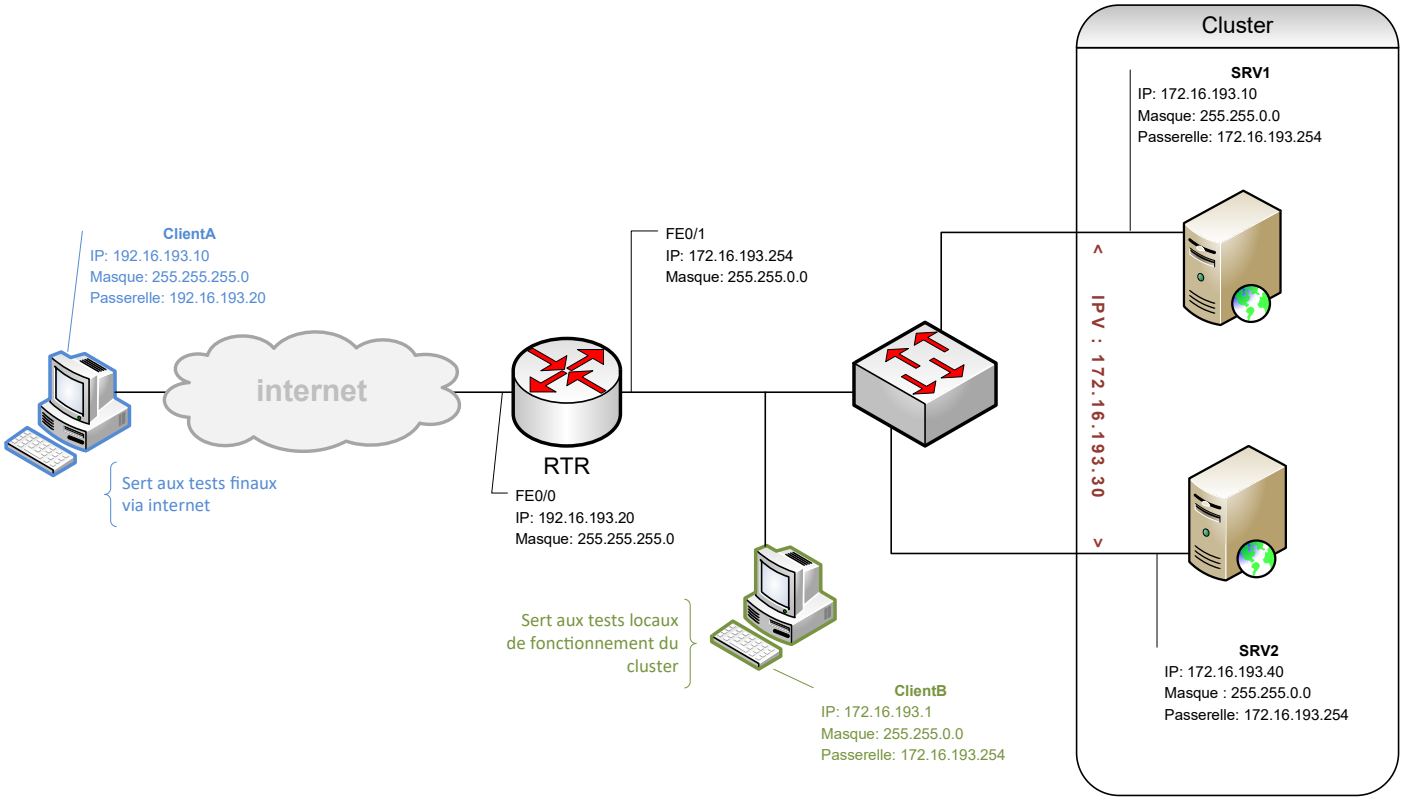
L'objectif de ce TP est de mettre en œuvre les mécanismes de haute disponibilité et de répartition des charges entre serveurs. Vous pourrez ainsi :

- Comprendre l'intérêt et la complémentarité de la haute disponibilité et de la répartition des charges ;
- Comprendre les mécanismes mis en œuvre pour le fonctionnement de ces mécanismes ;
- Comprendre les concepts utilisés au travers de ces mécanismes ;
- Savoir mettre en œuvre les mécanismes de la haute disponibilité et de répartition des charges, séparément ou conjointement ;
- Savoir lister les avantages et les limites des concepts de haute disponibilité et de répartition des charges.

