Évaluation: La haute disponibilité

SOMMAIRE

I.Définitions

1.1.Haute disponibilité :

La haute disponibilité ce sont toutes les dispositions visant à garantir la disponibilité d'un service et son bon fonctionnement 24h sur 24.

1.2.Pacemaker :

Pacemaker est le **gestionnaire de ressources de cluster (CRM)** qui agit comme le chef d'orchestre de l'infrastructure de haute disponibilité (HA). Son **rôle principal est d'assurer que les services critiques, comme le serveur web Apache, sont toujours disponibles en surveillant leur état en permanence**. Si un nœud du cluster ou un service échoue, Pacemaker le détecte immédiatement et ordonne le **basculement (failover)** en déplaçant automatiquement les ressources associées (l'adresse IP flottante et le service Apache) vers un nœud sain, tout en respectant les règles de dépendance définies (comme le groupe WebCluster), assurant ainsi une transition rapide et sans intervention humaine.

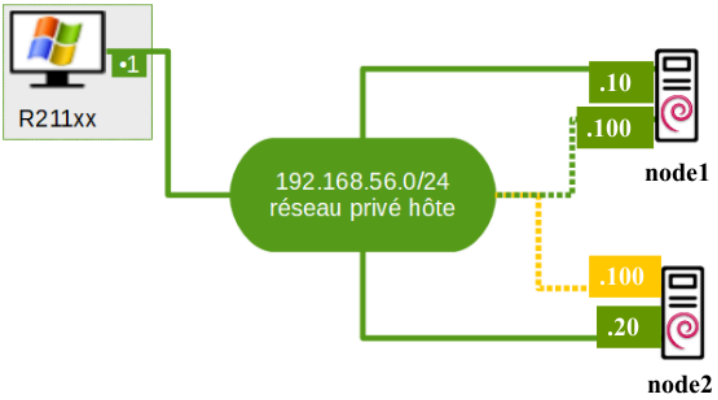
1.3.Corosync :

Corosync, **c'est le cœur qui maintient le contact entre les serveurs du cluster**. Il **gère la communication rapide entre toutes les machines (les nœuds) et s'assure qu'elles savent en temps réel qui est en ligne et qui est en panne**. Corosync est absolument essentiel car il fournit à Pacemaker l'information la plus critique : l'appartenance au cluster (qui est là) et la garantie du quorum (la majorité des nœuds actifs pour prendre des décisions). Sans Corosync, Pacemaker n'aurait aucune idée de l'état de l'infrastructure.

II.Objectif du projet: Mise en place de la haute disponibilité (HA)

L'objectif principal de ce travail est de mettre en place une solution de Haute Disponibilité (HA) robuste et automatisée pour le serveur web Apache2 entre les deux nœuds, **Node 1** et **Node 2**. Pour cela, on utilise la suite Corosync et Pacemaker afin de créer un cluster actif/passif. Cette architecture garantit la continuité de service : en cas de défaillance totale du nœud principal, l'adresse IP flottante (**192.168.56.100**) et le service Apache basculent automatiquement et instantanément sur le nœud de secours, assurant que l'accès au site web n'est jamais interrompu pour les utilisateurs.

III.Schéma de la mise en place de la haute disponibilité



Le schéma montre l'architecture Actif/Passif de notre cluster. Il se compose de Node 1 et Node 2 connectés au réseau privé. L'élément clé est l'Adresse IP flottante (**192.168.56.100**) : elle n'est active que sur un seul nœud à la fois (ici, elle est sur Node 1). Tous les utilisateurs accèdent au service via cette adresse unique. En cas de panne de Node 1, l'IP **.100** et le service Apache basculent immédiatement et automatiquement vers Node 2, garantissant la continuité du service sans que l'utilisateur ne change d'adresse.

