Mise en place d'un Cluster Web Haute Disponibilité avec HEARTBEAT

Savoir-faire

- **Installer et configurer des éléments d'infrastructure**
- **Déployer une solution d'infrastructure**
- **Administrer un système**
- **Administrer sur site et à distance des éléments d'une infrastructure**
- **Automatiser des tâches d'administration**
- **Tester l'intégration et l'acceptation d'une solution d'infrastructure**
- **Rédiger ou mettre à jour la documentation technique et utilisateur d'une solution d'infrastructure**

Contexte professionnel :

M. LAURENT, de la mairie de la ville des Abymes, a fait migrer le site web de la ville vers un système d'exploitation libre (opensource), et l'a hébergé sur l'un des serveurs Linux Debian Bookworm disponible au sein de l'infrastructure réseau et système actuelle. Ce projet a été mené par un Développeur de C2I Caraïbes, spécialiste dans la création d'applications métiers basées sur les technologies web traditionnelles et 3.0.

M. LAURENT désire disposer d'un serveur web de secours prêt à prendre le relais si le serveur web principal, hébergeant le site de la ville venait à ne plus être opérationnel.

Vous disposez, dans la ferme des serveurs, d'une machine virtuelle (et vous êtes en mesure d'en créer d'autres) sur laquelle se trouve le site web opérationnelle en HTTP.

La résolution des noms est prise en charge par un serveur DNS déjà configuré qui a (en interne) autorité sur la zone « ville-abymes.fr ».

Votre serveur web du site de la ville des Abymes fait partie de la zone « ville-abymes.fr », il est accessible par son nom pleinement qualifié déclaré sur le serveur DNS « www.ville-abymes.fr ».

Le site web est quant à lui accessible via l'URL : « http://www.ville-abymes.fr/index.html ».

MISSION : Vous êtes stagiaire au sein du service informatique de la ville des Abymes et vous avez pour objectif de construire, étape par étape, une solution totalement opérationnelle.

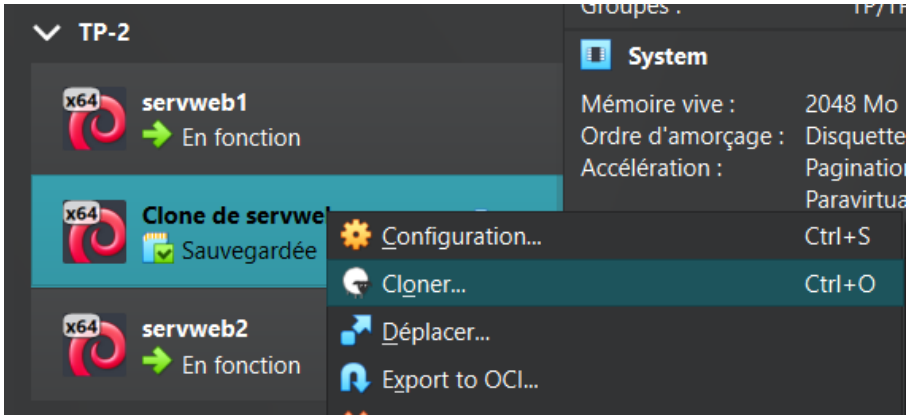
SOMMAIRE

Étape 1 : Création d'un second serveur web nommé « SRVWEB2 »

1.1 Clonage des machines virtuelles dans VirtualBox

1. Ouvrez **VirtualBox** et sélectionnez votre VM « SRVWEB ».
2. Cliquez sur **Machine** > **Cloner**.
3. Nommez le clone **srvweb_clone** et choisissez **Clonage intégral**.
4. Activez l'option « **Générer de nouvelles adresses MAC** ».
5. Une fois le clonage terminé, renommez la VM d'origine en **srvweb1**.
6. Clonez à nouveau **SRVWEB1** et nommez ce second clone **servweb2**, avec une nouvelle adresse MAC également.

À ce stade, vous avez deux serveurs web : srvweb1 et srvweb2.



Voici le serveur web cloné intégralement avec une génération de nouvelles adresse MAC et renommé afin de disposer d'un deuxième serveur web et l'original.

