Red Hat Enterprise Linux 7
High Availability Add-On Overview

Overview of the High Availability Add-On for Red Hat Enterprise Linux 7

Steven Levine

Red Hat Enterprise Linux 7 High Availability Add-On Overview

Overview of the High Availability Add-On for Red Hat Enterprise Linux 7

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

Legal Notice

Abstract

Red Hat High Availability Add-On Overview provides an overview of the High Availability Add-On for Red Hat Enterprise Linux 7. To expand your expertise in deploying a Red Hat High Availability cluster, you may be interested in the Red Hat High Availability Clustering (RH436) training course.

Table of Contents

Chapter 1. High Availability Add-On Overview

The High Availability Add-On is a clustered system that provides reliability, scalability, and availability to critical production services. The following sections provide a high-level description of the components and functions of the High Availability Add-On:

1.1. Cluster Basics

A cluster is two or more computers (called *nodes* or *members*) that work together to perform a task. There are four major types of clusters:

- Storage
- High availability
- Load balancing
- High performance

Storage clusters provide a consistent file system image across servers in a cluster, allowing the servers to simultaneously read and write to a single shared file system. A storage cluster simplifies storage administration by limiting the installation and patching of applications to one file system. Also, with a cluster-wide file system, a storage cluster eliminates the need for redundant copies of application data and simplifies backup and disaster recovery. The High Availability Add-On provides storage clustering in conjunction with Red Hat GFS2 (part of the Resilient Storage Add-On).

High availability clusters provide highly available services by eliminating single points of failure and by failing over services from one cluster node to another in case a node becomes inoperative. Typically, services in a high availability cluster read and write data (by means of read-write mounted file systems). Therefore, a high availability cluster must maintain data integrity as one cluster node takes over control of a service from another cluster node. Node failures in a high availability cluster are not visible from clients outside the cluster. (High availability clusters are sometimes referred to as failover clusters.) The High Availability Add-On provides high availability clustering through its High Availability Service Management component, **Pacemaker**.

Load-balancing clusters dispatch network service requests to multiple cluster nodes to balance the request load among the cluster nodes. Load balancing provides cost-effective scalability because you can match the number of nodes according to load requirements. If a node in a load-balancing cluster becomes inoperative, the load-balancing software detects the failure and redirects requests to other cluster nodes. Node failures in a load-balancing cluster are not visible from clients outside the cluster. Load balancing is available with the Load Balancer Add-On.

High-performance clusters use cluster nodes to perform concurrent calculations. A high-performance cluster allows applications to work in parallel, therefore enhancing the performance of the applications. (High performance clusters are also referred to as computational clusters or grid computing.)

Note

The cluster types summarized in the preceding text reflect basic configurations; your needs might require a combination of the clusters described.

Additionally, the Red Hat Enterprise Linux High Availability Add-On contains support for configuring and managing high availability servers *only*. It *does not* support high-performance clusters.

1.2. High Availability Add-On Introduction

The High Availability Add-On is an integrated set of software components that can be deployed in a variety of configurations to suit your needs for performance, high availability, load balancing, scalability, file sharing, and economy.

The High Availability Add-On consists of the following major components:

- Cluster infrastructure — Provides fundamental functions for nodes to work together as a cluster: configuration file management, membership management, lock management, and fencing.
- High availability Service Management — Provides failover of services from one cluster node to another in case a node becomes inoperative.
- Cluster administration tools — Configuration and management tools for setting up, configuring, and managing a the High Availability Add-On. The tools are for use with the Cluster Infrastructure components, the high availability and Service Management components, and storage.

You can supplement the High Availability Add-On with the following components:

- Red Hat GFS2 (Global File System 2) — Part of the Resilient Storage Add-On, this provides a cluster file system for use with the High Availability Add-On. GFS2 allows multiple nodes to share storage at a block level as if the storage were connected locally to each cluster node. GFS2 cluster file system requires a cluster infrastructure.
- Cluster Logical Volume Manager (CLVM) — Part of the Resilient Storage Add-On, this provides volume management of cluster storage. CLVM support also requires cluster infrastructure.
- Load Balancer Add-On — Routing software that provides high availability load balancing and failover in layer 4 (TCP) and layer 7 (HTTP, HTTPS) services. The Load Balancer Add-On runs in a cluster of redundant virtual routers that uses load algorithms to distribute client requests to real servers, collectively acting as a virtual server. It is not necessary to use the Load Balancer Add-On in conjunction with Pacemaker.

