RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88
http://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Sécurité Informatique

Présenté par :
Christelle **KOUTON**

Implémentation d'un système de gestion des incidents avec le framework de l'Ingénierie de Fiabilité des Sites (SRE)

Sous la supervision :
Dr. Arnaud **AHOUANDJINOU**
Enseignant Chercheur à l'UAC

Membres du jury :

Eugène C. EZIN	Professeur	UAC	Président
Emery ASSOGBA	Docteur	UAC	Examinateur
Arnaud AHOUANDJINOU	Docteur	UAC	Rapporteur

Année Académique : **2022-2023**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Liste des figures vi

Liste des tableaux vii

Liste des acronymes viii

Introduction 1

1 Revue de littérature 3

2 Présentation et choix des outils du système de gestion automatisé des incidents 11

3 Implémentation du système de gestion des incidents et test de fonctionnalité 16

Conclusion 36

Webographie 37

Table des matières 40

Dédicace

À
Mes parents

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voudrons témoigner toute notre reconnaissance. Nous remercions sincèrement :

- Professeur Eugène C. EZIN, Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) pour ses conseils et la formation qu'il nous a donnée ;
- Dr. Arnaud AHOUANDJINOU, le superviseur de ce travail pour son accord spontané à nous suivre, sa bienveillance, ses conseils et sa disponibilité ;
- tout le corps professoral de l'IFRI qui œuvre pour fournir les outils nécessaires à la réussite de nos études universitaires ;
- mon père Désiré M. KOUTON et ma mère Mathilde M. AGBOKPANKZO pour leur soutien indéfectible ;
- ma sœur Lauriane et mon frère Jaurès pour leur accompagnement dans la réalisation de ce travail.

Que tous ceux qui nous ont aidé trouvent ici l'expression de notre profonde gratitude.s

Résumé

Les systèmes informatiques rencontrent parfois des dysfonctionnements ou des problèmes qui perturbent leur fonctionnement normal. Pour minimiser l'interruption des services et maintenir une continuité opérationnelle, un processus de gestion des incidents est mis en place. Ce processus vise à assurer une résolution rapide et efficace des incidents dès leur apparition. Notre travail consiste à mettre en place un système de gestion des incidents en nous basant sur les pratiques de l'Ingénierie de Fiabilité des Sites. Pour y parvenir nous avons fait une étude sur l'ingénierie de fiabilité des sites. Nous avons ensuite fait une comparaison puis un choix des différents logiciels intervenant dans le processus de gestion des incidents. Le logiciel Kibana a été choisi pour la surveillance, Squadcast pour la réponse automatisée aux incidents, Slack pour la communication et Terraform pour la configuration automatisée. Des tests de fonctionnalités ont été effectués après l'installation, la configuration et l'intégration des outils choisis.

Mots clés : ingénierie, fiabilité, gestion des incidents

Abstract

Computer systems sometimes experience malfunctions or problems that disrupt their normal operation. To minimize service interruption and maintain operational continuity, an incident management process is put in place. This process aims to ensure rapid and effective resolution of incidents as soon as they occur. Our work consists of setting up an incident management system based on Site Reliability Engineering practices. To achieve this, we carried out a study on site reliability engineering. We then made a comparison and then a choice of the different software involved in the incident management process. Kibana software was chosen for monitoring, Squadcast for automated incident response, Slack for communication and Terraform for automated configuration. Functionality tests were carried out after the installation, configuration and integration of the chosen tools.

Key words: engineering, reliability, incident management

Liste des figures

1.1 Principes SRE [1] . . . 5
1.2 Étapes du processus de gestion des incidents [2] . . . 10

3.1 Configuration du nom du cluster et du noeud . . . 18
3.2 Configuration réseau Elasticsearch . . . 18
3.3 Configuration du Discovery . . . 18
3.4 Configuration de la sécurité minimale . . . 19
3.5 Elasticsearch status . . . 19
3.6 Configuration réseau Kibana . . . 20
3.7 Configuration du lien Elasticsearch . . . 20
3.8 Configuration du nom d'utilisateur et du mot de passe . . . 20
3.9 Configuration de l'alerting . . . 21
3.10 Kibana status . . . 22
3.11 Page d'authentification . . . 22
3.12 Activation du rechargement automatique . . . 23
3.13 Configuration du point de terminaison kibana . . . 23
3.14 Configuration du point de sortie Elasticsearch . . . 23
3.15 Metricbeat status . . . 24
3.16 Tableau de bord de Kibana . . . 24
3.17 Installation de Slack . . . 25
3.18 Interface de Slack . . . 25
3.19 Page de connexion de Squadcast . . . 26
3.20 Nom et propriétaire de la politique d'escalade . . . 26
3.21 Régle de la politique . . . 27
3.22 Définition du service . . . 27
3.23 Ajout d'une source d'alerte . . . 28
3.24 Création d'un connecteur webhook . . . 28
3.25 Intégration de Squadcast à Slack . . . 29
3.26 Configuration de Slack à un service . . . 29
3.27 Vérification de l'état du service Terraform . . . 30
3.28 Alerte de la surcharge de la bande passante . . . 31
3.29 Alerte de la surcharge du disque . . . 32
3.30 Alerte Squadcast . . . 32
3.31 Script pour la résolution de l'attaque DOS . . . 33
3.32 Alerte de la surcharge du disque . . . 33
3.33 Alerte de l'augmentation du trafic entrant . . . 34
3.34 Script pour la résolution de la surcharge du disque . . . 34

Liste des tableaux

2.1 Comparatif des outils de surveillance 12
2.2 Comparatif des outils de gestion incident 12
2.3 Comparatif des outils de communication à temps réel 13
2.4 Comparatif des outils de configuration 13

Liste des acronymes

APM :

Application Performance Management

CPU :

Central Processing Unit

HTTP :

HyperText Transfer Protocol

IT :

Information Technology

ITIL :

Information Technology Infrastructure Library

ITSM :

Information Technology Service Management

KPI :

Key Performance Indicator

SMS :

Short Message Service

SRE :

Site Reliability Engineering

Introduction Générale

Dans un monde numérique en constante évolution, la disponibilité et la fiabilité des services informatiques sont devenues des éléments essentiels pour les organisations cherchant à maintenir leur compétitivité et à répondre aux attentes croissantes des utilisateurs. Face à la complexité croissante des infrastructures et à la nécessité d'une réactivité accrue, la gestion automatisée des incidents émerge comme une solution prometteuse pour relever ces défis.

