Autorisation de diffusion de rapport

Remerciements

Resume

Abstract

Sommaire

Introduction

Depuis sa création en 1864, Société Générale met son expertise au service de ses clients et du financement de l’économie, avec l’ambition de devenir la banque relationnelle de référence.

Le but principal de la Société Générale est de servir ses clients et l’économie.

La Société Générale est l’un des tout premiers groupes européens de services financiers. Avec un ancrage solide en Europe et une présence dans les régions à fort potentiel, les 146 0000 collaborateurs du Groupe et de ses filiales, présents dans 66 pays, accompagnent au quotidien plus de 31 millions de clients particuliers, grandes entreprises, investisseurs institutionnels, dans le monde entier. Ils offrent une large palette de conseils et de solutions financières sur mesure qui s’appuie sur trois pôles métiers complémentaires.

Le premier pôle correspond à la banque de détail (en France), le deuxième regroupe la banque de détail et services financiers internationaux, le troisième pôle comprend la banque de financement et d’investissement, la banque privée et la gestion d’actifs et métiers titres.

J’effectue mon stage dans le troisième pôle métier, dans la banque de financement et d’investissement, qui est un acteur essentiel dans le financement de l’économie à travers son rôle d’intermédiaire auprès de deux types de clients : les émetteurs et les investisseurs. Elle accompagne ses clients sur leurs besoins stratégiques de long terme et à travers quatre services essentiels : le conseil, le financement, la couverture des risques et les solutions d’investissements.

Mon stage consiste à travailler sur une application de Covered Bond System (CBS), au sein de la SGCIB, dans le service ITEC/OSD/FCC….

L’application CBS contribue à l’amélioration de la qualité des données et du pilotage des structures de Covered Bonds.

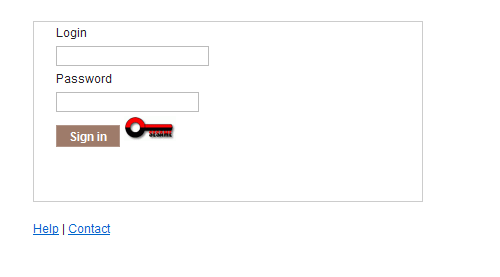
CBS est principalement un outil de monitoring de Covered Bonds, des données venant d’autres systèmes (Horus, GMC, .., etc.) sont injectées dans la base de données CBS via des imports de fichiers. Avant d’être insérées dans les tables de travail de l’application, les données sont importées dans des tables temporaires (<nomTable>\_TEMP\_CBS). Les données sont ensuite validées avant d’être importées dans les tables de travail.

Certaines tables sont auditées (@Audited, @AuditTable). Les tables temporaires sont également auditées car pendant la phase de contrôle des données importées, les utilisateurs peuvent mettre à jour un certain nombre d’information sur les dossiers en écarts. Cela permet d’auditer les modifications faites sur les données importées avant validation.

L’application a été initialement générée par SpringFuse Celerio (qui permet de générer du code Java représentant un modèle de données ainsi que les DAO (**D**ata **A**ccess **O**bject) et les services. Il peut aussi générer des tests unitaires), mais la suite a été codée manuellement.

1. Présentation de l’entreprise
2. Présentation de l’application CBS et de l’agilité

L’application CBS est une application web accessible à l’adresse suivante : [**https://cbs.fr.world.socgen/cbs/login/**](https://cbs.fr.world.socgen/cbs/login/)



1. Implémentation des demandes d’évolution sur l’application CBS
2. Software factory : Création de slaves Jenkins avec ITASS
3. Mise au point sur l’existant

Au sein d’ITEC, l’entité FCC/OSD/SCM gère les projets d’évolution ainsi que la maintenance de toutes les applications (TDA, CBS, B3S, SLA Web, ANTALIS) couvrant ces secteurs.

L’intégration continue est un ensemble de bonnes pratiques utilisé dans l’entité FCC/OSD/SCM afin d’améliorer la qualité du code et le produit final. En effet, à chaque modification du code source, on vérifie que le résultat des modifications ne produit pas de régression dans les applications. Le but principal est de détecter les problèmes d’intégration lors du développement. De plus, cela permet d’automatiser les exécutions des suites de tests, de voir l’évolution du développement du logiciel et d’améliorer la qualité du code source.

