Rapport SAE51

Rédigé par :

* Maxime VALLET
* Erwann MADEC
* Gabin PETITCOLAS

**Sommaire**

[I) Contexte 3](#_Toc188555877)

[A) Importance de la supervision des serveurs 3](#_Toc188555878)

[B) Avantages de la supervision 3](#_Toc188555879)

[C) Solution proposée 3](#_Toc188555880)

[D) Solutions similaires 3](#_Toc188555881)

[II) Solution développée 4](#_Toc188555882)

[A) Outils utilisés 4](#_Toc188555883)

[1) Environnement de développement 4](#_Toc188555884)

[2) Interface utilisateur 4](#_Toc188555885)

[3) Serveur 4](#_Toc188555886)

[4) Client 4](#_Toc188555887)

[5) Déploiement 4](#_Toc188555888)

[6) Tests de fonctionnement 4](#_Toc188555889)

[B) Estimation du travail à réaliser 4](#_Toc188555890)

[1) Méthode d’estimation du temps requis 4](#_Toc188555891)

[2) Estimation du temps requis 4](#_Toc188555892)

[III) Organisation 4](#_Toc188555893)

[IV) Pistes d’amélioration 4](#_Toc188555894)

I) Contexte

A) Importance de la supervision des serveurs

Les serveurs occupent une place centrale en entreprise car ils hébergent des données et services critiques à son bon fonctionnement. Il est donc nécessaire de les surveiller afin d’assurer la meilleure qualité de service et d’améliorer leur durée de fonctionnement.

En effet, parmi ce qui peut être hébergé on retrouve :

* Active Directory : comprend tous les profils utilisateurs des salariés et peut aussi stocker une partie variable de leurs données.
* Outils utilisés en interne : système de pointage, serveurs VPN notamment pour faire du télétravail ou encore des applications métier
* Site Web de l’entreprise

B) Avantages de la supervision

La supervision permet de surveiller des métriques précises afin de prévenir de futures pannes ou bien d’être informé de problèmes en temps réel.

Par exemple, nous pouvons monitorer la quantité de stockage utilisée afin de pouvoir être prévenu en avance et mettre en place des mesures préventives afin que cela ne se transforme pas en un problème d’une plus grande envergure (impossibilité d’utiliser le serveur sans purger des fichiers manuellement dans ce cas).

Dans le cas énoncé précédemment, cela laisserait assez de temps pour faire le tri dans ce qui est stocké ou alors de rajouter du stockage.

C) Solution proposée

Nous avons proposé un outil avec lequel on interagit par le biais d’une page Web afin de le rendre facilement accessible.

Il se base sur une architecture Client/serveur, c’est-à-dire qu’il y’a un serveur qui recueille et traite les données recueillies auprès des clients installés sur les machines surveillées.

Elle inclue un système d’authentification et de gestion des droits d’utilisateurs notamment pour fournir un accès granulaire aux machines qui sont supervisées.

D) Solutions similaires

Parmi les solutions existantes qui réalisent des tâches similaires, on retrouve des outils comme Zabbix ou OpenNMS.

La différence majeure entre l’outil que nous avons développé et ceux préexistant réside dans la façon dont il récupère les informations auprès des machines monitorées. En effet, nous utilisons Java (avec la bibliothèque Oshi) contrairement au protocole SNMP.

II) Solution développée

A) Outils utilisés

1) Environnement de développement

Afin de rendre le développement du projet le plus simple et accessible possible, nous avons opté pour des méthodes de déploiement variées et adaptées pour tous les membres du groupe.

Parmi ces méthodes de déploiement, on retrouve des VM contenant un environnement entièrement configuré, des instructions manuelles mais aussi un script qui installe le projet sans interaction nécessaire.

Voici le tableau contenu dans la documentation, permettant de guider la personne installant le projet vers la méthode d’installation la plus adaptée au contexte.

Une image contenant texte, capture d’écran, espace, ligne

Description générée automatiquement

Figure 1 : Méthodes d'installation

2) Interface utilisateur

L’interface utilisateur a été développée avec les outils suivants : HTML, CSS et JavaScript.

3) Serveur

Le choix qui a été fait côté serveur, a été d’utiliser des outils que nous avions utilisé lors de TP ou projets précédents afin d’optimiser au mieux le temps requis pour réaliser ce projet.

Pour le serveur Web, nous avons utilisé Nginx, un programme très populaire et simple d’utilisation. Sa principale tâche est de distribuer le contenu statique du site Web (HTML, CSS, JS et images).

Afin de pouvoir interagir avec le serveur, nous avons utilisé Tomcat, un serveur Web qui peux faire tourner un projet Java. Son objectif est de faire tourner l’intégralité du serveur (tout est développé en Java) et de permettre de permettre des interactions par le biais du serveur Web grâce à des Servlets.

Afin de stocker les données (utilisateurs, machines…), nous avons utilisé un serveur PostgreSQL.

4) Client

Le client se présente sous la forme d’un projet Java qui récupère les métriques ainsi que la configuration du PC grâce à la librairie nommée Oshi.

Pour les ordinateurs Windows, le programme est lancé par un script PowerShell au démarrage de la machine et pour Linux, il est lancé par Systemd.

5) Déploiement

Pour le déploiement, nous avons créé des conteneurs Docker pour chaque serveur.

Docker est un outil qui permet de créer des conteneurs qui contiennent chacun leur environnement ainsi qu’un unique programme sur lequel il s’exécute. L’objectif de Docker est de créer un environnement léger et déployable n’importe où sans problème de compatibilité (à partir du moment où il est possible d’installer Docker, il est possible de faire tourner le projet).

Un avantage majeur est aussi que les conteneurs Docker peuvent être déployés sur GitHub, rendant le partage du projet très simple.

6) Tests de fonctionnement

B) Estimation du travail à réaliser

1) Méthode d’estimation du temps requis

2) Estimation du temps requis

III) Organisation

IV) Pistes d’amélioration