Résumé Journalier

Joffrey Hérard

4 avril 2017

1 Les VMs

Liste de mes connaissances des différentes technologies de virtualisation

- VirtualBox
- VMWARE
- LXC
- Docker
- QEMU
- KVM
- HyperV
- Proxmox VE
- Oracle VM Server
- Hyper-V Containers

Réorganisation des différentes VM par tableau : Tout d'abord avant de savoir ce que l'on

Table 1 – Différents services de Virtualisations

Conteneurs	Hyperviseurs Types 1 et 2
LXC	Virtualbox
Docker	VMWare
Vagrant	Qemu
	KVM
Hyper-V Containers	Proxmox(Basé sur KVM)
	Oracle VM Server
	Hyper-V

retiendras on va déjà faire quelque définitions.

Définition 1 *Hyperviseur de type 1 : est un logiciel qui s'exécute directement sur une plateforme matérielle ; cette plateforme est alors considérée comme outil de contrôle de système d'exploitation. Un système d'exploitation secondaire peut, de ce fait, être exécuté au-dessus du matériel.*

Définition 2 *Hyperviseur de type 2 : Un hyperviseur de Type 2 est un logiciel qui s'exécute à l'intérieur d'un autre système d'exploitation. Un système d'exploitation invité s'exécutera donc en troisième niveau au-dessus du matériel. Les systèmes d'exploitation invités n'ayant pas conscience d'être virtualisés, ils n'ont pas besoin d'être adaptés.*

Définition 3 *Conteneur/ Isolateur : Un isolateur est un logiciel permettant d'isoler l'exécution des applications dans ce qui est appelé des contextes.L'isolateur permet ainsi de faire tourner*

plusieurs fois la même application dans un mode multi-instance, mais les environnements virtualisés ne sont pas complètement isolés.Cependant on ne peut pas vraiment parler de virtualisation de systèmes d'exploitation.

Quoi garder ? Choix sur les différentes VM : Justification des choix cas par cas :

LXC ✓	Docker ✓	HyperV ✗
VMWare ✗	VirtualBox ✓	Qemu ✗
Oracle VM Server ✗	Proxmox ✓	KVM ✗
Hyper-V Containers ✗		

Postulat 1 *LXC : Gestionnaire de Conteneurs Linux, je l'ai utilisé aussi souvent que Docker (NB : Docker basé sur LXC)*

Postulat 2 *Docker : Gestionnaire de Conteneurs, que j'ai utilisé très souvent. (NB : Docker basé sur LXC)*

Postulat 3 *VirtualBox : Utilisé de manière courante aussi, de plus c'est un Hyperviseur classique de Type 2.*

Postulat 4 *Proxmox VE : Utilisé d'un point de vue utilisateur, mais il complète la liste en étant un Hyperviseur de type 1.*

Postulat 5 *VMware : Je ne sais pas si l'on a une licence à notre disposition ?*

Postulat 6 *Qemu : Utilisé quelque fois cependant, basé sur KVM(Comme Proxmox VE) et XEN(Connu que de nom).*

Postulat 7 *Oracle VM Server : Jamais abordé ..et je ne sais pas si l'on a une licence à notre disposition ?*

Postulat 8 *KVM : HyperViseur de type 1. Proxmox étant basé sur KVM et étant lui aussi un Hyperviseur de Type 1,*

Postulat 9 *Hyper-V Containers : Jamais abordé .. ne connaissait pas son existence avant de faire de la recherche sur le sujet .*

Postulat 10 *Hyper-V : Jamais abordé.*

2 Les tests

Test de performance Les informations recueillies concernent les temps de réponse utilisateurs, les temps de réponse réseau et les temps de traitement d'une requête.

Test de montée en charge Il s'agit d'un test au cours duquel on va simuler un nombre d'utilisateurs sans cesse croissant de manière à déterminer quelle charge limite le système est capable de supporter

Test de dégradations Ce test peut tenir compte ou non de la cadence des itérations, la représentativité en termes d'utilisateurs simultanés vs. sessions simultanées n'étant pas ici un objectif obligatoire, s'agissant ici plutôt de déterminer les points de contention générés par chaque scénario fonctionnel.

Test de robustesse il s'agit de tests au cours desquels on va simuler une charge importante d'utilisateurs sur une durée relativement longue, pour voir si le système testé est capable de supporter une activité intense sur une longue période sans dégradations des performances et des ressources applicatives ou système.

Test aux limites Proche du test de capacité,

Benchmark Un benchmark est un logiciel "banc d'essai" permettant de mesurer les performances d'un système pour le comparer à un autre. Ces test doivent donc être équivalents, en conditions similaires, seul les résultats peuvent changer.

