Laboratoire Redundancy (90 min)

§0	Introduction	`sudo ./c 2`

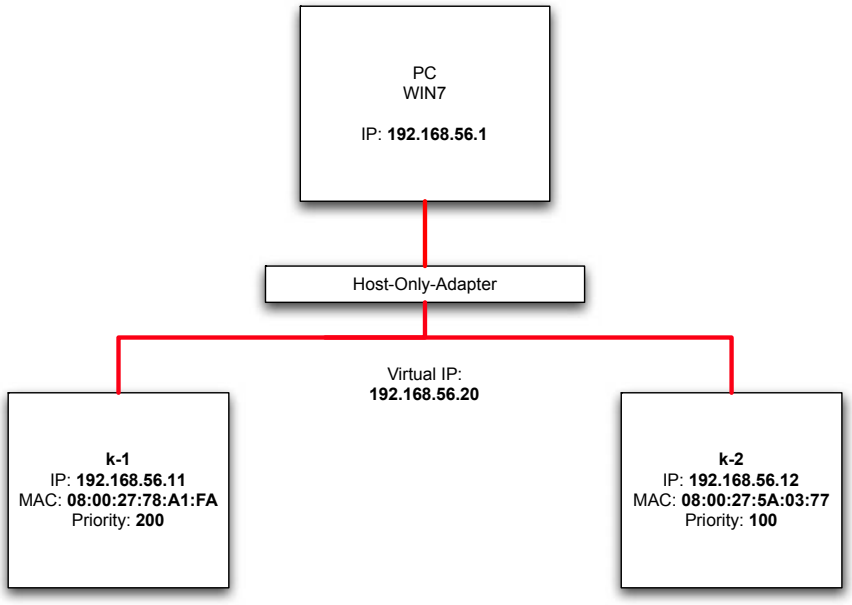
Cadre Les machines virtuelles utilisées sont basées sur la distribution CentOS 6.4 CLI

Session Ouvrir une **session Windows 7** administrateur : compte=**albert** password=**admin**

Action Copier sur le bureau le dossier **`\\10.2.1.1\doclabo\RSX\5_Redundancy`** contenant les fichiers utiles

§1	Disponibilité avec KeepAlived	30 min

Objectif Tester la tolérance à la panne d'un des nœuds k-1 et k-2

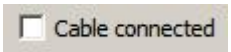


a) Préciser toutes les étapes dans un ordre logique
Penser à effectuer des tests unitaires avec ping, …
Désactiver le serveur DHCP intégré à Vbox → **`File – Preference – Network`**
Fichiers **`k-1.ova`** et **`k-2.ova`** à disposition
Utiliser **`/etc/keepalived/keepalived.conf`** pour connaître la config

Démarrer le nœud le moins prioritaire avec **`service keepalived start`**
Analyser les logs avec **`tail –f –n 50 /var/log/messages | grep keepalived`**
Utiliser **`ip addr show`** pour afficher l'adresse ip virtuelle

Démarrer l'autre nœud
Analyser les logs
Contrôler l'adr IP

b) Etudier le basculement en effectuant **`ping –i 0.3 xxx`** depuis generator.ova
Déterminer le nombre de paquets perdus suite à une déconnexion du réseau de k-1 depuis Vbox



Ne pas oublier de valider avec OK

c) Etudier l'acquisition vrrp faite avec Wireshark dans le scénario suivant :

- k-1 est Master, k-2 est slave
- Déconnexion de k-1 après 4 s

c1 A quelle fréquence le nœud Master envoit-il le paquet VRRP ?

c2 Quelles sont les 2 principales différences entre les paquets 4 et 5 ?

c3 Quel type d'adresses Eth et IP est utilisé par ces paquets VRRP ?

c4 A quoi sert le paquet 6 ?

