École Polytechnique de Montréal
Département Génie Informatique et Génie Logiciel

LOG8371: Ingénierie de la qualité en logiciel

Hiver 2019

Travail Pratique 1

Groupe : Québec

Matricule	Prénom & Nom
1827022	Hon-Leung Éric Chao
1733073	Alpha Oumar Diallo
2035541	Aladji Faye
2061707	Amara Sountoura

Présenté à : Aquilas Tchanjou Njomou

Polytechnique Montréal

17 Février 2020

Table des matières

1. Plan d’assurance

1.1 Introduction

L’outil que nous allons étudier au cours de ce travail pratique sera Elasticsearch. C’est un outil qui peut être vu et considéré comme un moteur de recherche, mais qui ne s’occupe pas uniquement du texte, mais aussi des chiffres et données lors de la recherche. Au cours de ce travail nous allons utiliser la dernière version de cet outil et nous allons sélectionner quatre modules pour lesquelles on va planifier un processus pour l’assurance qualité. Nous allons d’abord discuter de l’importance de sa qualité, ensuite la prochaine section sera consacré aux parties pertinentes du système. Celle qui suivra discutera des critères de qualités et enfin la dernière section parlera des critères de validation.

1.2 Importance de la qualité du logiciel

L’objectif principal de ce plan est de concevoir un ensemble de plans d’assurance qualité pour le logiciel ElasticSearch, plus spécifiquement de quatre de ses modules. Il nous convient ainsi d’élaborer des stratégies de test afin d’évaluer la qualité du logiciel selon des critères d’assurance qualité données. Tout ceci a pour but de répondre de manière concrète à la question suivante: quelle est l’importance de la qualité du logiciel ElasticSearch dans son contexte général?

Tout d’abord, ElasticSearch est un logiciel qui prend de plus en plus d'ampleur dans l’industrie et est maintenant utilisé par de grandes entreprises tel qu’Uber, Slack et Shopify, ainsi ce dernier manipule des données sensibles. Ces données ne doivent pas fuiter et donc le logiciel ne doit pas pouvoir modifier les documents/données sans action intentionnelle de l’utilisateur.

Par la suite, ElasticSearch a pour but d’afficher des informations précises, pertinentes et cohérentes; le but n’est pas seulement de trouver des résultats; une réponse incomplète ou incohérente peut aboutir à de lourdes conséquences (ex : si l’on veut mettre en vente un produit dans un magasin d’une ville en fonction de la popularité des magasins…)

Ensuite, le logiciel doit être, de base, fonctionnel : il doit pouvoir afficher les résultats rapidement, pour ne pas avoir à faire attendre l'utilisateur. Il doit aussi avoir une certaine fiabilité, c'est-à-dire, qu'il doit être capable de fonctionner correctement selon certaines conditions et pendant une durée raisonnable. Il doit également ne pas faillir ou du moins le moins souvent possible.

Finalement, il doit avoir une certaine aisance d'usage qui va permettre à l'utilisateur d'effectuer ses tâches avec facilité et efficacité. Tout ceci contribue à l'expérience utilisateur qui est important, car c'est ce dernier qui permet au produit de rester compétitive et accessible à tous.

1.3 Critères de qualité et mesure de validation

1.3.1 Présentation des Critères de qualités:

Notre plan d'assurance pour le logiciel **ElasticSearch** sera basé sur trois critères de qualités de la norme **ISO/IEC 25010** et de ses sous-critères. Afin de discerner, avec une meilleure clarté, notre plan plan d'assurance qualité, voici des définitions des critères et sous-critères que nous allons utiliser dans ce plan.

1. **Fonctionnalité (Aptitude fonctionnelle)**: ce critère mesure le degré auquel un système fournit des fonctionnalité qui répondent aux requis et besoins lorsqu'il est utilisé dans des conditions spécifiques. Cette caractéristique est composée des sous-caractéristiques suivantes:
    a. **Exactitude fonctionnelle**: ce sous-critère mesure le degré auquel un système à fournir des sorties de résultats d'opérations exactes et avec une certaine degré de précision.
    b. **Complétude fonctionnelle**: ce sous-critère mesure le degré auquel l'ensemble des fonctionnalités du système est capable de couvrir toutes les requis fonctionnels et non fonctionnels spécifiés par le client et les utilisateurs.
2. **Fiabilité**: ce critère mesure le degré auquel un système ou un composant est capable de fournir les services ou d'exécuter des tâches données dans des conditions spécifiées

pendant une durée fournie sans tomber en panne. Cette caractéristique est composée des sous-caractéristiques suivantes :

a. **Tolérance aux fautes**: ce sous-critère mesure le degré auquel un système ou un composant est capable de fonctionner comme prévu malgré la présence de défauts matériels ou logiciels.
b. **Récupérabilité**: ce sous-critère mesure le degré auquel un système est capable de récupérer des données lors de panne ou d'interruption sans directement affectées le système et de rétablir l'état de ce dernier.
c. **Disponibilité**: ce sous-critère mesure le degré auquel un système, un produit ou un composant est opérationnel et accessible lorsqu'il est requis pour son utilisation.