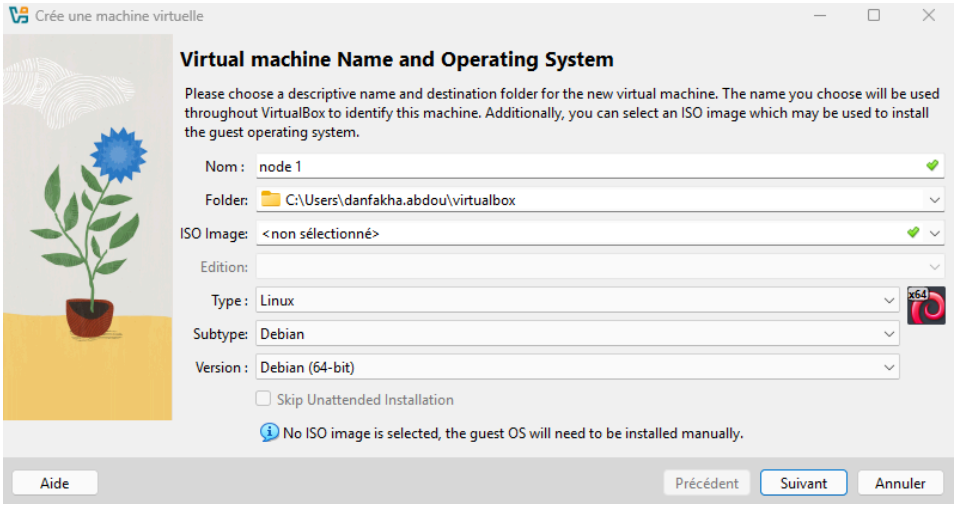
IV.Mise en oeuvre de la haute disponibilité

Cette section décrit toutes les étapes nécessaires à la mise en place du cluster de haute disponibilité, de la préparation initiale des systèmes à la configuration finale des ressources.

4.1. Préparation des noeuds (node1 et node2)

Pour commencer notre mise en œuvre nous allons créer une machine Debian (node 1) qu'on va ensuite cloner pour créer la deuxième machine (node2).

Pour cela nous allons aller dans **outils**, **nouvelle**, puis nous allons entrez toutes les informations nécessaires.



Une fois la machine créée nous allons commencer par changer le nom de la machine, pour cela nous devons taper la commande suivante: **nano /etc/hostname** puis nous allons changer le nom de la machine et mettre node1.

```
root@node1:~# nano /etc/hostname
```

```
node1
```

Une fois cela fait nous allons changer l'adresse ip de la machine pour cela pour cela nous allons taper la commande suivante: **nano /etc/network/interfaces**

```
root@node1:~# nano /etc/network/interfaces
```

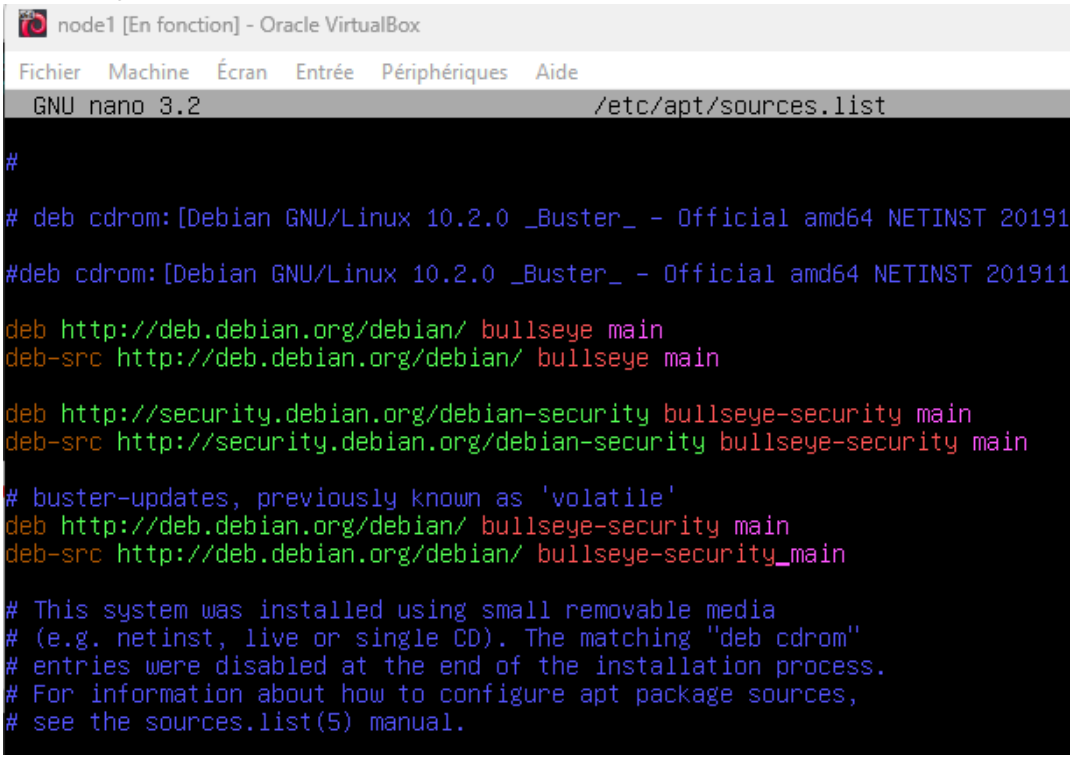
```
  GNU nano 3.2                          /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
address 192.168.56.10/24
gateway 192.168.56.254
```

