Victor Bienvenu
Matthias Hartmann

Compte rendu

SAE15

Sujet : 3 | Métriques : 1 et 3

Table des matières

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 et plus particulièrement GitHub, qui nous a permis de synchroniser notre travail pour que chacun puisse travailler de son côté.

Nous nous sommes réparti le travail en deux parties. La partie Bash, qui récupère les métriques, et la partie Python qui les affiche. Matthias a été assigné à la programmation des scripts Bash pendant que Victor a fait la programmation Python. Afin que Victor puisse avancer ses programmes dès le début du projet Matthias a écrit un script python qui générait des données aléatoires.

Lorsque la partie Bash fut finie, Matthias commença la mise en place du notebook jupyter en prenant les programmes python de Victor. Il les a corrigés lorsque cela était nécessaire, les a commentés, et les a modifiés pour qu'ils puissent être correctement utilisés avec le notebook.

Problème rencontré :

La partie Bash était complexe pour plusieurs points. Premièrement il a fallu récupérer les mesures de connectivité. C'est à dire l'Upload (téléversement), Download(Téléchargement) et le Ping (L'accessibilité). Le premier problème fut la récupération du téléversement. En effet afin de pouvoir téléverser un fichier sur un serveur il nous faut des droits d'écritures sur le dit serveur. Matthias ayant un serveur à disposition il a mis en place un environnement permettant de se connecter en SSH et d'écrire dans un dossier précis. De plus il nous a fallu écrire plusieurs autres scripts pour pouvoir avoir

des observations intéressantes. Ces scripts ont été long à faire car Matthias n'arrivait pas à trouver de moyens de changer la connectivité ou l'utilisation des ports directement en commandes. Il a néanmoins réussit à contourner le problème.

Dans la partie python, le choix des graphiques à faire a posé un problème car nous devions choisir des graphiques pertinents. Pour afficher ces graphiques nous avons utilisé matplotlib. Cependant il est n'est pas facile de trouver, dans la documentation de matplotlib, la fonction qui correspond à notre recherche sans avoir à chercher sur plusieurs pages.

Victor Bienvenu
Matthias Hartmann

Récupérer les métriques :

Les scripts Bash :

Métrique 1 : Les ports

```
#!/bin/bash

#Récuppération de la date en timestamp
date=$(date '+%s')

#Récuppération de la liste des ports ouverts avec netstat -nautA dont on retire l'entête avec
tail -n +3 puis on normalise les espaces avec tr -s ' '
infos=$(netstat -nautA inet | tail -n +3 | tr -s ' ')

#Par défaut la boucle 'for' parcourt un texte et utilise ' ' comme délimiteur. Nous le
changeons car nous devons délimiter par ligne.
#On retient le précédent délimiteur car nous modifons une variable environnementale et donc
nous devrons la remettre à la fin
before=$IFS
#On assigne le nouveau délimiteur, ici un saut de ligne
IFS=$'\n'

#On parcourt notre liste de ports
for ligne in $infos
do
   #On récuppère le socket local et distant dont on obtient le port distant et le port local
   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`

   #On établie une variable "protocol" qui aura comme valeur un numéro de port représentant
le protocol.
   protocol="d"
   #On regarde si notre port local est un port connu ou réservé (10000 est une valeur
arbitraire)
   if (( port_local > 10000 ));
   then
      #Si le port local est supérieur à 10000 on considère le port distant comme le port
représentant
      protocol=$port_distant
   else
      #Sinon on considère le port local.
      protocol=$port_local
   fi

   #Afin d'avoir des observations intéressantes on utilise le Script modifPorts.sh. S'il
renvoit "0" enregistre bien le port, sinon on ne l'enregistre pas.
   chemin=${0::-8}
   if [ "$(${chemin}modifPorts.sh)" == "0" ]
   then
      echo "$(echo $socket_local | cut -d ':' -f 1 ),$port_local,$protocol,$date" >> $1
   fi


done;

#On remet la configuration par défaut du 'For'
IFS=$before
exit 0;
```

Victor Bienvenu
Matthias Hartmann

Métrique 3 : La connectivité

```
#!/bin/bash
#On demande en paramètre du script 1) l'adresse de la machine/serveur à "ping" 2) l'adresse
d'un fichier à télécharger
if [ $# -gt 1 ]
then

    #On récupère le chemin du dossier dans lequel se trouve le script car on va devoir
utiliser plusieurs autres fichiers qui s'y trouvent
    chemin=${0::-15}

    #On créer une variable "key" qui contient le chemin vers la clé privée qui sera utiliser
pour se connecter au vps de Matthias
    key="${chemin}iutsshkey"

    #On récupère le temps de connectivité grace à un ping
    ping=$(ping -c 1 $1 | grep "time=" | cut -d " " -f 8 | tr -d "time=")

    #On récupère le débit descendant grace a la commande curl
    download_number=`curl -w '%{speed_download}\n' $2 |& tail -n 1`

    #On récupère le débit montant grace à l'upload de la clé privé sur le vps de Matthias.
    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}

    #Le débit montant récupéré est en Kbit/s donc on le converti en Ko/s
    upload_number=`bc <<< $upload_number*1000 `
    upload_number=`bc <<< $upload_number/8 `
    upload_number=`bc <<< $upload_number/1000 `
    #On récupère la date sous la forme d'un timestamp
    depoch=$(date +%s)

    #Afin d'avoir des observations intéressantes on utilise le Script modifConnectivity.sh.
Celui-ci ajoutera ou non une valeur aléatoire au valeur observée
    #De cette façon on aura pas de valeur trop constantes
    ping=$($modifier ping $ping)
    upload_number=$($modifier upload $upload_number)
    download_number=$($modifier download $download_number)

    #On affiche dans la sortie nos valeurs
    echo "$ping,$download_number,$upload_number,$depoch"
fi

exit 0;
```