3. **Maintenabilité**: ce critère mesure la capacité et l'effort auquel un système peut être amélioré, adapté ou corrigé aux changements de l'environnement et des exigences. Cette caractéristique est composée des sous-caractéristiques suivantes:
   a. **Testabilité**: ce sous-critère mesure le degré auquel il est facile de conduire des tests pour le système ou composant. Ceci implique des outils de tests et des rapports de tests.
   b. **Modifiabilité**: ce sous-critère mesure le degré auquel un système est capable d'être modifié de manière efficace et avec facilité sans introduire de défauts ni dégrader la qualité du produit existant.

4. **Efficacité/Performance**: ce critère mesure les exigences d'efficacité relatives qui sont liés aux ressources matérielles nécessaires à l'exécution de toutes les fonctionnalités du système, en conformité avec les autres exigences. Cette caractéristique est composée des sous-caractéristiques suivantes:

   a. **Utilisation de ressource**: ce sous-critère mesure la consommation et la durée qu'une ressource va utiliser lors de son exécution
   b. **Capacité** : ce sous-critère va mesurer la capacité de stockage des ressources ainsi que leurs durée de validité.
   c. **Comportement temporel** : ce sous critère mesure la capacité du système de fournir une réponse dans le temps et dans les conditions désirées.

1.3.2 Modules à tester:

Nous avons sélectionné quatre modules à tester qui vont parcourir notre plan d'assurance qualités.

1.3.2.1 Module : Transport-netty4

Ce module est celui du transport et est utilisé pour la communication entre les noeuds internes d'un cluster et ainsi que pour le *TransportClient* dans l'API Java d'Elasticsearch. Il est utilisé dans la communication asynchrone de données, que ça soit avec le protocole HTTP ou les opérations de diffusion ou de collecte d'informations dans le cas de la recherche dans Elasticsearch.

Critère	Sous-critère	Objectif	Mesure de validation
Fonctionnalité	Exactitude	Le module assure parfaitement la communication entre noeuds internes.	Communication >=99%
Fiabilité	Tolérance aux fautes	Les noeuds doivent pouvoir communiquer malgré une dégradation de la qualité du réseau.	aptitude du module à assurer la communication>85%
Fiabilité	récupérabilité	En cas de panne du réseau, la communication doit être parfaitement rétablie entre les noeuds dès que celle-ci est résolu .	Rétablissement de la communication après panne >= 99%
Maintenabilité	Testabilité	Il doit être possible et facile de tester tout le module.	Le module est couvert par 90% des tests approprié.
Maintenabilité	Modifiabilité	Pour des raisons d'amélioration, le	Facilité de mise à jour >90%

		module doit être facilement ajustable.	
Efficacité/performance	Comportement temporel	Le module doit être capable de fournir une réponse dans un délai raisonnable	Le délai de la réponse est reçu et couvre plus de 99% du module.
	Capacité	Il doit être possible d'obtenir la réponse d'une requête http selon la capacité du CPU.	Réponse obtenu respecte la capacité du CPU de la machine pour le protocole HTTP
	Utilisation de ressource	Les ressources du protocole doivent rester dans la norme.	99% des ressources comparer du protocole Http doivent etre les memes que celles du module.

1.3.2.2 Module : Reindex

Ce module permet d'extraire la source d'un document à partir l'index source et d'indexer les documents dans l'index de destination. Il permet également de travailler avec des ensembles de documents et d'en copier l'ensemble à partir de l'index source ou d'indexer l'ensemble.

Critère	Sous-critère	Objectif	Mesure de validation
Fonctionnalité	Correction fonctionnelle	La réindexation doit se faire dans l'index de destination précisée	Comparaison entre l'index précisé et l'index dans lequel a été fait l'indexation
	Complétude fonctionnelle	La réindexation doit bien se faire	Observation, dans le nouvel index, si la réindexation a bien été faite

Fiabilité	Tolérance aux fautes	Minimiser les erreurs/échecs de réindexation	taux d'échec < 95 %
	récupérabilité	Les réindexations doivent pouvoir se poursuivre après une interruption/défaillance du système; il faut pouvoir récupérer les informations des index	Récupération de 90 % des informations (nouveaux index, index de base) après une panne/défaillance
Maintenabilité	Testabilité	Possibilité de tester le module facilement	Les tests doivent couvrir au moins 90 % des méthodes du module, avec un taux de 90 % de réussite minimum
Efficacité/performance	Comportement temporel	Le module doit être capable de fournir une réponse dans un délai raisonnable	Le délai de la réponse est reçu et couvre plus de 99% du module.
	Capacité	Le nombre d'indexation dans les documents ne diminue pas la rapidité du CPU.	Réponse obtenu respecte la capacité du CPU.
	Utilisation de ressource	Les ressources dans le document indexé doivent être présentés.	99% des ressources se retrouve dans le document indexé.

