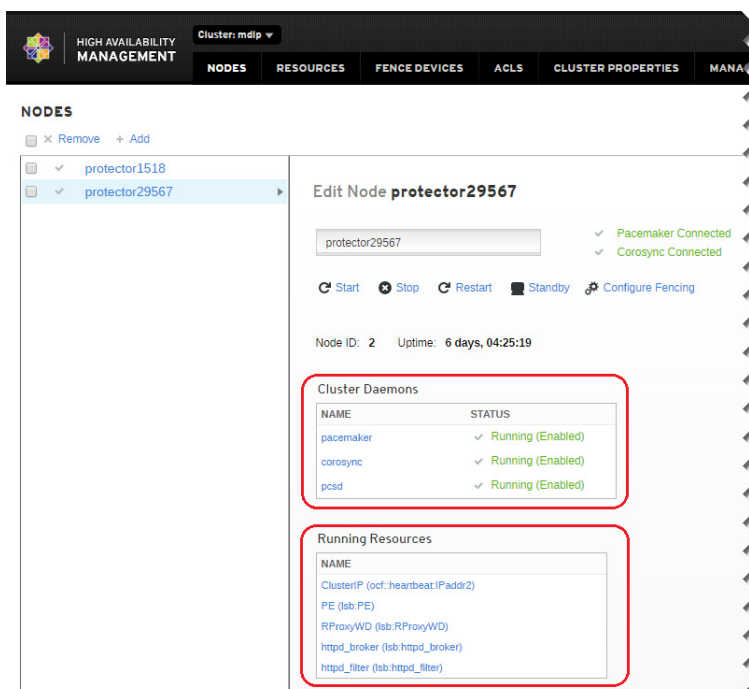
Update the “broker.conf” file any time the mobile agent is registered with a new instance of the Forcepoint Security Manager.

---

Continue with [Verify the cluster](#), page 18.

# Verify the cluster

To verify the cluster, select the Nodes tab in the graphical user interface and inspect the status of each node:



Items in green indicate that the cluster is configured and ready to run.

The active node should display a list of running resources.

Additional information about node status can be found in the “var/log/cluster” directory.

## Active/passive switch events

The cluster uses several signs to track its own state. These signs are eventually used to tell the cluster when something in the active node is broken. When this happens, then the cluster needs a passive node to take over.

The cluster monitors when:

1. The floating IP address becomes unavailable (including when the active node is rebooted).
2. Services are down (httpd\_filter, httpd\_broker, PolicyEngine and RProxyWD), using the “lsb” resources described above.
3. A machine stops delivering heartbeat traffic (i.e., the machine stops working suddenly).

After an active node goes down, when it is fully operational again, it automatically rejoins the cluster. Depending on why the node went down, the cluster manager decides whether to keep the node passive or make it active again.

# Complete set of required packages

This is the entire set of dependent packages that are silently installed as part of the deployment.

