Casanova Delfeil
de Germiny Vincent

README

I Collecte d'information:

Résumé:
Tâches réalisées:
5 sondes: utilisation du cpu, temperature, utilisation de la ram, utilisation des disques et le nombre d'utilisateurs.
la sonde collectant le nombre d'utilisateurs est exclusivement en bash, la sonde collectant les températures est en python, et les autres sondes mélanges du python et du bash.
En plus de collecter les informations principales (température, utilisation du cpu…)
Chacunes de ces sondes collectes des informations complémentaires utiles pour la gestions des fichiers de données ces infos sont: (le nom de la machine, la date de la collecte.)

Pour réaliser ces sondes, nous avons utilisé les librairies python:
-json: pour la mise en forme
-psutil: pour récupérer les données principales comme la température, utilisation du cpu, de la ram et des disques.
-datetime: pour collecter les informations de datation.
-os: pour récupérer le nom de la machine sur laquelle les informations sont collectés.

collecteur.sh:c'est le script qui permet de lancer les sondes puis de récupérer leurs infos et les stocker dans des fichiers json. On peut mettre en argument le nom d'un type de sonde si on ne veut exécuter que celle-ci. En absence d'argument toutes les sondes seront appelées. Le résultat de chaque type de sonde sera stocké dans un fichier json nommé nom_de_la_sonde.json. Si on refait appel au collecteur une nouvelle ligne est ajoutée à la fin des fichiers json correspondant.

destructeur.sh:permet d'effacer manuellement toute la base de données.

2:Stockage et collecte web

Comme expliqué précédemment tous les résultats des sondes seront stockés sur disque dur dans un dossier data contenant des fichiers json. Chaque fichier json agira comme un historique d'un type de sonde en particulier, chaque ligne de celui-ci représentant un appel du collecteur et contenant toutes les infos récoltées lors de cet appel .

garbageCollector.sh: Il supprimme les données trop anciennes dans la base de données. Il effacera tous les résultats de sondes dont l'appel a été effectué à une date antérieure à l'appel du garbageCollector. Le collecteur appelle le garbageCollector à chaque exécution avant de mettre à jour la base de données.

Le parseur web (recup.sh) n'est présent que sur le serveur. Il est initialisé lors de la configuration du serveur en téléchargeant une première fois les données d'alertes accessible sur le lien suivant: https://www.cert.ssi.gouv.fr/feed/
Le parseur web va par la suite retélécharger les données du même lien et comparer les deux versions téléchargées (la précédente et la nouvelle), si une nouvelle alerte est détectée, on récupère la différence entre ces deux fichiers et on la stocke dans un fichiers temporaire. Ensuite on analyse ce fichier pour en retirer des informations importante (comme la date de publication de l'alerte et le titre de celle-ci) qui seront stockées dans le répertoire data, dans un fichier alerteWeb.json.
Et le nouveau fichier html téléchargé prends la place de l'ancien.

De plus le script garbageCollectorWeb, supprime les informations trop anciennes en comparant la date actuelle à la date de publication de l'alerte, et si celle-ci est vieille de plus d'un mois, la ligne est supprimée.

Pour réaliser le parseur web, nous avons utilisé les librairies python:
-json, pour le format de la base de donnée
-datetime et timedelta: pour récupérer la date référencée sur le fichier json et la comparer avec la date actuelle (pour vérifier l'ancienneté de l'alerte)
-os: pour réaliser des modifications dans la structure des fichiers

---
3:Affichage et alerte
---

detect_crise.sh: ce script permet de prevenir l'utilisateur en cas de crise. Il parcours la base de données à la recherche de résultats de sondes ayant dépassé le seuil critique (seuil paramétrable dans le fichier config) et envoie toutes les infos de ces sondes par mail.

mail.sh:met en forme les données récupérées par la détection de crise et les envoie par mail.
Lors de la récupération des alertes web, si une nouvelle alerte est détectée, un mail est envoyé, signalant qu'une nouvelle alertée a étée détectée

display_info.sh: affiche pour chaque utilisateur les infos des sondes les plus récentes. Attention pour déterminer les infos les plus récentes le script ne se base pas sur la date mais sur l'ordre des lignes (la dernière étant normalement la plus récente) ce qui veut dire

qu’on peut avoir des résultats incorrects si on venait à modifier manuellement les fichiers json.

Lors de la configuration du serveur, il est demandé de saisir l’adresse mail de réception des mail, et l’adresse qui enverrais les mails, or cette dernière adresse doit être une adresse gmail car nous utilisons dans nos scriptes le protocole smtp de gmail.
Les scriptes gérant les mails d’alertes sont en python et utilisent les librairies:
-json: pour récupérer les informations concernant les différentes adresses mail,
-smtplib: pour envoyer les mails.

Le script graphe: permet de générer des graphes au format svg à partir des données présentes dans les fichiers contenant les informations des sondes.
Plusieurs arguments peuvent-être passés pour récupérer que certains graphes en particulier:
-vide: on fait tous les graphes possible de tous les utilisateurs
-1 argument:(type ou host) soit le type de graphe pour tous les utilisateurs
soit tous les graphes de tel machine
-2 arguments:(type, host) on fait le graphe de la machine et du type choisi
Ce scripte est en python et il utilise les librairies:
-json: pour récupérer les différentes informations des fichiers générés par les sondes,
-os: pour gérer les différents fichiers,
-pygal: pour générer les graphes au format svg,
-sys: pour gérer les différents arguments passés lors de l’appel du scripte.

4: Communication

Du côté des utilisateurs, nous avons le script requêtes qui se charge d’envoyer le contenu des fichiers json, remplis par les sonde à un serveur par le biais de requêtes http, vers différents fichiers type.php (selon le type d’infos envoyé:cpu, température, ram,...), l'adresse du serveur est saisie lors de la configuration de la partie utilisateur. Ces fichiers php sont situés dans le dossier web-service du serveur.

Sur la partie web Service du serveur, nous retrouvons différents fichiers .php recevant les requêtes http des différents utilisateurs, et complétant les fichiers json contenant les résultats des sondes selon les différents types de données (cpu, température,..) nous avons un fichier php par type de données.

Pour réaliser le script python gérant les requêtes, nous avons utilisé les librairies:
-json: pour récupérer l’adresse du serveur
-os:
-requests: pour envoyer des requêtes de type post à des fichiers .php situés sur un serveur

---
Configuration:

Deux fichiers de configurations sont disponibles (un pour le serveur et un pour l'utilisateur). Ceux-ci permettent de définir plusieurs paramètres nécessaires au bon fonctionnement de l'ensemble, pouvant varier selon le système et le choix de l'administrateur (plus de détails dans les commentaires du script configServ.sh et configUsers.sh).
Ces fichiers permettent également à l'administrateur de modifier la crontab afin d'exécuter certains scripts à intervalle régulier. Celui-ci peut saisir la fréquence désiré (au format cron) ou alors laisser la config de base (plus de détails dans les commentaires du script).

Pour faire fonctionner le système, il faut:
-Du côté des utilisateur déposer le répertoire Utilisateur, puis se rende sur celui-ci à l'aide d'un terminal et de la commande cd, puis il suffit de lancer le script configUsers.sh.

-Du côté serveur: il faut déposer le contenu du répertoire Serveur, dans un serveur web, puis de se rendre dans le répertoire et de lancer le script configServ.sh.