1.3.2.3 Module Percolator:

Le module Percolator permet d'envoyer des requêtes percolates qui peuvent être utilisées pour faire correspondre des requêtes stockées dans un index.

Critère	Sous-critère	Objectif	Mesure de validation
Fonctionnalité	complétude fonctionnelle	La percolation doit s'effectuer avec l'index de la requête	L'ensemble des requêtes reçues sont différentes pour chaque index
	Correction Fonctionnelle	Exactitude des resultats des documents reçus à partir des requêtes	Marge minimum de nombres d'échecs et nombres d'erreurs lors de l'envoie des documents
Fiabilité	Récupérabilité	Envoyer une requêtes avec un document et vérifier sa réception	Au moins 90% des requêtes sont récupérables
	Tolérance aux fautes	Minimiser les risques de fautes lors d'une percolation	90% des percolations sont réussies
Maintenabilité	Testabilité	Possibilité de tester le module facilement à l'aide de plusieurs tests	Les tests couvrent toutes les fonctionnalités du module
	Modifiabilité	Créer une requêtes et vérifier que celle-ci est bien modifiable	Modification possible sans défaillance
Efficacité/performance	Comportement temporel	Le temps d'envoi d'une requête percolate est raisonnable	99% des requêtes se font dans les délai qui correspondent à la capacité du CPU.
	Capacité	n.d.	n.d.
	Utilisation de ressource	Les ressources dans la percolation ne ralentissent pas le système.	Le système reste encore fonctionnelle après plusieurs percolation.

1.3.2.4 Module: Ingest-Geoip

Le module ingest-geoip est un module qui est en fait un processeur geoip, c'est-à-dire un middleware dans le processus d'ingestion de noeuds, qui permet d'ajouter des informations sur l'emplacement géographique des adresses IP, à partir des bases de données Maxmind. Ce processeur ajoute ces informations par défaut sous le champ geoip.

Critère	Sous-critère	Objectif	Mesure de validation
Fonctionnalité	Complétude fonctionnelle	Garantir la réalisation d'une ingestion Geoip selon toutes les options disponibles.	Comparer le résultat d'une requête selon sa configuration avec le résultat attendu.
	Exactitude Fonctionnelle	Exactitude des resultats des documents reçus à partir des requêtes	Marge minimum de nombres d'échecs et nombres d'erreurs lors de l'envoie des documents
Fiabilité	Tolérance aux fautes	Minimiser les erreurs/échecs d'ingestion GeoIp	95% des ingestions GeoIp sont réussis
Maintenabilité	Testabilité	Possibilité de tester le module facilement à l'aide de plusieurs tests	95% des requis fonctionnels sont couverts par des tests
	Modifiabilité	Créer une requêtes et vérifier que celles-ci soient bien modifiables.	Comparer le résultat d'une requête modifiée selon sa configuration avec le résultat attendu.
Efficacité/performance	Comportement temporel	Le temps d'envoi d'une requête qui modifie le processus est raisonnable	99% des requêtes se font dans les délai qui correspondent à la capacité du CPU.
	Capacité	processeur geoip dispose des capacités nécessaire	99% des requêtes se font dans les délai

			d'ingestion raisonnable.
	Utilisation de ressource	Les ressources dans les ingestion Geoip ne ralentissent pas le système.	Le système reste encore fonctionnelle après plusieurs ingestion.

1.4 Stratégie de validation

Afin de conduire les tests des modules nous avons besoin d'une stratégie de validation afin de pouvoir testé la validité et la durée des tests, de réduire les risques potentiels et d'essayer de prévoir le plus d'imprévus. Ceci permet également de structurer notre plan et d'être efficace et flexible dans notre approche d'assurance.

Pour ce faire nous allons tout d'abord s'assurer de maintenir des données au cours de tous les cycles des tests afin d'avoir le meilleur contrôle des cas de tests et du contexte de validation. De plus, la stratégie se voit flexible selon les critères, ainsi pour des critères de validations de fonctionnalités, nous allons, à l'aide des requis, identifier ce qui manque comme fonctionnalités. Avec ces derniers, nous pouvons également identifier les résultats erronés obtenu après exécution du logiciel et ainsi avoir une référence qui nous permet d'avoir un aperçu un peu plus objectif de l'état du système et d'identifier la marge d'erreur entre les valeurs attendues et obtenues.