1 Problématique

La fiabilité et la disponibilité des services fournis par une entreprise sont devenues des facteurs critiques pour leur succès et compétitivité. Les utilisateurs finaux, qu'ils soient clients ou employés, s'attendent à des services accessibles à tout moment et sans interruption. Traditionnellement, la gestion des incidents repose sur des équipes d'opérations informatiques dédiées, qui surveillent les systèmes et répondent aux problèmes au fur et à mesure qu'ils se produisent. Cependant, ce modèle montre rapidement ses limites face à la croissance des systèmes informatiques. Le Port Autonome de Cotonou s'appuie actuellement sur le modèle ITIL et utilise le logiciel de gestion de parc informatique GLPI pour gérer les incidents. Bien que cette approche traditionnelle soit efficace, elle peut être limitée par sa réactivité et sa dépendance aux interventions humaines. Il est donc nécessaire de mettre en place un système de gestion automatisé des incidents en utilisant l'Ingénierie de Fiabilité des Sites.

2 Contexte

Depuis quelques années, le Gouvernement béninois a affiché sa volonté de faire du numérique et de la digitalisation des outils au service du développement et accessibles à tous. Pour donner effet à cette volonté politique, le Port Autonome de Cotonou (PAC) a mis à la disposition de ses usagers plusieurs applications métiers pour les assister dans leurs différentes activités. On peut citer VTR21 pour la gestion des escales des navires, JDEdwards pour la gestion des ressources de l'entreprise, BESC pour la création de bordereau de suivi des cargaisons, PGOP pour la facturation, Talend ESB pour l'intégration des données, Accès plus (Accès camion, Accès piéton) pour la gestion des accès au port. Afin d'assurer la crédibilité des services fournis il est important de mettre en place un système de gestion efficace des incidents.

3 Objectif général

L'objectif général de notre projet est de mettre en place un système de gestion des incidents avec un framework SRE (Site Reliability Engineering) qui a pour fonction principale de prévenir et répondre aux incidents dans le but d'éviter toute interruption des services en fonction des besoins d'une entreprise.

4 Objectifs spécifiques

De manière spécifique, il s'agit de :

- mieux cerner les notions de gestion des incidents et framawork SRE ;
- comprendre le processus de gestion des incidents avec le framework SRE ;
- identifier des outils SRE utilisés pour la gestion des incidents ;
- mettre en place un système de gestion des incidents avec un framework SRE ;
- effectuer des tests de fonctionnalités ;

5 Les contributions des travaux

La mise en place du système permet de gérer et de résoudre de manière automatique les incidents ; c'est-à-dire utiliser des logiciels pour réduire le travail manuel et répétitif, afin d'améliorer la satisfaction des clients et de renforcer la crédibilité des services auprès des utilisateurs.

6 Organisation du mémoire

Le présent mémoire est composé de trois (03) chapitres :

- le premier chapitre présente une revue de littérature sur l'Ingénierie de Fiabilité des Sites. Il présente également, les processus de gestion des incidents ;
- le deuxième chapitre expose une étude comparative des outils intervenant dans le processus de gestion des incidents et leur rôle ;
- le troisième chapitre est consacré à la mise en œuvre de la solution et les tests de vérification ;

Chapitre 1

Revue de littérature

Introduction

Dans ce chapitre, nous présentons d'abord les principes fondamentaux de l'ingénierie de fiabilité des sites. Nous expliquons ensuite le rôle que joue l'ingénierie de fiabilité des sites dans les systèmes informatiques. Nous exposons enfin les étapes du processus de gestion automatisé des incidents.

1 Framework SRE

1.1 Définition

L'ingénierie de la fiabilité des sites (SRE) a été créée chez Google par **Ben Treynor Sloss** en 2003, c'est une discipline ou approche qui intègre des aspects de l'ingénierie logicielle et les applique aux problèmes d'infrastructure et d'exploitation [3]. SRE prend les tâches qui étaient historiquement effectuées par les équipes opérationnelles, souvent manuellement, et les confie à des ingénieurs ou à des équipes opérationnelles qui utilisent des logiciels et l'automatisation pour résoudre les problèmes et gérer les systèmes de production [4].

1.2 Les principes fondamentaux de l'ingénierie de fiabilité de site (SRE)

Le framework SRE est basé sur plusieurs principes que sont :

1.2.1 Acceptation du risque

La fiabilité est l'un des principaux objectifs de SRE. Cependant il est important de reconnaître qu'aucun système n'est parfaitement fiable. À un moment donné, des problèmes surviendront. Ces derniers entraînent des coûts associés à la garantie de la fiabilité, qu'ils soient financiers, temporels ou en termes de confiance des clients du service. SRE a pour rôle d'analyser les échecs et les risques pour comprendre comment rendre les services et les systèmes plus résilients
En acceptant le risque SRE s'efforce de rendre le système suffisamment fiable, mais pas plus que nécessaire. Il cherche à équilibrer le risque d'indisponibilité avec le budget d'erreur de l'entreprise à l'aide des commentaires des utilisateurs, des objectifs de niveaux de services [5] [6].

1.2.2 Les objectifs de niveaux de service (SLI, SLO, SLA)

SLI (Service Level Indicator) : ou indicateur de niveau de service est une mesure quantitative utilisée pour évaluer la performance ou la fiabilité d'un service informatique. Les SLIs sont généralement définis en fonction de performance clés du service, tels que la disponibilité, la latence, le débit, le taux d'erreur.
SLO (Service Level Objective) : ou objectif de niveau de service représente la valeur ou la plage cible des SLI sur un intervalle de temps convenue par le client. Les SLOs sont établis en respectant le budget d'erreur du client ou de l'entreprise.
SLA (Service Level Agreement) : ou accord de niveau de service est un contrat explicite ou implicite conclu avec le client qui définit le niveau de service qui va être fourni. Il contient également les conséquences du respect ou non des SLO.
Les objectifs de niveau de services (SLOs) représentent alors les objectifs de performances souhaités ou voulus par le client. Elles sont définies dans un accord de niveau de service (SLA) et mesurées avec aux indicateurs de niveau de service (SLI) qui sont les mesures brutes des performances du système actuel. Lorsqu'il est suivi, ce principe SRE contribue à assurer la fiabilité du système, une réalisation efficace du projet et un client satisfait[6] [7].