Pour que les modifications soient prises en compte nous allons faire un **reboot**

```
root@node1:~# reboot
```

Ensuite nous allons installer **apache**, **ssh**, **pacemaker** et **corosync.**
Avant de commencer l'installation nous allons faire un **apt update qui sert à mettre à jour la liste des logiciels disponibles sur internet, sans rien installer. C'est comme consulter le catalogue avant de faire les courses.**

Devant l'échec de la mise à jour des index de paquets **(apt update)**, le fichier **/etc/apt/sources.list** a été édité pour assurer la cohérence avec Debian 11 (Bullseye)**.** Les références obsolètes 'buster' ont été corrigées, et les lignes correspondant aux dépôts **security** ont été ajoutées pour garantir l'accès aux mises à jour.



Une fois cela fais nous devrons faire un **reboot** afin que les modifications soient bien prises en compte ensuite nous pourrons passer aux différentes installations.

```
root@node1:~# apt install apache2
```

```
root@node1:~# apt install ssh
```

```
apt install pacemaker corosync pcs
```

4.2.Initialisation du cluster

Avant de commencer l'initialisation du cluster. le fichier **/etc/corosync/corosync.conf** a été édité pour définir le nom du cluster, la liste

des nœuds et activer la gestion du quorum pour un environnement à deux nœuds.

```
root@node1:~# nano /etc/corosync/corosync.conf
```

Dans un premier temps nous avons nommé le cluster**.**

```
  GNU nano 3.2                          /etc/corosync/corosync.conf

# Please read the corosync.conf.5 manual page
totem {
        version: 2

        # Corosync itself works without a cluster name, but DLM needs one.
        # The cluster name is also written into the VG metadata of newly
        # created shared LVM volume groups, if lvmlockd uses DLM locking.
        cluster_name: clusterweb_

        # crypto_cipher and crypto_hash: Used for mutual node authentication
        # If you choose to enable this, then do remember to create a shared
        # secret with "corosync-keygen".
        # enabling crypto_cipher, requires also enabling of crypto_hash.
        # crypto works only with knet transport
        crypto_cipher: none
        crypto_hash: none
}
```

Ensuite nous avons taper cette commande **two_node: 1** qui permet de maintenir le quorum avec un seul nœud actif ce qui est crucial pour un cluster à 2 nœuds.

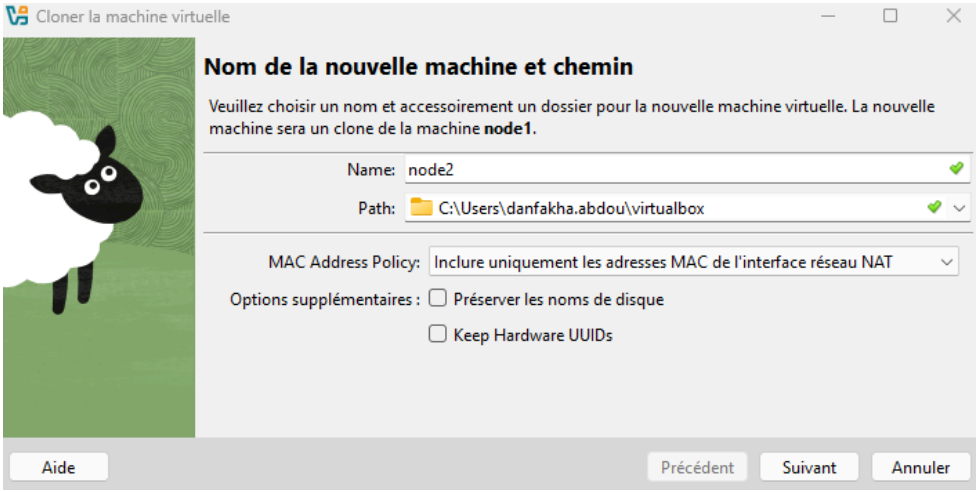
```
quorum {
        # Enable and configure quorum subsystem (default: off)
        # see also corosync.conf.5 and votequorum.5
        provider: corosync_votequorum
        two_node: 1
}
```