1.3. Pacemaker Overview

The High Availability Add-On cluster infrastructure provides the basic functions for a group of computers (called *nodes* or *members*) to work together as a cluster. Once a cluster is formed using the cluster infrastructure, you can use other components to suit your clustering needs (for example, setting up a cluster for sharing files on a GFS2 file system or setting up service failover). The cluster infrastructure performs the following functions:

- Cluster management
- Lock management

- Fencing
- Cluster configuration management

1.4. Pacemaker Architecture Components

A cluster configured with Pacemaker comprises separate 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)

The Pacemaker information daemon, which uses XML internally to distribute and synchronize current configuration and status information from the Designated Coordinator (DC) — a node assigned by Pacemaker to store and distribute cluster state and actions by means of the CIB — 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.

Each cluster node also includes a local resource manager daemon (LRMd) that acts as an interface between CRMd and resources. LRMd passes commands from CRMd to agents, such as starting and stopping and relaying status information.

Shoot the Other Node in the Head (STONITH)

Often deployed in conjunction with a power switch, STONITH acts as a cluster resource in Pacemaker 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.

1.5. Pacemaker Configuration and Management Tools

Pacemaker features two configuration tools for cluster deployment, monitoring, and management.

pcs

pcs can control all aspects of Pacemaker and the Corosync heartbeat daemon. A command-line based program, **pcs** can perform the following cluster management tasks:

- Create and configure a Pacemaker/Corosync cluster
- Modify configuration of the cluster while it is running
- Remotely configure both Pacemaker and Corosync remotely 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.

Chapter 2. Cluster Operation

This chapter provides a summary of the various cluster functions and features. From establishing cluster quorum to node fencing for isolation, these disparate features comprise the core functionality of the High Availability Add-On.

2.1. Quorum Overview

In order to maintain cluster integrity and availability, cluster systems use a concept known as *quorum* to prevent data corruption and loss. A cluster has quorum when more than half of the cluster nodes are online. To mitigate the chance of data corruption due to failure, Pacemaker by default stops all resources if the cluster does not have quorum.

Quorum is established using a voting system. When a cluster node does not function as it should or loses communication with the rest of the cluster, the majority working nodes can vote to isolate and, if needed, fence the node for servicing.

For example, in a 6-node cluster, quorum is established when at least 4 cluster nodes are functioning. If the majority of nodes go offline or become unavailable, the cluster no longer has quorum Pacemaker stops clustered services.

The quorum features in Pacemaker prevent what is also known as *split-brain*, a phenomenon where the cluster is separated from communication but each part continues working as separate clusters, potentially writing to the same data and possibly causing corruption or loss.

Quorum support in the High Availability Add-On are provided by a Corosync plug-in called **votequorum**, which allows administrators to configure a cluster with a number of votes assigned to each system in the cluster and ensuring that only when a majority of the votes are present, cluster operations are allowed to proceed.

In a situation where there is no majority (such as a two-node cluster where the internal communication network link becomes unavailable, resulting in a 50% cluster split), **votequorum** can be configured to have a *tiebreaker* policy, which administrators can configure to continue quorum using the remaining cluster nodes that are still in contact with the available cluster node that has the lowest node ID.

2.2. Fencing Overview

In a cluster system, there can be many nodes working on several pieces of vital production data. Nodes in a busy, multi-node cluster could begin to act erratically or become unavailable, prompting action by administrators. The problems caused by errant cluster nodes can be mitigated by establishing a *fencing* policy.

Fencing is the disconnection of a node from the cluster's shared storage. Fencing cuts off I/O from shared storage, thus ensuring data integrity. The cluster infrastructure performs fencing through the *STONITH* facility.