3 Retour sur la liste des choses à faire

Retour sur le point 3 En réalité, j'ai mis les use case à cette place, mais sur le schéma de la page suivante on voit que il ce fait en "parallèle ".Mais effectivement on pourrais le mettre en première positions puis faire l'ensemble des autres tache à faire. J'ai donc mis à jour ce correctif dans le graphe du précédent rapport journalier.

Retour sur le point 6 Parce que à ma connaissance, il y a probablement plusieurs éléments (A entendre différent services)déjà installer, et sachant que l'on doit faire des tests, il est nécessaire de "nettoyer la machine" afin de n'avoir que ce qui est utile à nos besoins, et pas avoir des Démons qui s'exécutent et donc utilisent des ressources.

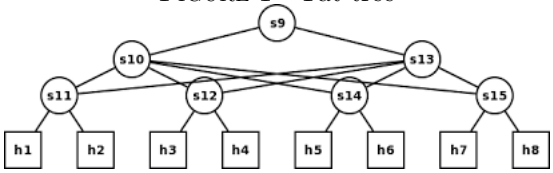
Retour sur le point 7

Pourquoi une sonde neutre Car les relevés feront objets de statistique on pourrait en sortir des données aberrant si la sonde influe elle même sur son propre relevé. Statistiquement si cette influence est constante elle peut en être extraite.

Pourquoi une sonde interne ? La encore Interne a quoi ? interne a chaque machine virtuelle/ Conteneurs ? Si on part sur le fait que il y ai une sonde sur chaque machines virtuelles crée.

- Si le nombre des machines virtuelle/ conteneurs tend vers l'infinie ça veut dire que le nombre de sonde est infinie lui aussi ? Il y a donc un besoin de répartitions et de récupération des données fournies par les sondes.
- Par contre si on exécute plusieurs machines virtuelles différentes, par exemple 50 Windows 20 Ubuntu 30 CentOS, il faut être capable de différencier les relevés d'une machine par rapport à une autre avec un ensemble d'informations qui reste à définir...
- Différenciation point à point de chaque relevé → On peut faire une structure de distribution des relevés si on le fait ainsi par exemple en prenant une architecture de fat-tree Ceci dit on interpréterait chaque nœud, comme une sonde qui soit communique avec une autre sonde soit "sonde" justement une machines.

FIGURE 1 – Fat tree



Pourquoi une sonde externe ? L'Intérêt serait d'avoir un seul gros amas de données en même endroit il ne resterait que à en extraire les informations désiré, par contre le temps de recherche peut etre long, plus il y a de donnée enregistrés(encore une fois cela reste à définir) il y aurait cependant un enregistrement qui serait par transmission d'information, un aspect Moniteur d'informations pour chaque vm a mettre en place ?

Retour sur le point 8 a)

Préparation de la machines physiques Une remise a zéro pour ne pas faussez les résultats/comparaisons, il faut une base identique, pour une neutralité de la base il faut que l'on parte du même constat pour toutes les technologies aue on compareras .

