Project Report On

High Availability Cluster using Pacemaker

Submitted in partial fulfillment for the award of

Post Graduate Diploma in High Performance Computing System Administration from **C-DAC ACTS (Pune)**

Guided by

Mr. Roshan Gami

Presented By

Mrs. Manisha Labade	**PRN:230940127006**
Mrs. Poonam Watpade	**PRN:230940127013**
Mr. Diwan Chand	**PRN:230940127033**
Mr. Mukut Raj Verma	**PRN:230940127042**
Ms. Sampada Kokane	**PRN:230940127046**
Ms. Sneha Kisku	**PRN:230940127055**

Centre of Development of Advanced Computing (C-DAC), Pune

TABLE OF CONTENTS

1. Abstract

A high-availability (HA) stack serves a crucial purpose: it ensures service availability and automatic recovery in case of problems by employing a redundant setup of two or more nodes. In the context of Linux, the state-of-the-art HA stack is built upon the Pacemaker cluster resource manager.

Key Aspect of high-availability solution:

1.Storage Layer:

- Single-Instance Storage: In this conventional approach, all cluster data resides in a centralized instance (often a volume on a Storage Area Network). Access to this data can be either active/passive (one node at a time) or active/active (multiple nodes simultaneously, typically using a shared-cluster filesystem like ext4). However, single-instance storage has a significant drawback: if data becomes inaccessible or is destroyed, server redundancy is compromised.
- Shared Storage: Shared Storage refers to a common storage space accessible by multiple users or devices. It allows them to store data and data sharing.

2.Cluster Communications Layer:

- Corosync provides messaging and membership services for the cluster. It facilitates communication between nodes, ensuring coordination and synchronization.

3.Resource Management Layer:

- Pacemaker acts as the cluster resource manager. It monitors resources (such as services, applications, and virtual IPs) and orchestrates their failover and recovery. Pacemaker ensures that services remain available even when individual nodes encounter issues.

4.Applications Layer:

- At the top layer, we have the actual applications and services running on the cluster. These applications benefit from the underlying layers' redundancy and automatic failover capabilities.

2. Introduction

This project aims to implement High availability cluster using the pacemaker. Pacemaker is an advanced, scalable high-availability cluster resource manager. It plays a crucial role in ensuring service availability and automatic recovery in case of problems. In today's rapidly evolving technological landscape, ensuring the continuous availability of critical services and applications is paramount for businesses and organizations. Downtime can lead to significant financial losses, reputation damage, and customer dissatisfaction. To address this challenge, High Availability (HA) clustering solutions have emerged as a cornerstone of modern IT infrastructure.

At the heart of HA clustering lies Pacemaker, an open-source cluster resource manager that orchestrates the operation of services and applications across multiple nodes, ensuring seamless failover and high uptime. This introduction aims to provide a foundational understanding of HA clustering with Pacemaker, highlighting its key concepts, benefits, and implementation considerations.

A High Availability Cluster is a group of interconnected servers, or nodes, orchestrated to work in tandem to deliver uninterrupted services. These clusters distribute workload and resources across nodes, ensuring redundancy and fault tolerance. In the event of a failure on one node, services seamlessly transition to another healthy node, averting downtime and maintaining service continuity.

A high availability (HA) cluster using Pacemaker is a configuration designed to ensure continuous availability of services or applications by automatically transferring their operation from one node to another in case of failure. Here's an introduction to setting up such a cluster using Pacemaker:

1. Node Setup: You start by setting up multiple nodes (physical or virtual servers) that will be part of the cluster. Each node should have the necessary software installed, including Pacemaker, Corosync (for communication between cluster nodes), and any applications you want to make highly available.

2. Resource Configuration: Identify the resources (services or applications) that need to be highly available. This could include web servers or any custom applications. Pacemaker supports a wide range of resource types.

3. Resource Agents: For each resource, define resource agents. Resource agents are scripts or binaries that Pacemaker uses to control the lifecycle of resources. These scripts manage starting, stopping, monitoring, and migrating resources between nodes.

4. Cluster Configuration: Configure Pacemaker to manage the cluster. This includes defining the cluster nodes, communication settings, fencing devices (to isolate failed nodes), and constraints (rules that govern resource placement and failover behavior).

5. Resource Constraints: Set up constraints to define resource dependencies and preferred locations. For example, you can specify that a web server resource should only run on nodes with a certain amount of CPU or memory available.

6.Testing and Monitoring: Test the cluster setup thoroughly to ensure that failover occurs as expected. Monitor the cluster regularly to detect and address any issues proactively.

7. Documentation and Training: Document the cluster configuration and failover procedures. Train the relevant personnel on how to manage and troubleshoot the cluster effectively.

Pacemaker provides a robust framework for building high availability clusters, but setting up and maintaining such clusters requires careful planning and expertise. It's essential to understand your application's requirements and design the cluster accordingly to achieve the desired level of availability.

3. High Availability Cluster

Highly Available (HA):

High Availability refers to a system or component that is continuously operational for a desirably long length of time.

Cores principles to HA:

- Elimination of single point of failures.
- Reliable crossover.
- Detection of failures as they occur.

Clustering:

A cluster is a set of computers working together on a single task. Which task is performed, and how that task is performed, differs from cluster to cluster.

Different kinds of clusters:

- High-availability clusters: Known as an HA cluster or failover cluster, their function is to keep running services as available as they can be.
    - Main Configuration:
        - Active-Active (where a service runs on multiple nodes).
        - Active-Passive (where a service only runs on one node at a time).
- Load-balancing clusters: All nodes run the same software and perform the same task at the same time and the requests from the clients are distributed between all the nodes.
- Compute clusters: Also known as high-performance computing (HPC) cluster. In these clusters, tasks are divided into smaller chunks, which then get computed on different nodes.
- Storage clusters: All nodes provide a single cluster file system that will be used by clients to read and write data simultaneously.