Ce TP utilise la solution LVS (Linux Virtual Server) qui comprend -notamment- les logiciels ipvs, Heartbeat, Keepalived, ... Le site http://www.linuxvirtualserver.org recense tous les logiciels liés à la haute disponibilité et la répartition des charges entre serveurs.

Partie 1 – Mise en place d'une solution de haute disponibilité avec Heartbeat

Architecture du réseau à mettre en place



Cette architecture sera mise en place en deux temps :

- d'abord le cluster et la haute disponibilité ;
- ensuite le routeur, la redirection d'adresse et de port et l'accès via internet.

Création des serveurs

Le réseau nécessite deux serveurs web sous Linux Debian 9. Ces serveurs web doivent être créés sous Esxi uniquement car Virtualbox ne supporte pas le type d'architecture que nous allons mettre en place.

Nous allons procéder en deux temps : d'abord créer un serveur modèle sous Debian, puis le dupliquer pour créer nos serveurs web.

- Créez un serveur virtuel **modeleD9** sous Esxi en configuration personnalisée : basé sur Linux Debian 9/Stretch 32bits, avec 1Go de mémoire vive, 20Go de disque dur alloué selon le besoin.
- Installez Debian 9 sur le serveur virtuel avec le cd-rom de la machine physique, sans oublier de le connecter au démarrage : utilisez la passerelle et le DNS 172.16.253.253 durant l'installation. Définissez les mots de passe usuels. Nommez votre serveur **masterGr** suivi de votre numéro de groupe.
- A la fin de l'installation, en mode console, mettez à jour la liste des paquets, avec la commande :
  `apt-get update`
- Eteignez votre machine modèle.
- Retirez le démarrage du cd-rom automatique.
- Dans le **datastore**, vous allez créer à partir de la racine **/** deux dossiers **srvweb1** et **srvweb2**, puis copier/coller à partir du répertoire **modeleD9** le (gros) fichier **.vmdk** (disque dur du serveur modèle) vers **srvweb1** puis vers **srvweb2**.
- Sortez du **datastore**.
- Créez les deux serveurs virtuels **srvweb1** puis **srvweb2** en leur donnant les mêmes spécifications techniques que le serveur modèle. Dans l'écran du choix de la taille du disque dur, après avoir mis 20Go, spécifiez d'utiliser un disque à partir du **datastore** et pointez sur le répertoire où se trouve le disque (**srvweb1** ou **srvweb2**) puis sur le disque .**vmdk**.
- Allumez les deux serveurs web et attribuez-leur une adresse IP différente pour qu'ils puissent accéder à internet. ☚ **ne mettez pas encore l'adresse du schéma.**
- Sur les deux serveurs web, installez le service **apache2** :
  `apt-get install apache2`
- Adaptez l'affichage de la page d'accueil de chacun des serveurs web en éditant la page par défaut et en écrivant juste après « *It Works !* » le nom réel du serveur **SRV1** ou **SRV2** :
  `nano /var/www/html/index.html`
- Vérifiez le bon fonctionnement du service web en saisissant **localhost** dans l'URL du navigateur de chaque serveur.
- Renommez les serveurs avec leurs noms respectifs **SRV1** ou **SRV2** :
  `nano /etc/hostname`

Le nom ne sera pris en compte que quand nous redémarrerons les machines, un peu plus tard.

Dorénavant les deux serveurs seront désignés comme étant **SRV1** ou **SRV2**, pour ne pas confondre avec les noms des machines virtuelles créées sous Esxi.

- Téléchargez et installez le package **heartbeat** sur chaque serveur web :
  `apt-get install heartbeat`

› Pour éviter tout conflit d'adresse IP et pour respecter l'architecture du schéma, isolez les deux serveurs en déconnectant votre commutateur du réseau du lycée.

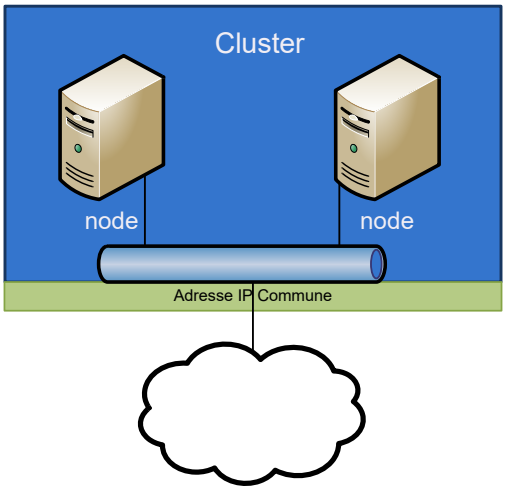
Les deux serveurs sont donc reliés au commutateur via la machine physique. Le commutateur sera relié plus tard, au routeur.