Finalement, le développement et l'analyse de qualité est un travail d'équipe, donc nous allons organiser des contrôles de code par l'équipe comme des revus par pairs ce qui va nous permettre d'avoir un pairs d'yeux qui vont ajouter une couche de validation et de non-biais en plus des outils automatisés.

1.4.1 Rôles et responsabilités

Rôles	Responsabilités
Client	- Approuvez le plan de tests - Approuvez le plan de développement logiciel
Gestionnaire du projet	- Rédiger le plan de développement - Mettre à jour le plan de validation - Rédiger le plan de validation - Superviser le projet
Développeurs	- Suivre les étapes du plan de développement et de validation - Générer un rapport de défaillance
Personnel de l’assurance qualité	- Aider à tester les itérations du logiciel - Vérifier version des documents livrés

2. Test

Voici le plan des tests des différents modules que nous allons suivre afin de conduire le processus de testing et de validation à partir des différents outils et cas de tests.

2.1 Plan de test

2.1.1 Module : Transport-netty4

Critère	Sous critère	Test	Description	Logiciel/Outil	Objectif de couverture
Fonctionnalité	Exactitude	Test unitaire	Vérifier que le module assure les fonctionnalités spécifiées dans les spécifications fonctionnelles	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	99%
Fiabilité	Tolérance aux fautes	Test unitaire	Vérifier jusqu'où le module peut assurer la communication malgré des anomalies réseaux	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	98%
Fiabilité	récupérabilité	Test unitaire	Vérifier jusqu'où le module peut rétablir la communication malgré des anomalies réseaux	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	99%
Maintenabilité	Testabilité	Test unitaire	Vérifier qu'il est possible de	Tests unitaires	99%

			soumettre le module à des tests à tout moment	inclus dans le code source d'elasticsearch (gradle)	
Maintenabilité	Modifiabilité	Test de régression	Vérifier qu'une modification n'a pas eu d'effets de bord sur le module	Jenkins	98%
Efficacité/performance	Comportement temporel	LoadTesting	Vérifer le temps de réponse à mesure que les données augmentent	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Capacité	Scalability testing	Vérifier en ajoutant graduellement différents données que le système ne plante pas	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Utilisation de ressource	Test de volume	Vérifier à quel point le système est performant avec plusieurs données	LoadView	Il faut couvrir 95% des exigences non fonctionnelles

2.1.2 Module : Reindex

Critère	Sous critère	Test	Description	Logiciel/Outil	Objectif de couverture
Fonctionnalité	Complétude fonctionnelle	Test unitaire	Tester l'ensemble des fonctionnalité	Tests unitaires inclus dans le code source d'elasticsearch	Il faut couvrir 95% des exigences

			s du module	(gradle)	fonctionnelles
	Correction fonctionnelle	Test unitaire	Tests pour déterminer si les résultats sont bien conformes à la commande effectuée	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	Il faut que 99% des requêtes retournent une sortie correcte
Fiabilité	Tolérance aux fautes	Test d'acceptation de fautes	Observer le nombre d'erreurs (réindexations non abouties, erreur dans l'index de destination…)	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	Assurer moins de 5 % de fautes globales (réindexations non abouties, erreurs d'index)
	Récupérabilité	Tests de récupérabilité	Observer si les indexations sont bien faites et/ou si les destinations sont enregistrées, le temps mis pour récupérer les réindexations	Forecast	Temps de récupération de 10 min après une erreur, récupérer 90 % des indexations et réindexations en cours
Maintenabilité	Testabilité	Tests unitaires	On observe si on peut tester les principales caractéristiques du module	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	95% des requis fonctionnels sont couverts par des tests
Efficacité/performance	Comportement temporel	LoadTesting	Vérifer le temps de réponse à mesure que	LoadView	Il faut couvrir 95% des exigences

			les données augmentent		non fonctionnelles
	Capacité	Test d'endurance	Vérifier en ajoutant graduellement différents données que le système ne plante pas	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Utilisation de ressource	Test de volume	Vérifier à quel point le système est performant avec plusieurs données	LoadView	Il faut couvrir 95% des exigences non fonctionnelles

2.1.3 Module Percolator :

Critère	Sous-critère	Test	Description	Logiciel/Outil	Objectif de couverture
Fonctionnalité	complétude fonctionnelle	Test fonctionnelles	L'ensemble des requêtes reçues sont différentes pour chaque index	Squash TA	couvrir 95% des exigences fonctionnelles
	Correction Fonctionnelle	Test Unitaires	Implémenter des tests unitaires pour une requêtes et vérifier sa réception	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	couvrir 95% des requis
Fiabilité	Récupérabilité	Test unitaires	Utiliser les tests unitaires pour confirmer la percolation	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	couvrir 95% des requis