4. Features of High Availability Clusters

- Detection and recovery of machine and application-level failures
- Supports practically any redundancy configuration
- Supports both quorate and resource-driven clusters
- Configurable strategies for dealing with quorum loss (when multiple machines fail)
- Supports application startup/shutdown ordering, regardless of which machine(s) the applications are on
- Supports applications that must/must-not run on the same machine
- Supports applications which need to be active on multiple machines
- Supports applications with multiple modes (eg. master/slave)

Properties	iSCSI	node1	node2	node3	Virtual IP
OS	CentOS 7	CentOS 7	CentOS 7	CentOS 7	CentOS 7
vCPU	1	1	1	1	1
Memory	1GB	1GB	1GB	1GB	1GB
Disk	20GB+20GB	30GB	30GB	30GB	30GB
FQDN	http://ec2-65-0-131-40.ap-south-1.compute.amazonaws.com	http://ec2-13-201-190-82.ap-south-1.compute.amazonaws.com	http://ec2-13-232-74-191.ap-south-1.compute.amazonaws.com	http://ec2-3-110-174-149.ap-south-1.compute.amazonaws.com	http://ec2-13-233-192-88.ap-south-1.compute.amazonaws.com
Hostname	iscsi.aws	Node1.aws	node2.aws	node3.aws	virtual.aws
IP Address (Internal)	172.31.7.159	172.31.2.149	172.31.2.26	172.31.2.51	172.31.10.4
IP Address (External)	DHCP	DHCP	DHCP	DHCP	DHCP

5. Benefits of High Availability Clusters

Implementing an HA cluster with Pacemaker offers several compelling benefits:

- Enhanced Reliability: By distributing services across multiple nodes, HA clustering improves system reliability and mitigates the impact of hardware failures, software errors, and other disruptions.

- Continuous Availability: Pacemaker's automated failover capabilities enable services to remain accessible even in the event of node failures or maintenance activities, maximizing uptime and productivity.

- Scalability: HA clusters can scale horizontally by adding additional nodes to accommodate growing workloads, ensuring optimal performance and resource utilization.

- Cost-Effectiveness: While HA clustering requires initial investment in hardware and configuration, the long-term cost savings from reduced downtime and improved productivity justify the expenditure.

6. System Requirement

- User Interface:
  - EC2 Instance Centos 7 (t2.micro)

- Software Requirement:
  - AWS
  - Pacemaker

- Hardware Requirement:
  - Centos 7 (t2.micro), 30GB HD, 1 GB RAM

- Used Languages:
  - Shell Scripting
  - HTML

7. System Architecture

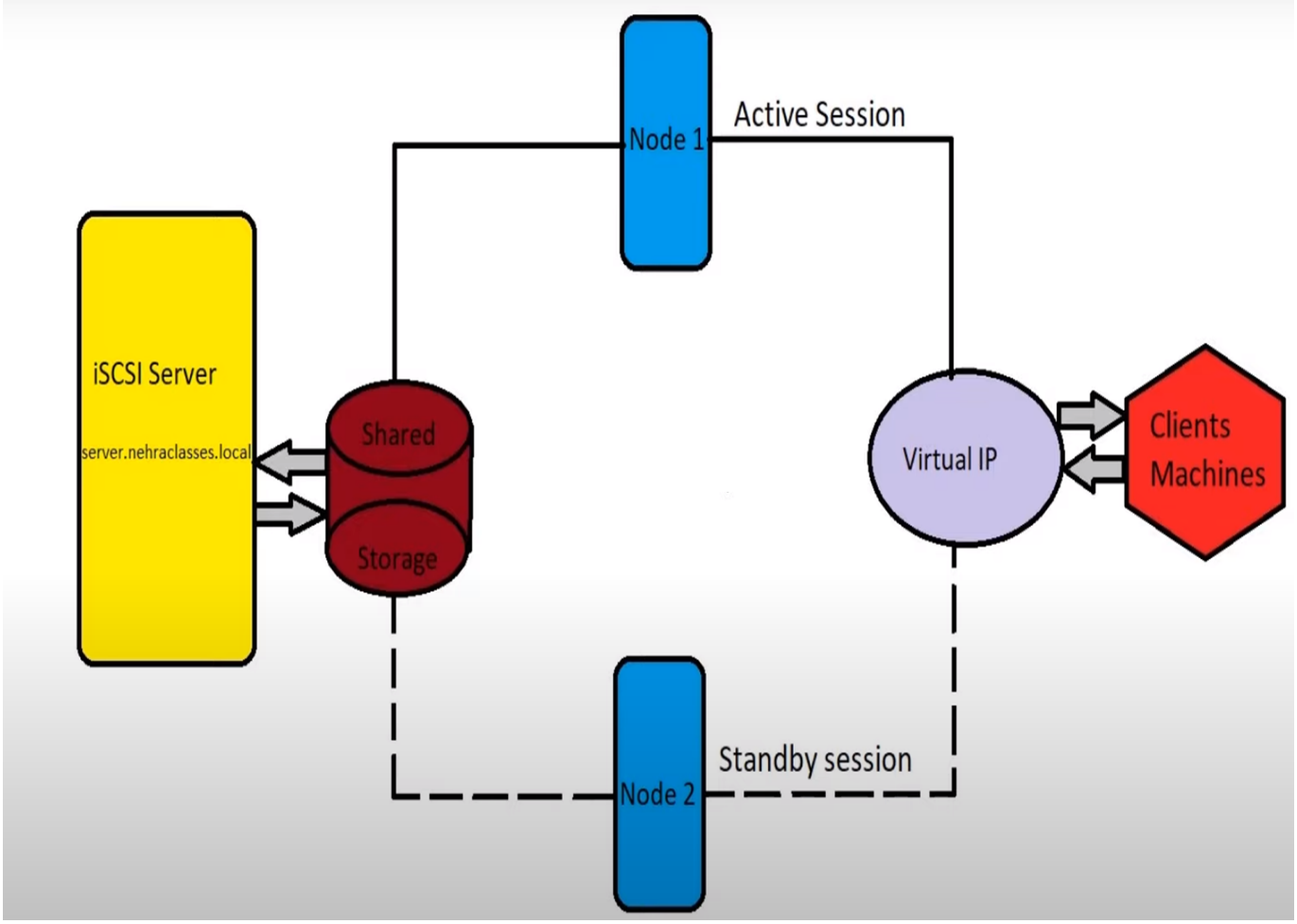
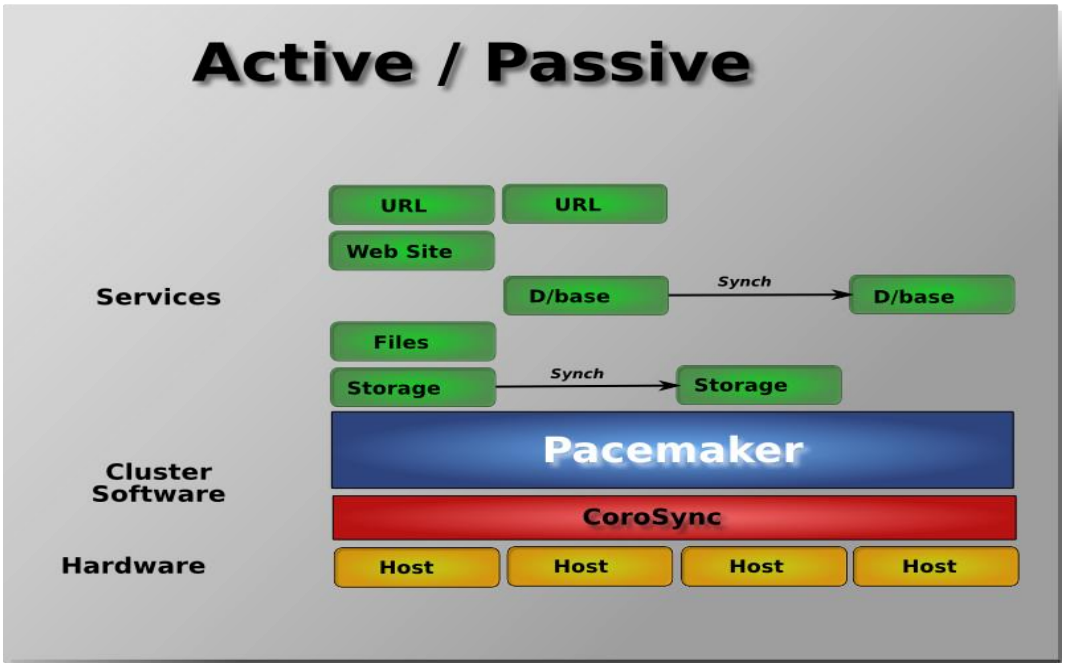
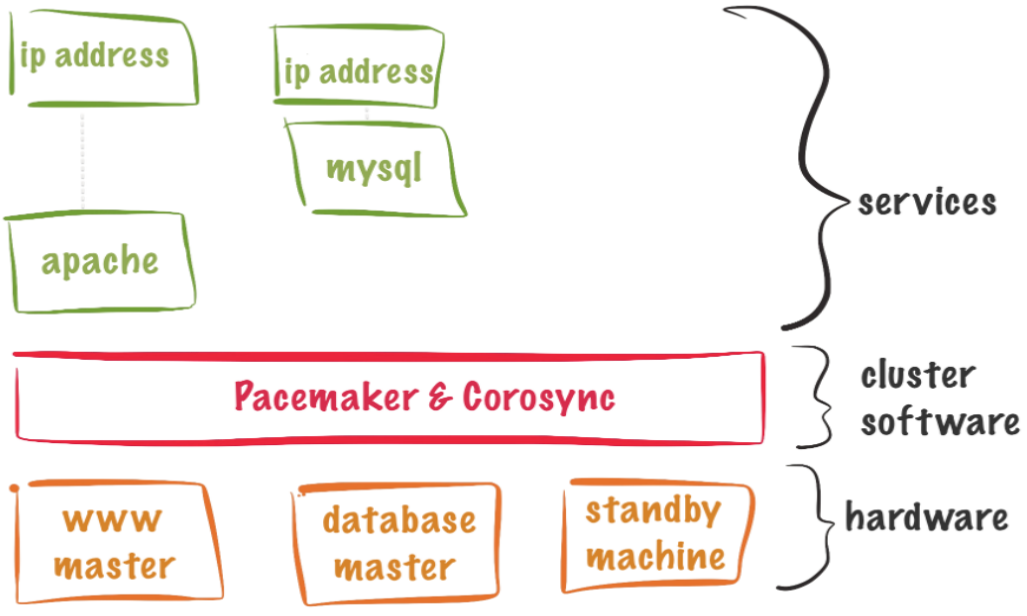


