Faire un script bash dans le but de remplacer le projet réaliser par un ancien stagiaire car le projet qu'il a mis en place date de 2020 réalisé durant le stage de 1ère annèe:

Ce projet consiste à faire un fping sur différentes adresses ip et de récupérer des informations qui sont l'ip, le pourcentage de paquet perdu, le temps minimum de réponses, le temps moyens et le temps maximum.

Dans un premier j'ai dû installer maria dB et phpMyAdmin et créer la base de données où il y aura les données de stocker :

https://www.it-connect.fr/installer-phpmyadmin-sur-debian-11-et-apache/

Pour créer un script Bash on peut se mettre dans n'importe quel dossier, dans un terminal on utilise nano avec le nom du fichier suivi de l'extension sh.

Pour écrire ce script on y aller étapes par étapes :

#On récupère les ip dans les fichiers dans le dossier DATA, on effectue un fping sur toutes les ip. Les résultats du fping sont stocker dans le fichier resultats

```
for i in $(find $path -type f -exec cat *)

do

fping -c 1 -q --all $i >> resultats 2>&1

done
```

#On récupère les informations que l'on souhaite dans le résultat du ping. On utilise la commande cut pour avoir chaque partie séparément pour qu'elle soit stocké dans des variables pour les rappeler plus tard.

```
while read line; do

echo "${line}"

ip=$(echo $line | cut -d' ' -f 1)

lost=$(echo $line | cut -d' ' -f 5 | cut -d'/' -f 3 | cut -d'%' -f 1 | cut -d'/' -f 3)

min=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 1)

avg=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 2)

max=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 3)

echo "$ip $lost $min $avg $max"

done < resultats
```

Ensuite ces données on les transferts vers la base de données

/usr/bin/mariadb -u$dbuser -p$dbpasswd $dbname -e "INSERT INTO $dbtabname (ip, lost, min_ms, avg_ms, max_ms, IDclient, IDproduit, IDcontrat, nom_client) VALUES ('$ip', '$lost', '$min', '$avg', '$max', '$id_client', '$id_produit', '$id_contrat', '$nom_client');"

Après j'ai dû rajouter les informations à propos des clients qui sont l'id client, l'id produit, l'id contrat et le nom du client dans les fichiers dans le dossier data qui sont séparés de l'ip et entre elles par des deux points. Il a fallu modifier la commande qui cherche les adresses ip pour que le script qui est situé à gauche du premier deux-points pour qu'elle devienne :

find $path -type f -exec awk -F ':' '{print $1}' {} \;

Cela m'a permis de découvrir de nouvelles commandes comme sed (supprime ou remplace des caractères), cat (afficher le contenu d'un fichier texte), find -type f -exec (permet de rechercher dans un dossier), cut (séparés les données que l'on veut afficher), awk (séparés les données que l'on veut afficher)

Finalement le script ressemble à cela :

```
#!/bin/bash

#Récupère les ip dans les fichiers dans le dossier DATA, situé à gauche du premier deux-points et effectue un fping sur toutes les ip

for i in $(find $path -type f -exec awk -F ':' '{print $1}' {} \;)

do

fping -c 1 -q --all $i >> resultats 2>&1

done


#Récupère les informations que l'on souhaite dans le résultat du ping et les insèrent dans la base de données

while read line; do

echo "${line}"

ip=$(echo $line | cut -d' ' -f 1)

lost=$(echo $line | cut -d' ' -f 5 | cut -d'/' -f 3 | cut -d'%' -f 1 | cut -d'/' -f 3)

min=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 1)

avg=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 2)

max=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 3)

#echo "$ip $lost $min $avg $max"
```

```
#Récupère des informations à propos des clients dans les fichiers dans le dossier DATA et les insèrent dans la base de données

#ID client

echo > idclient

for a in $(find $path -type f -exec cat {} \; | grep $ip)

do

echo "$a" >> idclient 2>&1

id_client=$(cat idclient | grep $ip | cut -d':' -f 2)

done


#ID produit

echo > idproduit

for b in $(find $path -type f -exec cat {} \; | grep $ip)

do

echo "$b" >> idproduit 2>&1

id_produit=$(cat idproduit | grep $ip | cut -d':' -f 3)

done


#ID contrat

echo > idcontrat

for c in $(find $path -type f -exec cat {} \; | grep $ip)

do

echo "$c" >> idcontrat 2>&1

id_contrat=$(cat idcontrat | grep $ip | cut -d':' -f 4)

done


#Nom client

echo > nomclient

for d in $(find $path -type f -exec cat {} \; | grep $ip)

do

echo "$d" >> nomclient 2>&1
```

```
#Dans le nom client il ne peut pas y avoir d'espace

nom_client=$(cat nomclient | grep $ip | cut -d':' -f 5)

done


/usr/bin/mariadb -u$dbuser -p$dbpasswd $dbname -e "INSERT INTO $dbtabname (ip, lost, min_ms, avg_ms, max_ms, IDclient, IDproduit, IDcontrat, nom_client) VALUES ('$ip', '$lost', '$min', '$avg', '$max', '$id_client', '$id_produit', '$id_contrat', '$nom_client');"

done < resultats

rm résultats
```