	Tolérance aux fautes	Test d'acceptation de fautes	Possibilité de tester le module facilement à l'aide de plusieurs tests	Tableau de comparaison	Il faut avoir 99% des fonctions qui sont dans les autres logiciels
Maintenabilité	Testabilité	Test unitaires	Créer des requêtes et vérifier que celle-ci sont bien reçue	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	couvrir 95% des requis
	Modifiabilité	Test de régression	Plan d'intégration	Jenkins et travis	couvrir tous les modules
Efficacité/performance	Comportement temporel	LoadTesting	Vérifer le temps de réponse à mesure que les données augmentent	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Capacité	Test de fatigue	Vérifier en ajoutant graduellement différents données que le système ne plante pas	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Utilisation de ressource	Test de volume	Vérifier à quel point le système est performant avec plusieurs données	LoadView	Il faut couvrir 95% des exigences non fonctionnelles

2.1.4 Module Ingest GeoIp:

Critère	Sous critère	Test	Description	Logiciel/Outil	Objectif de couverture
Fonctionnalité	Complétude fonctionnelle	Test unitaire	Ensemble des cas de tests qui	Tests unitaires inclus dans le	Couvrir plus de 95% des options de

			prennent en compte les différents cas de paramètres et d'options incluant les cas nuls et invalides	code source d'elasticsearch (gradle)	requêtes et des cas de fonctionnalités.
	Exactitude fonctionnelle	Test unitaire	Ensemble des cas de tests qui testent les différents aspects du système; de sa configuration jusqu'à la livraison des résultats.	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	Au moins 95% des requêtes retournent des résultats justes.
Fiabilité	Tolérance aux fautes	Test unitaire	Ensemble des cas de tests qui implémentent des cas d'exceptions avec une sortie spécifique.	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	Au moins 95% des tests doivent passer et donc retourner une sortie correcte.
Maintenabilité	Testabilité	Test unitaire	Ensemble des cas de tests qui couvrent les requis fonctionnels de l'ingestion et du traitement GeoIp	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	Au moins 95% des requis fonctionnels sont couverts par des tests
	Modifiabilité	Test unitaire	Ensemble des cas de tests qui	Tests unitaires inclus dans le	Au moins 95% des requêtes

			permettent l'ajout de nouveaux paramètres, clefs et qui en affirme la validité	code source d'elasticsearch (gradle)	modifiés retournent un résultat juste.
Efficacité/performance	Comportement temporel	LoadTesting	Vérifer le temps de réponse à mesure que les données augmentent	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Capacité	Scalability testing	Vérifier en ajoutant graduellement différents données que le système ne plante pas	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Utilisation de ressource	Spike testing	Vérifier à quel point le système est performant avec plusieurs données	LoadView	Il faut couvrir 95% des exigences non fonctionnelles

2.2 Rapport des tests :

2.2.1 Rapport Module Percolator :

Résultat des tests unitaires :

Test Summary

123	0	0	1m43.26s	100%
tests	failures	ignored	duration	successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
org.elasticsearch.percolator	123	0	0	1m43.26s	100%

Test Summary

123	0	0	1m43.26s	100%
tests	failures	ignored	duration	successful

Packages **Classes**

Class	Tests	Failures	Ignored	Duration
org.elasticsearch.percolator.CandidateQueryTests	12	0	0	13.052s
org.elasticsearch.percolator.PercolateQueryBuilderTests	20	0	0	7.306s
org.elasticsearch.percolator.PercolateQueryTests	1	0	0	0.497s
org.elasticsearch.percolator.PercolateWithNestedQueryBuilderTests	21	0	0	26.374s
org.elasticsearch.percolator.PercolatorFieldMapperTests	20	0	0	38.829s
org.elasticsearch.percolator.PercolatorHighlightSubFetchPhaseTests	2	0	0	1.653s
org.elasticsearch.percolator.PercolatorMatchedSlotSubFetchPhaseTests	3	0	0	3.892s
org.elasticsearch.percolator.PercolatorQuerySearchTests	5	0	0	4.272s
org.elasticsearch.percolator.QueryAnalyzerTests	38	0	0	1.914s
org.elasticsearch.percolator.QueryBuilderStoreTests	1	0	0	5.466s

2.2.2 Rapport module Reindex :

Résultats des tests unitaires :

Test Summary

174 tests	0 failures	0 ignored	9m41.66s duration

100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
org.elasticsearch.index.reindex	144	0	0	9m36.53s	100%
org.elasticsearch.index.reindex.remote	30	0	0	5.135s	100%