Diagram: System Architecture of High Availability Cluster using Pacemaler

8. Process Flow Diagram



- Services
- Cluster Software
- Hardware



9. Git Concepts

Git is a version control system that is used to manage and track changes made to source code and other files. It was created by Linus Torvalds in 2005 to manage the development of the Linux operating system. With Git, developers can track changes to their code, collaborate with others on a project, and maintain multiple versions of their codebase. Git uses a distributed model, which means that every developer has a complete copy of the codebase on their local machine. This allows developers to work offline and independently, and then synchronize their changes with others when they're ready.
Git also provides features like branching and merging, which allow developers to create parallel versions of their codebase and merge changes made by multiple developers back into the main codebase. These features make it easier to manage complex development workflows and collaborate on large projects.
Overall, Git is a powerful and widely used tool for managing software development projects.

git init

The command git init is used to create an empty Git repository.
After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.
git init

git add

Add command is used after checking the status of the files, to add those files to the staging area.
Before running the commit command, "git add" is used to add any new or modified files. git add .

git commit

The commit command makes sure that the changes are saved to the local repository.
The command "git commit –m <message>" allows you to describe everyone and help them understand what has happened.
git commit -m “commit message”

git-commit git status

The git status command tells the current state of the repository.
The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

git config

The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.

When git config is used with --global flag, it writes the settings to all repositories on the computer.

git config --global user.name “any user name” git config --global user.email <email id> gitcon

git branch

The git branch command is used to determine what branch the local repository is on. The command enables adding and deleting a branch.

Create a new branch
git branch <branch_name>

List all remote or local branches git branch -a

Delete a branch
git branch -d <branch_name>

git checkout

The git checkout command is used to switch branches, whenever the work is to be started on a different branch.

The command works on three separate entities: files, commits, and branches.

Checkout an existing branch git checkout <branch_name>

Checkout and create a new branch with that name git checkout -b <new_branch>

git merge

The git merge command is used to integrate the branches together. The command combines the changes from one branch to another branch.

It is used to merge the changes in the staging branch to the stable branch. git merge <branch_name>

Basic git commands:

git remote

The git remote command is used to create, view, and delete connections to other repositories. The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

git remote add origin <address> git remote

git clone

The git clone command is used to create a local working copy of an existing remote repository.