› Attribuez l'adresse IP définitive aux deux serveurs avec la passerelle du schéma.
› Redémarrez les deux serveurs pour que leurs noms soient pris en compte et pour vérifier la bonne configuration IP.

Configuration du cluster

Les deux serveurs sont à présents fonctionnels. Ils fonctionnent de façon autonome, nous allons les mettre en cluster.
L'idée générale pour assurer la disponibilité d'un service est de faire fonctionner plusieurs machines (deux au minimum) en même temps. Ces machines forment ce qu'on appelle un *cluster* et chaque machine est un *node* du cluster. Chacune des machines va vérifier si les autres répondent toujours en prenant le *pouls* de chacune des autres. Si une machine cesse de fonctionner, les autres assurent le service.

Une fois le cluster configuré, on y accède au travers d'une seule et unique adresse IP qui est celle du cluster ; qui lui-même est composé de plusieurs *nodes*.



Pour commencer, nous allons déclarer le nom et l'adresse IP de l'autre serveur du cluster dans le fichier de résolution locale de chaque serveur.

› Editez le fichier de résolution local de **SRV1** :
```
nano /etc/hosts
```
› Au besoin adaptez la ligne 127.0.1.1, puis ajoutez au-dessous le nom et l'adresse de **SRV2** :
```
127.0.0.1  localhost
127.0.1.1  SRV1
172.16.193.40    SRV2
```

› Vérifiez la bonne résolution des noms en faisant un ping sur **SRV1** puis **SRV2** :

```
ping SRV1
ping SRV2
```

› Editez le fichier de résolution local de **SRV2** :

```
nano /etc/hosts
```

› Au besoin adaptez la ligne 127.0.1.1, puis ajoutez au-dessous le nom et l'adresse de **SRV1** :

```
127.0.0.1  localhost
127.0.1.1  SRV2
172.16.193.10    SRV1
```

› Vérifiez la bonne résolution des noms en faisant un ping sur **SRV1** puis **SRV2** :

```
ping SRV1
ping SRV2
```

Dorénavant, on peut désigner nos serveurs en utilisant leurs noms à la place de leurs adresses IP.

La configuration basique d'HeartBeat repose sur trois fichiers fondamentaux. Les trois fichiers de configuration sont strictement identiques sur les deux serveurs. Il convient donc de les copier sur chacune des machines du cluster. Dans notre exemple, il s'agit de **SRV1** et **SRV2**.

› Nous allons créer, sur **SRV1**, le fichier de configuration d'Heartbeat :

```
nano /etc/ha.d/ha.cf
```

› Complétez le fichier, avec les informations suivantes :

```
bcast          enp0s3          ☚ c'est normalement la carte que vous avez configurée.
debugfile      /var/log/ha-debug
logfile        /var/log/ha-log
logfacility    local0
keepalive      2
deadtime       10
warntime       6
initdead       60
udpport        694
node           SRV1
node           SRV2
auto_failback off
```

Examinons ce fichier de configuration en détails (les durées sont en seconde par défaut, mais peuvent être mises en millisecondes en ajoutant ms) :

Elément	Explications
`bcast`	indique l'interface réseau par laquelle on va effectuer la prise de pouls.
`debugfile`	indique le fichier de débogage à utiliser.
`logfile`	indique le log d'activité à utiliser (à consulter en cas d'erreurs).
`logfacility`	indique que l'on utilise la facilité syslog en plus.
`keepalive`	indique le délai entre deux battements de pouls en secondes.
`deadtime`	indique le temps nécessaire avant de considérer un nœud comme étant mort.
`warntime`	indique le délai avant d'envoyer un avertissement pour les pouls en retard.
`initdead`	indique un deadtime spécifique pour les configurations où le réseau met un certain temps à démarrer. initdead est normalement deux fois plus grand que deadtime (au minimum).
`udpport`	indique le port à utiliser pour la prise de pouls.
`node`	renseigne le nom des machines faisant partie du cluster. Ce nom doit être identique à celui retourné par la commande **hostname**.
`auto_failback`	indique le comportement à adopter si le *node* maître revient dans le cluster.