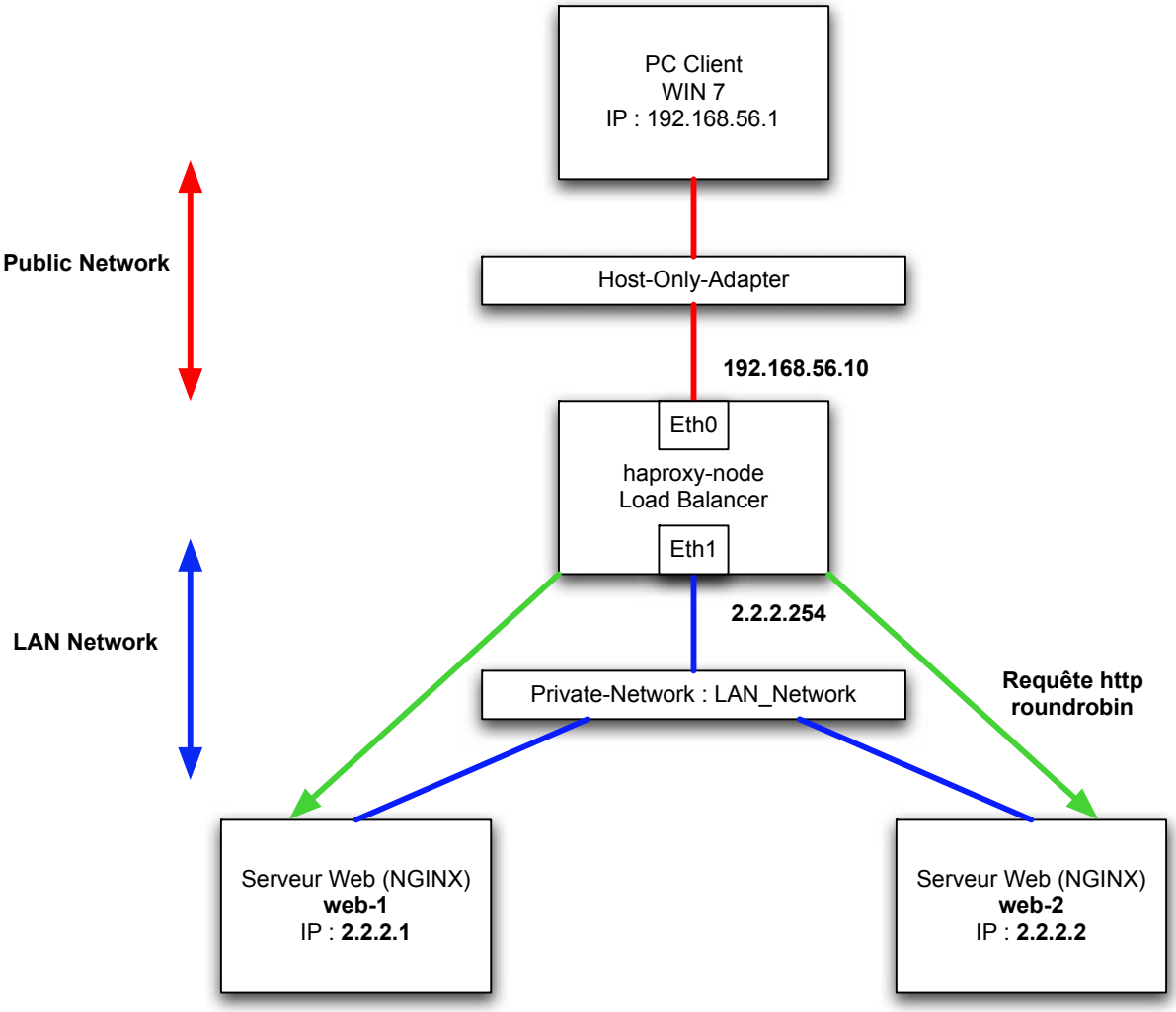
c5 Quelle est la condition pour qu'un nœud slave devienne master ?

c6 Peut-on facilement ajouter un nœud supplémentaire ?

c7 Comment contrer l'attaque consistant à activer un nœud malveillant ?

§2	**Répartition de charge avec HAProxy**	**40 min**

Objectif Etudier la répartition de charge entre 2 serveurs web NGINX
NGINX v1.6.2 : http://fr.slideshare.net/baohx2000/nginx-14817175



a) Répondre aux questions à partir du fichier de config. **`/etc/haproxy/haproxy.cfg`** copié dans le partage

a1 Quelle est la requête http coté client ?

a2 Quel serveur va répondre ?

a3 Comment la charge est-elle répartie ?

b) Tester la répartition de charge
Fichiers **`haproxy-node.ova`**, **`web-1.ova`** et **`web-2.ova`** à disposition
Préciser toutes les étapes dans un ordre logique
HAProxy démarre automatiquement mais peut être relancé avec **`service haproxy restart`**
Tester avec le navigateur IE

c) Analyser les statistiques depuis http://192.168.56.10:8080/stats
Rechercher les champs qui montrent le mécanisme de round-robin
Dans generator.ova, utiliser trafic_generator.sh qui produit 5000 requêtes

