RT0801 : Scripting

Plateforme et paramètres

Ces TP de scripting sont à réaliser sur les machines du lab RT0801 TP présent sur le Remote Labz.

Remarque : Le TP Test se fera obligatoirement sur la plateforme RemoteLabz

Script bash de configuration

Écrivez un script *bash* non interactif effectuant une configuration **persistante** d'une machine. Ce script prendra en paramètre les informations suivantes :

- Nom à affecter à l'hôte
- Identifiant de la carte réseau à configurer
- Adresse de la carte réseau
- Adresse de la passerelle
- Adresse du DNS

Le script réalisera les opérations suivantes :

1. Affectation du nom de l'hôte
2. Vérification de l'existence de l'interface réseau
3. Désactivation de la carte réseau
4. Modification de l'adresse de la carte réseau
5. Activation de l'interface réseau, et si nécessaire du service réseau
6. Modification de l'adresse du DNS
7. Test de connexion au réseau

Des tests seront effectués à chaque étape, et une sortie d'erreur sera effectuée en cas de problème, vous différencierez les erreurs.

Reprises des scripts du cours

Vous trouverez à la fin des transparents du cours les 3 descriptions suivantes.
Développez successivement les scripts correspondants en bash et en python.

Récupération d'informations sur un utilisateur

Description du script :

- Pas de paramètres
- Création d'un nouveau fichier à chaque utilisation, les anciens fichiers ne doivent pas être détruits !
- Opérations :
    - Affichage et enregistrement : de l'utilisateur ; de l'id de l'utilisateur ; des groupes de l'utilisateur

Un script de sauvegarde

Pas testé

Description du script :

- Paramètres
  - le nom de l'archive
  - nom du répertoire à sauvegarder
  - adresse du serveur de sauvegarde
  - login et password du compte
- Opérations
  - L'archive est réalisée avec tar
  - Après archivage, copie à travers du réseau de l'archive (vous installerez un serveur *sftp* en local)

Un script d'installation

Pas testé

Description du script

- Paramètres
  - le nom de l'archive
  - le répertoire de base d'installation
- Opérations
  - Vérification des existences et des droits
  - Test du type d'archive (zip, tar, tgz) (Uniquement sur l'extension)
  - Utilisation de l'archiveur correspondant

Gestion des conteneurs

Pour les scripts suivants, vous allez utiliser des conteneurs LXC. Dans cette partie, vous allez écrire des scripts bash qui permettront la mise en place de différents conteneurs. Vous avez le choix du type de conteneur que vous allez utiliser : debian, ubuntu, alpine …

Les conteneurs seront montés selon une architecture NAT sur l'interface réseau de l'hôte.

Scripts de création / destruction de l'environnement

Écrivez les 4 scripts qui assurent la mise en place l'environnement d'exécution des conteneurs :

1. `inst_cont_env.sh` : installera l'ensemble des packages nécessaires à l'utilisation de conteneurs et à la création d'un bridge spécifique ;
2. `set_net.sh` : assurera la création du bridge permettant la connexion des futurs conteneurs, vous êtes libre du type de bridge (brctl, ip, openvswitch) que vous allez utiliser ;
3. `restore_net.sh` : assurera la destruction du bridge créé par le script `set_net.sh`, ainsi que la remise en place de la configuration d'origine ;
4. del_cont_env.sh : assurera l'effacement des packages installés par le script `inst_cont_env.sh`

Scripts de création / destruction de conteneur

Écrivez les 4 scripts qui permettent la gestion des conteneurs :

1. `creat_container.sh` : assure la création d'un conteneur LXC. Ce script prendra en paramètre l'ensemble des paramètres du conteneur : template, adresse réseau, configuration système (mémoire, CPU …), mot de passe root … Ce script assurera la modification du fichier de configuration du conteneur.
2. `start_container.sh` : assure le démarrage d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt. Vous pourrez étendre à l'utilisation des expressions régulières pour désigner les conteneurs.
3. `stop_container.sh` : assure l'arrêt d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt. Vous pourrez étendre à l'utilisation des expressions régulières pour désigner les conteneurs.
4. `del_container.sh` : assure la destruction d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt. Vous pourrez étendre à l'utilisation des expressions régulières pour désigner les conteneurs.