	Si on met la valeur on, lorsque le *node* maître revient dans le cluster, tout est transféré sur ce dernier. Si on met la valeur off, les services continuent à tourner sur l'esclave même lorsque le maître revient dans le cluster. La valeur off permet de faire un retour à la normale manuellement lorsque la charge de production est moins importante

Maintenant, nous allons définir le *node* maître, l'adresse IP du cluster et les services devant être assurés. Dans un premier temps (configuration basique oblige), nous allons uniquement mettre en place l'adresse IP du cluster.

› Toujours sur **SRV1**, créez le fichier de configuration des ressources d'Heartbeat :
`nano /etc/ha.d/haresources`
› Complétez le fichier, avec les informations suivantes :
`SRV1 IPaddr::172.16.193.30/16/enp0s3`

Par la suite, nous pourrons indiquer également quel service est associé à l'IPV du cluster avec :
`SRV1 IPaddr::172.16.193.30 apache2`

On peut partager plusieurs adresses IPV au sein d'un cluster avec, par exemple :
`SRV1 IPaddr::172.16.193.30/16/enp0s3 IPaddr:: 172.16.193.31/16/enp0s3`

Nous allons à présent gérer l'authentification mutuelle des *nodes*. Cela leur permet de se reconnaître entre eux et éventuellement d'éviter l'immiscion d'un serveur pirate.

› Toujours sur **SRV1**, créez le fichier de configuration de l'authentification :
```
nano /etc/ha.d/authkeys
```
› Complétez le fichier, avec les informations suivantes :
```
auth 1
1 md5 "phrase secrete du cluster"
2 crc
```
› Protégez le fichier en réservant l'accès au seul utilisateur *root*, avec la commande :
```
chmod 600 /etc/ha.d/authkeys
```

Le mot clé `auth` indique quel est le système d'authentification que l'on va utiliser. Si le lien n'est pas sûr physiquement, il est nécessaire d'utiliser un chiffrement *md5* avec une *chaîne clé* comme nous avons choisi avec `auth 1` (1 étant le choix de md5). L'autre méthode, avec *CRC* (le choix numéro 2) n'est pas sécurisée. On peut également utiliser *SHA-1* (option : `3 sh1 "autre phrase secrete du cluster"`). L'ordre des options n'est pas important.

› Configurez le second *node* **SRV2** en **créant exactement les mêmes fichiers et les mêmes contenus**.

Les deux serveurs sont maintenant prêts. Nous allons redémarrer les services Heartbeat, dans l'ordre.

› Sur **SRV1** (le serveur primaire), relancez Heartbeat avec la commande :
```
service heartbeat stop
service heartbeat start
```
› Sur **SRV2** (le serveur secondaire), relancez Heartbeat avec la commande :
```
service heartbeat stop
service heartbeat start
```
› Sur **les deux serveurs** vérifiez l'éventuelle présence d'une carte virtuelle enp0s3:0.

Test du cluster

Dans un premier temps, nous allons tester localement le fonctionnement du cluster avec une station (**ClientB**) reliée au commutateur auquel sont raccordés les deux serveurs.

› Branchez une station (**ClientB**) au commutateur, lancez une machine virtuelle (peu importe le système d'exploitation).
› Configurez l'adresse IP et la passerelle de **ClientB** comme indiqué sur le schéma.
› Testez le fonctionnement du cluster avec un ping sur l'adresse IP du cluster.
› Testez le fonctionnement du service avec le navigateur et en saisissant l'adresse IP du cluster comme URL.

Vous devriez voir la page web du serveur primaire **SRV1**.

› Déclenchez une panne en éteignant la carte réseau du serveur primaire **SRV1** avec :
```
ifdown enp0s3
```
› Au bout d'une vingtaine de secondes, actualisez la page de votre navigateur.

Vous devriez voir la page web du serveur secondaire **SRV2**.

› Rallumez la carte du serveur primaire et voyez s'il reprend du service en actualisant la page web du navigateur au bout d'une vingtaine de secondes.

Il se peut que le serveur **SRV2** reste toujours actif. Il faudra peut-être éteindre les services Heartbeat et les relancer dans l'ordre.