1.2.3 Eliminer le labeur

L'élimination du labeur consiste à réduire le travail fastidieux, répétitif liée à la gestion des services informatiques en automatisant les tâches considérées comme laborieuses. Ceci permet aux équipes SRE de se consacrer à des tâches plus importantes qui ajoutent de la valeur au système telles que le développement de nouvelles fonctionnalités, l'amélioration de la fiabilité [6] [7].

1.2.4 Surveillance des systèmes distribués

La surveillance des systèmes distribuée fait référence à la collecte, à l'analyse et à la visualisation des données de performance et de disponibilité d'un système informatique à partir de plusieurs emplacements et composants.
La surveillance est essentielle dans SRE, elle permet de s'assurer que les SLOs sont respectés par conséquent que les services fonctionnent comme prévu. Elle sert également à identifier le moment où des problèmes surviennent afin qu'ils puissent être résolus immédiatement. La surveillance fournit une historique des performances et peut donner un aperçu de ce qui constitue un problème ponctuel par rapport à un problème systémique plus large. SRE décrit quatre métriques principales pour toute surveillance :
Latence : le délai nécessaire à un service pour répondre à une demande
Trafic : quantité de demande ou de charge sur un système
Erreurs : le nombre de services qui ne parviennent pas à répondre à une demande
Saturation : degré de dégradation d'un service et de ses ressources utilisables [6] [7].

1.2.5 Automatisation

Dans le but de limiter autant que possible le travail manuel, SRE considère l'automatisation comme un élément crucial de l'évolutivité et de la fiabilité du système. Trouver un moyen d'éliminer l'effort humain entraîne une augmentation de la vitesse d'exécution des différentes tâches, ainsi plus de membres de l'équipe pourront se concentrer sur les tâches qui nécessitent une intervention

humaine. De plus à mesure que les services évoluent et deviennent de plus en plus distribués, ils deviennent de plus en plus difficiles à gérer. L'automatisation des tâches répétitives à tous les niveaux, qu'il s'agisse de tests, de déploiement de logiciels, de réponse aux incidents ou simplement de communication entre les équipes, offre des avantages immédiats, d'efficacité et surtout de cohérence [6] [7].

1.2.6 Ingénierie des versions

En tant qu'un des principes SRE, l'ingénierie des versions fait référence à la fourniture de logiciels d'une manière cohérente et reproductible.
L'ingénierie des versions, également connue sous le nom de gestion des versions ou versionnement, est un processus utilisé dans le développement logiciel pour contrôler et gérer les différentes versions d'un logiciel tout au long de son cycle de vie. Cela implique généralement la création, la gestion et la publication de différentes versions d'un logiciel, avec des objectifs tels que la gestion des modifications, le suivi des fonctionnalités ajoutées ou modifiées, la correction des bugs et l'amélioration de la qualité du logiciel. L'ingénierie des versions comprend également des pratiques telles que le contrôle de version, la gestion des branches, la fusion de code et la création de déploiements et de packages logiciels pour différentes plates-formes. En résumé, l'ingénierie des versions vise à assurer la cohérence, la traçabilité et la qualité du logiciel tout au long de son évolution[6] [7].

1.2.7 Simplicité

Le principe de simplicité dans Site Reliability Engineering (SRE) met l'accent sur la création et la gestion de systèmes informatiques aussi simples que possible. Cela signifie concevoir des architectures simples, éliminer les processus complexes et se concentrer sur des mesures claires et significatives pour évaluer la fiabilité des systèmes. En simplifiant les systèmes et les processus, les équipes SRE peuvent réduire les risques d'incidents, améliorer la stabilité opérationnelle et faciliter la maintenance à long terme. En bref, la simplicité est essentielle pour garantir des systèmes informatiques fiables et faciles à gérer [6] [7].

La figure 1.1 présente les principes SRE.

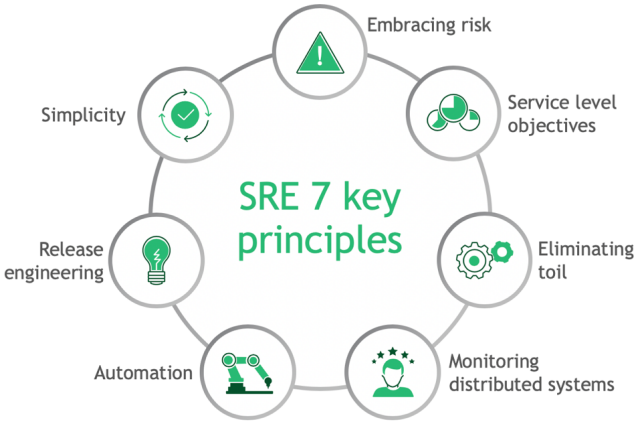


FIGURE 1.1 : Principes SRE [1]

1.3 Les principaux objectifs du framework SRE

L'ingénierie de fiabilité des sites vise principalement à garantir la fiabilité et la disponibilité des systèmes informatiques. Pour cela, SRE se concentre sur plusieurs objectifs clés. Tout d'abord, SRE cherche à maximiser la fiabilité des systèmes en minimisant les temps d'arrêt et en maintenant des performances optimales, assurant ainsi une expérience utilisateur satisfaisante. Ensuite, SRE cherche à automatiser les tâches opérationnelles autant que possible pour réduire les erreurs humaines, accélérer les déploiements et améliorer l'efficacité globale des opérations informatiques. De plus, SRE s'efforce de concevoir des systèmes informatiques qui peuvent évoluer facilement pour répondre à la croissance des besoins et des charges de travail, tout en maintenant des performances optimales. Enfin, SRE met en place des processus efficaces de gestion proactive des incidents, permettant une détection rapide, une réponse appropriée et une résolution efficace des problèmes. En combinant ces objectifs, SRE vise à créer un environnement informatique fiable, robuste et évolutif [8].

