Mini projet

Présentation générale

L'objectif de ce mini-projet et de réaliser, en binôme, une plate-forme logiciel de monitoring d'un parc informatique. Cette plate-forme, une fois installée, permettra à l'administrateur système que vous êtes de connaître rapidement à chaque fois que vous le demandez l'état détaillé de l'ensemble des serveurs faisant partis du parc et les dernières alertes du C.E.R.T. (http://www.cert.ssi.gouv.fr/). Le logiciel permettra aussi d'être prévenu par e-mail en cas de situation de crise. Une situation de crise étant par exemple : quand un serveur n'a pas donné signe de vie depuis plus de 30 minutes, quand un disque dur atteint 100% d'occupation ou lorsque la RAM est utilisée à 100% , etc …

Ce mini projet doit être réalisé en BASH et en Python, la proportion de chacun des langages dans le projet reste à votre appréciation. Cependant, certain modules auront un langage imposé. Vous pourrez utilisez l'ensemble de la librairie standard python et tous les binaires installés par défaut sur une distribution Ubuntu serveur. Les librairies tiers python et bash sont acceptées si vous avez l'accord explicite de votre tuteur. Nous maintiendrons une liste officiel des librairies utilisable sur le forum du cours sur e-uapv.

Rendu

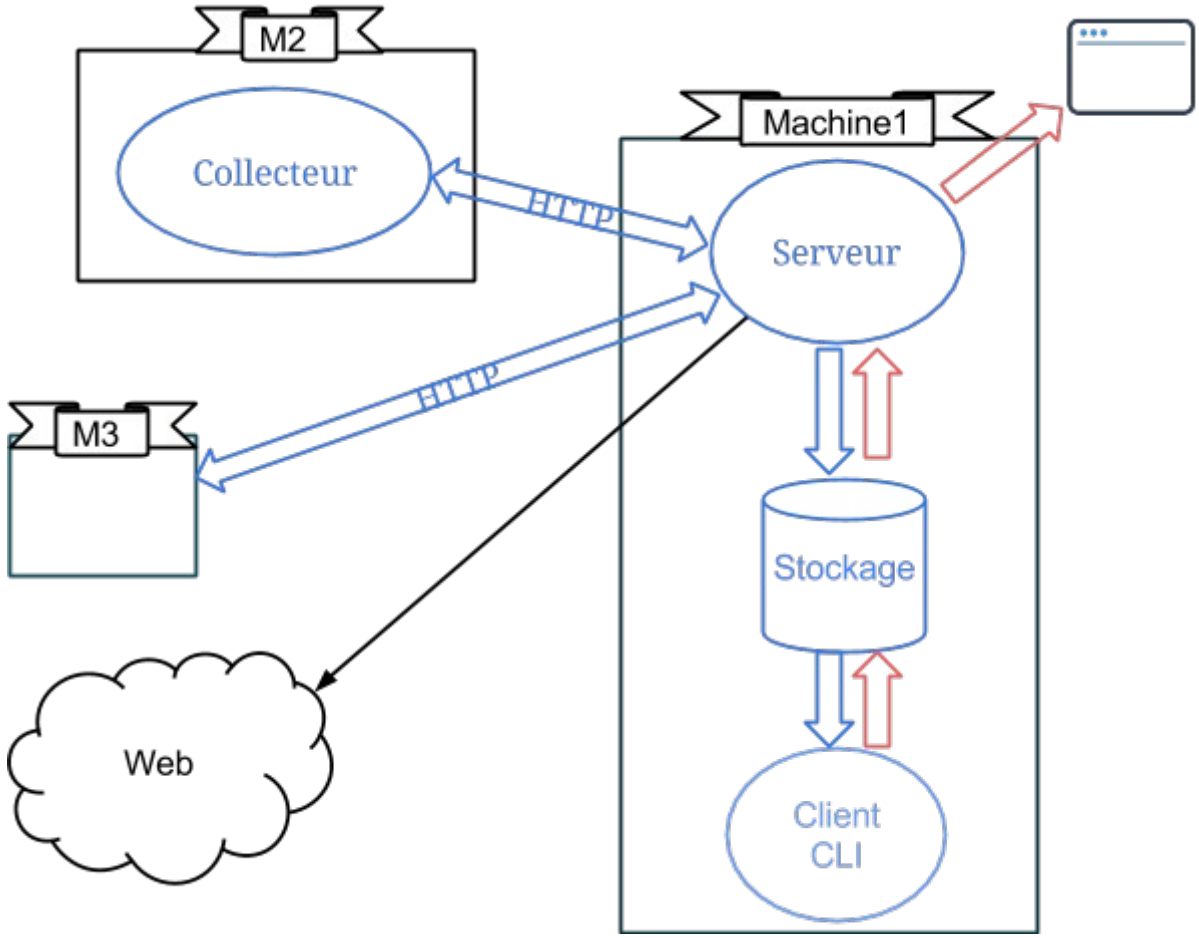
Le projet est à réaliser en binôme. Il vous sera demandé de nous rendre une archive avec l'intégralité de votre code source. Vous devez joindre également un README expliquant les options de fonctionnement de votre projet, les fonctionnalités bonus que vous avez choisis de développer et les librairies que vous avez utilisées.