- À l'aide de la commande « **hostnamectl set-hostname srvweb2** » on change le nom du serveur web et pour vérifier la commande « **hostname** ».
- Pour le plan d'adressage IP ce sera « **sudo nano /etc/network/interfaces** »

auto enp0s3
iface enp0s3 inet static
address 192.168.10.15|| .10.14 (= l'interface réseau du srvweb2 et srvweb1)
netmask 255.255.255.0
gateway 192.168.10.1

- Et **redémarrer** le service réseau avec « **systemctl restart networking** » pour vérifier le plan d'adressage IP, pour vérifier on entre la commande « **ip a** ».

À faire sur les deux VM cloné (servweb 1 et srvweb2).

« srvweb1 »

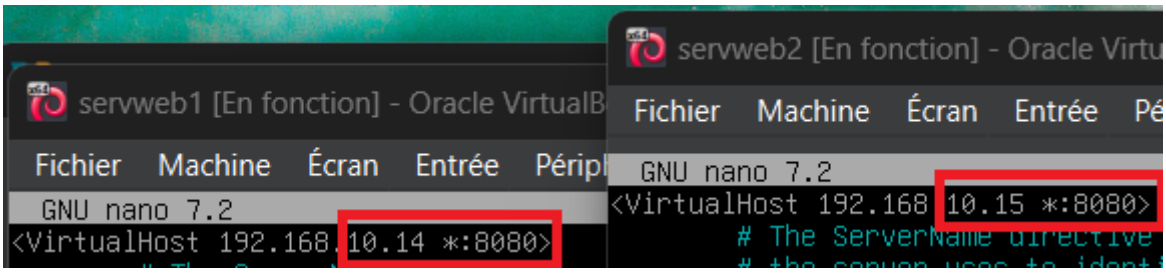
```
root@srvweb:/etc/apache2# systemctl restart networking
root@srvweb:/etc/apache2# hostname
srvweb1
root@srvweb:/etc/apache2# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d6:be:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.14/24 brd 192.168.10.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed6:be41/64 scope link
       valid_lft forever preferred_lft forever
```

« srvweb2 »

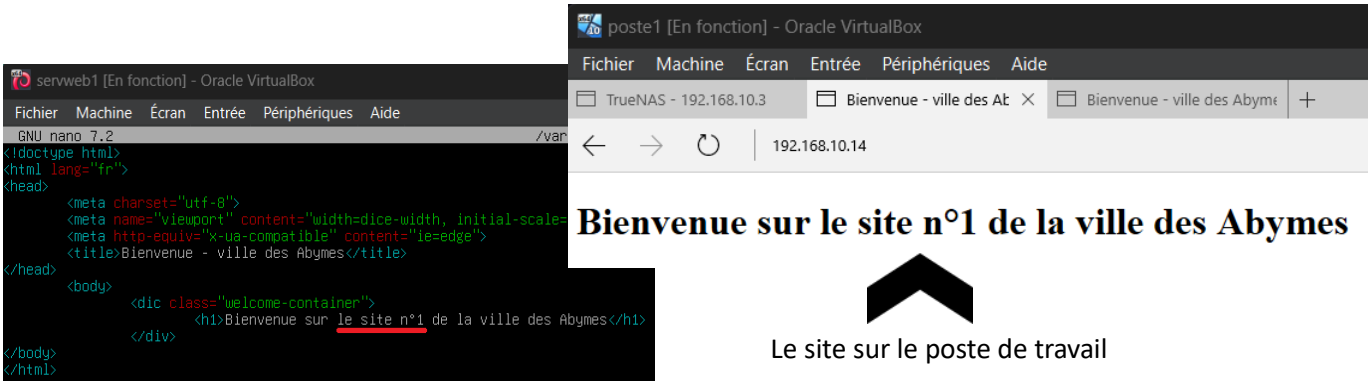
```
root@srvweb:/etc/apache2# systemctl restart networking
root@srvweb:/etc/apache2# hostname
servweb2
root@srvweb:/etc/apache2# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d6:be:41 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.15/24 brd 192.168.10.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed6:be41/64 scope link dadfailed tentative
       valid_lft forever preferred_lft forever
```

Étape 2 : Mise à jour des fichiers de configuration d'Apache 2.0 des deux serveurs web

- Dans cette étape, il s'agira de revoir les configurations du fichier « nano **/etc/apache2/sites-available/villeabymes.conf** » et modifier l'adresse IP pour chaque serveur web.