The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

git clone <remote_URL>

git pull

The git pull command is used to fetch and merge changes from the remote repository to the local repository.

The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

git pull <branch_name> <remote URL>

git-pull git push

The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.

The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

git push -u origin master

10.AWS

AWS stands for Amazon Web Services. AWS is one, most popular and ever-evolving cloud computing platform developed by Amazon. This cloud computing platform not only allows the computation but also offers many other services in order to make it much easier for a developer to develop an application and also allows users to focus primarily on the business logic as it automatically manages most of the application components such as Operating system, hardware needed, memory required, etc with “pay as you go” pricing scheme.

High availability of an application means how fast the application responds to the server or to the customer. Here availability means the application is available for the customer. Higher availability results in lower latency. So today in this article we will discuss various strategies to attain High availability on an application created on AWS.

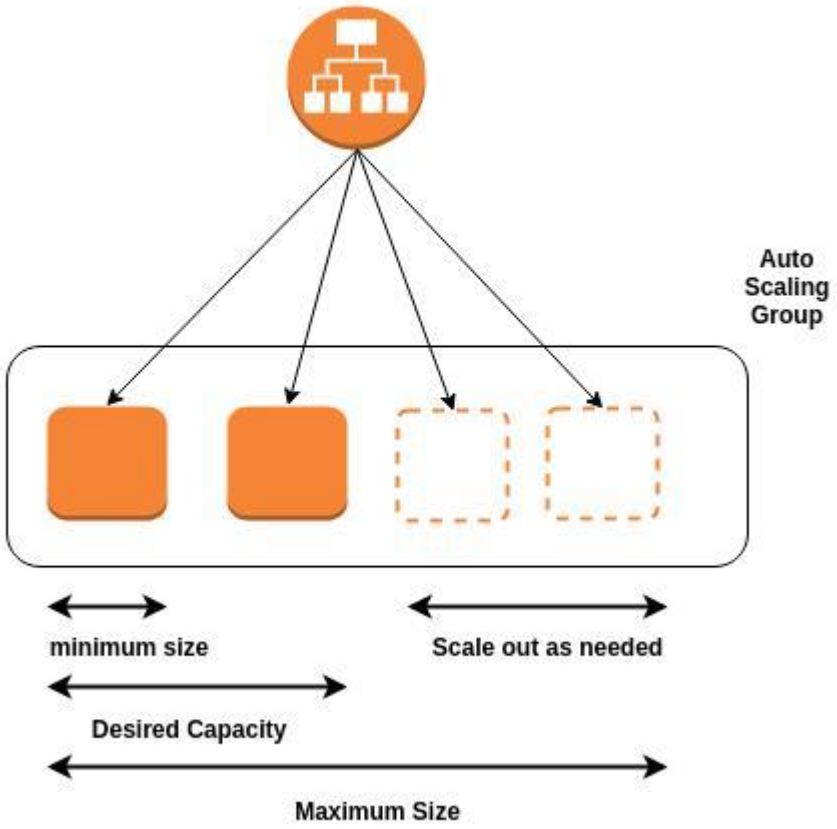
- Region:

A region is a separate, self-contained geographic area where all the resources needed for an application are contained. AWS has regions spread in the whole world. These regions are further divided into an availability zone. The availability zone is the different and separate isolated spaces within a region containing all the resources. So when you create an application, An availability zone from a specific region is selected for launching that application. Choosing the right region is an important factor.

- Availability Zone:

As mentioned above, regions are geographic areas and are located in big cities such as Sydney, London, etc. These regions contain many availability zones which might be a distance apart but within the same region. These availability zones also contribute to achieving the high availability of applications.

So to compute or to develop an application, the developer must launch an Elastic Compute Cloud (EC2) instance. These EC2 instances are launched in a specific availability zone of a specific region but suppose what happens to the application if due to some reason that specific availability zone is not available. Although AWS ensures no data loss but your application will not be available. So in order to tackle this situation, and to increase availability, 1 must launch 2 separate EC2 instances in 2 different availability zones.



- EC2 Auto Scaling:

Elastic Cloud Compute Auto Scaling is one of the many services offered by AWS. EC2 Autoscaling contributes to achieving higher availability by automatically removing or adding instances. The other two: Finding a suitable region and more than 1 instance approaches the primary level approaches and doesn't help much but when it comes to heavy traffic, EC2 Auto Scaling finds its application. To understand better suppose you have chosen the right region and are using different EC2 instances within the same region but different availability zones. But suddenly the traffic of your application increases so much so that the available instance is unable to handle the traffic. Then the EC2 auto-scaling will add new instances to divide the traffic. This will return the availability as more users can use the application at the same time.

11.iSCSI

iSCSI, which stands for Internet Small Computer Systems Interface or iSCSI is an Internet Protocol-based storage networking standard for linking data storage facilities. iSCSI provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network.

iSCSI (Internet Small Computer System Interface) is a protocol that allows SCSI commands to be sent over a network. It enables the remote access of storage devices, such as hard drives and tape drives, over a TCP/IP network. iSCSI is commonly used in data storage and server environments, providing a way to access storage resources over long distances without the need for physical proximity. It's often utilized in storage area networks (SANs) and other enterprise storage solutions.

Internet Small Computer Systems Interface or iSCSI is an Internet Protocol-based storage networking standard for linking data storage facilities. iSCSI provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network. iSCSI facilitates data transfers over intranets and to manage storage over long distances. It can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval.

The protocol allows clients (called initiators) to send SCSI commands (CDBs) to storage devices (targets) on remote servers. It is a storage area network (SAN) protocol, allowing organizations to consolidate storage into storage arrays while providing clients (such as database and web servers) with the illusion of locally attached SCSI disks. It mainly competes with Fibre Channel, but unlike traditional Fibre Channel which usually requires dedicated cabling, iSCSI can be run over long distances using existing network infrastructure. iSCSI was pioneered by IBM and Cisco in 1998 and submitted as a draft standard in March 2000.

12.Pacemaker

We will use pacemaker and corosync to configure High Availability Cluster. Pacemaker is a cluster resource manager, that is, a logic responsible for a life-cycle of deployed software — indirectly perhaps even whole systems or their interconnections — under its control within a set of computers and driven by prescribed rules.

It achieves maximum availability for your cluster services by detecting and recovering from node- and resource-level failures by making use of the messaging and membership capabilities provided by your preferred cluster infrastructure (either Corosync or Heartbeat), and possibly by utilizing other parts of the overall cluster stack.