Cette intégration repose sur la mise en place d’une brique logicielle, qui permet l’automatisation de certaines taches telle que la compilation, le lancement des tests unitaires et fonctionnels, clean code et DB refresh. L’exécution de cet ensemble de tâches se fait à chaque changement dans le code. Le code source est un code partagé issue d’un répertoire Git (dans la configuration des jobs Jenkins, nous passons l’url du répertoire ainsi que la branche à compiler). Ainsi, à chaque fois qu’on commit nos modifications, les taches sont lancées automatiquement. En ce qui concerne les tests d’intégrations, on utilise JUnit pour valider les applications. L’outil d’intégration utilisé est Jenkins (fork de hudson).

Cependant, les applications de l’entité FCC/OSD/SCM se partageaient le même slave Jenkins (fcc-srvparosdp01-nt) pour les lancements des jobs Jenkins, en environnement de développement. Ce slave correspond à un serveur Windows composé de quartes cœurs.

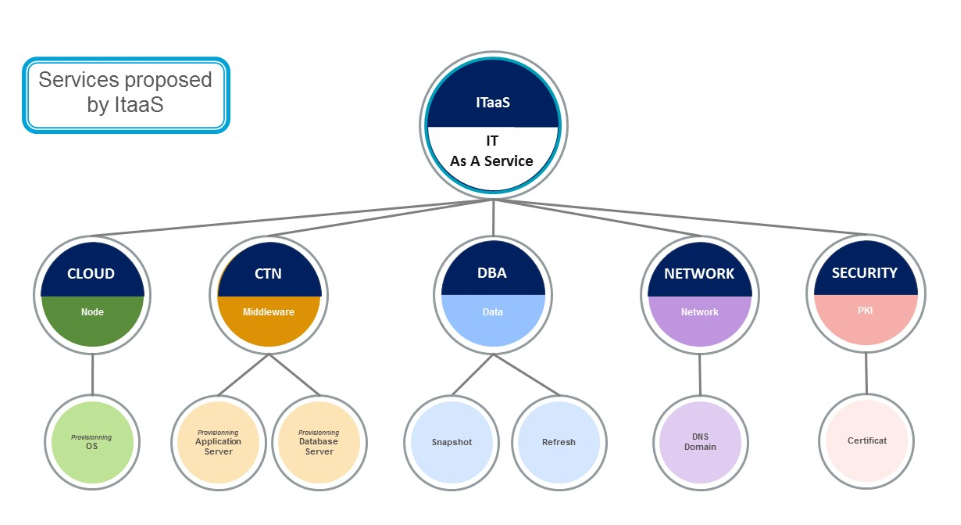
Ainsi, lancer plusieurs jobs en même temps sur ce slave prenait du temps et l’ensemble des jobs ne se lançaient pas en même temps.

1. Quelles évolutions pour quels impacts ?

Le choix a tout d’abord été de créer un slave Jenkins pour chaque application existante. Ainsi, chaque application possédera son propre serveur pour pouvoir lancer ses propres jobs.

Cela permettra de gagner du temps dans la résolution des demandes d’exécution des différentes tâches sur toutes les applications. Le panel d’applications gérées étant particulièrement important, il est nécessaire d’avoir un slave Jenkins pour chaque application.

Pour cela on a utilisé ITAAS (IT As A Service), qui offre un panel de services comme nous pouvons le voir sur ce schéma :



Le service qu’on utilise est le Cloud, qui nous permet de créer nos serveurs linux, qui sont ensuite utilisés comme slave Jenkins.

// Mon travail TDA et CBS

Trois serveurs linux ont été crées. Un serveur pour l’environnement de développement pour CBS, un serveur pour l’environnement UAT, ainsi que sur l’environnement HOM pour TDA.

1. Processus de création d’un Slave Jenkins

Avec ITaas on crée un nœud linux en choisissant :