Packages **Classes**

Class	Tests	Failures	Ignored	Duration
org.elasticsearch.index.reindex.AsyncBulkByScrollActionTests	27	0	0	0.783s
org.elasticsearch.index.reindex.BulkByScrollParallelizationHelperTests	1	0	0	1.069s
org.elasticsearch.index.reindex.BulkIndexByScrollResponseTests	1	0	0	0.037s
org.elasticsearch.index.reindex.CancelTests	6	0	0	1m14.92s
org.elasticsearch.index.reindex.ClientScrollableHitSourceTests	4	0	0	0.174s
org.elasticsearch.index.reindex.DeleteByQueryBasicTests	11	0	0	1m59.51s
org.elasticsearch.index.reindex.DeleteByQueryConcurrentTests	2	0	0	15.216s
org.elasticsearch.index.reindex.ReindexBasicTests	5	0	0	1m8.22s
org.elasticsearch.index.reindex.ReindexFailureTests	3	0	0	39.557s
org.elasticsearch.index.reindex.ReindexFromRemoteBuildRestClientTests	2	0	0	0.161s
org.elasticsearch.index.reindex.ReindexFromRemoteWhitelistTests	10	0	0	1.761s
org.elasticsearch.index.reindex.ReindexFromRemoteWithAuthTests	4	0	0	27.946s
org.elasticsearch.index.reindex.ReindexMetadataTests	5	0	0	0.336s
org.elasticsearch.index.reindex.ReindexRestClientSslTests	4	0	0	2.618s
org.elasticsearch.index.reindex.ReindexScriptTests	10	0	0	0.338s
org.elasticsearch.index.reindex.ReindexSingleNodeTests	1	0	0	2.743s
org.elasticsearch.index.reindex.ReindexSourceTargetValidationTests	6	0	0	0.091s
org.elasticsearch.index.reindex.ReindexVersioningTests	9	0	0	1m14.01s
org.elasticsearch.index.reindex.RestReindexActionTests	2	0	0	0.084s
org.elasticsearch.index.reindex.RethrottleTests	6	0	0	42.249s
org.elasticsearch.index.reindex.RetryTests	4	0	0	29.930s
org.elasticsearch.index.reindex.RoundTripTests	4	0	0	0.059s
org.elasticsearch.index.reindex.TransportRethrottleActionTests	6	0	0	0.469s
org.elasticsearch.index.reindex.UpdateByQueryBasicTests	4	0	0	37.083s
org.elasticsearch.index.reindex.UpdateByQueryMetadataTests	1	0	0	0.171s
org.elasticsearch.index.reindex.UpdateByQueryWhileModifyingTests	1	0	0	36.442s
org.elasticsearch.index.reindex.UpdateByQueryWithScriptTests	5	0	0	0.543s
org.elasticsearch.index.reindex.remote.RemoteInfoTests	1	0	0	0.017s
org.elasticsearch.index.reindex.remote.RemoteRequestBuildersTests	9	0	0	0.288s
org.elasticsearch.index.reindex.remote.RemoteResponseParsersTests	1	0	0	0.032s
org.elasticsearch.index.reindex.remote.RemoteScrollableHitSourceTests	19	0	0	4.798s

2.2.3 Rapport module Ingest-Geolp:

Résultats des tests unitaires :

Test Summary

36 tests	0 failures	0 ignored	55.816s duration	100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
org.elasticsearch.ingest.geoip	36	0	0	55.816s	100%

Test Summary

36 tests	0 failures	0 ignored	55.816s duration	100% successful

Packages **Classes**

Class	Tests	Failures	Ignored	Duration
org.elasticsearch.ingest.geoip.GeoIpProcessorFactoryTests	15	0	0	5.905s
org.elasticsearch.ingest.geoip.GeoIpProcessorNonIngestNodeTests	1	0	0	20.988s
org.elasticsearch.ingest.geoip.GeoIpProcessorTests	17	0	0	28.492s
org.elasticsearch.ingest.geoip.IngestGeoIpPluginTests	3	0	0	0.431s

2.24 Rapport module Transport-netty4:

Test Summary

105 tests	0 failures	1 ignored	1m25.79s duration	100% successful

Ignored tests | Packages | Classes

SimpleNetty4TransportTests. testTimeoutPerConnection

Generated by Gradle 6.1.1 at Feb. 17, 2020, 11:38:13 a.m.

Test Summary

105 tests	0 failures	1 ignored	1m25.79s duration	100% successful

Ignored tests | **Packages** | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
org.elasticsearch.http.netty4	18	0	0	34.190s	100%
org.elasticsearch.transport	1	0	0	3.804s	100%
org.elasticsearch.transport.netty4	86	0	1	47.794s	100%

Generated by Gradle 6.1.1 at Feb. 17, 2020, 11:38:13 a.m.