2 SRE un cadre de travail ITSM

2.1 ITSM

2.1.1 Définition

L'ITSM (Information Technology Service Management) est une approche stratégique de la gestion IT, qui met l'accent sur la création de valeur pour les clients. L'ITSM définit un ensemble de processus et de pratiques utilisés pour planifier, concevoir, fournir, exploiter et contrôler les services informatiques au sein d'une organisation, en alignant ces services sur les besoins métier et les objectifs stratégiques.

2.1.2 Les principaux objectifs de l'ITSM

La gestion des services informatiques vise à garantir que les services informatiques répondent aux besoins de l'entreprise de manière efficace et efficiente. Les principaux objectifs de l'ITSM comprennent premièrement la fourniture de services de haute qualité en alignement avec les besoins de l'entreprise, assurant ainsi la satisfaction des utilisateurs et des clients. Deuxièmement, l'ITSM cherche à optimiser les processus et les ressources informatiques pour améliorer l'efficacité opérationnelle, réduire les coûts et maximiser la valeur ajoutée pour l'entreprise. Troisièmement, l'ITSM vise à garantir la disponibilité et la sécurité des services informatiques en mettant en place des processus de gestion des incidents, des problèmes et des changements. Enfin, l'ITSM s'efforce d'améliorer continuellement les services informatiques grâce à la collecte de données, à l'analyse des performances et à la mise en œuvre d'initiatives d'amélioration continue. En combinant ces objectifs, l'ITSM contribue à optimiser les opérations informatiques et à soutenir les objectifs commerciaux de l'organisation.

2.1.3 SRE un cadre ITSM

La gestion des services informatiques vise à fournir des services informatiques de haute qualité qui répondent aux besoins de l'entreprise, en alignant les services sur les objectifs métier, en optimisant les processus et les ressources informatiques, et en assurant la disponibilité et la sécurité des services. L' ingénierie de fiabilité des sites se concentre principalement sur la fiabilité et la disponibilité des

services informatiques, en veillant à minimiser les temps d'arrêt et à maintenir des performances optimales pour assurer une expérience utilisateur satisfaisante.

L'ingénierie de fiabilité des sites et la gestion des services informatiques sont deux approches différentes mais complémentaires pour la gestion des services informatiques. Bien que SRE ne soit pas traditionnellement considéré comme un cadre ITSM, il peut certainement être utilisé comme cadre ITSM pour la gestion des services informatiques.

Le SRE met l'accent sur la fiabilité des services informatiques en utilisant des pratiques d'ingénierie logicielle, d'automatisation et de gestion des risques pour garantir des niveaux de service élevés. Cette orientation vers la fiabilité des services est également un objectif clé de ITSM et se traduit par la satisfation du client . Mais encore SRE encourage une culture d'apprentissage et d'amélioration continue à travers des examens post-mortem, des rétrospectives d'incidents et des initiatives d'ingénierie pour résoudre les problèmes sous-jacents. Cette approche est compatible avec le principe d'amélioration continue de ITSM. SRE utilise des métriques telles que le temps de disponibilité, le temps moyen de résolution et le taux d'incidents pour évaluer la fiabilité des services. Ces mesures de performance sont alignées avec les KPI utilisés dans ITSM pour évaluer la qualité des services informatiques. Dans l'ensemble nous pouvons considérer SRE comme un cadre ITSM.

3 SRE et DevOps

3.1 DevOps

Le DevOps est un mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (dev) et de l'administration des infrastructures informatiques (ops), notamment l'administration système. Il vise à créer une culture et un environnement dans lesquels la conception, les tests et la diffusion de logiciels peuvent être réalisés rapidement, fréquemment et efficacement [9].

3.2 Relation entre SRE et DevOps

L'ingénierie de fiabilité des sites et le DevOps sont deux approches clés dans le domaine de l'ingénierie logicielle et de la gestion des opérations informatiques. Bien qu'ils aient des différences, ils partagent plusieurs similitudes. Tout d'abord, les deux méthodologies mettent l'accent sur la collaboration et la communication entre les équipes de développement et d'exploitation. Ils visent à favoriser une culture de responsabilité partagée, où les équipes travaillent ensemble pour atteindre des objectifs communs de développement, de déploiement et de maintenance des logiciels. De plus, SRE et DevOps valorisent tous deux l'automatisation des processus pour accélérer les déploiements, améliorer la fiabilité et réduire les erreurs humaines.

Cependant, malgré ces similitudes, il existe des différences notables entre SRE et DevOps. Par exemple, SRE se concentre spécifiquement sur la fiabilité des systèmes en production, en mettant l'accent sur la disponibilité, la latence, les performances et d'autres aspects clés. En revanche, DevOps est plus axé sur la collaboration entre les équipes de développement et d'exploitation pour favoriser un déploiement continu et une livraison agile des logiciels. De plus, alors que SRE met l'accent sur la réduction des risques et la résolution proactive des incidents, DevOps cherche à optimiser l'ensemble du cycle de vie du développement logiciel, de la conception à la livraison.

Bien que SRE et DevOps aient des approches différentes, ils partagent un objectif commun de fournir des logiciels de haute qualité de manière efficace et fiable. Ensemble, ils forment un partenariat puissant pour créer un environnement opérationnel robuste et agile, où la collaboration, l'automatisation et l'amélioration continue sont au cœur de la culture organisationnelle [10].

4 La gestion des incidents

4.1 Incident informatique

4.1.1 Définition

Un incident informatique est un événement qui met en péril de manière réelle ou imminente, sans autorisation légale, la confidentialité, l'intégrité ou la disponibilité des informations ou d'un système d'information. Il constitue une violation ou une menace imminente de violation des politiques de sécurité, des procédures de sécurité ou des politiques d'utilisation acceptable [11].