Pacemaker is a powerful tool for achieving high availability (HA) in server infrastructures. Let’s delve into the details:

Corosync and Pacemaker:
Corosync provides cluster membership and messaging capabilities, allowing servers to communicate as a cluster.
Pacemaker acts as an open-source cluster resource manager (CRM), coordinating resources and services within the cluster to ensure high availability.

High Availability Setup:
The goal is to create an active/passive HA setup using Corosync, Pacemaker, and a Reserved IP on DigitalOcean.

The setup involves two Centos 7 servers:
The primary (active) server handles traffic.
The secondary (passive) server takes over if the primary fails.

Working:
The Reserved IP points to the primary server.
If Pacemaker detects primary server unavailability, the secondary server takes over.
The secondary server reassigns the Reserved IP via the DigitalOcean API.
Incoming traffic is then directed to the secondary server.

Steps to Set Up HA:
Create 2 Droplets to receive traffic.
Configure Reserved IP and assign it to one of the Droplets.
Install and configure Corosync.
Install and configure Pacemaker.
Test failover to ensure seamless transition.
Optionally, configure Nginx for load balancing.

Pacemaker is a cluster resource manager. It achieves maximum availability for your cluster services and resources by making use of the cluster infrastructure's messaging and membership capabilities to deter and recover from node and resource-level failure.

In high availability, a resource can be something as simple as an IP address that "floats" between cluster nodes, or something as complex as a database instance with a very intricate configuration. Put simply, a resource is anything that the cluster starts, stops, monitors, recovers or moves around. Cluster resource management is what performs these tasks for us—in an automated, transparent, highly configurable way. The canonical cluster resource manager in high-availability Linux is Pacemaker.

Pacemaker is a spin-off of Heartbeat, the high-availability stack formerly driven primarily by Novell (which then owned SUSE) and IBM. It re-invented itself as an independent and much more community-driven project in 2008, with developers from Red Hat, SUSE and NTT now being the most active contributors.

Pacemaker provides a distributed Cluster Information Base (CIB) in which it records the configuration and status of all cluster resources. The CIB replicates automatically to all cluster nodes from the Designated Coordinator (DC)—one node that Pacemaker automatically elects from all available cluster nodes.

The CIB uses an XML-based configuration format, which in releases prior to Pacemaker 1.0 was the only way to configure the cluster—something that rightfully made potential users run away screaming. Since these humble beginnings, however, Pacemaker has grown into a tremendously useful, hierarchical, self-documenting text-based shell, somewhat akin to the “virsh” subshell that many readers will be familiar with from libvirt. This shell—unimaginatively called “crm” by its developers—hides all that nasty XML from users and makes the cluster much simpler and easier to configure.

In Pacemaker, the shell allows us to configure cluster resources—no surprise there—and operations (things the cluster does with resources). Besides, we can set per-node and cluster-wide attributes, send nodes into a standby mode where they are temporarily ineligible for running resources, manipulate resource placement in the cluster, and do a plethora of other things to manage our cluster. Finally, Pacemaker's Policy Engine (PE) recurrently checks the cluster configuration against the cluster status and initiates actions as required.

The PE would, for example, kick off a recurring monitor operation on a resource (such as, “check whether this database is still alive”); evaluate its status (“hey, it's not!”); take into account other items in the cluster configuration (“don't attempt to recover this specific resource in place if it fails more than three times in 24 hours”); and initiate a follow-up action (“move this database to a different node”). All these steps are entirely automatic and require no human intervention, ensuring quick resource recovery and maximum uptime.

At the cluster resource management level, Pacemaker uses an abstract model where resources all support predefined, generic operations (such as start, stop or check the status) and produce standardized return codes. To translate these abstract operations into something that is actually meaningful to an application, we need resource agents.

13.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):

STONITH is the Pacemaker fencing implementation. It acts as a cluster resource in Pacemaker that processes fence requests, forcefully shutting down nodes and removing them from the cluster to ensure data integrity. STONITH is configured in the CIB and can be monitored as a normal cluster resource.

14.Pacemaker configuration and management tools

The High Availability Add-On features two configuration tools for cluster deployment, monitoring, and management.

pcs

The pcs command-line interface controls and configures 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 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.