Synchronisation de fichiers en bash

On souhaite développer une solution de synchronisation de fichiers. Dans cette partie nous aurons besoin de 3 conteneurs LXC C1, C2 et C3, qui seront créés avec les scripts précédents. La synchronisation sera dirigée par un utilisateur, qui lancera explicitement un script, et non par un processus en tâche de fond.

On souhaite mettre en place une solution de synchronisation. C1 jouera le rôle de serveur de sauvegarde. Sur C2 et C3 on définira des répertoires à synchroniser. Lors d'une phase de synchronisation, chacun des fichiers à synchroniser sera comparé avec sa version sur le serveur. Si les deux fichiers sont identiques, rien ne sera effectué. Si les deux fichiers ne sont pas identiques, le plus récent remplacera le plus ancien, et ce quelle que soit la localisation du plus ancien, sur C1, C2 ou C3.

Proposez un ensemble de scripts permettant :

- L'ajout de répertoires ou de fichiers à la liste des sauvegardes / synchronisations ;
- La synchronisation d'un fichier, d'un répertoire ou de l'ensemble des éléments enregistrés pour être synchronisés ;

À chaque exécution d'une synchronisation, le script affichera l'état de la synchronisation sur C2 et C3, et sauvegarder dans un fichier les opérations réalisées sur C1.

Une fois les scripts écrits, vous les implanterez dans les conteneurs, et effectuerez un exemple d'exécution. Cet exemple d'exécution sera piloté par un script depuis la machine hôte.

Une solution de supervision en python

On souhaite mettre en place une solution simple de supervision en python. Le but est de collecter à intervalle régulier différentes informations sur les machines, ceci à l'aide de sondes développées en python. Dans cette partie nous aurons besoin de 3 conteneurs C1, C2 et C3, qui seront créés avec les scripts précédents. C1 jouera le rôle de centralisateur.

Les sondes à développer permettent :

- L'*uptime*, ainsi que la lecture de la charge mémoire et CPU, ainsi que l'occupation de chacune des partitions
- La lecture des 3 processus qui consomment au moment de la lecture le plus de CPU, ainsi que les 3 processus qui consomment le plus de mémoire

L'ensemble des informations seront collectées et centralisées par un script sur C1. Assurera donc la sauvegarde des informations, mais aussi leur mise en forme dans le cadre d'un serveur WEB. Une page html sera créée par machine supervisée, celle-ci sera mise à jour lors de chaque collecte de données.

Écrivez l'ensemble des scripts nécessaire à la réalisation de ce système de supervision, ainsi que le script bash de déploiement, ce dernier étant exécuté, une fois les conteneurs lancés, depuis la machine hôte.

Provisionning

Dans cette partie, vous allez utiliser des outils de provisionning. Dans chaque cas, le serveur sera implanté sur l'hôte. Vous utiliserez en priorité les outils offerts par les solutions de provisionning plutôt que des scripts externes en python ou en bash.

Ansible

Une fois que vous aurez lancé les scripts de nettoyage des installations, réalisez différents *playbooks* ansible réalisant les opérations des scripts développés dans la partie *gestion des conteneurs*. Vous développerez aussi bien des *playbooks* pour la gestion de l'environnement que la gestion des conteneurs.

En exploitant les *playbooks* précédemment écrits, écrivez un ou des *playbooks* réalisant les mêmes opérations que celle réalisées dans le cadre des scripts de synchronisation de fichiers en *bash*.

SaltStack

Une fois que vous aurez lancé les scripts de nettoyage des installations, réalisez différents states *saltstack* réalisant les opérations des scripts développés dans la partie *gestion des conteneurs*. Vous développerez aussi bien des states pour la gestion de l'environnement que la gestion des conteneurs.

En exploitant les states précédemment écrits, écrivez un ou des states réalisant les mêmes opérations que celle réalisées dans le cadre des scripts de synchronisation de supervision en python.