4.1.2 Les catégories d'incidents informatiques

Les incidents informatiques peuvent être classés en trois catégories. D'abord nous avons les incidents qui touchent les logiciels et les applications tels que : le ralentissement des logiciels, le plantage des applications, les erreurs des programmes. Ensuite les incidents concernant le matériel comme : les problèmes des disques durs, le blocage des imprimantes, le dysfonctionnement des serveurs. Enfin les demandes de service telles que : les problèmes d'accès à un compte utilisateur, l'oubli de l'identifiant ou du mot de passe [11].

4.2 La gestion des incidents

4.2.1 Définiton

La **gestion des incidents** est un aspect clé de la gestion des services informatiques (ITSM). C'est le processus de gestion du cycle de vie des incidents. Elle s'assure que l'exploitation normale des services soit rétablie le plus rapidement possible et que l'impact sur le business soit réduit au minimum [11].

La **gestion automatisée des incidents** fait référence à l'utilisation de technologies et de processus automatisés pour détecter, trier, diagnostiquer, résoudre et documenter les incidents informatiques de manière rapide et efficace. Cette approche vise à minimiser les interventions manuelles, à réduire les temps de résolution et à améliorer la disponibilité des services informatiques.

4.2.2 Processus de gestion des incidents

Le processus de gestion des incidents est un ensemble d'étapes visant à réduire les temps d'arrêt et à minimiser l'impact sur la productivité des employés en cas d'incident. Le processus est composé de six étapes.

- **Etape 1 :** Identification de l'incident

La première étape consiste à détecter et enregistrer l'incident. L'incident est enregistré avec certaine informations telles que : le nom, l'identifiant, la date et l'heure de détection, une description [12][13].

- **Etape 2 :** Catégorisation de l'incident

  La catégorisation consiste à affecter une catégorie et au moins une sous-catégorie à l'incident. Pour catégoriser nous ajoutons un mot clé ou décrivons la nature du problème. Par exemple, si nous rencontrons un bug logiciel causé par un problème de développement, nous pouvons simplement l'associer à la catégorie « Développement » [12][13].

- **Etape 3 :** Hiérarchisation de l'incident

  La hiérarchisation ou priorisation consiste à classer les incidents par ordre d'importance en fonction de leur priorité. La priorité d'un incident peut être déterminée en fonction de son impact sur les utilisateur et de son urgence. L'impact d'un incident indique l'étendue des dégâts que le problème causera à l'utilisateur ou à l'entreprise. L'urgence d'un incident indique le délai dans lequel l'incident doit être résolu [12][13].

- **Etape 4 :** Réponse à l'incident

  La réponse à l'incident ce fait en deux temps. Nous diagnostiquons d'abord le problème. Il s'agit là de remonter à la racine de l'incident pour identifier les causes exactes et profondes.Un incident dont la vraie cause n'a pas été identifiée, peut être mal résolu. Et l'incident peut ressurgir à nouveau peu de temps après. Ensuite nous passons à la résolution de l'incident proprement dite en appliquant le traitement adéquat pour l'éliminer. Il peut arriver que cette étape fasse l'objet de plusieurs escalades ou remontés entre intervenants pour aboutir à la solution définitive [12][13].

- **Etape 5 :** Vérification de la résolution

  Le traitement appliqué pour résoudre l'incident doit être testé. Ce n'est pas une étape toujours requise, mais fortement recommandée. Elle permet de s'assurer que c'est bien le bon traitement qui a été appliqué [12][13].

- **Etape 6 :** Clôture de l'incident et post-mortem

  Une fois la vérification de la résolution effectuée, il faut marquer l'incident comme étant clos. C'est l'étape finale. Suite à cela, la base de connaissances de l'organisation doit être mis à jour avec les données de l'incident. Ces données constituent un bon rapport de gestion d'incident. Toute clôture d'incident doit être soldée par un rapport [12][13].

La figure 1.2 présente les étapes détaillées du processus de gestion des incidents.

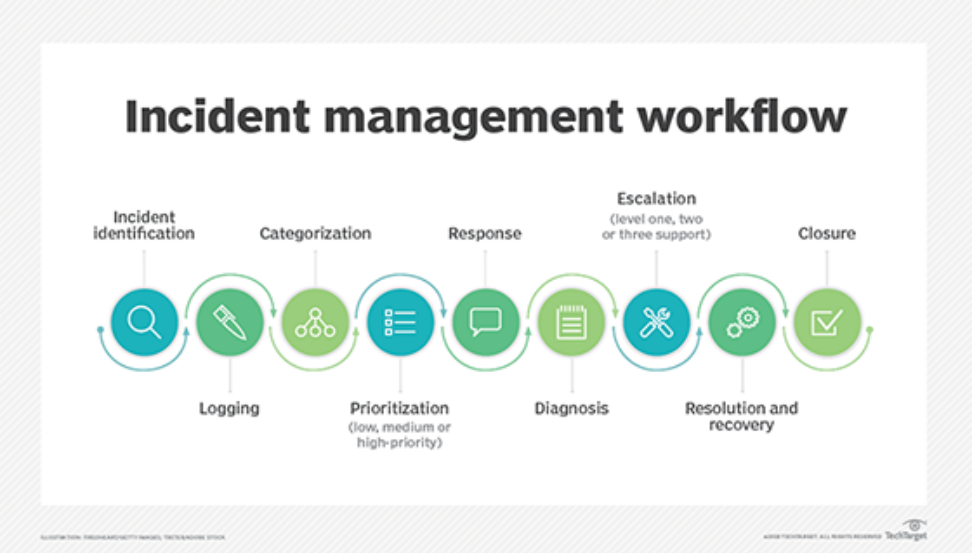


FIGURE 1.2 : Étapes du processus de gestion des incidents [2]

Conclusion

Ce chapitre nous a permis de fournir un aperçu approfondi sur le framework SRE une approche innovante en matière de gestion des services informatiques, à travers des définitions, des principes, des pratiques et des cas d'utilisation de SRE.

Chapitre 2

Présentation et choix des outils du système de gestion automatisé des incidents

Introduction

Dans ce présent chapitre, nous allons présenter les types de logiciel intervenant dans notre système de gestion des incidents. Ensuite nous allons faire une étude comparative des différents logiciels et faire un choix.