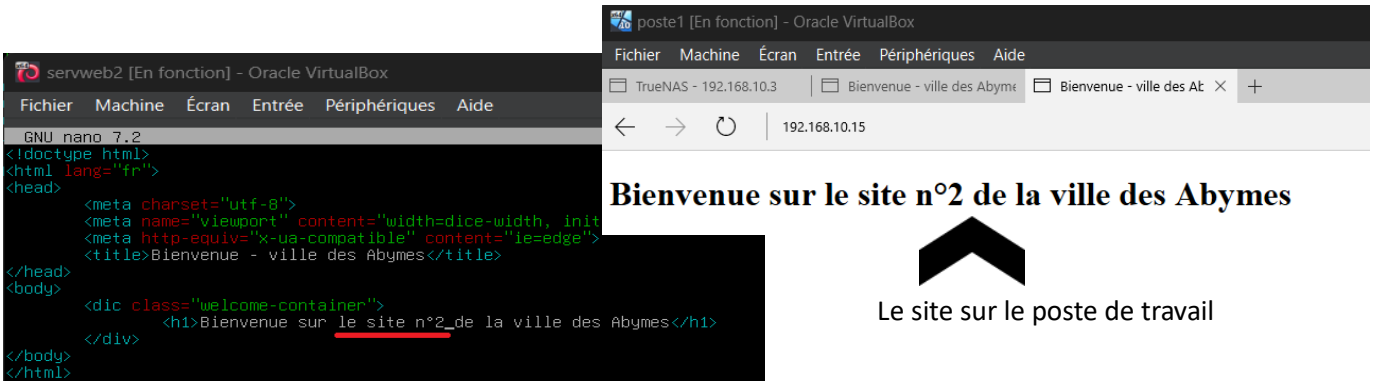


- Sur chaque serveur web, on modifie la page d'accueil avec la commande « **nano /var/www/villeabymes/index.html** » du site de la ville des Abymes, comme suit :
    - SRVWEB1 : « Bienvenue sur le site n°1 de la ville des Abymes »



Le site sur le poste de travail

- SRVWEB2 : « Bienvenue sur le site n°2 de la ville des Abymes »



Le site sur le poste de travail

Étape 3 : Création et configuration du cluster de serveurs web avec Hearbeat

Hearbeat s'appuie sur le nom d'hôte des serveurs participant au cluster afin de garantir la communication entre eux.

➢ **Configuration du fichier « /etc/hosts » :** Le fichier « **/etc/hosts** » permet de définir des correspondances entre des adresses IP et des noms d'hôtes pour garantir la communication entre les serveurs du cluster.

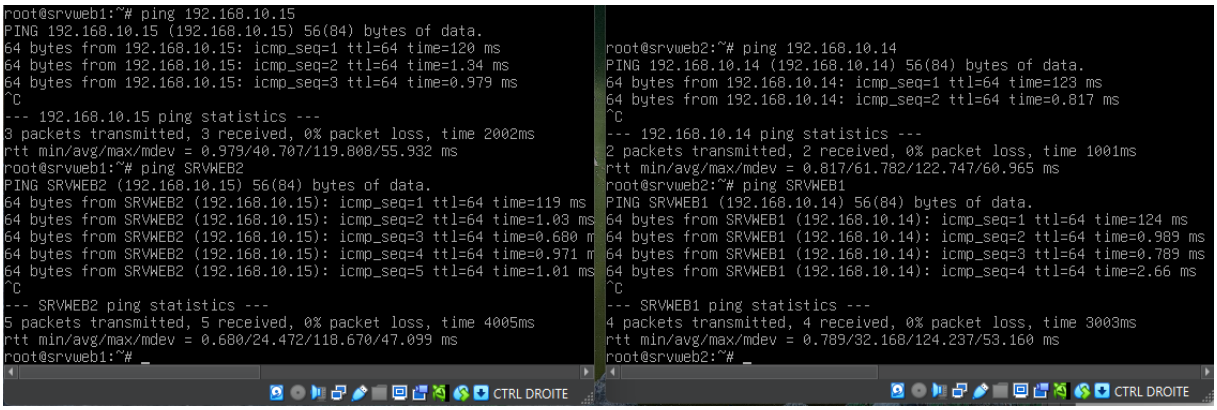
Sur la VM **srvweb1**, on accède au fichier grâce à la commande « **nano /etc/hosts** », une fois dans le fichier on modifie le nom d'hôte de l'adresse IP **127.0.1.1** qui est **Debian** en « **SRVWEB1** ». Puis on ajoute l'adresse IP et le nom d'hôte du « **srvweb2** » = **192.168.10.15 SRVWEB2.**