Pour finir nous avons défini explicitement les adresses IP (192.168.56.10 et 192.168.56.20) de communication pour chaque nœud.

```
nodelist {
        node {
                ring0_addr: 192.168.56.10
                name: node1
                nodeid: 1
        }
        node {
                ring0_addr: 192.168.56.20
                name: node2
                nodeid: 2
        }
        # ...
}
```

4.3. Création et Post-Configuration de Node 2 (Clonage)

Après avoir configuré l'environnement de base et les fichiers de configuration Pacemaker/Corosync sur **Node 1**, la machine virtuelle a été **clonée** pour créer **Node 2**. Le clonage a entraîné l'héritage de l'adresse IP et du nom d'hôte de **Node 1**, nécessitant les ajustements suivants pour que **Node 2** puisse rejoindre le cluster :



Une fois que le clone à été créé nous avons changer le nom ainsi que l'adresse IP de la machine, une fois que cela à été effectué nous avons changer le fichier hosts des 2 machines pour cela il faut taper la commande suivante **nano /etc/hosts** puis nous avons ajouté ses lignes suivantes:

127.0.0.1 localhost
192.168.56.10 Node1
192.168.56.20 Node2

```
root@node2:~# nano /etc/hostname
```

```
node2
```

```
root@node2:~# nano /etc/network/interfaces
```

```
  GNU nano 3.2                                /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s3
iface enp0s3 inet static
address 192.168.56.20/24
gateway 192.168.56.254
```

```
nano /etc/hosts
```

```
  GNU nano 3.2                                      /etc/hosts

127.0.0.1       localhost
192.168.56.10   node1
192.168.56.20   node2
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

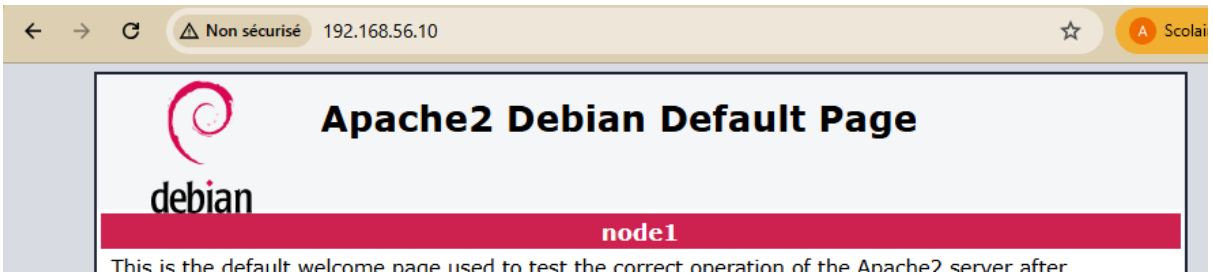
A la fin de chaque modifications il faut bien sûr ne pas oublier de **reboot** afin que les modifications soient bien prises en compte.

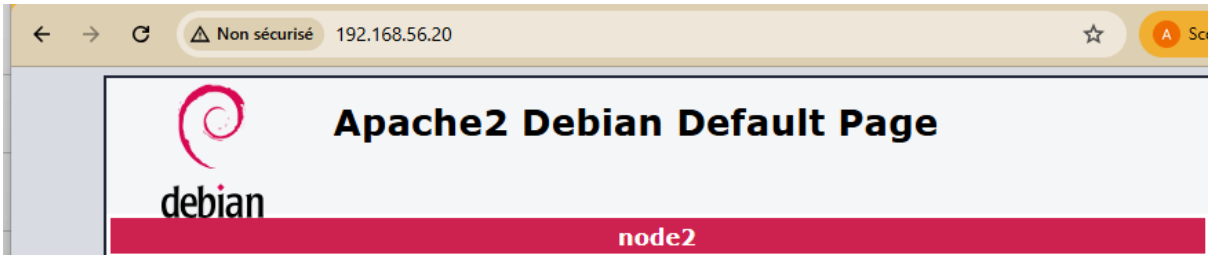
4.4. Personnalisation des Services pour le Test

Pour pouvoir valider visuellement le bon basculement des ressources entre les deux nœuds, les pages d'accueil Apache par défaut ont été modifiées sur chaque serveur pour indiquer clairement le nom du nœud.

