Article
Lorenzo Scalese · Oct 14, 2022 6m de lecture

Exemple d'utilisation d'IRIS Embedded Python avec Azure Service Bus (ASB)

Présentation générale

Il y a trois ans, nous avons commencé à utiliser Azure Service Bus (ASB) comme solution de messagerie d'entreprise. Elle est utilisée pour publier et consommer des données entre de nombreuses applications de l'entreprise. Comme le flux de données est complexe et les données d'une application sont généralement nécessaires à plusieurs applications, le modèle "éditeur" ---> "abonnés multiples" était parfaitement adapté. L'utilisation d'ASB dans l'organisation donne des dizaines de millions de messages par jour, tandis que la plate-forme IRIS a environ 2-3 millions de messages par jour.

Le problème d'ASB

Lorsque nous avons commencé l'intégration d'ASB, nous avons constaté que le protocole AMQP n'est pas " prêt à l'emploi " pour le déploiement d'IRIS. Nous avons donc cherché une solution alternative pour pouvoir communiquer avec l'ASB.

La solution

Nous avons développé un service Windows local (en .NET et Swagger) qui effectuait la communication réelle avec l'ASB. Il était installé sur la même machine qu'IRIS. Nous avons envoyé les messages à ASB à ce service local (en utilisant : localhost et en faisant une API REST), et ce service a fait le reste. Cette solution a bien fonctionné pendant quelques années, mais il était très difficile de surveiller le trafic, de déboguer les messages "perdus" et de mesurer la performance globale. Le fait que nous avions ce service Windows local comme "un homme au milieu" n'est pas toujours la meilleure architecture.

Programme d'accès anticipé à IRIS Embedded Python (EAP)

On m'a demandé (ou on m'a proposé, je ne sais plus) de participer au programme de diffusion anticipée (ERP) pour le programme Embedded Python. C'était très excitant pour moi, de pouvoir tester de nouvelles fonctionnalités, de donner du feedback à l'équipe de développement. C'était également très agréable de participer activement au développement de ce produit et de l'influencer. À ce stade, nous avons commencé à tester le programme Embedded Python pour l'ERP et nous avons décidé de vérifier si nous pouvions élargir le programme Embedded Python pour résoudre des problèmes "réels". Prendre la "connectivité directe entre IRIS et ASB" dans ce but était tout à fait logique.

Le POC (preuve de concept****)

Nous avons commencé à coder la solution et avons découvert que l'utilisation de la bibliothèque ASB de Microsoft pour Python nous faciliterait la vie (et le codage). L'étape initiale était de développer une fonction qui peut se connecter à un sujet spécifique dans ASB et recevoir des messages. Cela a été fait assez rapidement (1 jour de développement) et ensuite nous sommes passés à la phase de "publication" (envoi à ASB).

Quelques jours supplémentaires nous ont permis de développer une " enveloppe " complète pour ces fonctions d'envoi et de réception : nous avons construit ce qui suit :

- Une "base de transit" adéquate pour contrôler le flux des messages entrants et sortants
- Un mécanisme central pour stocker le trafic d'entrée et de sortie de l'ASB afin de pouvoir disposer de statistiques et de mesures de performance appropriées
- Une page de surveillance du CSP pour activer/désactiver les services, montrer les statistiques et donner des alertes en cas de problème

La mise en œuvre

Configuration préalable

Avant d'utiliser les bibliothèques ASB de Microsoft pour Python, il est nécessaire de les installer dans un dossier dédié .. \python.
Nous avons choisi d'utiliser ../mge/python/ mais vous pouvez utiliser n'importe quel autre dossier (initialement c'était un dossier vide).
Ces commandes doivent être exécutées à partir d'un CMD élevé (sous Windows) ou en utilisant un utilisateur élevé (sous Linux) :
.. \bin \irispip install --target .. \mgr \python asyncio
.. \bin \irispip install --target .. \mgr \python azure.servicebus

Réception de messages à l'ASB (consommation)

Nous avons une méthode ClassMethod avec quelques paramètres :

- Le topiId - L'ASB est divisée en thèmes à partir desquels vous consommez des messages
- subscriptionName - Nom de l'abonnement à Azure
- connectionString - Pour pouvoir vous connecter à la rubrique à laquelle vous êtes abonné(e)

Notez que nous utilisons [Language = python] pour indiquer que cette ClassMethod est écrite en Python (!)
ClassMethod retrieveASB(topicId As %Integer = 1, topicName As %String = "", subscriptionName As %String = "", connectionString As %String = "", debug As %Integer = 1, deadLetter As %Integer = 0) As %Integer [Language = python]

Pour utiliser la bibliothèque ASB, nous devons d'abord importer les bibliothèques ServiceBusClient et ServiceBusSubQueue :
d' azure.servicebus importer ServiceBusClient,ServiceBusSubQueue

pour pouvoir interagir avec IRIS (par exemple pour exécuter du code), nous devons également :
importer iris

À ce stade, nous pouvons utiliser les bibliothèques ASB :
avec ServiceBusClient.fromconnectionstring(connectionString) en tant que client:
avec client.getsubscriptionreceiver(topicName, subscriptionName, maxwaittime=1, subqueue=subQueue) en tant que récepteur :

pour msg dans récepteur:

À ce stade, nous avons un objet Python "msg" (flux) que nous pouvons transmettre (en tant que flux bien sûr) à IRIS et stocker directement dans la base de données :
result = iris.cls("anyIRIS.Class").StoreFrom Python(stream)

Envoi de messages à l'ASB (publication)

Nous avons une ClassMethod avec quelques paramètres :

- topicName - La rubrique (Topic) sur laquelle nous voulons publier (ici, nous devons passer le nom et non l'identifiant)
- connectionString - Pour pouvoir vous connecter à la rubrique à laquelle vous êtes abonné(e)
- JSONmessage - le message que nous voulons envoyer (publier)

Notez que nous utilisons [Language = python] pour indiquer que cette ClassMethod est écrite en Python (!)
ClassMethod publishASB(topicName As %String = "", connectionString As %String = "", JSONmessage As %String = "") As %Status [Language = python]

Pour utiliser la bibliothèque ASB, nous devons d'abord importer les bibliothèques ServiceBusClient et ServiceBusMessage :
from azure.servicebus import ServiceBusClient, ServiceBusMessage

L'utilisation des bibliothèques de l'ASB est alors très facile :
try:
résultat=1

avec ServiceBusClient.fromconnectionstring(connectionString) en tant que client:

avec client.getqueuesender(topicName) en tant qu'expéditeur:

singlemessage = ServiceBusMessage(JSONmessage)

sender.sendmessages(singlemessage)

sauf Exception comme e:

publier(e)

résultat=0

retourner le résultat

Avantages de l'utilisation de la connectivité directe de l'ASB

- Beaucoup plus rapide que l'"ancienne alternative" du "service local de Windows"
- Facile à surveiller, à collecter des statistiques, à déboguer les problèmes
- La mise hors service de l'"homme du milieu" (service local de Windows) réduit un "point de défaillance" potentiel
- La possibilité de gérer automatiquement les "lettres mortes" pour toute rubrique et de faire en sorte que l'ASB renvoie ces messages à la rubrique de l'abonné

Références

Je tiens à remercier @David Satorres (notre développeur principal pour IRIS) pour son énorme contribution à la conception, au codage et aux tests. Sans son aide, ce projet n'aurait pas été possible.

#Azure #Embedded Python #InterSystems IRIS