Package: **pacemaker.x86\_64 1.1.13-10.el7\_2.4**

Dependency	Provider
/bin/bash	bash.x86_64 4.2.46-20.el7_2
/bin/sh	bash.x86_64 4.2.46-20.el7_2
/usr/bin/perl	perl.x86_64 4:5.16.3-286.el7
corosync	corosync.x86_64 2.3.4-7.el7_2.3
libbz2.so.1()(64bit)	bzip2-libs.x86_64 1.0.6-13.el7
libc.so.6(GLIBC_2.4)(64bit)	glibc.x86_64 2.17-106.el7_2.8
libcfg.so.6()(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcfg.so.6(COROSYNC_CFG_0.82)(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcib.so.4()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libcmmap.so.4()(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcmmap.so.4(COROSYNC_CMAP_1.0)(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcorosync_common.so.4()(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcpng.so.4()(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcpng.so.4(COROSYNC_CPG_1.0)(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libcrmcluster.so.4()(64bit)	pacemaker-cluster-libs.x86_64 1.1.13-10.el7_2.4
libcrmcommon.so.3()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libcrmservice.so.3()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libdbus-1.so.3()(64bit)	dbus-libs.x86_64 1:1.6.12-14.el7_2
libdl.so.2()(64bit)	glibc.x86_64 2.17-106.el7_2.8
libglib-2.0.so.0()(64bit)	glib2.x86_64 2.42.2-5.el7
libgnutls.so.28()(64bit)	gnutls.x86_64 3.3.8-14.el7_2
libgnutls.so.28(GNUTLS_1_4)(64bit)	gnutls.x86_64 3.3.8-14.el7_2
liblrm.so.1()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libltdl.so.7()(64bit)	libtool-ltdl.x86_64 2.4.2-21.el7_2
libncurses.so.5()(64bit)	ncurses-libs.x86_64 5.9-13.20130511.el7
libpam.so.0()(64bit)	pam.x86_64 1.1.8-12.el7_1.1
libpam.so.0(LIBPAM_1.0)(64bit)	pam.x86_64 1.1.8-12.el7_1.1
libpe_rules.so.2()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libpe_status.so.4()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libpengine.so.4()(64bit)	pacemaker-libs.x86_64 1.1.13-10.el7_2.4
libpthread.so.0()(64bit)	glibc.x86_64 2.17-106.el7_2.8

Dependency	Provider
libqb > 0.17.0	libqb.x86_64 0.17.1-2.el7.1 libqb.i686 0.17.1-2.el7.1
libqb.so.0()(64bit)	libqb.x86_64 0.17.1-2.el7.1
libquorum.so.5()(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
libquorum.so.5(COROSYNC_QUORUM_1.0)(64bit)	corosynclib.x86_64 2.3.4-7.el7_2.3
librt.so.1()(64bit)	glibc.x86_64 2.17-106.el7_2.8
libstonithd.so.2()(64bit)	pacemaker-lb.x86_64 1.1.13-10.el7_2.4
libtinfo.so.5()(64bit)	ncurses-lb.x86_64 5.9-13.20130511.el7
libtransitioner.so.2()(64bit)	pacemaker-lb.x86_64 1.1.13-10.el7_2.4
libuuid.so.1()(64bit)	libuuid.x86_64 2.23.2-26.el7_2.3
libxml2.so.2()(64bit)	libxml2.x86_64 2.9.1-6.el7_2.3
libxslt.so.1()(64bit)	libxslt.x86_64 1.1.28-5.el7
pacemaker-cli = 1.1.13-10.el7_2.4	pacemaker-cli.x86_64 1.1.13-10.el7_2.4
pacemaker-cluster-lb = 1.1.13-10.el7_2.4	pacemaker-cluster-lb.x86_64 1.1.13-0.el7_2.4 pacemaker-cluster-lb.i686 1.1.13-10.el7_2.4
pacemaker-lb = 1.1.13-10.el7_2.4	pacemaker-lb.x86_64 1.1.13-10.el7_2.4 pacemaker-lb.i686 1.1.13-10.el7_2.4
perl(Getopt::Long)	perl-Getopt-Long.noarch 2.40-2.el7
python >= 2.4	python.x86_64 2.7.5-39.el7_2
resource-agents	resource-agents.x86_64 3.9.5-54.el7_2.17 glibc.x86_64 2.17-106.el7_2.8
rtld(GNU_HASH)	glibc.i686 2.17-106.el7_2.8
systemd	systemd.x86_64 219-19.el7_2.13

Package: pcs.x86\_64 0.9.143-15.el7.centos

Dependency	Provider
/bin/sh	bash.x86_64 4.2.46-20.el7_2
/usr/bin/env	coreutils.x86_64 8.22-15.el7_2.1
/usr/bin/python2	python.x86_64 2.7.5-39.el7_2
/usr/bin/ruby	ruby.x86_64 2.0.0.598-25.el7_1
corosync	corosync.x86_64 2.3.4-7.el7_2.3
initscripts	initscripts.x86_64 9.49.30-1.el7_2.3
libc.so.6(GLIBC_2.4)(64bit)	glibc.x86_64 2.17-106.el7_2.8
libcrypt.so.1()(64bit)	glibc.x86_64 2.17-106.el7_2.8
libdl.so.2()(64bit)	glibc.x86_64 2.17-106.el7_2.8
libgcc_s.so.1()(64bit)	libgcc.x86_64 4.8.5-4.el7
libgcc_s.so.1(GCC_3.0)(64bit)	libgcc.x86_64 4.8.5-4.el7
libm.so.6()(64bit)	glibc.x86_64 2.17-106.el7_2.8

