Hugues Justin

Baux Mattéo

Galvani Guillaume

Alves Lucas

Plan de test Semalynx/SemaOS

La société C-MA-4 est une société qui offre à ses clients des services d'infogérance, d'audit, de conseil et d'intégration de services informatiques. La société souhaite développer un nouveau produit, une « box » installée chez chaque client pour surveiller leur réseau, collecter des données et permettre une gérance à distance. Cette application présente sur les SemaBox se nommera SemaOS. Les techniciens de C-MA-4 pourront se connecter sur une interface WEB nommée SemaLynx, cette application WEB hébergé directement sur le réseau de C-MA-4 pourra recueillir l'intégralité des informations de la SemaBox et de son réseau et pouvoir être utilisée pour piloter à distance certaines fonctionnalités de SemaOS.

1) Fonctionnalités :

Dans un premier temps, nous devons lister toutes les fonctionnalités présentes dans nos applications SemaOS et SemaLynx :

Fonctionnalité SemaOS :

- Possibilité de lancer un scan réseau
- Possibilité de lancer un scan de débit (Download/Upload en Mbps)
- Possibilité de récolter les informations réseau de la Semabox (IP publique, IP privée, Hostname)
- Possibilité d'effectuer un test de ping (latence)
- Le test de ping se lance automatiquement toutes les 5 secondes
- Envoie des données récoltés grâce à des routes
- Reboot de la machine

Fonctionnalité SemaLynx:

- Tableau de bord
- Liste des SemaBox
- Receuil des informations des routes Flask de SemaOS
- Redémmlarage à distance de la Semabox

1) Cas d'utilisation :

SemaOS :

Scan réseau :

- L'utilisateur peut effectuer un scan réseau depuis l'interface de SemaOS, il peut effectuer cette manipulation en ayant un accès internet valide, cela lui retournera alors le résultat de son scan réseau

- L'usager peut effectuer un scan réseau depuis l'interface de SemaOS, il peut effectuer cette manipulation en ayant un accées internet invalide, cela lui retournera alors une erreur mais cela n'empchera pas le fonctionnement global de l'application.

- L'usager pourra retrouver le résultat de son p^récedent scan réseau sur l'onglet "Résultats du dernier test".

Scan de débit :

- L'usager peut effectuer un scan de débit depuis l'interface de SemaOS, il peut effectuer cette manipulation en ayant un accées internet valide et pendant que l'API fonctionne (service externe), cela lui retournera alors le résultat de son test de débit avec une entrée Download et une entrée Upload.
- L'usager peut effectuer un scan de débit depuis l'interface de SemaOS, il peut effectuer cette manipulation en ayant un accées internet invalide ou pendant que l'API n'est pas accessible (service externe), cela lui retournera alors une erreuir mais cela ne bloquera pas le reste de l'application.
- L'usager pourra retrouver le résultat de son p^récedent test de débit sur l'interface d'acceuil

IP publique, IP privée, Hostname :

- L'utilisateur doit pouvoir voir ces infomrations sur l'interface de SemaOS, ceci se produit sans besoin d'action de la part de l'usager.

Test de ping:

- Le test de ping toi pouvoir analyser la latence de la connexion de la SemaBox
- Le test de ping doit s'effectuer toutes les 5 secondes
- Le résultat du test de ping sera afficher sur le tableau de bord de SemaOS

Routes :

- Toutes les données récoltés par les foncttionnalités précedente doivent être envoyées dasn un serveur WEB créer par une route Flask
- Une route qui écoute une requête POST doit permettre de redemarrer la SemaBox

SemaLynx :

Liste des semabox :

- L'utilisateur doit avoir accès à une liste de toutes les SemaBox qu'il peut manager

Tableau de bord :

- L'utilisateur ddoitr pouvoir accéder à un tableau de bord sur chacune des semabox après avoir cliquer sur celle-ci

Recceuil des inforamations SemaOS :

- L'appliaction doit pouvoir récupérer les informations présentes dasn les routes Flask de SemaOS
- Les données doivent être continuellment mise à jour

Requet POST pour redémarrage de la SemaBox:

- Une route doit être créer coté semalynx pour envoyer une requête POST sur la route d'écoiute du SemaOS, suite à cela, la SemaBox doit redémarrer

2) Tests :

Maintenant que nous avaons énumérer les différentes fonctionalités et les différents cas d'utilisation de nos applications, nous allons pouvoir imaginer le type de test approprié pour répondre à nos problèmatqiues.

SemaOS :

Scan réseau :

Afin de tester notre scan réseau, nous allons utiliser des tests unitaires, en effet nous allons tester la fonction scan() de façon solitaire afin d'isoler cette fonctionnalité.