Test Summary

105 tests	0 failures	1 ignored	1m25.79s duration	100% successful

Ignored tests | Packages | **Classes**

Class	Tests	Failures	Ignored	Duration	Success rate
org.elasticsearch.http.netty4.Netty4BadRequestTests	1	0	0	6.593s	100%
org.elasticsearch.http.netty4.Netty4CorsTests	5	0	0	1.549s	100%
org.elasticsearch.http.netty4.Netty4HttpPipeliningHandlerTests	4	0	0	1.667s	100%
org.elasticsearch.http.netty4.Netty4HttpServerPipeliningTests	1	0	0	6.618s	100%
org.elasticsearch.http.netty4.Netty4HttpServerTransportTests	7	0	0	17.763s	100%
org.elasticsearch.transport.CopyBytesSocketChannelTests	1	0	0	3.804s	100%
org.elasticsearch.transport.netty4.ByteBufBytesReferenceTests	34	0	0	0.889s	100%
org.elasticsearch.transport.netty4.Netty4SizeHeaderFrameDecoderTests	2	0	0	0.872s	100%
org.elasticsearch.transport.netty4.Netty4UtilsTests	4	0	0	0.040s	100%
org.elasticsearch.transport.netty4.NettyTransportMultiPortTests	4	0	0	0.269s	100%
org.elasticsearch.transport.netty4.SimpleNetty4TransportTests	42	0	1	45.724s	100%

Nous observons (à part pour transport-netty4 où il y a un test ignoré) que tous les tests sont achevés avec succès à 100 %.

3. Plan d'intégration continue

Le développement et la conception de projets modernes d'envergures exige l'application du processus d'intégration, de livraison et de déploiement continue. Dans cet ensemble, l'intégration continue touche particulièrement au code et fonctionnalités du programme et plus particulièrement aux tests et à l'assurance qualité , car en effet, l'intégration continue est un ensemble de processus d'automatisation de l'intégration des différents modules d'un système et des tests de code à chaque nouvel ajout (commit) de modifications de code par le biais d'un outil de contrôle et gestion de version.

Ainsi, il est important pour nous de conduire un plan d'intégration continue pour ElasticSearch afin d'y avoir un réel impact applicatif de notre plan d'assurance qualité.

Pour ce faire, nous allons utiliser Git comme outil de gestion de version afin de pouvoir effectuer des commits dans une branche commune qui contient l'intégralité du programme, que ça soit dev ou master. Lors d'un merge avec une des branches communes, le logiciel va passer par des étapes du pipeline qui va lui être appliqué. Ainsi nous allons relier l'outil Jenkins avec Github afin de permettre le fait que l'ajout d'une modification qui va passer par notre pipeline d'intégration continue et qui va effectuer nos séries de tests des quatre fonctionnalités afin de garantir qu'il n'y ait pas de régression.

4. Ajout d'un nouveau module

4.1 Module: lang-painless

Painless est un langage de script simple et sécurisé conçu spécifiquement pour être utilisé avec Elasticsearch. La syntaxe Painless est similaire à la syntaxe Java avec quelques fonctionnalités supplémentaires telles que la typage dynamique et les initialiseurs de tableau.

4.2 Critères et métriques de validation

Critère	Sous-critère	Objectif	Mesure de validation
Fonctionnalite	Exactitude	Fournir un langage de script sécurisé et expressif.	Expressivité >=90%
Fiabilité	Tolérance aux fautes	Le langage doit supporter quelques erreurs syntaxiques	aptitude à corriger les fautes de frappes >80%
Maintenabilité	Testabilité	Il doit être possible et facile de tester tout le module	Le module est couvert par 90% des tests approprié.
Maintenabilité	Modifiabilité	Le langage doit être facilement extensible	Facilité de mise à jour >90%
Efficacité/performance	Comportement temporel	Le temps d'envoi d'une requête qui modifie le processus est raisonnable	99% des requêtes se font dans les délai qui correspondent à la capacité du CPU.
	Capacité	n.d.	n.d.
	Utilisation de ressource	n.d.	n.d.