Afin d'améliorer la reprise on pourra améliorer la directive dans `/etc/ha.d/haresources` sur chaque serveur avec :

```
SRV1     IPaddr::172.16.193.30      apache2
```

Il faudra également changer la directive dans `/etc/ha.d/ha.cf` sur chaque serveur avec :

```
auto_failback on
```

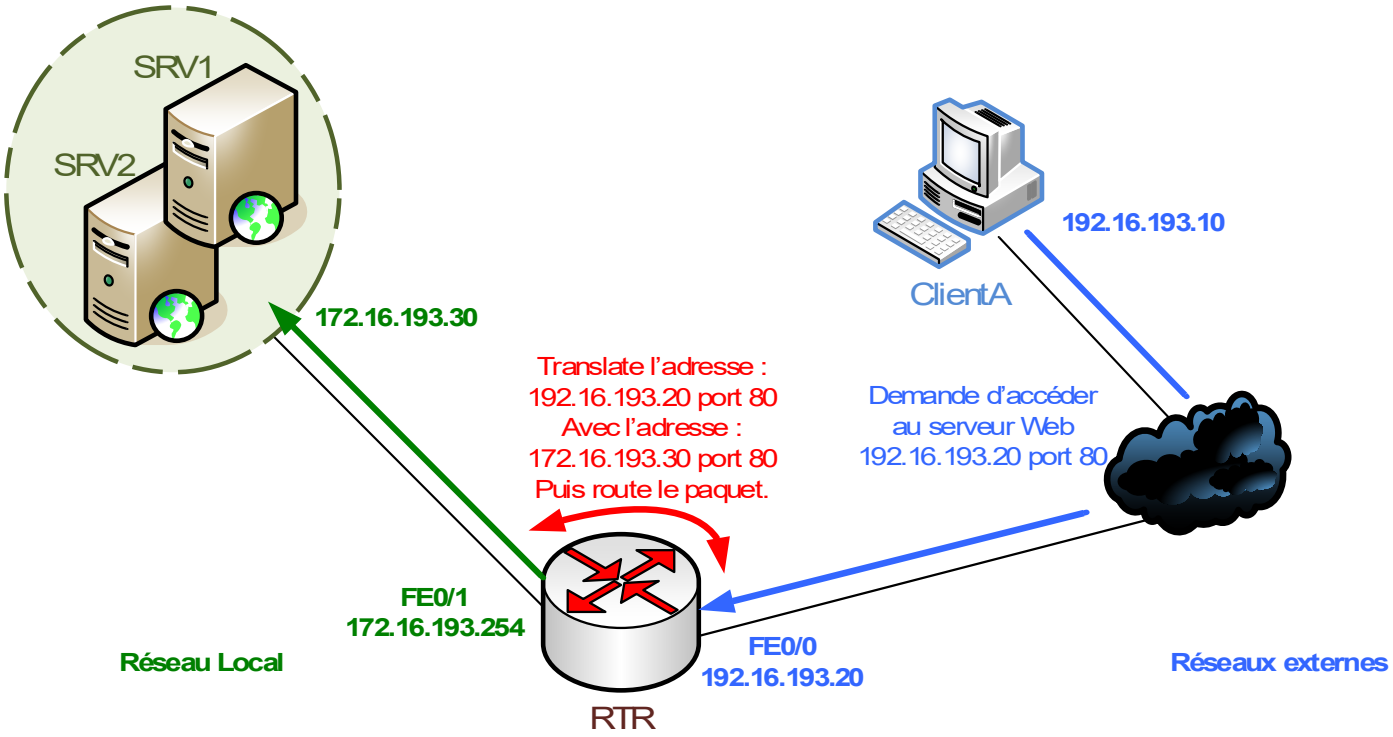
Mise en place d'un routeur avec redirection de port

Cette solution de cluster n'est jamais accessible directement via internet. Les serveurs sont généralement placés dans une DMZ ou un réseau dédié, derrière un (ou plusieurs) routeur(s).
Les adresses, souvent de classe privée, utilisées par le cluster ne sont pas accessibles d'internet, c'est l'adresse IP publique externe qui est sollicitée. Le routeur va donc assumer la charge des requêtes web provenant des internautes et les rediriger vers le serveur web. C'est de la redirection d'adresse (et de port).

←Translation→

Cluster	Routeur
IP : 172.16.193.30	IP : 192.16.193.20
Port : 80	Port : 80

Chaque demande que reçoit le routeur, il la redirige vers le cluster.