```
root@node1:~# nano /var/www/html/index.html
```

```
GNU nano 3.2                        /var/www/html/index.html

        Apache2 Debian Default Page
      </span>
    </div>
<!--    <div class="table_of_contents floating_element">
      <div class="section_header section_header_grey">
        TABLE OF CONTENTS
      </div>
      <div class="table_of_contents_item floating_element">
        <a href="#about">About</a>
      </div>
      <div class="table_of_contents_item floating_element">
        <a href="#changes">Changes</a>
      </div>
      <div class="table_of_contents_item floating_element">
        <a href="#scope">Scope</a>
      </div>
      <div class="table_of_contents_item floating_element">
        <a href="#files">Config files</a>
      </div>
    </div>
-->
    <div class="content_section floating_element">


      <div class="section_header section_header_red">
        <div id="about"></div>
        node1
      </div>
      <div class="content_section_text">
        <p>
              This is the default welcome page used to test the correct
              operation of the Apache2 server after installation on Debian systems.
```





4.5. Synchronisation de l'Heure (NTP)

Une horloge synchronisée sur les deux nœuds est cruciale pour la communication Corosync. La commande **timedatectl status** a permis de vérifier que le service NTP était actif et que l'horloge système était bien synchronisée sur chaque serveur.

```
root@node1:~# timedatectl status
               Local time: jeu. 2025-11-20 14:41:40 CET
           Universal time: jeu. 2025-11-20 13:41:40 UTC
                 RTC time: jeu. 2025-11-20 13:41:41
                Time zone: Europe/Paris (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Ce statut confirme que les conditions de temps sont remplies pour un fonctionnement stable du cluster.

4.6.Configuration des ressources et tests de validation

Dans un premier temps nous avons désactivé le démarrage automatique en tapant la commande **systemctl disable apache2**. Cette étape est essentielle. Elle empêche Apache de démarrer au lancement du système d'exploitation. Pacemaker doit être le **seul** responsable** du démarrage et de l'arrêt du service HA pour garantir la haute disponibilité.

```
root@node1:~# systemctl disable apache2
```

Ensuite nous avons arrêté le service en tapant la commande **systemctl stop apache2** qui assure que le service est bien arrêté avant que Pacemaker ne le prenne en charge.

```
root@node1:~# systemctl stop apache2
```

Une fois cela fait nous avons défini l'adresse IP virtuelle (**192.168.56.100**) qui "flotte" entre les nœuds. L'option **op monitor interval=10s** demande à Pacemaker de vérifier l'état de cette IP toutes les 10 secondes.

```
root@node1:~# pcs configure primitive VirtulIP ocf:heartbeat:IPaddr2 \params ip=192.168.56.100 cidr_
netmask=24 op monitor interval=10s
```

La commande ne s'effectuais pas correctement donc nous avons tapé la commande suivante: **dpkg -l pcs** cette commande permet de savoir dans quel dossier se trouve exactement pcs afin que nous puissions taper la commande directement depuis le bon répertoire.
Puis nous avons changé la commande initiale en changeant primitive par **resource**.

```
root@node1:~# dpkg -l pcs
Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé/W=attend-traitement-déclen
|/ Err?=(aucune)/besoin Réinstallation (État,Err: majuscule=mauvais)
||/ Nom           Version         Architecture Description
+++-==============-===============-============-================================
ii  pcs           0.10.8-1+deb11u1 all          Pacemaker Configuration System
lines 1-6/6 (END)
```

```
root@node1:/usr/sbin# pcs resource create Virtual_IP ocf:heartbeat:IPaddr2 ip=192.168.56.100 cidr_ne
tmask=24 op monitor interval=10s
```

Ensuite nous avons défini le service Apache **(systemd:apache2**) comme une ressource HA. L'option **op monitor interval=20s** permet à Pacemaker de surveiller l'état d'Apache toutes les 20 secondes.