15.Corosync

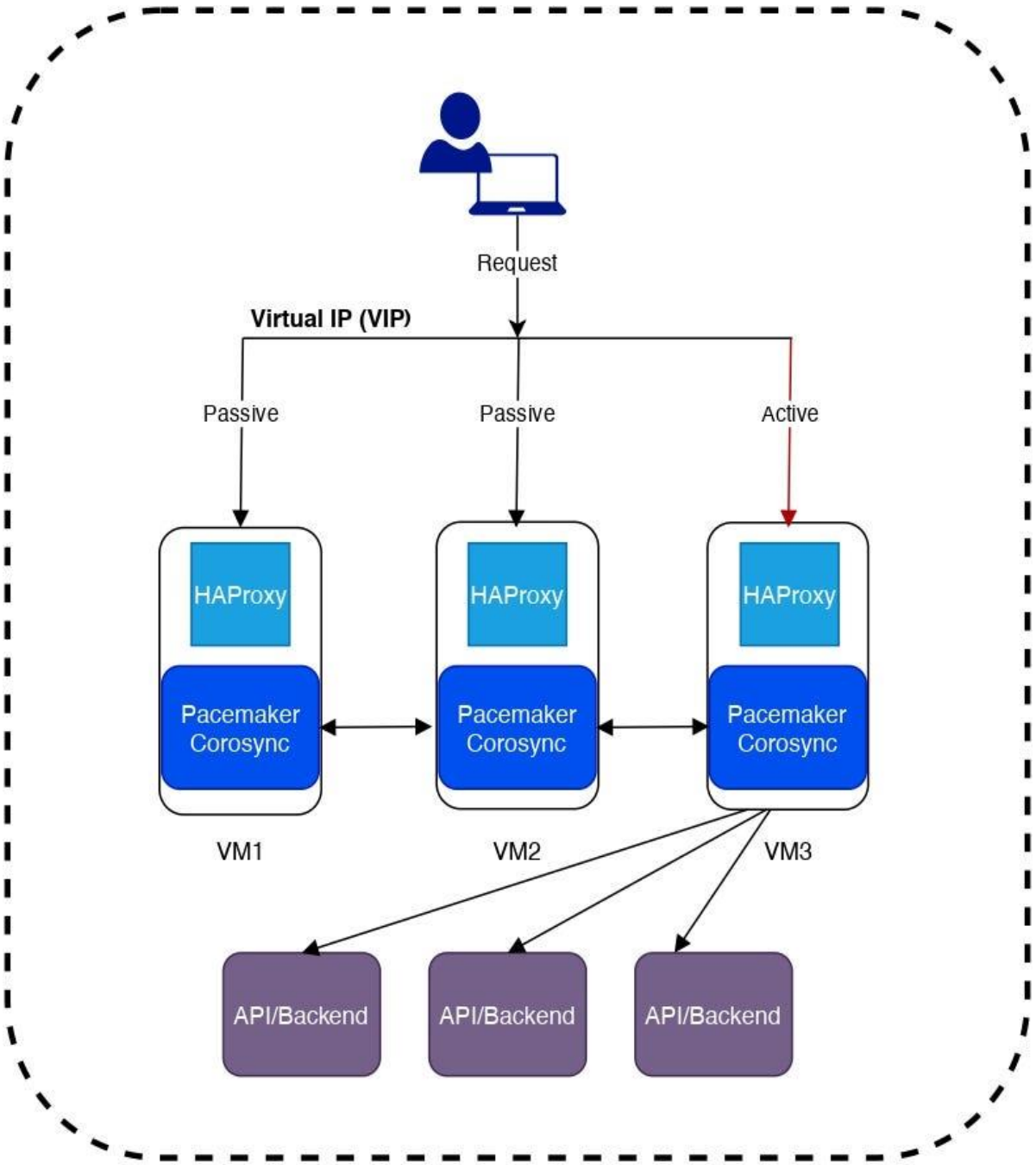
Corosync is an open source program that provides cluster membership and messaging capabilities, often referred to as the messaging layer, to client servers. Pacemaker is an open source cluster resource manager (CRM), a system that coordinates resources and services that are managed and made highly available by a cluster. In essence, Corosync enables servers to communicate as a cluster, while Pacemaker provides the ability to control how the cluster behaves. It provides cluster membership and messaging capabilities, allowing servers to communicate as a cluster.

Corosync is the component - and a daemon of the same name - that serves the core membership and member-communication needs for high availability clusters. It is required for the High Availability Add-On to function.

In addition to those membership and messaging functions, corosync also:

- Manages quorum rules and determination.
- Provides messaging capabilities for applications that coordinate or operate across multiple members of the cluster and thus must communicate stateful or other information between instances.
- Uses the kronosnet library as its network transport to provide multiple redundant links and automatic failover.

16. Setup Pacemaker/Corosync Cluster



This figure explains how to setup Pacemaker/Corosync cluster for highly available, scalable resource manager and HAProxy load balancer for splitting traffic to backend.

Technology stack:

- HAProxy:
  HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications.

- Pacemaker:
  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.

A Pacemaker stack is built on five core components:

- libQB — core services (logging, IPC, etc)
- Corosync — Membership, messaging and quorum
- Resource agents — A collection of scripts that interact with the underlying services managed by the cluster
- Fencing agents — A collection of scripts that interact with network power switches and SAN devices to isolate cluster members
- Pacemaker itself

- Virtual IP:
  VIP floats between virtual machines or baremetal machines of pacemaker cluster. VIP receives the request and HAProxy passes the request to target API/backend.

17. HA Cluster Configuration

---

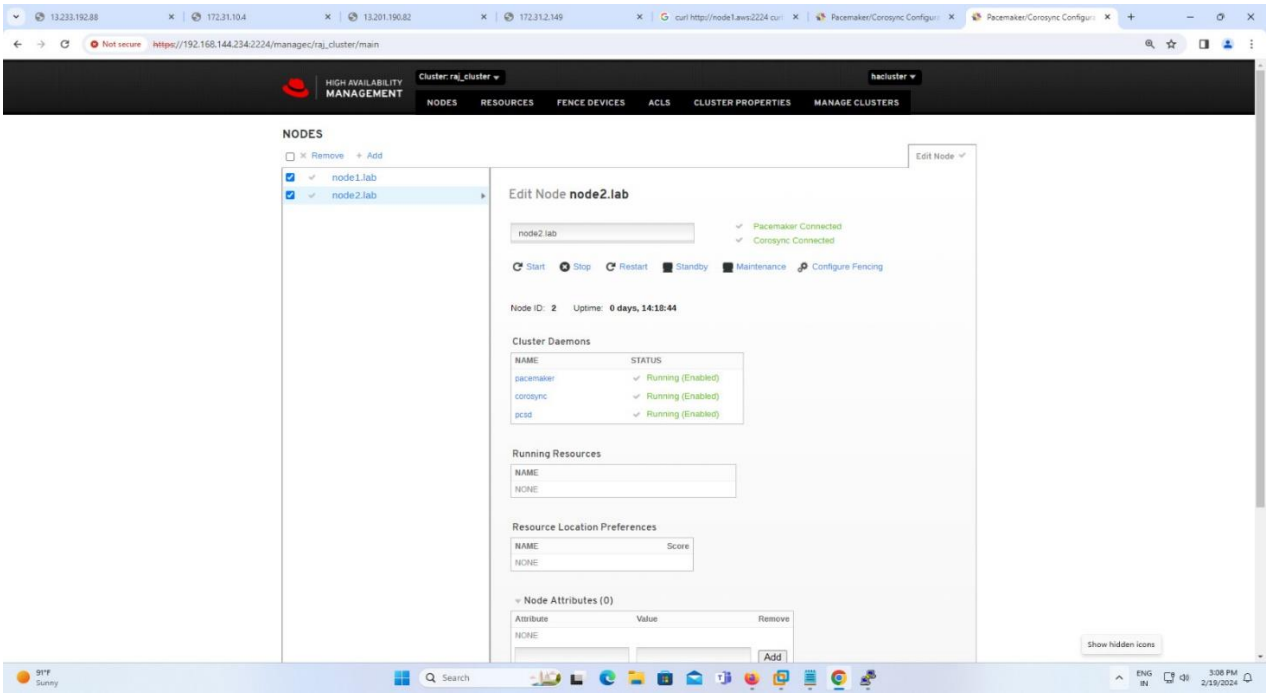
1. #HA Cluster Configuration CentOS 7:
2. #Environment:
3. #Here, we will configure a failover cluster with Pacemaker to make the Apache (web) server as a highly available application.
4. #Here, we will configure the Apache web server, filesystem, and networks as resources for our cluster.
5. #For a filesystem resource, we would be using shared storage coming from iSCSI storage.
6. #CentOS 7 High Availability Cluster Infrastructure
7. #Host Name IP Address OS Purpose
8. #node1.lab 192.168.1.126 CentOS 7 Cluster Node 1
9. #node2.lab 192.168.1.119 CentOS 7 Cluster Node 2
10. #iscsi.lab 192.168.1.109 CentOS 7 iSCSI Shared Storage
11. #virtual.lab 192.168.1.112 CentOS 7 Virtual Cluster IP
12. #on storage:
13. yum install targetcli lvm2
14. #on nodes:
15. yum install -y iscsi-initiator-utils lvm2
16. #on storage:
17. yum install -y targetcli lvm2 iscsi-initiator-utils lvm2
18. fdisk -l | grep -i sd -----list the available disks in the iSCSI server
19. #create an LVM on the iSCSI server to use as shared storage for our cluster nodes.
20. pvcreate /dev/sdb
21. vgcreate vg_iscsi /dev/sdb
22. lvcreate -l 100%FREE -n lv_iscsi vg_iscsi
23. #Node 1:
24. cat /etc/iscsi/initiatorname.iscsi
25. ##InitiatorName=iqn.1994-05.com.redhat:121c93cbad3a
26. #Node 2:
27. cat /etc/iscsi/initiatorname.iscsi
28. ##InitiatorName=iqn.1994-05.com.redhat:827e5e8fecb
29. #Enter the below command to get an iSCSI CLI for an interactive prompt.
30. #on storage:
31. targetcli
32. cd /backstores/block
33. create iscsi_shared_storage /dev/vg_iscsi/lv_iscsi -----Created block storage object iscsi_shared_storage using /dev/vg_iscsi/lv_iscsi.
34. cd /iscsi