Ainsi on reproduit les mêmes configurations sur la VM **srvweb2** mais l'adresse IP et le nom d'hôte à modifier ajoute seront ceux du **srvweb1.**

Ce qui nous donne :

Configuration sur **srvweb1**	Configuration sur **srvweb2**
127.0.0.1 localhost 127.0.1.1 SRVWEB1 192.168.X.15 SRVWEB2	127.0.0.1 localhost 127.0.1.1 SRVWEB2 192.168.X.14 SRVWEB1

- Pour tester la connectivité on fera :
  - ping SRVWEB2 > depuis SRVWEB1
  - ping SRVWEB1 > depuis SRVWEB2



Maintenant, nous installons **Heartbeat** avec la commande « **apt install heartbeat** » mais avant, mettre à jour « **apt update** » et à niveau « **apt upgrade** » les paquets sur chaque serveur web.

A. **Configuration des fichiers de Heartbeat**

Les fichiers de configuration sont situés dans « **nano /etc/ha.d/** ».

- **ha.cf** : Configuration principale du cluster.
- **haresources** : Définition des ressources gérées par Heartbeat.
- **authkeys** : Fichier d'authentification entre les nœuds.

1. Création, édition et paramétrage du fichier « **/etc/ha.d/ha.cf** » :
   - bcast enp0s3
   - warntime 4
   - deadtime 5
   - initdead 15
   - keepalive 2
   - auto_failback on
   - node SRVWEB1
   - node SRVWEB2

<u>Explication des paramètres :</u>

- **bcast enp0sX** : Utilise le broadcast pour la communication entre les nœuds.
- **warntime 4** : Temps avant un avertissement de défaillance.
- **deadtime 5** : Temps avant de déclarer un nœud inactif.
- **initdead 15** : Délai initial pour considérer un nœud inactif.
- **keepalive 2** : Intervalle d'envoi des messages de vie.
- **auto_failback on** : Retour automatique au nœud principal.
- **node** : Liste des nœuds du cluster.

2. Création, édition et paramétrage du fichier « **/etc/ha.d/authkeys** » :
    - auth 1
    - 1 md5 macle

Explication :

- **auth 1** : Utilisation de la première méthode d'authentification.
- **1 md5 macle** : Utilisation du hash MD5 avec une clé.

A l'aide de la commande « **chmod 600 /etc/ha.d/authkeys** », changez les droits d'accès au fichier « **authkeys** » en « **600** ». La commande saisie permet de protège le fichier de toute lecture ou modification non autorisée.

3. Création, édition et paramétrage du fichier « **/etc/ha.d/haresources** » :

SRVWEB1 IPaddr::192.168.X.4/24/enp0s3:0 apache2

Explication :

- **SRVWEB1** : Nœud maître qui héberge initialement la ressource.
- **IPaddr::192.168.X.4/24/enp0sX:0** : Adresse IP flottante du cluster.
- **apache2** : Service géré par Heartbeat.

Arrêter le service : **systemctl stop heartbeat**
Démarrer le service : **systemctl start heartbeat**
Vérifier l'état du service : **systemctl status heartbeat**

srvweb1

```
root@srvweb1:~# systemctl stop heartbeat
root@srvweb1:~# systemctl restart heartbeat
root@srvweb1:~# systemctl status heartbeat
● heartbeat.service - Heartbeat High Availability Cluster Communication and Membership
     Loaded: loaded (/lib/systemd/system/heartbeat.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-17 01:01:07 CET; 9s ago
       Docs: man:heartbeat(8)
             http://www.linux-ha.org/wiki/Documentation
   Main PID: 1073 (heartbeat)
      Tasks: 4 (limit: 2315)
     Memory: 7.0M
        CPU: 45ms
     CGroup: /system.slice/heartbeat.service
             ├─1073 "heartbeat: master control process"
             ├─1076 "heartbeat: FIFO reader"
             ├─1077 "heartbeat: write: bcast enp0s3"
             └─1078 "heartbeat: read: bcast enp0s3"
```

