Création du Go

Pour créer un plugin pour telegraf, on doit respecter un format de fichier précis, il doit y avoir au moins **4 fonctions** et **1 structure**:

- **type Powertop struct { }** : C'est la structure de notre fichier CSV, ainsi que celle qui sera envoyée dans notre base de données
- **func (p *Powertop) Description string { }** : On va ici indiquer une description de notre fichier dans un **return « »**.
- **func (p *Powertop) SampleConfig() string { }** : On va pouvoir indiquer des variables personnelles qui seront utilisées dans notre fichier de configurations. Ici on ne passe pas de variables, on va donc le laisser vide avec **return « »**.
- **func (p *Powertop) Gather(acc telegraf.Accumulator) error { }** : C'est le coeur de notre programme, c'est ici que l'on écrire notre code et envoyer nos données dans notre base de données.
- **func init() { }** : C'est la première fonction lancée de notre programme, c'est à partir de là que l'on va lancer toutes nos fonctions, grâce aux pointeurs que l'on a indiqué.

Gather

Dans un premier temps, on importe le fichier. S'il n'y a pas de fichier, une erreur est retournée.

```
csvFile, err := os.Open("/opt/telegraf/plugins/inputs/powertop/powertop.csv")
if err != nil {
    log.Fatal(err)
}
```

Ensuite, on lit le fichier, puis on ferme le fichier, car il est conservé dans la mémoire.

```
reader := csv.NewReader(csvFile)
defer csvFile.Close()
```

On rendre maintenant dans une boucle. On commence par lire le contenu du fichier, que l'on stocke dans la variable **itt** (pour itération). Si l'erreur retournée est en fait la fin du fichier, on sort de la boucle avec un **break**, sinon on affiche l'erreur.

```
for {
    itt, error := reader.Read()
    if error == io.EOF {
        break
    } else if error != nil {
        log.Fatal(error)
    }
```

Ensuite, on va mapper les données lues du CSV dans la variable **fields**. Etant donné que chaque colonne correspond à une donnée, on indique son nom, suivi de son identifiant **itt** avec l'itération => **itt[*]**

```
fields := map[string]interface{}{
    "Usage":    itt[0],
    "Events":   itt[1],
    "Category": itt[2],
    "Desc":     itt[3],
    "PW":       itt[4],
}
```

Finalement, on envoie le contenu de **fields** dans la base de données. On indique d'abord un nom, ici **powertop**, ensuite on indique le **fields**, puis un **tag**. Etant donné qu'on ne donne pas de tag, vu que par défaut il retourne le nom de la machine, ce que nous voulons, on indique seulement **nil**.

```
acc.AddFields("powertop", fields, nil)
```

powertop.csv

Ici, on va extraire le fichier généré par notre powertop en csv, on ne va conserver que le top 10.

Il est séparé en 5 catégories :

- **Usage** : le pourcentage du processeur utilisé
- **Events** : Les IOPS
- **Category** : Le type de processus qui tourne
- **Description** : Le PID et le chemin du programme exécuté
- **PW** : La consommation électrique

Difficultés

- powertop.go

Error: Path

Une difficulté rencontrée était celle du chemin du fichier, si le fichier **powertop.csv** n'est pas dans le répertoire du binaire, telegram nous ressort une erreur, d'ou l'intérêt d'indiquer une valeur absolue pour le chemin.

Error: string

Les données envoyées étant considérées comme des **string**, on ne peut donc pas récupérer la données, il faut donc changer la donnée pour **float64**.

- /usr/bin/telegraf

Binaire

Pour exécuter le binaire, le choix a été fait de l'ajouter dans le /usr/bin pour pouvoir le lancer de n'importe ou, mais cependant le **globalnet.conf** ne fonctionne pas, il faut donc indiquer une valeur absolue pour le chemin.

/dev/null

Si on lance le script encapsulé dans un **/bin/bash « »**, l'indicateur **>/dev/null 2>&1**, qui permet de ne pas avoir de retour dans le terminal (sinon telegraf ne libère pas le conteneur) et le **&** à la fin (pour lancer le programme dans un processus en tâche de fond) ne fonctionne pas, il faut donc s'en passer. Si on lance le script encapsulé dans un **/bin/bash « »**, l'indicateur **>/dev/null 2>&1**, qui permet de ne pas avoir de retour dans le terminal (sinon telegraf ne libère pas le conteneur) et le **&** à la fin (pour lancer le programme dans un processus en tâche de fond) ne fonctionne pas, il faut donc s'en passer.