- Reliez le routeur au commutateur via son interface FastEthernet0/1 et à la station **ClientA** (station physique qui doit être différente de celle de **ClientB**, avec n'importe quelle machine virtuelle) via son interface *FastEthernet0/0*.
- Configurez l'adresse IP et la passerelle de **ClientA** comme indiqué sur le schéma.
- Allumez le routeur et entrez en mode de configuration globale.
- Configurez et activez la carte *FastEthernet0/0* du routeur avec l'adresse IP 192.16.193.20 et son masque.
- Configurez et activez la carte *FastEthernet0/1* du routeur avec l'adresse IP 172.16.193.254 et son masque.
- Activez le routage en mode de configuration globale.
  `ip routing`
- Vérifiez la table de routage en mode `enable`.
  `show ip route`
- Vérifiez la connectivité avec la commande ping, entre **ClientA** et **ClientB**.
- Configurez la redirection d'adresse et de port en mode de configuration globale avec la commande :
  `ip nat inside source static tcp 172.16.193.30 80 interface FastEthernet 0/0 80`
- Activez la redirection en sortie sur la carte *FastEthernet0/0* avec la commande :
  `ip nat outside`
- Activez la redirection en entrée sur la carte *FastEthernet0/1* avec la commande :
  `ip nat inside`

Tests de la solution finalisée

› A partir de **ClientA**, testez le fonctionnement du service avec le navigateur et en saisissant l'adresse IP externe du routeur comme URL.

Vous devriez voir la page web du serveur primaire **SRV1**.

› Déclenchez une panne en éteignant la carte réseau du serveur primaire **SRV1** avec :
`ifdown enp0s3`

On peut aussi déclencher une panne en arrêtant le service **apache2** sur **SRV1**.

› Au bout d'une vingtaine de secondes, actualisez la page de votre navigateur.

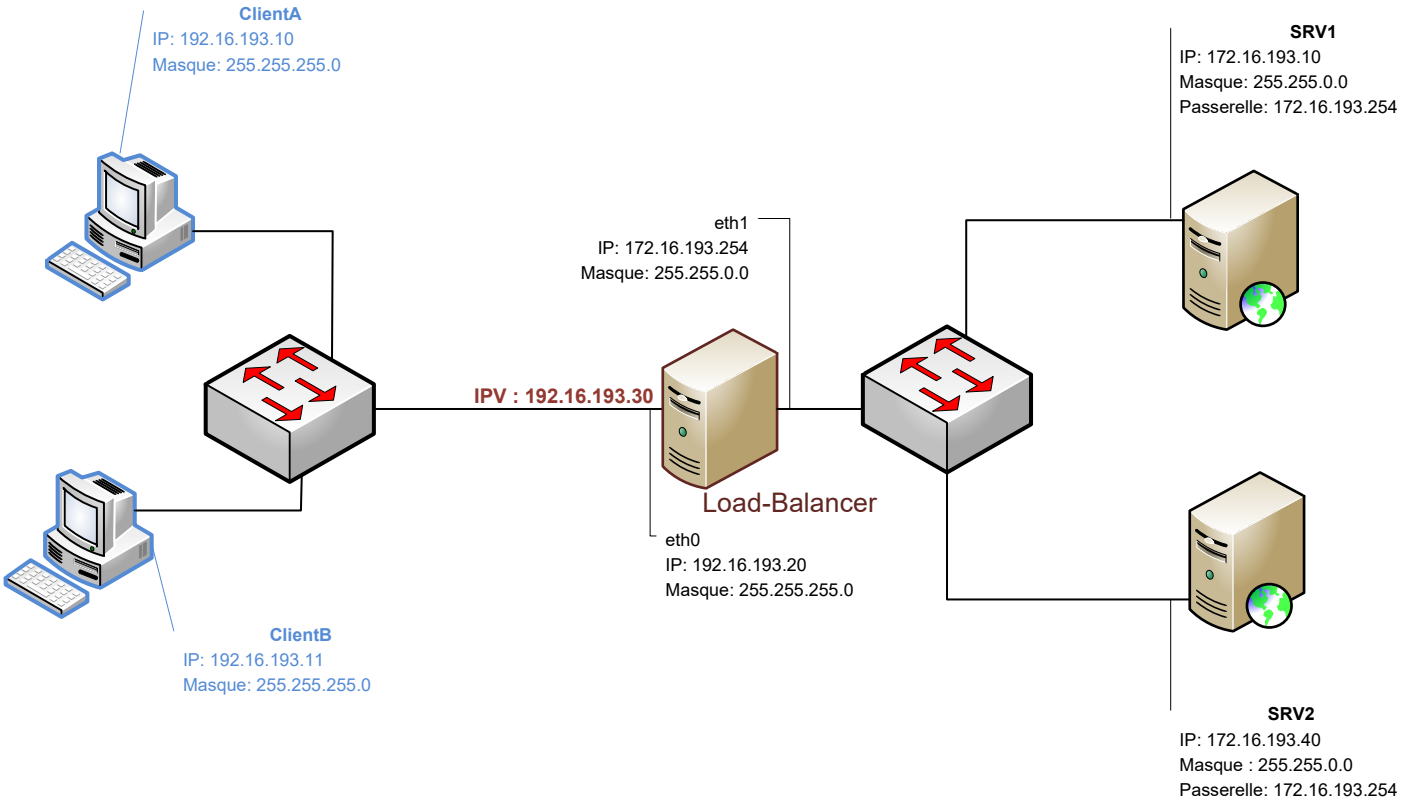
Vous devriez voir la page web du serveur secondaire **SRV2**.