Notation

Si vous ne faites que les fonctionnalités de base, votre note maximale sera 15 (à condition bien sûr que tout soit juste et fonctionnel). Les 5 points manquants jusqu'à 20 seront rajoutés suivant les fonctionnalités bonus que vous rajouterez.

Description spécifique

Le logiciel sera découpé en quatre morceaux indépendants mais interconnectés. De cette manière si une briques pose problèmes le logiciel peut toujours être un ensemble cohérent et fonctionnel. L'absence d'une brique ne sera donc pas bloquante lors de la poursuite du TP.



I. Collecte d'information

Le premier morceaux est un collecteur (ou sonde) qui va être installé sur chacune des machines du parc à surveiller. Le rôle de la sonde est de récupérer et afficher/envoyer à intervalles réguliers (toutes les 1 à 5 minutes) et de façon formatées les informations de la machine monitorée.

Taches à réaliser :

- Mettre en place au moins trois sondes collectant un ensemble d'informations sur le système. (CPU / disque / RAM / nombre de process / nombre de users connecté …)

Contraintes :

- Au moins une extraction d'information en Bash
- Au moins une extraction d'information en Python
- Au moins une extraction d'information en mixte

Options Bonus :

- Le nombre et la complexité des informations collectées seront pris en compte.

Librairies :

- psutil (py)

II. Stockage & Collecte web

Le second élément est un gestionnaire de stockage et d'archivage qui est capable de recevoir des informations dans un format d'échange défini, de stocker ces informations de garder un historique de taille défini et de nettoyer les informations obsolètes. Cet outil devra aussi permettre un accès simple et exploitable aux informations qu'il contient.

Taches à réaliser :

- Un moteur de stockage de données avec gestion d'historique
- Un parseur web (http://www.cert.ssi.gouv.fr/) qui va récupéré la dernière alerte CERT et l'envoi au moteur de stockage

Contraintes :

- Les données doivent être stockées sur disque dur
- Les données trop anciennes doivent être supprimées

Options Bonus :

- L'utilisation de format standard ou d'un moteur de base de données sans serveur
- Prévoir un cas de sauvegarde/restauration
- L'ajout d'une nouvelle machine dans le réseaux ne nécessite pas de modification manuelle du code ou de la base.

Librairies :

- rrdtool (un outils Génial pour gérer les time série très puissant c'était un Standard dans l'industrie pendant les année !) (py et bash)
- sqlite (py et bash)
- (g)dbm (py et bash) (une Bdd NoSQL ancêtre du sql qui est ultra vielle et fonctionne sur tout les système)

III. Affichage & Alerte

Le troisième package est un script de visualisation. C'est une interface de visualisation en ligne de commande qui affichera pour chaque serveur le dernier état connu, un graphe d'évolution pour les données aillant un historique, ainsi que la date depuis la dernière communication. Ce package intègre aussi des éléments de contrôles lui permettant de détecter une situation de crise et de prévenir l'administrateur par e-mail.

Taches à réaliser :

- Un module de détection de situation de crise
- Un module d'envoi de mail
- Un module d'affichage

Contraintes :

- Le module d'affichage doit être capable afficher dans le terminal les informations pour toutes les machines connues.
- Il affichera aussi les historiques sous forme de graphique

Options Bonus :

- Affichage en couleur ou on mode interactif (top)
- Critère de la situation de crise configurable
- Taille maximum de l'historique configurable
- Contenu de l'e-mail paramétrable (fichier template)
- Envoie des email via le serveur smtp de l'université

Librairies :

- rrdtool (python et bash)
- gnuplot
- pygal

IV. Communication

Le quatrième volet est une couche de communication (web-service) qui permettra à l'ensemble des différents éléments de communiquer de manière distante.(il remplacera un système de communication local mis en place dans les TP précédents). Cette couche permettra à un ensemble de N machines monitorés (noeud) de communiquer avec un serveur de consultation (tête). Ce package peut aussi fournir en plus de web-services un interface de consultation web

qui reprendra les mêmes informations que le client CLI avec des graphiques plus imagé et potentiellement un complément d'information.
Pour ce module l'utilisation de librairies tiers est encouragée.

Taches à réaliser :

- Un module web-service qui attend les requêtes des sondes et communique avec le module de stockage
- Un module de requêtes qui est utilisé par les sondes pour communiquer avec le module web-service

Contraintes :

- La communication se fait via le protocole http
- Il affichera aussi les historiques sous forme de graphique

Options Bonus :

- Un module d'affichage web qui reprend les informations du client CLI.

Librairies :

- netcat (bash)
- httpie (bash)
- wget (bash)
- curl (bash)
- saxon (bash)
- mush (template enfin bash)
- flask (python)
- request (python)
- beautifulsoup4 (python)
- jinja (python)
-

Timeline

Vous aurez 5 séances de TP pour faire le projet, et il est attendu de vous que vous travaillez également l'équivalent de 3 séances chez vous, soit au total 21h de travail. Voici certains repères, donnés à titre indicatif, pour vous aider à planifier votre travail :

- Séance 1 : Collecte d'information et stockage
- Séance 2 : Collecte web et alerte
- Séance 3 : Affichage
- Séance 4 : Communication
- Séance 5 : Finalisation

N'oubliez de faire des backups de vos travaux, ce serait dommage d'avoir 0 parce que les données auraient été curieusement effacées la veille du rendu...