When Pacemaker determines that a node has failed, it communicates to other cluster-infrastructure components that the node has failed. STONITH fences the failed node when notified of the failure. Other cluster-infrastructure components determine what actions to take, which includes performing any recovery that needs to done. For example, DLM and GFS2, when notified of a node failure, suspend activity until they detect that STONITH has completed fencing the failed node. Upon confirmation that the failed node is fenced, DLM and GFS2 perform recovery. DLM releases locks of the failed node; GFS2 recovers the journal of the failed node.

Node-level fencing through STONITH can be configured with a variety of supported fence devices, including:

- Uninterruptible Power Supply (UPS) — a device containing a battery that can be used to fence devices in event of a power failure
- Power Distribution Unit (PDU) — a device with multiple power outlets used in data centers for clean power distribution as well as fencing and power isolation services
- Blade power control devices — dedicated systems installed in a data center configured to fence cluster nodes in the event of failure
- Lights-out devices — Network-connected devices that manage cluster node availability and can perform fencing, power on/off, and other services by administrators locally or remotely

Chapter 3. Red Hat High Availability Add-On Resources

This chapter provides a summary of Red Hat High Availability resources and their operation.

3.1. Red Hat High Availability Add-On Resource Overview

A *cluster resource* is an instance of program, data, or application to be managed by the cluster service. These resources are abstracted by *agents* that provide a standard interface for managing the resource in a cluster environment. This standardization is based on industry approved frameworks and classes, which makes managing the availability of various cluster resources transparent to the cluster service itself.

3.2. Red Hat High Availability Add-On Resource Classes

There are several classes of resource agents supported by Red Hat High Availability Add-On:

- LSB — The Linux Standards Base agent abstracts the compliant services supported by the LSB, namely those services in **`/etc/init.d`** and the associated return codes for successful and failed service states (started, stopped, running status).
- OCF — The Open Cluster Framework is superset of the LSB (Linux Standards Base) that sets standards for the creation and execution of server initialization scripts, input parameters for the scripts using environment variables, and more.
- systemd — The newest system services manager for Linux based systems, systemd uses sets of unit files rather than initialization scripts as does LSB and OCF. These units can be manually created by administrators or can even be created and managed by services themselves. Pacemaker manages these units in a similar way that it manages OCF or LSB init scripts.
- Upstart — Much like systemd, Upstart is an alternative system initialization manager for Linux. Upstart uses jobs, as opposed to units in systemd or init scripts.
- STONITH — A resource agent exclusively for fencing services and fence agents using STONITH.
- Nagios — Agents that abstract plug-ins for the Nagios system and infrastructure monitoring tool.

3.3. Monitoring Resources

To ensure that resources remain healthy, you can add a monitoring operation to a resource's definition. If you do not specify a monitoring operation for a resource, by default the pcs command will create a monitoring operation, with an interval that is determined by the resource agent. If the resource agent does not provide a default monitoring interval, the pcs command will create a monitoring operation with an interval of 60 seconds.

3.4. Resource Constraints

You can determine the behavior of a resource in a cluster by configuring *constraints*. You can configure the following categories of constraints:

- location constraints — A location constraint determines which nodes a resource can run on.
- order constraints — An order constraint determines the order in which the resources run.

- colocation constraints — A colocation constraint determines where resources will be placed relative to other resources.

As a shorthand for configuring a set of constraints that will locate a set of resources together and ensure that the resources start sequentially and stop in reverse order, Pacemaker supports the concept of resource groups.

3.5. Resource Groups

One of the most common elements of a cluster is a set of resources that need to be located together, start sequentially, and stop in the reverse order. To simplify this configuration, Pacemaker supports the concept of *groups*.

You create a resource group with the **pcs resource** command, specifying the resources to include in the group. If the group does not exist, this command creates the group. If the group exists, this command adds additional resources to the group. The resources will start in the order you specify them with this command, and will stop in the reverse order of their starting order.

Appendix A. Upgrading From Red Hat Enterprise Linux 6 High Availability Add-On

This appendix provides an overview of upgrading Red Hat Enterprise Linux High Availability Add-On from release 6 to release 7.