1 Les outils SRE

L'Ingénierie de Fiabilité des Sites (SRE) utilisent une variété de logiciels pour assurer la gestion et la fiabilité des systèmes informatiques. Pour notre système de gestion nous allons utilisé les outils qui suivent.

1.1 APM (Application Performance Management) ou outils de surveillance

Les APM ou logiciels de gestion de performances surveillent, gèrent la performance et la disponibilité des applications logicielles dans des environnements informatiques.

Un outil de surveillance est un logiciel utilisé pour surveiller et suivre en temps réel les performances, l'état et les activités d'un système informatique, d'une application, du trafic réseau ou d'un équipement [14].

Ces deux logiciels sont utilisés pour identifier les problèmes ou anomalies dans le processus de gestion aux incidents

Le tableau 2.1 illustre une étude comparative des outils de surveillance Datadog, New Relic et Kibana populaire dans l'Ingénierie de Fiabilité de Sites, en se basant sur les critères suivant : l'alerting, la personnalisation des tableaux de bord, les plateformes ; l'intégration, la surveillance en temps réel, la visualisation et l'exploration des données, la gestion des logs, l'open source [15].

TABLE 2.1 : Comparatif des outils de surveillance

Critères	Datadog	New Relic	Kibana
Alerting	oui	oui	oui
Tableau de bord personnalisé	Peu avancé	Peu avancé	Avancé
Plateforme	Multiplateforme	Multiplateforme	Multiplateforme
Intégration	Large	Large	Peu large
Surveillance en temps réel	Oui	Oui	Oui
Visualisation et exploration des données	Peu avancé	Peu avancé	Avancé
Gestion des logs	Peu avancé	Peu avancé	Avancé
Surveillance des métriques	Bonne	Bonne	Excellente
Open source	Non	Non	Oui

1.2 Système automatisé de réponse aux incidents

Un système automatisé de réponse aux incidents ou logiciel de gestion d'incidents est une plateforme logicielle conçue pour détecter, analyser et réagir aux incidents informatiques de manière automatisée, sans intervention humaine directe ou avec une intervention minimale. Ces systèmes sont largement utilisés dans les opérations informatiques pour assurer une réactivité accrue, réduire les temps d'arrêt et améliorer la fiabilité des services. Il permet de catégoriser, de hiérarchiser et d'assigner un incident à une équipe automatiquement.

Le tableau 2.2 présente une comparaison des logiciels de gestion d'incidents Pager Duty, Opsgenie et Squadcast utilisés pour SRE. La comparaison est faite sur les critères qui suivent : politique d'escalade, tableau de bord, gestion des SLO, utilisation complexe, plateforme, intégration, automatisation, coût [16].

TABLE 2.2 : Comparatif des outils de gestion incident

Critères	Pager Duty	Opsgenie	Squadcast
Politique d'escalade	Oui	Oui	Oui
Tableau de bord	Oui	Oui	Oui
Gestion des SLO	Non	Non	Oui
Gestion des gardes	Oui	Oui	Oui
Utilisation complexe	Oui	Oui	Non
Plateforme	Multiplateforme	Multiplateforme	Multiplateforme
Intégration	Large	Large	Peu large
Automatisation	Avancé	Limité	Avancé
Coût	Peu aborbable	Peu aborbable	Aborbable

1.3 Outils de communication à temps réel

Un outil de communication est une plateforme logicielle conçue pour faciliter l'échange d'informations, à travers des messages instantané, des appels audio vidéo entre des individus, des équipes

d'une organisations [14]. Dans le processus de gestion des incidents, ils sont utilisés pour la communication entre les membres de l'équipe de résolution, les utilisateurs du service.

Le tableau 2.3 fait une comparaison entre les outils de communication Microsoft Teams, Telegram et Slack. La comparaison est faite selon les critères qui suivent : générateur de flux de tavail, appel d'équipe, messagerie instantanée, canaux de discussion, plateforme, intégration, automatisation [17] [18].

TABLE 2.3 : Comparatif des outils de communication à temps réel

Critères	**Microsoft Teams**	**Telegram**	**Slack**
Générateur de flux de tavail	Oui	Non	Oui
Messagerie instantanée	Oui	Oui	Oui
Canaux de discussion	Oui	Oui	Oui
Appel d'équipe	Oui	Oui	Oui
Plateforme	Multiplateforme	Multiplateforme	Multiplateforme
Intégration	Limité	Limité	Large
Automatisation	Peu avancé	Limité	Avancé

1.4 Outils de configuration et d'automatisation

Un outil de configuration et d'automatisation est une plateforme logicielle conçue pour gérer et automatiser les configurations, les déploiements, les tâches répétitives et les processus opérationnels dans les environnements informatiques [14]. Ces outils sont largement utilisés pour simplifier la gestion des infrastructures, améliorer l'efficacité opérationnelle et réduire les erreurs humaines.

Le tableau 2.4 est un comparatif des outils de configuration et d'automatisation Ansible, Saltstack et Terraform. Les critères de comparaison sont les suivants : infrastructure as code, provisioning, agentless, écosystème cloud, état de l'infrastructure, scalabilité, vérification de configuration, planification des modifications [19] [20].

TABLE 2.4 : Comparatif des outils de configuration

Critère	**Ansible**	**SaltStack**	**Terraform**
Infrastructure as Code	Partiellement	Partiellement	Complet
Provisioning	Limité	Limité	Fort
Agentless	Oui	Non	Oui
Écosystème Cloud	Support limité	Support limité	Support étendu
État de l'infrastructure	Non	Non	Oui
Scalabilité	Moyenne	Haute	Haute
Vérification de configuration	Non	Non	Oui
Planification des modifications	Non	Non	Oui

2 Présentation des solutions choisies

2.1 Kibana

Kibana pour sa capacité à fournir des visualisations puissantes et des tableaux de bord personnalisables. Son statut open source et sa grande extensibilité offrent une flexibilité permettant de répondre aux besoins spécifiques de surveillance et d'analyse des incidents, tout en favorisant l'intégration avec d'autres outils et sources de données. Cette combinaison de fonctionnalités rend Kibana adéquat à la détection précoce des problèmes, à l'analyse approfondie des incidents et à la prise de décisions rapides pour minimiser les temps d'arrêt et optimiser les performances du système.