srvweb2

```
root@srvweb2:~# systemctl stop heartbeat
root@srvweb2:~# systemctl restart heartbeat
root@srvweb2:~# systemctl status heartbeat
● heartbeat.service - Heartbeat High Availability Cluster Communication and Membership
     Loaded: loaded (/lib/systemd/system/heartbeat.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-17 01:05:02 CET; 3s ago
       Docs: man:heartbeat(8)
             http://www.linux-ha.org/wiki/Documentation
   Main PID: 29146 (heartbeat)
      Tasks: 4 (limit: 2315)
     Memory: 7.0M
        CPU: 41ms
     CGroup: /system.slice/heartbeat.service
             ├─29146 "heartbeat: master control process"
             ├─29149 "heartbeat: FIFO reader"
             ├─29150 "heartbeat: write: bcast enp0s3"
             └─29151 "heartbeat: read: bcast enp0s3"
```

Étape 4 : Testez la haute disponibilité du site web de la ville des Abymes

À l'aide de la commande « **ping** », testez la communication réseau entre les postes de travail et chaque serveur web :

- **$ping 192.168.X.14**
- **$ping 192.168.X.15**
- **$ping 192.168.X.4**

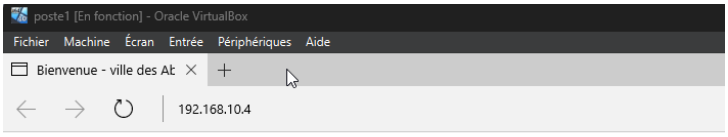
```
C:\Users\jdupont>ping 192.168.10.4

Envoi d'une requête 'Ping'  192.168.10.4 avec 32 octets de données :
Réponse de 192.168.10.4 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.10.4 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.10.4 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.4 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.10.4:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 1ms, Moyenne = 0ms

C:\Users\jdupont>ping 192.168.10.14

Envoi d'une requête 'Ping'  192.168.10.14 avec 32 octets de données :
Réponse de 192.168.10.14 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.14 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.14 : octets=32 temps=3 ms TTL=64
Réponse de 192.168.10.14 : octets=32 temps=1 ms TTL=64

Statistiques Ping pour 192.168.10.14:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 3ms, Moyenne = 1ms

C:\Users\jdupont>ping 192.168.10.15

Envoi d'une requête 'Ping'  192.168.10.15 avec 32 octets de données :
Réponse de 192.168.10.15 : octets=32 temps=1 ms TTL=64
Réponse de 192.168.10.15 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.15 : octets=32 temps<1ms TTL=64
Réponse de 192.168.10.15 : octets=32 temps=1 ms TTL=64

Statistiques Ping pour 192.168.10.15:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 1ms, Moyenne = 0ms
```

Que constatez-vous suite aux résultats de chaque commande « **ping** » ? Qu'en déduisez-vous ?

On constate que les srvweb1 et srvweb1 sont accessible grâce au réponse reçues donc il fonctionne correctement. Pour le service Heartbeat on voit qu'il est joignable, ce qui signifie qu'il est actif donc la haute disponibilité est opérationnelle.

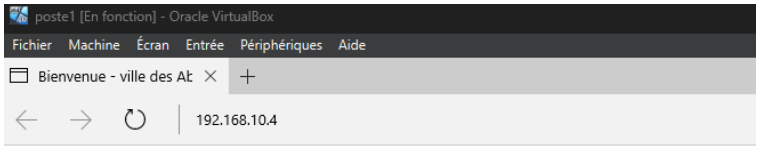
Maintenant, pour tester la « haute disponibilité » du site web de la ville des Abymes depuis l'un des postes de travail nous proposons un scénario.

- Sur le **poste de travail** le « site n°1 » est accessible en **192.168.10.4**.



Bienvenue sur le site n°1 de la ville des Abymes

•Pour la suite nous allons arrêter le **srvweb1** et en retournant sur le **poste de travail** nous verrons que le « site n°2 » a pris le relais toujours en **192.168.10.4**.



Bienvenue sur le site n°2 de la ville des Abymes

Conclusion

Nous avons mis en place une infrastructure de **haute disponibilité** pour le site web de la ville des Abymes en utilisant **Heartbeat** sur deux serveurs web sous Debian.
Grâce à la configuration du cluster, nous assurons la continuité de service : en cas de défaillance d'un serveur, le second prend automatiquement le relais sans interruption visible pour l'utilisateur.
Cette solution simple mais efficace garantit une meilleure fiabilité et une meilleure accessibilité du site web, ce qui est essentiel pour des services en ligne critiques.