Compte rendu

SAE15

Objectif du projet :

Lors de ce projet nous devions produire un outil de surveillance de serveur réseaux. En outre l’outil doit pouvoir surveiller la connectivité avec d’autres appareil et les ports qui y sont connectés.

Pour nos tests nous avons utilisés le serveur iziram.fr

Organisation du travail :

Pour ce projet, nous avons utilisé l’outil git, qui nous permet de synchroniser le travail que chacun produit de son côté.

Nous nous sommes réparti le travail de façon a ce que la partie Bash, qui récupère les métriques, et la partie python qui les affiches avancent de façon équitable. Pour cela nous avons créer un programme qui simule la récupération des données dans le même style que le script Bash final.

Récupérer les métriques :

Métrique 1 :

Pour voir les métriques il a fallu les récupérer. Un script bash fut créer pour avoir les données qui concernent la connectivité.

La script prend en paramètre l’adresse web du serveur pour le quel il faut tester la connectivité(ping). Il prend également en paramètre le chemin du fichier à télécharger pour avoir le download.

Le problème du upload est qu’il faut posséder les droits d’écriture sur le serveur. Donc pour cette partie du script nous ne pouvons pas permettre à l’utilisateur de fournir l’accès au serveur. Nous utilisons alors le serveur iziram.fr qui possède une configuration spécialement faite pour ce projet.

if [ $# -gt 1 ]

then

    chemin=${0::-15}

    key="${chemin}iutsshkey"

    ping=$(ping -c 1 $1 | grep "time=" | cut -d " " -f 8 | tr -d "time=")

    download\_number=`curl -w '%{speed\_download}\n' $2 |& tail -n 1`

    upload\_number=`scp -vi $key $key iut@iziram.fr:/home/iut/upload |& tail -n 2 | head -n 1 | cut -d " " -f 5`

    upload\_number=${upload\_number::-1}

    upload\_number=`bc <<< $upload\_number\*1000 `

    upload\_number=`bc <<< $upload\_number/8 `

    upload\_number=`bc <<< $upload\_number/1000 `

    depoch=$(date +%s)

    echo "$ping,$download\_number,$upload\_number,$depoch"

fi

Le script affiche dans la sortie standard des valeurs dans l’ordre suivant :

Ping, download, upload, temps.

440, 771, 60, 1638191858

Métrique 2 :

Les données sur les ports sont récupérées par un autre script :

#!/bin/bash

date=$(date '+%s')

infos=$(netstat -nautA inet | tail -n +3 | tr -s ' ')

before=$IFS

IFS=$'\n'

for ligne in $infos

do

    socket\_local=`echo $ligne | cut -d " " -f 4`

    socket\_distant=`echo $ligne | cut -d " " -f 5`

    port\_local=`echo $socket\_local | cut -d ':' -f 2`

    port\_distant=`echo $socket\_distant | cut -d ':' -f 2`

    protocol="d"

    if (( port\_local > 10000 ));

    then

       protocol=$port\_distant

    else

       protocol=$port\_local

    fi

    echo "$(echo $socket\_local | cut -d ':' -f 1 ),$port\_local,$protocol,$date"

done;

IFS=$before

exit 0;

Le script renvoie dans la sortie standard les informations dans cet ordre :

Interface, port, Protocol, temps.

10.1.1.1, 45533, 8080, 1638188558

Stocker les informations :

Comme je vous l’ai expliqué les scripts ne retourne leur résultat que dans la sortie standard, or pour pouvoir afficher les informations nous devons les disposer dans un fichier scv, qui, par la suite sera repris dans le programme en python. Pour cela crontab sera utiliser. Crontab est un outil qui permet de faire une action a un moment précis sans avoir à le faire manuellement.

Insérer partie qui montre l’utilisation de crontab