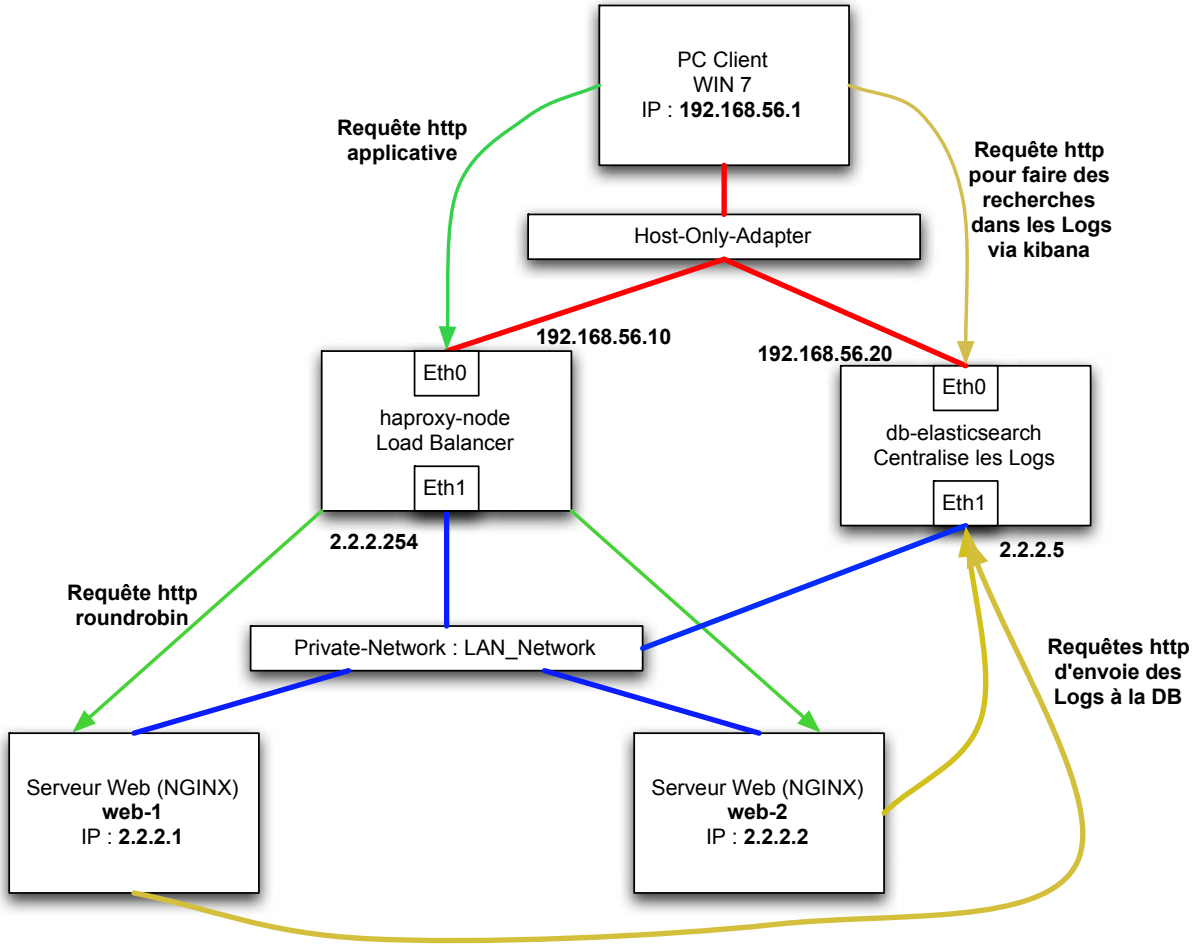
d) Quels sont les autres algorithmes supportés ?

e) Etudier l'acquisition Wireshark check
Qu'en déduisez-vous ?
Contrôler le bon fonctionnement du check dans les statistiques

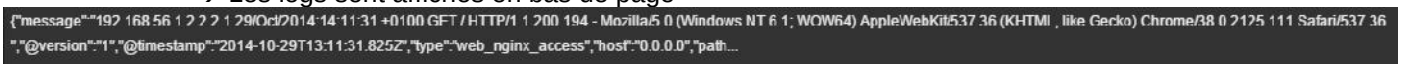
f) A quoi sert le paramètre forwardfor dans les logs du serveur web ?