2.2 Squadcast

Squadcast se distingue par son interface intuitive et sa mise en place rapide, offrant une approche simplifiée qui peut réduire les courbes d'apprentissage et les coûts opérationnels. Contrairement à PagerDuty et Opsgenie, qui peuvent être plus complexes et coûteux, Squadcast propose une tarification plus abordable et une fonctionnalité robuste pour la gestion des alertes, la coordination des incidents, et la collaboration en temps réel. De plus, Squadcast fournie des fonctionnalités d'automatisation avancées, assure un excellent support client, ce qui est crucial pour une adoption réussie et un dépannage efficace. Ces avantages font de Squadcast une alternative attrayante pour les équipes qui souhaitent une solution efficace sans les complexités et les coûts associés à d'autres outils de gestion des incidents.

2.3 Slack

Slack est un choix pertinent comme outil de communication dans un système de gestion des incidents en raison de sa capacité à faciliter une communication rapide et efficace entre les membres de l'équipe en temps réel. Avec ses fonctionnalités de messagerie instantanée, de gestion des canaux de discussion et d'intégrations avec d'autres outils de gestion des incidents, d'automatisation des tâche quotidienne Slack permet aux équipes de coordonner rapidement leurs efforts, de partager des mises à jour critiques et de collaborer de manière transparente pour résoudre les incidents. Son interface conviviale et ses options de personnalisation offrent également une flexibilité pour adapter les flux de communication aux besoins spécifiques de l'équipe, ce qui contribue à une gestion proactive et efficace des incidents, réduisant ainsi les temps d'arrêt et améliorant la résolution des problèmes.

2.4 Terraform

Choisir Terraform au détriment d'Ansible ou de SaltStack peut être justifié par plusieurs avantages clés. Terraform excelle dans la gestion de l'infrastructure en tant que code (IaC) grâce à son approche déclarative et à son système de gestion d'état. Contrairement à Ansible et SaltStack, qui se concentrent principalement sur la configuration des systèmes existants et l'orchestration, Terraform maintient un fichier d'état persistant qui lui permet de suivre l'état actuel de l'infrastructure. Cela facilite la planification précise des modifications et assure que les ressources sont créées et configurées conformément à la configuration désirée. De plus, Terraform offre un langage de configuration puissant et flexible. Cette capacité à gérer des infrastructures complexes de manière scalable et sa capacité à prévisualiser

les changements avant leur application en font un choix de prédilection pour les environnements de grande envergure.

2.5 Brève description de la solution

Il s'agit de mettre en place un système de gestion des incidents en utilisant la méthode SRE (Site Reliability Engineering). Le processus de gestion des incidents est automatisé par l'utilisation de logiciels permettant de faire la détection rapide, la mobilisation d'équipe, l'analyse approfondie, la communication transparente et l'apprentissage continu pour garantir la fiabilité et la disponibilité des services.

La détection rapide se fait par un système de surveillance qui détecte un incident lié à un service en production. Cela déclenche une alerte dans un système de gestion des incidents, tel que Squadcast. Le logiciel de gestion d'incident reçoit l'alerte et applique des règles d'astreinte configurée pour déterminer l'équipe d'intervention appropriée. Selon les critères de configuration, l'alerte est acheminée à l'équipe SRE responsable de ce service spécifique. L'équipe SRE est informée en temps réel de l'incident via des notifications par un outil de communication tel que slack. Les membres de l'équipe prennent connaissance de l'incident et se mobilisent pour le résoudre. Ils commencent par une analyse préliminaire de l'incident pour comprendre la cause possible. Ils consultent les journaux système, les métriques de performance et d'autres sources d'informations pertinentes pour obtenir des indices sur la cause de l'incident. Pendant l'analyse de l'incident, l'équipe SRE communique activement avec d'autres parties prenantes concernées, telles que les développeurs d'applications ou les équipes de support, pour partager des informations sur l'incident en cours, les impacts potentiels et les actions en cours. La cause principale de l'incident identifié, l'équipe prend des mesures pour le résoudre. Cela peut inclure des actions telles que le redémarrage des services, la modification de la configuration, la mise à jour du code. Une fois que les mesures correctives sont mises en place, l'équipe SRE surveille attentivement les systèmes et effectue des tests pour vérifier si l'incident est résolu et si la stabilité est rétablie.

Conclusion

Dans ce chapitre, nous avons fait une étude comparative des différents logiciels intervenant dans le système de gestion en nous basant sur différents critère.

Chapitre 3

Implémentation du système de gestion des incidents et test de fonctionnalité

Introduction

Dans ce chapitre nous présentons la mise en place de la solution et les tests de fonctionnalités effectués. Dans un premier temps nous allons décrire brièvement le fonctionnement du système. Ensuite nous abordons les étapes d'installation et de configuration des outils du système de gestion et enfin nous allons réaliser les tests de fonctionnalités.

1 Mode de fonctionnement du système de gestion d'incident

Notre système de gestion des incidents est constitué de différents logiciels intervenant chacun dans une étape du processus de gestion automatisée des incidents. Le système est composé de logiciels de supervision : **Elasticsearch**, **Kibana** et l'agent **Metricbeat** de la **Suite Elastic**, d'un système de réponse automatisée aux incidents : **Squadcast**, d'un logiciel de gestion de configuration : **Terraform** et d'un logiciel de communication : **Slack**. Le système fonctionne de la manière suivante : Metricbeat collecte des données sur des métriques précis qu'il envoie à Elasticsearch. Elasticsearch stocke et analyse les données reçues, qui seront affichées sous forme de documents, de graphes, de diagrammes par Kibana. En fonction des données visualisées dans Kibana des règles d'alerte sont créée. En cas d'anomalies autrement dit de non respect des règles un message d'alerte provenant de Kibana est envoyé à Squadcast puis transféré vers Slack et est assigné à une équipe. L'équipe en charge de la résolution de l'incident utilise Slack pour échanger des informations et informer les parties prenantes. Terraform et Squadcast sont utilisés pour la résolution de l'incident.