4.3 Plan de tests

Critère	Sous-critère	Test	Description	Logiciel/Outil	Objectif de couverture
Fonctionnalite	Exactitude	Test unitaire	Vérifier que le module fournit un langage sécurisé et assez expressif	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	99%
Fiabilité	Tolérance aux fautes	Test unitaire	Vérifier que des erreurs de saisies (Mauvaises syntaxes) sont automatiquement corrigées	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	90%
Maintenabilité	Testabilité	Test unitaire	Vérifier qu'il est possible de soumettre le module à des tests à tout moment	Tests unitaires inclus dans le code source d'elasticsearch (gradle)	99%
Maintenabilité	Modifiabilité	Test de régression	Vérifier qu'une modification n'a pas eu d'effets de bord sur le module	Jenkins	98%
Efficacité/performance	Comportement temporel	LoadTesting	Vérifer le temps de réponse à mesure que	LoadView	Il faut couvrir 95% des exigences non

			les données augmentent		fonctionnelles
	Capacité	Scalability testing	Vérifier en ajoutant graduellement différents données que le système ne plante pas	LoadView	Il faut couvrir 95% des exigences non fonctionnelles
	Utilisation de ressource	Test de volume	Vérifier à quel point le système est performant avec plusieurs données	LoadView	Il faut couvrir 95% des exigences non fonctionnelles

4.4 Résultat des tests:

Class org.elasticsearch.painless.WhenThingsGoWrongTests

all > org.elasticsearch.painless > WhenThingsGoWrongTests

21	0	1	4.199s
tests	failures	ignored	duration

100% successful

Tests | Standard output

Test	Duration	Result
testBadBoxingCast	0.135s	passed
testBadStringEscape	0.145s	passed
testBogusParameter	0.208s	passed
testCanNotOverrideRegexEnabled	0.104s	passed
testDynamicArrayWrongIndex	0.151s	passed
testDynamicListWrongIndex	0.195s	passed
testDynamicNPE	0.132s	passed
testDynamicWrongArgs	0.209s	passed
testIllegalDynamicMethod	0.234s	passed
testInfiniteLoops	0.479s	passed
testInvalidIntConstantSuggestsLong	0.124s	passed
testInvalidShift	0.153s	passed
testLoopLimits	0.355s	passed
testNullPointer	0.291s	passed
testOutOfMemoryError	-	ignored
testQuestionSpaceDotIsNotNullSafeDereference	0.133s	passed
testRCurlyNotDelim	0.100s	passed
testRegexDisabledByDefault	0.250s	passed
testRegularUnexpectedCharacter	0.217s	passed
testScriptStack	0.306s	passed
testStackOverflowError	0.169s	passed

4.5 Garantit de la qualité du système:

Suite à l'ajout de ce nouveau module (lang-painless) le plan de qualité qui avait été établie sera mise à jour afin de maintenir une bonne qualité du système. Pour ce faire nous ajouterons le plan qualité de ce nouveau module au plan qualité déjà présent.
Pour ce qui est du test, dans un premier temps, nous effectuerons des Tests Unitaires sur ce module, puis des tests d'intégration avec le reste du système dans un second temps et enfin des tests systèmes (tests suites).

5. Vidéo de démonstration de l'intégration et déploiement du système

Nous avons rencontré une erreur récurrente lors de la mise en place de l'intégration continue d'ElasticSearch v7.6.0. En effet, le build se termine toujours avec l'erreur :

```
JAVA8_HOME required to run tasks:
  :modules:reindex:oldEs2Fixture
  :modules:reindex:oldEs1Fixture
  :modules:reindex:oldEs090Fixture
```

qui est une erreur lié au module d'ElasticSearch. Cette erreur apparaît que ça soit avec Jenkins ou Travis CI.

Voici le lien de la démonstration de l'intégration et déploiement de notre système:
https://www.youtube.com/watch?v=_w3rXrgLWLI&feature=youtu.be

Références:

A. A. Auteur et B. B. Auteur. (Année) Titre. [En ligne]. Disponible : URL

Sam Guckenheimer. (2017) What is Continuous Integration?. [En ligne]. Disponible : https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration

ISO25000. (2019) ISO/IEC 25010. [En ligne]. Disponible : https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0

guru99.com. Non-functional testing. [En ligne]. Disponible : https://www.guru99.com/non-functional-testing.html

Painless scripting language https://www.elastic.co/guide/en/elasticsearch/reference/master/modules-scripting-painless.html

Elasticsearch B.V. Documentation (2020) Painless API. [En ligne]. Disponible: https://www.elastic.co/guide/en/elasticsearch/painless/master/painless-api-reference.html

Elasticsearch B.V. Documentation (2020) GeoIP Processor. [En ligne]. Disponible: https://www.elastic.co/guide/en/elasticsearch/reference/master/geoip-processor.html

Elasticsearch B.V. Documentation (2020) Percolate Query. [En ligne]. Disponible: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-percolate-query.html

Elasticsearch B.V. Documentation (2020) Reindex API. [En ligne]. Disponible: https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-reindex.html

Elasticsearch B.V. Documentation (2020) Transport. [En ligne]. Disponible: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-transport.html