Reprise du script et amélioration lors du stage de 2nde année :

J'ai repris ce projet que j'avais commencé durant mon stage à PobRun durant laquelle j'avais effectué une semaine chez Voganet avec comme consigne :

- au lieu de lire une liste d'ip dans un fichier je vais te connecter à une base de données qui contient la liste d'IP

avec cette commande j'ai pu obtenir la liste des IP : mysql -u ping -p<mot de passe BDD> radius -h <ip base de donnée BDD> -N -B -e "SELECT value FROM <nom de la table> WHERE attribute='DHCP-Your-IP-Address'"

- pour chaque IP on lance un fping

- je stocke les résultats dans une BDD maria DB (MySQL) situé sur une VM à laquelle je suis connecté en SSH

- on exploite les résultats dans Grafana, le but est d'arriver à une page similaire à celle qui s'affiche sur un écran

J'ai modifié le script en conséquence :

```
# Recupere les IP d'une base de donnees et effectue un fping sur toutes les IP

mysql -u ping -p<mdp> radius -h <ip base de données> -N -B -e "SELECT value FROM <nom table> WHERE attribute='DHCP-Your-IP-Address'" | while IFS= read -r i

do

   /usr/bin/fping -c 1 -q --all "$i" >> resultats 2>&1

done

#Recupere les informations que l'on souhaite dans le resultat du ping et les inserent dans la base de donnee

while read line; do

        ip=$(echo $line | cut -d' ' -f 1)
```

```
lost=$(echo $line | cut -d' ' -f 5 | cut -d'/' -f 3 | cut -d'%' -f 1 | cut -d'/' -f 3)

min=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 1)

avg=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 2)

max=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 3)


/usr/bin/mariadb -u$dbuser -p$dbpasswd $dbname -e "INSERT INTO $dbtabname (ip, lost, min_ms, avg_ms, max_ms) VALUES ('$ip', '$lost', '$min', '$avg', '$max') ON DUPLICATE KEY UPDATE lost='$lost', min_ms='$min', avg_ms='$avg', max_ms='$max';"

done < resultats

rm resultats
```

Je n'ai pas utilisé le « for » cette fois mais le « while » car lorsque j'exécutai le script sa mettait fping commande introuvable ou error in syntax of 'fping' token.

J'ai modifié la requête SQL pour qu'elles permettent d'insérer de nouvelles données et de mettre à jour des données lorsqu'un enregistrement existe déjà dans une table pour cela j'ai utilisé l'instruction ON DUPLICATE KEY UPDATE. J'ai modifié la table car on ne peut pas utiliser l'instruction précédente toute seule, avec l'utilisation de UNIQUE KEY sur les adresses IP pour pas que toute la table soit modifiée. Celle-ci permet d'identifier de manière unique les enregistrements. La clé unique garantit l'unicité des colonnes dans la base de données. Il peut exister plusieurs clés uniques dans une table comparé à la clé primaire où il n'y en a qu'une seule par table.

J'avais essayé de lire dans le fichier ips.txt et à chaque ligne, faire un fping. J'avais utilisé xargs qui permettait de faire 40 pings en parallèle pour toutes les IP avec 10 pings pour chaque IP. J'ai fait plusieurs tests avec xargs pour trouver la bonne valeur pour que le script dure une minute. Le problème avec cette solution, c'est qu'elle causait des erreurs dans le fichier résultats. Sur une ou plusieurs ligne(s), le résultat du fping se mettait à la suite d'un résultat précédent et se coupait au milieu de la phrase de la seconde réponse, puis se mettait ensuite à la ligne. Ce qui faisait n'importe quoi lors du découpage pour récupérer les informations que je voulais récupérer et lors de l'insertion dans la base de données.

Voici la commande que j'ai utilisée :

cat ips.txt | xargs -P 40 -I {} /usr/bin/fping -c 10 -q -f ips.txt --all "{}" > resultats 2>&1

J'ai recherché dans la documentation de fping et j'ai vu qu'il y avait une option qui permettait de lire un fichier et d'effectuer les pings. J'ai dû modifier la requête SQL pour que lorsqu'un nouvel enregistrement est réalisé dans la base de données sur la VM, on sache quand il a été fait.