* Taille : Medium -2vCPU avec 2GB de RAM
* Environnement : Dev

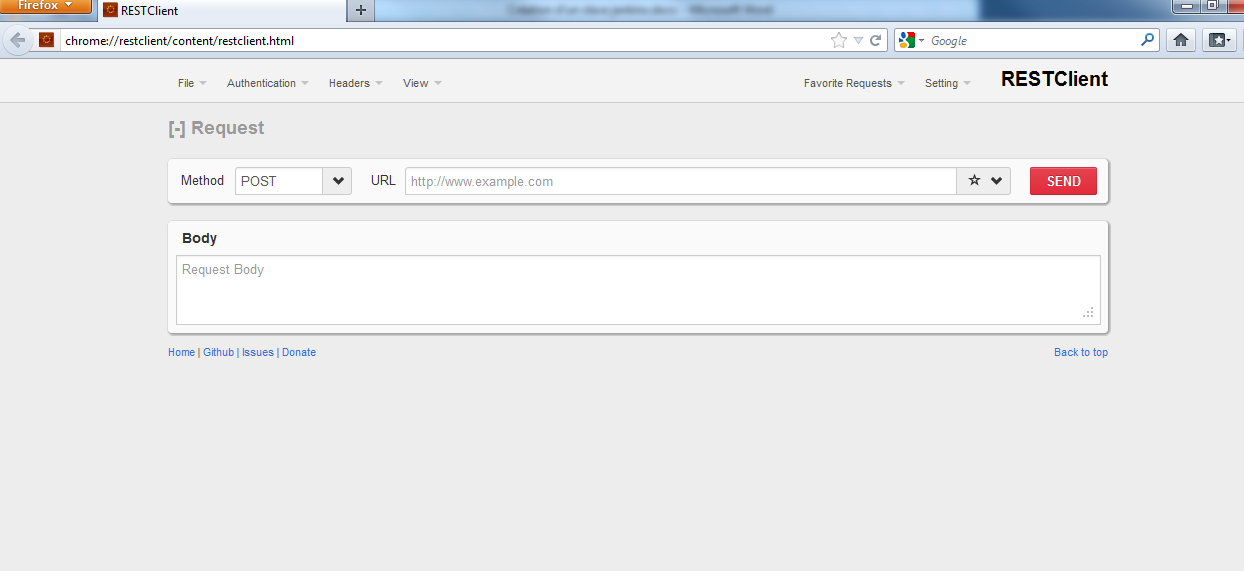
De plus, on fournit une liste d’email. Des notifications sont, ainsi, envoyées par la suite des évènements suivant :

* Lorsque le nœud a été provisionné
* Lorsque le nœud se rapproche de l’expiration. Plusieurs e-mails sont envoyés à l’approche de la date d’expiration :
* 7 jours avant
* 5 jours avant
* 2 jours avant
* 1 jour avant
* Quand il y a moins de 24 heure restantes

Ensuite, on écrit un script Json dans lequel on spécifie tous les paramètres, ce qui correspond à faire une «*request* ».

1. Rest Client

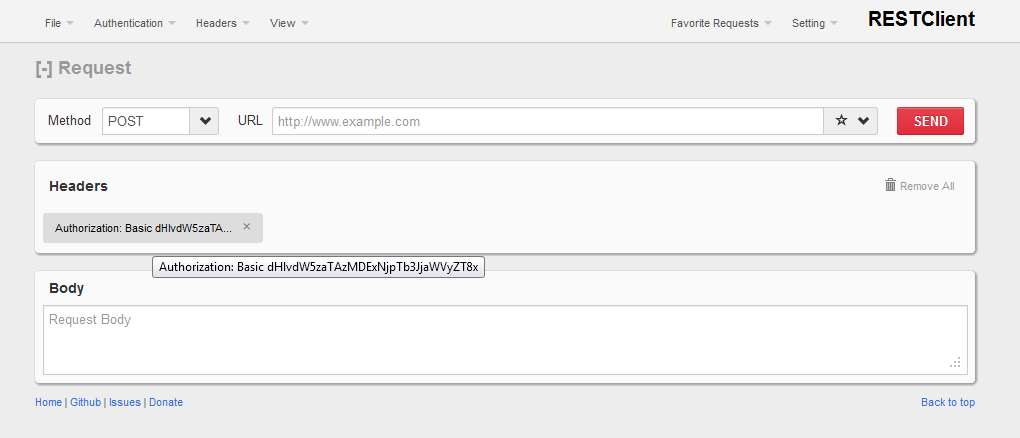
Tout d’abord, je configure Rest Client. [www.restclient.net](http://www.restclient.net) est un plug-in Firefox, testé avec succès avec l’API ITaas.