§3	Centralisation et analyse des logs	10 min

Objectif Utiliser ces VMs pour étudier les logs



a) Importer depuis le dossier Log_Part puis démarrer ces 4 VMs
Se connecter à Kibana depuis Chrome : http://192.168.56.20/kibana
Depuis un second onglet faire une requête applicative http://192.168.56.10 ; typer sur F5 F5 F5
Retourner à l'onglet de Kibana ; clic sur le lien here
→ Les logs sont affichés en bas de page



Le service logstash sur chaque serveur web lit ce fichier de log **`/var/log/nginx/access.log`**

Les données sont filtrées dans les catégories suivantes de la DB elasticsearch : **`{ "message" => "%{IPORHOST:clientip} %{IPORHOST:web_server} %{HTTPDATE:timestamp} %{WORD:verb} %{NOTSPACE:request} HTTP/%{NUMBER:httpversion} %{NUMBER:response} %{NUMBER:bytes} - %{GREEDYDATA:agent}"}`**

Kibana visualise ces catégories

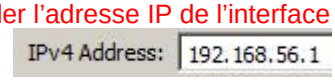
Clic sur le lien 

b) Analyser les logs produites lors des 5000 requêtes → voir §2c)

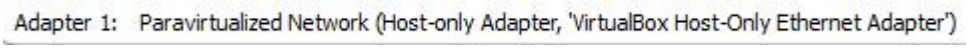
Corrigé

1a)

Désactiver le serveur DHCP
Contrôler l'adresse IP de l'interface Host-only



Importer k-1.ova et k-2.ova
Prendre connaissance des descriptions
username, password, adresse IP,
Contrôler la config réseau dans Vbox
Une seule interface active



Démarrer ces 2 VMs
Contrôler leur config réseau
ifconfig
Contrôler la connectivité avec chaque nœud
Win7 : **ping 192.168.56.11**
Win7 : **ping 192.168.56.12**
Ouvrir **/etc/keepalived/keepalived.conf** avec cat
IP_virtuelle = 192.168.56.20
Contrôler la connectivité
Win7 : **ping 192.168.56.20 → timeout**

Démarrer le nœud k-2 qui est le moins prioritaire
[root@k-2 ~]# service keepalived start
Contrôler la connectivité
Win7 : **ping 192.168.56.20 → ok (attendre évent. 30 secondes)**
Qui a répondu ?
Win7 : **arp –a → adr Eth = xx-xx-xx-xx-xx-77 donc VM k-2**
Analyser les logs

```
[root@k-2 ~]# tail -f /var/log/messages | grep keepalived
VRRP : Entering BACKUP STATE
VRRP : Transition to MASTER STATE
VRRP : Entering MASTER STATE  Ce nœud est logiquement Master puisqu'il est seul
VRRP : Setting protocol VIPs → le nœud ajoute l'adr IP virtuel
VRRP : Sending gratuitous ARPs on eth0 for 192.168.56.20
```

Contrôler la config IP du noeud

```
ip addr show →  inet 192.168.56.12/24 ... eth0
                inet 192.168.56.20/32 ... eth0
```

Démarrer le nœud k-1 qui est le plus prioritaire
[root@k-1 ~]# service keepalived start
Contrôler la connectivité
Win7 : **ping 192.168.56.20 → ok**
Qui a répondu ?
Win7 : **arp –a → adr Eth = xx-xx-xx-xx-xx-fa donc VM k-1**
→ le nœud k-1 est devenu logiquement maître car il possède une priorité supérieure
Analyser les logs

```
[root@k-1 ~]# tail –f –n 50 /var/log/messages | grep keepalived
VRRP : Entering BACKUP STATE
VRRP : Forcing a new MASTER election
VRRP : Transition to MASTER STATE
VRRP : Entering MASTER STATE
VRRP : Setting protocol VIPs
VRRP : Sending gratuitous ARPs on eth0 for 192.168.56.20

[root@k-2 ~]# tail –f –n 50 /var/log/messages | grep keepalived
...
VRRP : Entering BACKUP STATE → nœud 2 n'est plus maître
VRRP : removing protocol VIPs
```

Contrôler la config IP du nœud k-1

ip addr show → inet 192.168.56.11/24 ... eth0
inet 192.168.56.20/32 ... eth0

Contrôler la config IP du nœud k-2

ip addr show → inet 192.168.56.12/24 ... eth0

1b) 13 paquets sont perdus dans l'exemple ci-dessous (effectué sur mon Lenovo avec CPU i7)

```
64 bytes from 192.168.56.20: icmp_seq=35 ttl=64 time=1.02 ms
64 bytes from 192.168.56.20: icmp_seq=36 ttl=64 time=1.03 ms
64 bytes from 192.168.56.20: icmp_seq=50 ttl=64 time=2.56 ms
64 bytes from 192.168.56.20: icmp_seq=51 ttl=64 time=1.00 ms
```