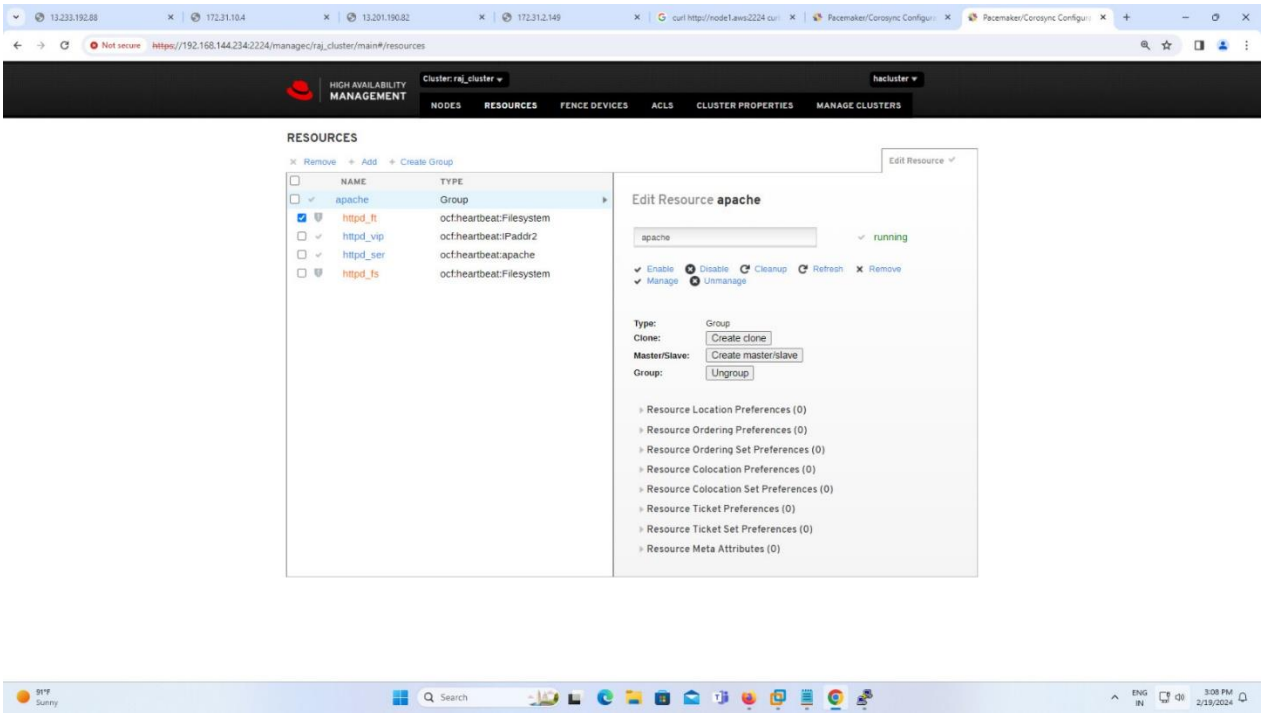
35. create(Created target iqn.2003-01.org.linux-iscsi.storage.x8664:sn.eac9425e5e18.
    a. Created TPG 1.
    b. Global pref auto_add_default_portal=true
    c. Created default portal listening on all IPs (0.0.0.0), port 3260.)
36. cd<iqn.2003-01.org.linux-iscsi.storage.x8664:sn.eac9425e5e18>/tpg1/acls ----< Change as per the output of previous command>
37. create iqn.1994-05.com.redhat:121c93cbad3a -----<< Node 1>>
38. create iqn.1994-05.com.redhat:827e5e8fecb ------<< Node 2>>
39. cd/iscsi/<iqn.2003-01.org.linux-iscsi.storage.x8664:sn.eac9425e5e18>/tpg1/luns
40. create /backstores/block/iscsi_shared_storage ------(Created LUN 0.
    a. Created LUN 0->0 mapping in node ACL iqn.1994-05.com.redhat:827e5e8fecb
    b. Created LUN 0->0 mapping in node ACL iqn.1994-05.com.redhat:121c93cbad3a)
41. cd /
42. ls
43. saveconfig
44. exit
45. systemctl enable target
46. systemctl restart target
47. firewall-cmd --permanent --add-port=3260/tcp
48. firewall-cmd --reload
49. #Both Nodes:
50. iscsiadm -m discovery -t st -p <ip of storage machine>
51. iscsiadm -m node -T <iqn.2003-01.org.linux-iscsi.storage.x8664:sn.73638ebfea25> <192.168.144.202> -l ----iqn and ip of storage machine
52. lsblk
53. #Node1:
54. pvcreate /dev/sdb
55. vgcreate vg_apache /dev/sdb
56. lvcreate -n lv_apache -l 100%FREE vg_apache
57. #Node2:
58. pvscan
59. vgscan
60. lvscan
61. lvdisplay /dev/vg_apache/lv_apache
62. #Both nodes:
63. vi /etc/hosts ---add ip address and hostnames of both nodes
64. yum-config-manager --set-enabled HighAvailability
65. yum install -y pcs fence-agents-all pcp-zeroconf