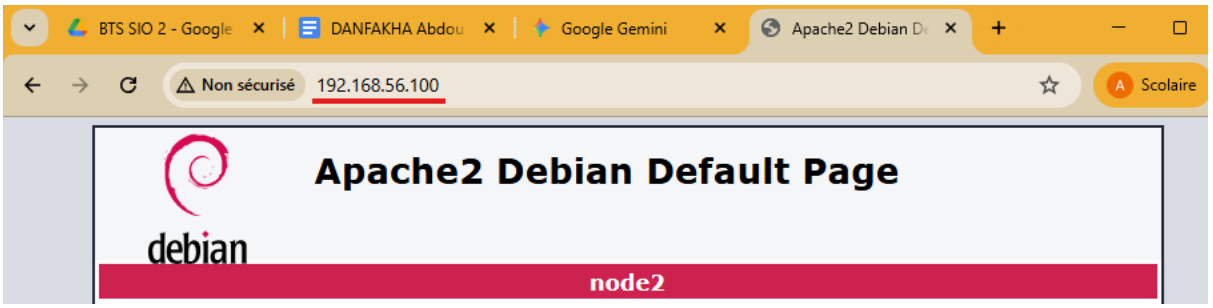
```
pcs resource create WebServer systemd:apache2 op monitor interval=20s
```

Pour fini pour vérifier que le cluster est en ligne et que les ressources ont bien été démarré nous avons taper la commande pcs status.

```
root@node1:~# pcs status
Cluster name: clusterweb
Cluster Summary:
  * Stack: corosync
  * Current DC: node2 (version 2.0.5-ba59be7122) - partition with quorum
  * Last updated: Fri Nov 21 13:56:00 2025
  * Last change:  Fri Nov 21 13:55:50 2025 by root via cibadmin on node1
  * 2 nodes configured
  * 2 resource instances configured

Node List:
  * Online: [ node1 node2 ]

Full List of Resources:
  * Resource Group: WebCluster:
    * Virtual_IP        (ocf::heartbeat:IPaddr2):        Started node2
    * WebServer (systemd:apache2):       Started node2

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Une fois toutes ses étapes faite nous avons tester une navigation vers l'adresse IP flottante, la navigation vers l'adresse IP flottante **(192.168.56.100)** ne renvoyait aucune page web. Cela est dû au fait que le service Apache **(WebServer)** n'avait pas d'ordre de démarrage défini et restait arrêté.Pour corriger cela et lier les dépendances, les deux ressources sont regroupées dans un Groupe de Ressources nommé **WebCluster**.