Modifier les métriques :

Comme vous avez pu le voir dans les scripts, il y a des « modifiers » se sont des scripts bash écrites pour modifier les observations afin de les rendre plus intéressantes.

Il y a au total 3 scripts et un fichier de configuration.

modifPorts.sh :

```
#!/bin/bash

#On récupère le chemin pour pouvoir accéder au fichier de configuration.
chemin=${0::-13}

#On lit le fichier et on prend le deuxième chiffre qui s'y trouve, il correspond à un booléen
disant si on doit ou non modifier les observations des ports.
conf=$(cat ${chemin}modif.conf | cut -d " " -f 2)

#Si le chiffre est à 1 cela veut dire que l'on doit modifier le port.
if [ "$conf" == "1" ]
then
    #La modification est pensée comme suit : il y a une chance sur 10 pour que le port qui
devait être affiché ne s'affiche pas.
    if [ "$(($RANDOM % 10))" == "0" ]
    then
        #On sort 1 dans la sortie pour signifier que le port ne doit pas être affiché
        echo 1
    else
        #Sinon on sort 0.
        echo 0
    fi
else
    echo 1
fi

exit 0;
```

Victor Bienvenu
Matthias Hartmann

modifConnectivity.sh :

```
#!/bin/bash
#le script demande 2 paramètre, 1) le type de mesure que l'on modifie 2) la mesure.
if [ $# -gt 1 ]
then
    #On récupère le chemin pour pouvoir accéder au fichier de configuration.
    chemin=${0::-20}

    #On lit le fichier et on prend le premier chiffre qui s'y trouve, il correspond à un
booléen disant si on doit ou non modifier les observations des mesures.
    conf=$(cat ${chemin}modif.conf | cut -d " " -f 1)
    if [ "$conf" == "1" ]
    then
        if [ "$1" == "ping" ]
        then
            #Si on modifie un ping on doit ajouter une valeur positive inférieur à 500
            rdm=$((1 + $RANDOM % 500))
            val=$(bc <<< $rdm+"$2");
            echo $val;
        else
            #Sinon on modifie les autres valeurs en ajoutant un nombre aléatoire entre -200
et 599
            rdm=$((-200 + $RANDOM % 600))
            val=$(bc <<< $rdm+$2);
            echo ${val#-}
        fi
    else
        #Si on ne modifie pas les mesures on renvoit juste la mesure non modifiée
        echo $2
    fi
fi
exit 0;
```

modifAll.sh :

```
#!/bin/bash

#On récupère le chemin pour pouvoir accéder au fichier de configuration.
chemin="${0::-11}modif.conf"

#On met le contenu du fichier dans une variable puis on vide le fichier.
confs=$(<${chemin})
> ${chemin}

#Pour chaque booléen du fichier de configuration on inverse sa valeur
for i in $confs
do
    if [ "$i" == "1" ]
    then
        #On utilise un echo -n pour ajouter la valeur dans le fichier sans passer de nouvelle
lignes
        echo -n "0 " >> $chemin
    else
        echo -n "1 " >> $chemin
    fi
done

exit 0;
```

modif.conf :

```
1 1
```

modif.conf est un fichier qui est utilisé par modifConnectivity et modifPort pour savoir quand est-ce qu'il faut modifier les valeurs. Cela permet de contrôler la modification des observations et donc d'avoir un mélange de vraies et de fausses valeurs.

Ce contrôle est réalisé grâce au script modifAll.sh qui inverse les booléens. Ce script est activé par CRON toutes les 5 minutes.

Répétition des observations :

Pour répéter nos observations sur une période nous avons utilisé CRON.

Nous avons alors créé 3 nouveaux cron jobs :

```
* * * * * /home/iut/observ/connectivity.sh >> /home/iut/observ/connectivityData.csv
* * * * * /home/iut/observ/ports.sh /home/iut/observ/portsData.csv
*/5 * * * * /home/iut/observ/modifAll.sh
```

Ces CRON jobs permettent de faire les choses suivantes :

- Observer toutes les minutes l'activité et la connectivité du serveur
- Modifier le fichier de configuration des modifications toutes les 5 minutes