A.1. Overview of Differences Between Releases

Red Hat Enterprise Linux 7 High Availability Add-On introduces a new suite of technologies that underlying high-availability technology based on Pacemaker and Corosync that completely replaces the CMAN and RGManager technologies from previous releases of High Availability Add-On. Below are some of the differences between the two releases. For a more comprehensive look at the differences between releases, refer to the appendix titled "Cluster Creation with rgmanager and with Pacemaker" from the *Red Hat Enterprise Linux High Availability Add-On Reference*.

- Configuration Files — Previously, cluster configuration was found in the **/etc/cluster/cluster.conf** file, while cluster configuration in release 7 is in **/etc/corosync/corosync.conf** for membership and quorum configuration and **/var/lib/heartbeat/crm/cib.xml** for cluster node and resource configuration.

- Executable Files — Previously, cluster commands were in **ccs** by means of a command line, **luci** for graphical configuration. In Red Hat Enterprise Linux 7 High Availability Add-On, configuration is done by means of **pcs** at the command line and the **pcsd** Web UI configuration at the desktop.

- Starting the Service — Previously, all services including those in High Availability Add-On were performed using the **service** command to start services and the **chkconfig** command to configure services to start upon system boot. This had to be configured separately for all cluster services (**rgmanager**, **cman**, and **ricci**. For example:

  ```
  service rgmanager start
  chkconfig rgmanager on
  ```

  For Red Hat Enterprise Linux 7 High Availability Add-On, the **systemctl** controls both manual startup and automated boot-time startup, and all cluster services are grouped in the **pcsd.service**. For example:

  ```
  systemctl start pcsd.service
  systemctl enable pcsd.service
  pcs cluster start -all
  ```

- User Access — Previously, the root user or a user with proper permissions can access the **luci** configuration interface. All access requires the **ricci** password for the node.

  In Red Hat Enterprise Linux 7 High Availability Add-On, the **pcsd** Web UI requires that you authenticate as user **hacluster**, which is the common system user. The **root** user can set the password for **hacluster**.

- Creating Clusters, Nodes and Resources — Previously, creation of nodes were performed with the **ccs** by means of a command line or with **luci** graphical interface. Creation of a cluster and adding nodes is a separate process. For example, to create a cluster and add a node by means of the command line, perform the following:

```
ccs -h node1.example.com --createcluster examplecluster
ccs -h node1.example.com --addnode node2.example.com
```

In Red Hat Enterprise Linux 7 High Availability Add-On, adding of clusters, nodes, and resources are done by means of **pcs** at the command line, or the **pcsd** Web UI. For example, to create a cluster by means of the command line, perform the following:

```
pcs cluster setup examplecluster node1 node2 ...
```

- Cluster removal — Previously, administrators removed a cluster by deleting nodes manually from the **luci** interface or deleting the **cluster.conf** file from each node

  In Red Hat Enterprise Linux 7 High Availability Add-On, administrators can remove a cluster by issuing the **pcs cluster destroy** command.

Appendix B. Revision History

Revision 3.1-2	**Mon Oct 17 2016**	**Steven Levine**
Version for 7.3 GA publication.		
Revision 3.1-1	**Wed Aug 17 2016**	**Steven Levine**
Preparing document for 7.3 Beta publication.		
Revision 2.1-5	**Mon Nov 9 2015**	**Steven Levine**
Preparing document for 7.2 GA publication.		
Revision 2.1-1	**Tue Aug 18 2015**	**Steven Levine**
Preparing document for 7.2 Beta publication.		
Revision 1.1-3	**Tue Feb 17 2015**	**Steven Levine**
Version for 7.1 GA		
Revision 1.1-1	**Thu Dec 04 2014**	**Steven Levine**
Version for 7.1 Beta Release		
Revision 0.1-9	**Tue Jun 03 2014**	**John Ha**
Version for 7.0 GA Release		
Revision 0.1-4	**Wed Nov 27 2013**	**John Ha**
Build for Beta of Red Hat Enterprise Linux 7		
Revision 0.1-1	**Wed Jan 16 2013**	**Steven Levine**
First version for Red Hat Enterprise Linux 7		

Index

quorum, Quorum Overview