Le script final sans aucun bug s'exécute en environ 32 secondes :

```
#!/bin/bash

dbuser=root
```

```
dbpasswd=root

dbname=ping

dbtabname=info

# Execute la requete SQL pour obtenir toutes les adresses IP dans un fichier texte

mysql -u ping -p<mdp BDD> radius -h <ip BDD> -N -B -e "SELECT value FROM <nom table> WHERE attribute='DHCP-Your-IP-Address'" > ips.txt

# Execute fping a partir du fichier ips.txt pour toutes les adresses IP avec 10 pings chacune, les resultats sont stockes dans un fichier

/usr/bin/fping -c 10 -q -f ips.txt >> resultats 2>&1

# Supprime le fichier temporaire contenant les adresses IP

rm ips.txt

#Recupere les informations que l'on souhaite dans le résultat du ping et les insèrent dans la base de donnée

while read line; do

    ip=$(echo $line | cut -d' ' -f 1)

    lost=$(echo $line | cut -d' ' -f 5 | cut -d'/' -f 3 | cut -d'%' -f 1 | cut -d'/' -f 3)

    min=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 1)

    avg=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 2)

    max=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 3)

    #echo "$ip $lost $min $avg $max"

/usr/bin/mariadb -u$dbuser -p$dbpasswd $dbname -e "INSERT INTO $dbtabname (ip, lost, min_ms, avg_ms, max_ms, datetime) VALUES ('$ip', '$lost', '$min', '$avg', '$max', current_timestamp()) ON DUPLICATE KEY UPDATE lost='$lost', min_ms='$min', avg_ms='$avg', max_ms='$max', datetime=current_timestamp();"

done < resultats

rm resultats
```

Je veux que mon script se lance toutes les minutes, donc j'utilise cron. Pour créer une nouvelle règle cron, il faut faire crontab -e, et pour voir les règles existantes, il faut faire crontab -l. La règle cron à ajouter est : * * * * * /home/voganet/script.sh.

Cependant, le problème que j'ai rencontré sur grafana c'est que MySQL n'est pas une base de données temporelle ; elle permet juste d'avoir les dernières données. Donc je me suis tourné vers la base de données temporelle InfluxDB.

Pour l'installation d'InfluxDB, j'ai utilisé la version 1 plutôt que la version 2, qui possède une interface graphique et d'autres options. Cependant, j'avais juste besoin d'une version simple, c'est pourquoi j'ai choisi cette version. Pour l'installer, j'ai utilisé la commande suivante :

wget https://dl.influxdata.com/influxdb/releases/influxdb_1.8.10_amd64.deb

dpkg -i influxdb_1.8.10_amd64.deb

Il faut ensuite créer un utilisateur :

influx create user voganet with password '${INFLUDB_PASSWORD}' with all privileges;

Pour ne pas avoir le mot de passe en clair dans le terminal j'ai utilisé les variables d'environnement qui se trouve le fichier bashrc

nano ~/.bashrc

Dans le fichier on met les variables sous cette forme :

export INFLUXDB_PASSWORD="<mdp>"

Pour que les résultats prennent effets il faut faire : source ~/.bashrc

On se connecte ensuite à influxDB :

influx -username voganet -password ${INFLUXDB_PASSWORD}

On crée notre première base de données :

CREATE DATABASE ping ;

InfluxDB fonctionne différemment des bases de données relationnelles comme MySQL. Il n'y a pas de tables mais des 'measurements' (mesures), pas de concepts de clé primaire, clé secondaire, etc., mais des 'tags' pour classer les données. Les données numériques sont classées dans des 'fields' (champs).

Measurement (Mesure) : La mesure peut être comparée à une table dans une base de données relationnelle. C'est la catégorie principale à laquelle appartiennent les données. Une mesure représente généralement un type spécifique de données collectées.

Tags (Étiquettes) : Les étiquettes sont utilisées pour identifier et indexer les données. Elles sont similaires aux clés primaires dans une base de données relationnelle, mais les étiquettes dans InfluxDB sont facultatives et peuvent être utilisées de manière plus flexible. Elles permettent de filtrer et d'organiser les données en fonction de métadonnées spécifiques.

Fields (Champs) : Les champs contiennent les valeurs réelles associées aux points de données. Chaque champ peut être considéré comme une colonne qui stocke une mesure spécifique.