› Rallumez la carte du serveur primaire et voyez s'il reprend du service en actualisant la page web du navigateur au bout d'une vingtaine de secondes.

Partie 2 - Mise en place d'une solution de répartition de service avec IPVS

> Dans cette partie, nous resterons au niveau local. Nous allons utiliser la configuration mise en place précédemment pour des raisons de simplicité.
> Nous utiliserons un routeur logiciel sous Linux comme répartiteur.

Architecture du réseau à mettre en place



Le commutateur situé entre le répartiteur (**Load-Balancer**) et les deux serveurs web est intégré dans Esxi : vous pourrez observer, dans la section « réseau » d'Esxi, un commutateur virtuel.

Création des serveurs

> Les deux serveurs ont une configuration identique que dans la partie numéro 1. Vous pouvez les reprendre en prenant soin d'arrêter Heartbeat avec : `service heartbeat stop`. En ce cas vous pouvez passer à l'étape suivante.

Ou…

> Vous pouvez copier/coller votre modèle sous Esxi et configurer à nouveau deux serveurs.

- Créer deux serveurs sous Linux Debian 9.
- Téléchargez et installez Apache 2.
- Adaptez l'affichage de la page d'accueil de chacun des serveurs web en éditant la page par défaut et en écrivant juste après « It Works ! » le nom réel du serveur **SRV1** ou **SRV2**.

- Vérifiez le bon fonctionnement du service web en saisissant **localhost** dans l'URL du navigateur de chaque serveur.
- Renommez les serveurs avec leur nom respectif **SRV1** ou **SRV2**.
- Pour éviter tout conflit d'adresse IP et pour respecter l'architecture du schéma, isolez les deux serveurs en déconnectant votre commutateur du réseau du lycée.
- Attribuez l'adresse IP définitive aux deux serveurs avec la passerelle du schéma.
- Redémarrez les deux serveurs pour que leurs noms soient pris en compte et pour vérifier la bonne configuration IP.

Création du répartiteur (Load-Balancer)

- Eteignez vos deux serveurs web temporairement pour éviter les potentiels conflits d'adresse IP.
- Connecter le commutateur au réseau du lycée pour accéder à internet (nécessaire pour installer le service de répartition des charges).
- Sous Esxi, copier le modèle **modeleD9** dans un nouveau dossier **repartiteur1**.
- Créez le serveur virtuel **repartiteur1** en lui donnant les mêmes spécifications techniques que le serveur modèle. Dans l'écran du choix de la taille du disque dur, après avoir mis 20Go, spécifiez sur le **datastore** le répertoire où se trouve le disque **repartiteur1** puis le disque .**vmdk**.
- Ajoutez une seconde carte à votre répartiteur.
- Allumez le répartiteur et attribuez-lui une adresse IP différente pour qu'il puisse accéder à internet.
  ☚ ne mettez pas encore l'adresse du schéma.
- Téléchargez et installez le package IPVS sur le répartiteur :
  `apt-get install ipvsadm`
- Pour éviter tout conflit d'adresse IP et pour respecter l'architecture du schéma, isolez les deux serveurs et le répartiteur en déconnectant votre commutateur du réseau du lycée.
- Redémarrez les deux serveurs web.
- Attribuez les deux adresses IP définitives aux deux cartes du répartiteur selon les indications du schéma.
- Renommez le serveur avec son nom **Load-Balancer** :
  `nano /etc/hostname`
- Redémarrez le serveur pour que son nom soit pris en compte et pour vérifier la bonne configuration IP.