66. firewall-cmd --permanent --add-service=high-availability
67. firewall-cmd --add-service=high-availability
68. firewall-cmd --reload
69. passwd hacluster
70. systemctl start pcsd
71. systemctl enable pcsd
72. # node1:
73. pcs cluster auth node1.lab node2.lab
74. pcs cluster setup --name project_cluster --start node1.lab node2.lab
75. pcs cluster enable --all
76. pcs cluster status
77. pcs status
78. pcs property set stonith-enabled=false ----to disable fence devices
79. #Both nodes:
80. yum install -y httpd
81. vim /etc/httpd/conf/httpd.conf ---(at the last copy following line)
82. <Location/server-status>
83. SetHandler server-status
84. Require local
85. </Location>
86. /bin/systemclt reload httpd.sevice > /dev/null 2>/dev/null || true
87. /usr/sbin/httpd -f /etc/httpd/conf/httpd.conf -c "PidFile /var/run/httpd.pid" -k graceful > /dev/null 2>/dev/null || true
88. #node1:
89. mount /dev/vg_apache/lv_apache /var/www/
90. mkdir /var/www/html
91. mkdir .var/www/cgi-bin
92. mkdir /var/www/error
93. restorecon -R /var/www
94. cat <<-END >/var/www/html/index.html ----(in file copy the following code)
95. <html>
96. <body>Hello, Welcome! this page is for project by HPCSA group no.1</body>
97. </html>
98. END
99. cat /var/www/html/index.html
100. unmount /var/www
101. #Both nodes:
102. firewall-cmd --permanent --add-service=http
103. firewall-cmd --reload
104. #node1:
105. pcs resource create httpd_fs Filesystem device="/dev/mapper/vg_apache-lv_apache" directory="/var/www" fstype="ext4" --group apche

106. pcs resource create httpd_vip IPaddr2 ip=192.168.144.203 cidr_netmask=24 --group apache
107. pcs resource create httpd_ser apache configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-status" --group apache
108. pcs status
109. #web browser:
110. ##search virtual machine ip address
111. ##search virtual machine ip address with 2224 default port
112. Node1: pcs node standby node1.lab

18. Result


HIGH AVAILABILITY MANAGEMENT
Cluster: raj_cluster
hacluster
NODES
RESOURCES
FENCE DEVICES
ACLS
CLUSTER PROPERTIES
MANAGE CLUSTERS
NODES
Remove
Add
node1.lab
node2.lab
Edit Node
Edit Node node2.lab
Pacemaker Connected
Corosync Connected
Start
Stop
Restart
Standby
Maintenance
Configure Fencing
Node ID: 2
Uptime: 0 days, 14:18:44
Cluster Daemons
NAME
STATUS
pacemaker
Running (Enabled)
corosync
Running (Enabled)
pcsd
Running (Enabled)
Running Resources
NONE
Resource Location Preferences
Score
Node Attributes (0)
Attribute
Value
Remove
Add
Show hidden icons


HIGH AVAILABILITY MANAGEMENT
Cluster: raj_cluster
hacluster
NODES
RESOURCES
FENCE DEVICES
ACLS
CLUSTER PROPERTIES
MANAGE CLUSTERS
RESOURCES
Remove
Add
Create Group
Edit Resource
NAME
TYPE
apache
Group
httpd_fs
ocf:heartbeat:Filesystem
httpd_vip
ocf:heartbeat:IPaddr2
httpd_ser
ocf:heartbeat:apache
Edit Resource apache
running
Enable
Disable
Cleanup
Refresh
Remove
Manage
Unmanage
Type:
Group
Clone:
Create clone
Master/Slave:
Create master/slave
Group:
Ungroup
Resource Location Preferences (0)
Resource Ordering Preferences (0)
Resource Ordering Set Preferences (0)
Resource Colocation Preferences (0)
Resource Colocation Set Preferences (0)
Resource Ticket Preferences (0)
Resource Ticket Set Preferences (0)
Resource Meta Attributes (0)

19. Conclusion

In conclusion, implementing a high availability (HA) cluster using Pacemaker offers a robust and reliable solution for ensuring continuous availability of critical services and applications. Pacemaker's automatic failover capabilities, scalability, resource management features, and ease of management make it a preferred choice for building resilient infrastructure.

By effectively managing cluster resources and automatically responding to failures, Pacemaker minimizes downtime and maximizes system uptime, ultimately enhancing the overall reliability and performance of the IT environment.

High Availability clusters powered by Pacemaker represent a cornerstone in ensuring the resilience and continuity of critical services in today's digital landscape. By automating failover processes, managing resources efficiently, and offering flexibility in configuration, Pacemaker enables organizations to mitigate the risk of downtime and maximize service uptime. As businesses continue to prioritize availability and reliability, HA clusters with Pacemaker emerge as indispensable components of robust IT infrastructures.

At this point High Availability Cluster project with Pacemaker is being configured successfully.

20.References

Here are some references that can be used for further reading on implementing High Availability Cluster using Pacemaker:

1. https://www.ibm.com/docs/en/ibm-mq/9.3?topic=availability-defining-pacemaker-cluster-ha-group

2. https://clusterlabs.org/pacemaker/

3. https://www.redhat.com/sysadmin/rhel-pacemaker-cluster

4. https://medium.com/@prateekbansalind/pacemaker-in-linux-ensuring-heartbeats-of-high-availability-services-111b6ae1f0c7

5. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_administration/ch-startup-haaa