**Dependency**

libpam.so.0()(64bit)  
 libpam.so.0(LIBPAM\_1.0)(64bit)  
 libpthread.so.0()(64bit)  
 libpthread.so.0(GLIBC\_2.2.5)(64bit)  
 librt.so.1()(64bit)  
 librt.so.1(GLIBC\_2.2.5)(64bit)  
 libruby.so.2.0()(64bit)  
 libstdc++.so.6()(64bit)  
 libstdc++.so.6(CXXABI\_1.3)(64bit)  
 libstdc++.so.6(CXXABI\_1.3.1)(64bit)  
 libstdc++.so.6(GLIBCXX\_3.4)(64bit)  
 openssl  
 pacemaker  
 pacemaker-cli  
 psmisc  
 python(abi) = 2.7  
 python-clufter  
 rtld(GNU\_HASH)  
  
 ruby >= 2.0.0  
 systemd

**Provider**

pam.x86\_64 1.1.8-12.el7\_1.1  
 pam.x86\_64 1.1.8-12.el7\_1.1  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glibc.x86\_64 2.17-106.el7\_2.8  
 ruby-libs.x86\_64 2.0.0.598-25.el7\_1  
 libstdc++.x86\_64 4.8.5-4.el7  
 libstdc++.x86\_64 4.8.5-4.el7  
 libstdc++.x86\_64 4.8.5-4.el7  
 libstdc++.x86\_64 4.8.5-4.el7  
 openssl.x86\_64 1:1.0.1e-51.el7\_2.7  
 pacemaker.x86\_64 1.1.13-10.el7\_2.4  
 pacemaker-cli.x86\_64 1.1.13-10.el7\_2.4  
 psmisc.x86\_64 22.20-9.el7  
 python.x86\_64 2.7.5-39.el7\_2  
 python-clufter.x86\_64 0.50.4-1.el7  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glibc.i686 2.17-106.el7\_2.8  
 ruby.x86\_64 2.0.0.598-25.el7\_1  
 systemd.x86\_64 219-19.el7\_2.13

Package: **resource-agents.x86\_64 3.9.5-54.el7\_2.17**

**Dependency**

/bin/bash  
 /bin/sh  
 bc  
 libc.so.6(GLIBC\_2.4)(64bit)  
 libglib-2.0.so.0()(64bit)  
 procps-ng  
  
 psmisc  
 rtld(GNU\_HASH)

**Provider**

bash.x86\_64 4.2.46-20.el7\_2  
 bash.x86\_64 4.2.46-20.el7\_2  
 bc.x86\_64 1.06.95-13.el7  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glib2.x86\_64 2.42.2-5.el7  
 procps-ng.x86\_64 3.3.10-5.el7\_2  
 procps-ng.i686 3.3.10-5.el7\_2  
 psmisc.x86\_64 22.20-9.el7  
 glibc.x86\_64 2.17-106.el7\_2.8  
 glibc.i686 2.17-106.el7\_2.8

# Troubleshooting mobile agent clustering

## Error while creating a new cluster

---

Check all the steps in [Configure the node agent, page 8](#).

Make sure the **pcsd** service is running on every node.

## Error while configuring the cluster

---

The cluster system dumps log records to the “/var/log/messages” file. Use these messages to troubleshoot the cluster.

© 2019 Forcepoint. Forcepoint and the FORCEPOINT logo are trademarks of Forcepoint. All other trademarks used in this document are the property of their respective owner.