Ce mauvais résultat est dû à Vbox
La méthodologie de mesure est intéressante

La disponibilité du nœud provoque également une perte de paquets

```
64 bytes from 192.168.56.20: icmp_seq=133 ttl=64 time=1.02 ms
64 bytes from 192.168.56.20: icmp_seq=134 ttl=64 time=1.00 ms
64 bytes from 192.168.56.20: icmp_seq=155 ttl=64 time=1.00 ms
64 bytes from 192.168.56.20: icmp_seq=156 ttl=64 time=0.994 ms
```

1c1 A quelle fréquence le nœud Master envoit-il le paquet VRRP ?
Intervalles de 1 seconde entre les paquets 1, 2, 3, 4, …, 12, 13, 14
Pour faciliter la mesure : View – Time Display Format – Seconds Since Previous Captured Packet

1c2 Quelles sont les 2 principales différences entre les paquets 4 et 5 ?
Paquet 4 est envoyé par 192.168.56.11 = k-1 avec champ Priority = 200
Paquet 5 est envoyé par .12 = k-2 avec 100 (backup VRRP router)

1c3 Quel type d'adresses Eth et IP est utilisé par ces paquets VRRP ?
Multicast

1c4 A quoi sert le paquet 6 ?
A mettre à jours le(s) cache(s) cache ARP du(des) client(s)

1c5 Quelle est la condition pour qu'un nœud slave devienne master ?
Le nœud slave ne reçoit plus d'annonce chaque seconde

1c6 Peut-on facilement ajouter un nœud supplémentaire ?
Oui il suffit de configurer correctement /etc/keepalived/keepalived.conf

1c7 Comment contrer l'attaque consistant à activer un nœud malveillant ?
Activer une authentification

2a1 Quelle est la requête http coté client ?
http://192.168.56.10

2a2 Quel serveur va répondre ?
2.2.2.1 ou 2.2.2.2

2a3 Comment la charge est-elle répartie ?
A tour de rôle (roudrobin)

2b) Importer haproxy-node.ova, web-1.ova et web-2.ova
Prendre connaissance des descriptions
username, password, adresse IP,
Contrôler la config. réseau
Démarrer ces 3 VMs

Win7 – IE http://**192.168.56.10** **Basculement ok avec F5 et Ctrl F5**

Remarque : le navigateur Chrome fonctionne différemment : basculement avec F5 ; pas de basculement avec Ctrl F5
Fonctionnement ok si outil développeur activé avec F12

2c) La partie verte renseigne par serveur
Nous constatons une répartition de 50% sur chaque serveur au niveau session et byte
Illustration avec 4 requêtes

Sessions					Bytes	
Cur	Max	Limit	Total	LbTot	In	Out
0	1	2 000	4		1 428	1 700
0	1	-	2	2	714	850
0	1	-	2	2	714	850
0	1	2 000	4	4	1 428	1 700

2d) Quels sont les autres algorithmes supportés ?
Recherche avec Google **algorithm haproxy.cfg**
Lien http://www.haproxy.org/download/1.5/doc/configuration.txt
→ balance algorithm

- roundrobin
- leastconn → le serveur qui a le moins de connexion
- source IP adr afin d'offrir une sorte de persistence entre IP_Client et IP_Server

2e) Etudier l'acquisition Wireshark check
Qu'en déduisez-vous ?
établissement TCP depuis HAProxy (2.2.2.254) sur vers Web2 (2.2.2.2) toutes les 2 secondes
établissement TCP depuis HAProxy (2.2.2.254) sur vers Web1 (2.2.2.1) toutes les 2 secondes
décalage de 1 seconde entre les 2 flux précédents

requête http get suit chaque établissement

Les statistiques indiquent le mode

LastChk
L7OK/200 in 1ms
L7OK/200 in 1ms

Remarque Il est possible de modifier le fichier de configuration pour supprimer l'un des 2 modes puis de vérifier avec tcpdump et les statistiques

2f) A quoi sert le paramètre **forwardfor** ?
Selon http://www.haproxy.org/download/1.4/doc/configuration.txt (slide 9)
Since HAProxy works in **reverse-proxy mode**, the servers see its IP address as their client address.
This is sometimes annoying when the client's IP address is expected in server logs.
To solve this problem, the well-known HTTP header "X-Forwarded-For" may be added by HAProxy to all requests sent to the server.
This header contains a value representing the client's IP address.

Exemple

```
2.2.2.254 - - [29/Sep/2014:18:52:55 +0200] "GET / HTTP/1.1" 200 194
"-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
AppleWebKit/537.78.2 (KHTML, like Gecko) Version/7.0.6
Safari/537.78.2" "192.168.56.1"
```