Figure

La première étape correspond à l’authentification :

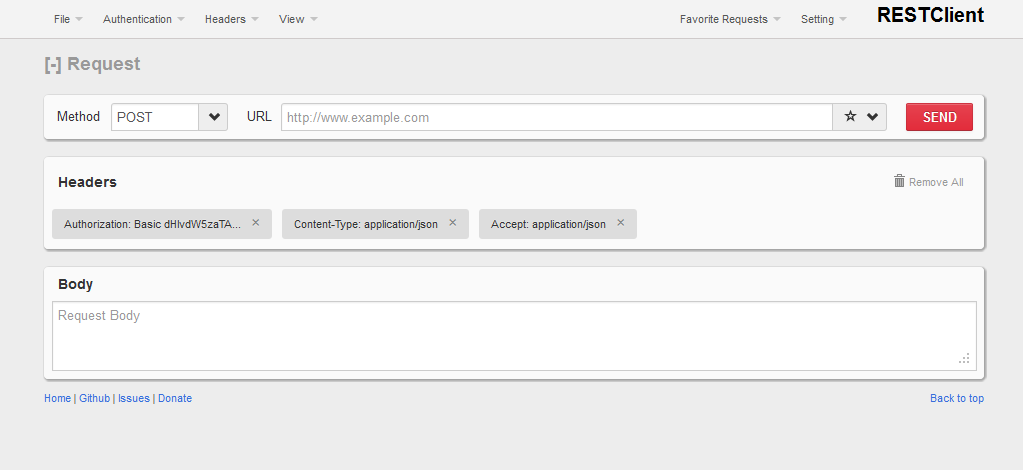
* Dans le menu « Authentication », on sélectionne « Basic Authentication »
* Le login et mot de passe Windows sont utilisés
* La case « Remember me » peut être cochée aussi
* Lorsqu’on clique sur « OK », l’authentification apparait dans la section Headers



Figure

La deuxième étape correspond à la configuration des en-têtes :

* Dans le menu « Headers », on choisit « custum Header »
* Le premier en-tête à ajouter est :
* Nom -> Accept
* Valeur -> application/json
* Le second en-tête à ajouter est :
* Nom -> Content-Type
* Valeur -> application/json
* On coche « Save to favorite »
* En cliquant sur « OK », les en-têtes apparaissent dans la section « Headers »



A la troisième étape, on choisit la méthode d’envoi du formulaire, la méthode POST sera utilisée dans notre cas.

Pour finir, dans la section « Body » on écrit notre requête, un script Json où sont définis tous les paramètres de l’API.

1. Le corps de la requête

Dans cette partie, je vais expliquer toutes les étapes pour configurer tous les paramètres d’un serveur linux.

La documentation de l’API fournit un exemple de script :



Pour configurer le serveur linux dans le Cloud pour CBS, nous avions dû fixer certains paramètres :

* Size : ”M”
* Disk size : “60”, pour les dossiers cotenants maven/git/java …
* Lease days : “-1” afin d’avoir un contrat infini
* environment type : “dev”
* Centrify zone : “zd\_fr\_cbs\_01” crée par GTS group
* Description : Jenkins salve for CBS
* User group : “cbsdev” crée GTS group
* service account : “cbsdev02”
  + On fait la demande de création sur la zone CBS, en spécifiant l’utilisateur "cbsdev02" et le groupe associé (cbsdev). Ensuite, une fois que l’utilisateur est crée, s\_fr\_\_cbsdev02 est ajouté dans le groupe AD de CBS.

En effet, il y’a un groupe AD Windows pour avoir le droit de créer des VM dans le Cloud (SGLOUD) qui est le même pour les actions sur l’API REST de l’ITaas.

* app trigram : « CBS »
* region : eu-fr-paris
* availability zone : eu-fr-paris-1
* subnet : 192.160.64.0/21
* notifymail : [olivier.terrien@sgcib.com, thin-hinane.younsi@sgcib.com](mailto:olivier.terrien@sgcib.com,%20thin-hinane.younsi@sgcib.com)
* master\_os : RHEL 6.7 x64 MKT
* label : slave\_jenkins\_linux\_cbs

Enfin, on lance la procedure dans le Client Rest, avec l’URL:

<https://itaas-prod.fr.world.socgen:10443/v2/nodes>

Nous obtenons la réponse ci-dessus, qui nous renvoie un « id » et «creation\_job\_id ». Ceux-ci nous permettent de lancer une requête GET pour avoir le nom d’hôte du serveur.  Le nom d’hote sera utilisé dans DeployIt.



1. Création d’une clef SSH
2. Conclusion
3. Bilan personnel

Bibliographie