Dans le script, il faut modifier la requête pour qu'elle utilise celle d'InfluxDB plutôt que celle de MySQL. Celle-ci utilise la commande curl pour faire une requête HTTP à la base de données.

```
# Construisez la commande curl avec la date et des conditions pour chaque variable

curl_command="curl -i -XPOST 'http://localhost:8086/write?db=$dbname' -u
$dbuser:${INFLUXDB_PASSWORD} --data-binary 'ping,ip=$ip"
```

#Ces conditions garantissent que seules les parties correspondant aux variables ayant une valeur définie sont incluses dans la commande curl finale. Cette partie est conçue pour gérer les données nulles, car InfluxDB n'aime pas avoir des données vides dans sa requête. Si des données sont vides, InfluxDB affiche un message d'erreur et n'effectue pas la requête

```
if [ -n "$lost" ]; then

        curl_command+=",lost=$lost"

fi

if [ -n "$min" ]; then

        curl_command+=",min_ms=$min"

fi

if [ -n "$avg" ]; then

        curl_command+=",avg_ms=$avg"

fi

if [ -n "$max" ]; then

        curl_command+=",max_ms=$max"
```

```
fi

curl_command+=" value=3538.3773' > /dev/null 2>&1"

# Executez la commande curl

eval "$curl_command"
```

Dans la fin de la requête, il y a curl_command+=" value=3538.3773' > /dev/null 2>&1", c'est-à-dire que l'on redirige le résultat pour chaque requête pour qu'elle ne soit pas affichée dans le terminal.

Avant de rediriger, on vérifie si les données sont correctement insérées. Lorsque les données sont correctement insérées sans aucun problème, InfluxDB affiche 204 | No content.

Quand les données sont enregistrées, la colonne 'time' est mise à jour au format Unix, c'est-à-dire le temps en secondes depuis le 1er janvier 1970.

J'ai remarqué avec un routeur de test que lorsqu'il était débranché, sur Grafana il n'y avait pas 100 % de paquets perdus, mais les dernières valeurs enregistrées quand il était encore branché. Le problème venait du fait que lorsque des variables étaient vides, InfluxDB ne prenait pas en compte les requêtes et n'affichait pas de message d'erreur qui aurait pu permettre d'identifier le problème.

Ce que j'ai fait, c'est que si une variable est vide, la valeur est remplacée par la valeur 0. Dans la base de données, j'ai pu maintenant avoir 'lost = 100', 'min = 0', 'max = 0', et 'avg = 0'.

```
# Assignez vos valeurs par défaut a ces variables si elles sont vides

min_default="0"

avg_default="0"

max_default="0"

# Verifiez si les variables sont vides et affectez les valeurs par defaut si necessaire

min="${min:-$min_default}"

avg="${avg:-$avg_default}"

max="${max:-$max_default}"
```

Créer une nouvelle table à partir d'un CRM

J’utilise la même base de données que le premier projet c’est à dire influxDB mais avec une autre table (measurement).

Je reprends la base du script que j’avais déjà fait : Faire un script Bash dans le but de remplacer le projet réaliser par un ancien stagiaire car le projet qu’il a mis en place date de 2020

```
#!/bin/bash

dbuser=voganet

dbname=ping

#source /home/voganet/.bashrc

# Assignez vos valeurs par defaut a ces variables si elles sont vides

min_default="0"

avg_default="0"

max_default="0"


# Execute la requete HTTP pour obtenir toutes les adresses IP dans un fichier texte

curl -k -s https://app.amanela.com/voganet/api/api.php --data > /home/voganet/IP.txt


# Execute fping a partir du fichier ips.txt pour toutes les adresses IP avec 10 pings chacune, les resultats sont stockes dans un fichier

/usr/bin/fping -c 10 -q -f /home/voganet/IP.txt >> /home/voganet/results 2>&1


# Supprime le fichier temporaire contenant les adresses IP

rm /home/voganet/IP.txt


# Recupere les informations que l'on souhaite dans le resultat du fping et les inserent dans la base de donnee

while read -r line; do
```

```
ip=$(echo $line | cut -d' ' -f 1)

lost=$(echo $line | cut -d' ' -f 5 | cut -d'/' -f 3 | cut -d'%' -f 1 | cut -d'/' -f 3)

min=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 1)

avg=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 2)

max=$(echo $line | cut -d' ' -f 8 | cut -d'/' -f 3)



# Verifiez si les variables sont vides et affectez les valeurs par defaut si necessaire

min="${min:-$min_default}"

avg="${avg:-$avg_default}"

max="${max:-$max_default}"



# Construisez la commande curl avec la date et des conditions pour chaque variable (l'espace entre les variables permet d'enregistrer ip en tant que tag et les autres donnees en field)

curl -i -XPOST "http://localhost:8086/write?db=$dbname" -u "$dbuser:${INFLUXDB_PASSWORD}" --data-binary "amanela,ip=$ip lost=$lost,min_ms=$min,avg_ms=$avg,max_ms=$max" > /dev/null 2>&1



done < /home/voganet/results

rm /home/voganet/results
```

La différence réside dans le fait que, au lieu de récupérer les adresses IP dans une base de données MySQL, on les récupère à partir d'un CRM (Customer Relationship Management) en utilisant une requête HTTP. Une autre différence se situe dans la requête pour insérer dans InfluxDB, où dans l'argument --data-binary, on remplace la mesure 'ping' par un autre nom pour enregistrer ces données dans une autre table.