```
root@node1:/usr/sbin# pcs resource group add WebCluster Virtual_IP WebServer
```

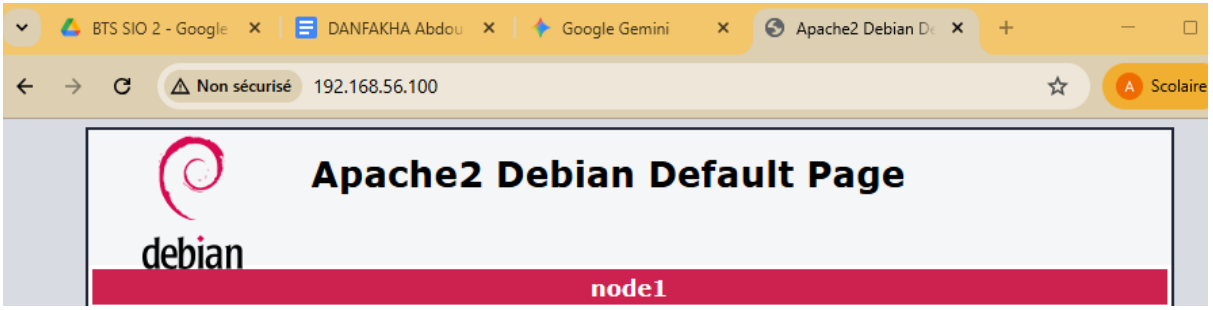


Malgré le démarrage réussi, la navigation vers l'adresse IP flottante **192.168.56.100** n'affichait pas la page Web attendue, confirmant un problème de configuration.
Pour régler le problème nouus avons tapér la commande suivante:

pcs resource move WebCluster node1
Cette commande permet de simuler un basculement du service sans avoir à éteindre le nœud actif.

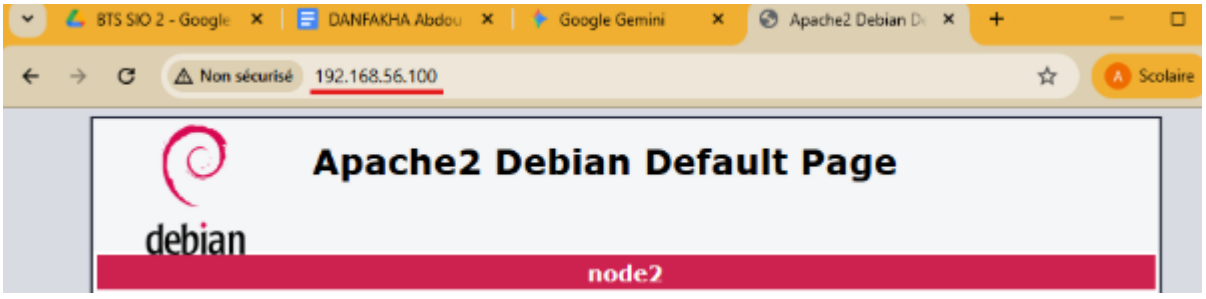
```
root@node1:~# pcs resource move WebCluster node1
```

Une fois cette commande tapée on peut voir que la migration a réussi. L'adresse IP flottante a basculé vers Node 1, ce qui a été confirmé en tapant **ip a** sur Node 1 (affichant l'IP **.100**) et en accédant à l'URL, qui affichait la page personnalisée de **Node 1**.




```
root@node1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1
000
    link/ether 08:00:27:53:d3:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.10/24 brd 192.168.56.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet 192.168.56.100/24 brd 192.168.56.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe53:d3bb/64 scope link
       valid_lft forever preferred_lft forever
root@node1:~#
```

Si on **arrête le service Pacemaker sur Node 1** (le nœud qui héberge actuellement le service) et qu'on rafraîchit la page web (via l'adresse flottante **192.168.56.100**), le cluster détectera l'absence du service sur Node 1 et ordonnera le **basculement (failover)**. On devrait alors **automatiquement voir la page Apache de Node 2**, prouvant que la haute disponibilité est fonctionnelle.



V.Problèmes rencontrés

Comme souvent avec ce type de solution, l'installation ne s'est pas faite sans accroc. La première difficulté a été de s'assurer que les deux serveurs pouvaient communiquer correctement, notamment après le clonage qui a causé des conflits d'identité (adresses IP et clés SSH en double). Une fois la base réseau stable, le cluster a rencontré une erreur bloquante lors de la migration des services (Error: unable to get cib), ce qui a obligé à faire un diagnostic poussé pour vérifier que Corosync fonctionnait bien et à redémarrer les démons de Pacemaker pour rétablir la communication. Enfin, nous avons dû affiner la configuration des ressources en groupe, car l'adresse IP flottante seule ne suffisait pas à rendre le service Apache accessible.

VI.Conclusion

En résumé, l'objectif d'atteindre la haute disponibilité pour le service Apache est atteint. Malgré les obstacles rencontrés sur le chemin (conflits d'identité après le clonage et l'erreur de communication CIB), l'architecture basée sur Corosync et Pacemaker est désormais parfaitement opérationnelle. Les tests de migration manuelle et les simulations de panne ont confirmé que notre groupe de ressources bascule automatiquement et proprement entre Node 1 et Node 2. Nous avons donc mis en place un site web qui est résilient aux pannes et qui assure une continuité de service totale via l'IP flottante, remplissant ainsi le cahier des charges initial.