› Editez le fichier de configuration de IPVS avec la commande :
`nano /etc/default/ipvsadm`
› Mettez `true` pour charger les règles IPVS au démarrage.
› Mettez `none` car notre serveur est autonome (ni master, ni slave).

Le fichier `/etc/default/ipvsadm` permet une configuration du service IPVS mais il faudra le compléter avec les « règles » de répartition de charge.

› Créez une carte réseau virtuelle avec la commande :
`nano /etc/network/interfaces`
› Saisissez les paramètres de la carte réseau virtuelle (appelée eth0:30) :
```
auto enp0s3:30
iface enp0s3:30 inet static
address 192.16.193.30
netmask 255.255.255.255
```
› Démarrez la carte réseau virtuelle avec la commande :
`ifup enp0s3:30`
› Activez le routage avec la commande :
`echo 1 > /proc/sys/net/ipv4/ip_forward`

Cette activation peut se faire au démarrage du serveur en dé-commentant la ligne `net.ipv4.ip_forward=1` dans le fichier : `/etc/sysctl.conf`

› Paramétrez le cluster avec les commandes :
```
ipvsadm --add-service -t 192.16.193.30:http -s rr
ipvsadm -a -t 192.16.193.30:http -r 172.16.193.10:http -m -w 1
ipvsadm -a -t 192.16.193.30:http -r 172.16.193.40:http -m -w 1
```

Ces paramètres peuvent être placés dans le fichier : `/etc/sysctl.conf`
En ajoutant les lignes :
```
# ipvsadm.rules
-A -t 192.16.193.30:http -s rr
-a -t 192.16.193.30:http -r 172.16.193.10:http -m -w 1
-a -t 192.16.193.30:http -r 172.16.193.40:http -m -w 1
```

L'option `-m` sert à faire du *masquerading* pour que le réseau derrière le répartiteur soit masqué des réseaux extérieurs (NAT).
L'option `-w` fixe le poids (*weight*) de chacun des serveurs réels (ici leur poids est identique), si le poids est différent il faudra remplacer le *Round Robin* (rr) aléatoire par un *Weighted Round Robin* (wrr). Quand les serveurs réels sont dans le même réseau que le répartiteur, le routage est désactivé et on remplace l'option `-m` par l'option `-g`.

Les méthodes de répartition sont donc :

- **rr** pour la méthode *Round Robin* purement aléatoire ;
- **wrr** pour la méthode *Weighted Round Robin* ;
- **lc** pour la méthode *Least-Connection* consistant à regarder préalablement le nombre de connexions établies sur le serveur ;
- **wlc** pour la méthode *Weighted Leats-Connection* prend en compte le poids attribué aux serveurs.

On peut effacer tous les paramètres avec : `ipvsadm --clear`

On peut effacer une règle avec, par exemple :

```
ipvsadm -d 192.16.193.30:http -r 172.16.193.10:http
```

› Vérifiez le paramétrage avec la commande :
`ipvsadm -L`

› Vérifiez le paramétrage sans résolution des noms avec la commande (plus rapide) :
`ipvsadm -Ln`

› Configurez l'adresse IP et la passerelle de **ClientA** comme indiqué sur le schéma.

Test du répartiteur de charge

› Sur le répartiteur, surveillez la répartition des charges entre les deux serveurs avec la commande suivante :

```
watch -n 1 'ipvsadm -ln'
```

› A partir de **ClientA**, testez le fonctionnement du service avec le navigateur et en saisissant l'adresse IPV du répartiteur comme URL.

Vous devriez voir la page web du serveur primaire **SRV1** ou secondaire **SRV2**.

› A partir de **ClientB**, testez le fonctionnement du service avec le navigateur et en saisissant l'adresse IPV du répartiteur comme URL.

Vous devriez voir la page web du serveur primaire **SRV1** ou secondaire **SRV2**.

› Sur le répartiteur, la surveillance de répartition des charges vous affiche les connexions actives ou inactives.

Si vous avez d'autres navigateurs installés sur les stations vous pourrez vérifier le caractère aléatoire de la répartition des charges.