Le test unitaire devra :

- Vérifier que la fonction teste la connexion internet

Si internet :

- La fonction devra utiliser la librairie nmap afin d'effectuer son scan réseau
- Suite à ce scan réseau, la fonction devra essayer de joindre les adresses IP trouvées aux potentiels nom d'hôte. (Si pas de nom trouvé affichera unknown)
- Une fonction regex devra récupérer les informations qui nous intéressent
- Une fois les résultats récupéré et traduit, ils doivent être stockés dans un fichier .txt afin qu'il soit lu et envoyé sur SemaLynx

Si pas internet :

- Affiche une erreur sans paralyser le reste de l'application

Scan de débit :

Afin de tester notres test de débit bous allons égalkemnt utliser des tests unitaires, en effet nous allons tester la fonction test_debit() de façon solitaire afin d'isoler cette fonctionnalité.

Le test unitaire devra :

- Vérifier que la fonction teste la connexion internet

Si internet :

- La fonction devra démarrer l'exécutable speedtest
- Une fois le scan terminé, les résultats seront analysés grâce à une fonction regex afin de récupérer les informations qui nous intéressent (en l'occurrence les valeurs download et upload)
- Une fois les résultats récupéré et traduit, ils doivent être stockés dans un fichier .txt afin qu'il soit lu et envoyé sur SemaLynx
- Les résulats doivent être afficher dans le tableau de bord et envoyé à SemaLynx

Si pas internet :

- Affiche une erreur sans paralyser le reste de l'application

Test de ping :

Le test de ping aura le même scénario de test que les fonctionnalités précédentes, avec des tests unitaires.

Routes des variables :

Le test unitaire devra :

- Vérifier la création de la route concerner (/variables ou /scan_result)
- Une fois la route créer elle devra “héberger” une url (serveur web flask)
- Les variables seront ensuite transformées en JSON afin d’être envoyé sur cette URL
- Les résultats doivent être stocker dans celui-ci

Route de restart :

Le test unitaire devra :

- Vérifier la création de la route concerner (/restart)
- Une fois la route créer elle devra “héberger” une url (serveur web flask)
- Vérification de l’écoute de la requête POST
- Lors de l’arrivée d’une requête POST, effectue une commande de reboot dans un CMD en arrière-plan.

Une fois tous ces tests unitaires réalisés, nous allons pouvoir mettre en place un test d’intégration, nous allons donc tester une à une chacune des fonctionnalités ensemble afin de vérifier le von fonctionnement de notre application. Le test se déroulera dans cet ordre :

- IP privée, IP publique, Hostname
- Test de ping
- Scan réseau
- Test de débit
- Test des routes variables
- Test de la route restart

Semalynx :

Liste des SemaBox :

Pour faire fonctionner la liste des semabox nous utilisons la lecture d’un fichier CSV répertoriant les semabox de notre réseau avec leurs adresses IP. La fonction devra lire ce fichier pour mettre à jour les informations de la liste des semabox.

Nous allons dans ce cas utiliser un test unitaire afin de vérifier le bon fonctionnement de notre fonction.

Le test se déroulera comme ceci :

- La fonction doit ouvrir un fichier CSV
- Elle lit son contenu

- Elle compare soin contenu avec les informations actuelles

Si les informations sont les mêmes :

- Ne fait rien

Si les informations sont différentes :

- Actualise ses informations
- Enregistre les nouvelles informations dans le fichier CSV (fichier de l'état actuel de la liste)

Recueil des informations des SemaBox (route SemaOS) :

Pour récupérer les informations présentes sur les routes de SemaOS, nous allons également créer des routes du côté de SemaLynx qui feront des requête GET sur les URL contenant les variables, nous pourrons ainsi récupérer les données qui nous intéressent.

Le test unitaire devra :

- Vérifier la création des routes concerner (/var_result /scan_result)
- Une fois la route créer elle devra envoyer une requête GET sur l'URL des routes SemaOS
- Lors de la récupération des données, les affiches sur le tableau de bord grâce à TKINTER

Route restart :

Enfin, nous devons tester le bon fonctionnement de la route /restart, nous avions du coté SemaOS, créer une route /restart. Cette route nous permet d'écouter une requête POST afin d'exécuter une commande de reboot en arrière-plan. Nous avions déjà testé le bon fonctionnement de l'écoute de la requête POST et de l'exécution de la commande reboot, nous pouvons donc nous permettre de tester seulement l'envoie de requête POST du coté SemaLynx.

Un test unitaire sera encore utilisé pour cela :

- Vérifier la création des routes concerner (/restart_agent)
- Une fois la route créer elle devra envoyer une requête POST sur l'URL de la route /restart de SemaOS

Une fois que nous avons terminé tous ces tests, nous pouvons comme précédemment utiliser un test d'intégration afin de tester toutes ces fonctionnalités ensemble.

Lorsque que tous les tests unitaires et d'intégration sont réalisés, nous pouvons maintenant faire un test end to end afin de tester nos applications ensemble dans un environnement proche de celui de production.

Si nous voulons mettre à jour notre application à l'avenir, nous devrons effectuer en plus des tests présentés précédemment effectué un test de non-régression pour s'assurer que les nouvelles fonctionnalités n'empiètent pas sur celle précédemment sorties.