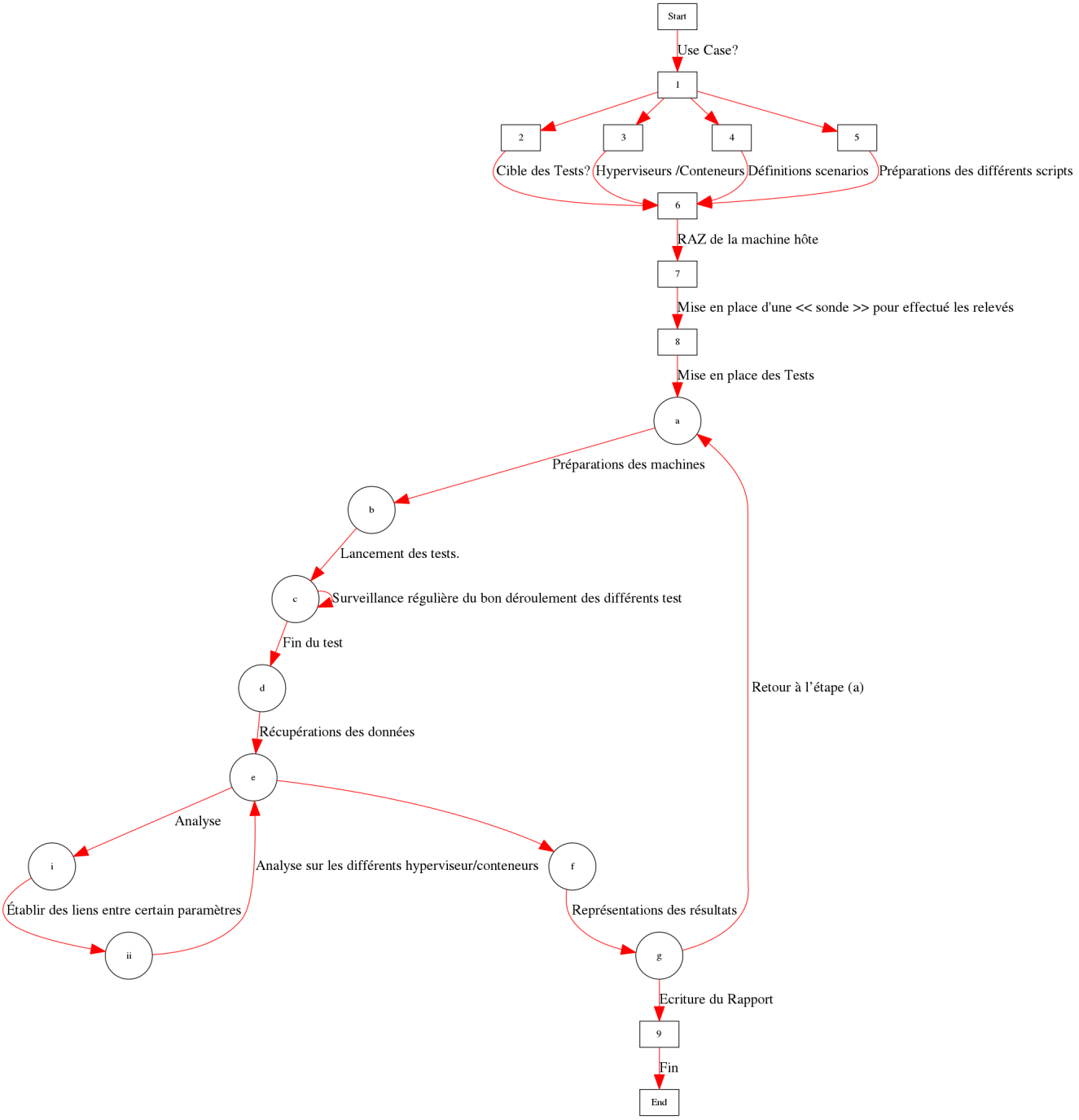
Préparation des machines virtuelles Il faudra probablement préparé des scripts de lancement de machines virtuelles pour nous en donné une certaine quantités (A déterminé).

Retour sur le point 8 c Si l'on suppose que tout est parfait, effectivement il n'y pas forcément besoin de surveiller régulièrement ! Par contre savoir ou le test en ai si il y a eu un crash d'une VM, l'idée serait de mettre un message dans un fichier.log pour savoir si des services se sont arrêter et surtout que on avait pas prévu .

Listes des choses à faire Mis a jour 4/04/2017

1. Définitions des cas d'utilisations.
2. Choix des Hyperviseurs de Type 1 ? Type 2 ? Conteneurs ?
3. Choix cible des Tests.
4. Définitions des différents scenarios associé au différent schéma ..
5. Préparations des différents scripts associés aux scenarios pour les différent hyperviseur, conteneurs
6. Remise à zero de la machine hôte.
7. Mise en place d'une « sonde » pour effectué les relevés de sorte à être des plus neutre dans les enregistrements.
8. Mise en place des Tests
   (a) Préparations de la machine physique et des machines virtuelles.
   (b) Lancement des tests.
   (c) Surveillance régulière du bon déroulement des différents test.
   (d) Récupérations des données .
   (e) Analyse des résultats sur la cohérence .
      i. Établir des liens entre certain paramètres → les confirmers par d'autre test ?
      ii. Analyse sur les différents hyperviseur/conteneurs sur les mêmes scenarios
   (f) Représentations des résultats pour chaque batterie de tests. (Courbes) et les interprétations.
   (g) Retour à l'étape (a) si nécessaire sinon étapes suivantes.
9. Écriture du rapport au fur et mesure.

FIGURE 2 – Graphe des dépendances



Références

[1] Documentation Docker https://docs.docker.com/.

[2] Hyper-V Containers https://docs.microsoft.com/en-us/virtualization/windowscontainers.

[3] Linux Containers https://linuxcontainers.org/fr/.

[4] Virtual Box https://www.virtualbox.org/.

[5] Proxmox VE https://www.proxmox.com/en/.

[6] VmWare http://www.vmware.com/fr.html.

[7] QEMU http://www.qemu-project.org/.

[8] Hyper-V https://www.microsoft.com/fr-fr/cloud-platform/server-virtualization.

[9] KVM https://www.linux-kvm.org/.

[10] Wikipedia sur les Tests https://fr.wikipedia.org/wiki/Test_de_performance.

[11] VM Mark https://www.vmmark.com/products/vmark.html.

[12] VM history par IBM http://www.vm.ibm.com/history/.