2 Installation et configuration des logiciels

2.1 Suite elastic

La suite elastic est un ensemble de logiciels que sont Elasticsearch, Logstash, Kibana et les agents Beat. Elle permet de collecter, de stocker, d'analyser et de visualiser des données structurées et non structurées à grande échelle. Elasticsearch et kibana sont installés sur une machine Ubuntu desktop 22.04.3 et l'agent Metricbeat sur une machine Ubuntu server 22.04.3.

2.1.1 Elasticsearch :

INSTALLATION

Pour l'installation nous allons avec les commandes suivantes : [21] :

- Importer la clé GPG publique Elastic

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-
   elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/
   elasticsearch-keyring.gpg
```

- Installer le paquet apt-transport-https

```
$ sudo apt-get install apt-transport-https
```

- Ajouter la liste des dépôts Elastic dans le répertoire sources.list.d

```
$ echo "deb [signed-by=/usr/share/keyrings/elasticsearch-
   keyring.gpg] https://artifacts.elastic.co/packages/7.x/apt
    stable main" | sudo tee /etc/apt/sources.list.d/elastic
   -7.x.list
```

- Mettre à jour la liste de paquets et installer Elasticsearch

```
$ sudo apt-get update && sudo apt-get install elasticsearch
```

CONFIGURATION

Dans le fichier de configuration de Elasticsearch ***/etc/elasticsearch/elasticsearch.yml*** nous allons [22] :

- définir le nom du cluster et du noeud

  La figure 3.1 présente la définition du nom du cluster et du noeud

```
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: ES
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: node-1
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#
```

FIGURE 3.1 : Configuration du nom du cluster et du noeud

- configurer les paramètres réseau

  Pour configurer les paramètres réseau nous allons définir le port et les adresses IP par lesquels Elasticsearch écoute et reçoit respectivement des données. Dans notre configuration, Elasctisearch reçoit des données de n'importe quelle machine contenue dans le même réseau via le port 9200.

  La figure 3.2 présente la configuration réseau Elasticsearch

```
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: 0.0.0.0
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
```

FIGURE 3.2 : Configuration réseau Elasticsearch

- configurer le **discovery.seed_hosts** et le **cluster.initial_master_nodes**.

  La figure 3.3 présente la configuration du discovery.

```
# ---------------------------------- Discovery ---------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: ["127.0.0.1", "192.168.8.110"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
cluster.initial_master_nodes: ["node-1"]

#discovery.type: single-node
#
# For more information, consult the discovery and cluster formation module documentation.
#
```

FIGURE 3.3 : Configuration du Discovery

- configurer la sécurité minimale en ajoutant les lignes suivantes [23] :

xpack.security.enabled permet d'activer les fonctionnalités de securité d'Elasticsearch sur notre noeud. L'activation des fonctionnalités de sécurité d'Elasticsearch active l'authentification de base qui permet d'accéder à Kibana en se connectant avec nom d'utilisateur et mot de passe.

xpack.security.transport.ssl.enabled active TLS/SSL sur la couche de transport, qu'utilisent les nœuds pour communiquer entre eux.

xpack.security.authc.api_key.enabled active le service de clé API intégré néccessaire pour la création des règle d'alerte. Cette commande intervient dans la configuration de la fonctionnalité d'alerte.

La figure 3.4 présente la configuration de la sécurité minimale.

```
# ---------------------------------- Security ----------------------------------
#
#                                    *** WARNING ***
#
# Elasticsearch security features are not enabled by default.
# These features are free, but require configuration changes to enable them.
# This means that users don't have to provide credentials and can get full access
# to  the cluster. Network connections are also not encrypted.

xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
#xpack.security.http.ssl.enabled: false
xpack.security.authc.api_key.enabled: true
#
# To protect your data, we strongly encourage you to enable the Elasticsearch security features.
# Refer to the following documentation for instructions.
#
```

FIGURE 3.4 : Configuration de la sécurité minimale

Après la configuration nous allons :

- Activer Elasticsearch et autoriser son initialisation à chaque démarrage du système

```
1  $ sudo systemctl start elasticsearch
2  $ sudo systemctl enable elasticsearch
```

- Vérifier l'état de Elasticsearch

```
1  $ sudo systemctl status elasticsearch
```

La figure 3.5 présente le status de Elasticsearch.

```
root@christelle-VirtualBox:/etc/elasticsearch# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-02-29 13:44:22 WAT; 1h 11min ago
       Docs: https://www.elastic.co
   Main PID: 881 (java)
      Tasks: 61 (limit: 4599)
     Memory: 1.8G
        CPU: 10min 3.836s
     CGroup: /system.slice/elasticsearch.service
             ├─ 881 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.n>
             └─2384 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

févr. 29 13:37:34 christelle-VirtualBox systemd[1]: Starting Elasticsearch...
févr. 29 13:39:39 christelle-VirtualBox systemd-entrypoint[881]: févr. 29, 2024 1:39:38 PM sun.util.locale.provider.LocaleProviderA>
févr. 29 13:39:39 christelle-VirtualBox systemd-entrypoint[881]: WARNING: COMPAT locale provider will be removed in a future release
févr. 29 13:44:22 christelle-VirtualBox systemd[1]: Started Elasticsearch.
lines 1-16/16 (END)
```

FIGURE 3.5 : Elasticsearch status

2.1.2 Kibana

INSTALLATION[24]

Les paquets Elastic étant déjà disponible sur le système, nous allons installé Kibana directement.

```
$ sudo apt-get install kibana
```

CONFIGURATION

Dans le fichier de configuration de Kibana ***/etc/kibana/kibana.yml*** nous allons [25] :

- configurer le numéro de port et l'adresse IP

  L'adresse IP représente la machine par laquelle ou lesquelles Kibana reçoit les données et le port représente le port sur lequel il écoute. Dans notre configuration Kibana reçoit des données de n'importe quelle machine et écoute le port sur 5601.

  La figure 3.6 présente la configuration du port et de l'adresse IP.

```
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "0.0.0.0"
```

FIGURE 3.6 : Configuration réseau Kibana

- configurer le lien des requêtes Elasticsearch

  La figure 3.7 présente la configuration du lien Elasticsearch.

```
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://192.168.8.110:9200"]
```

FIGURE 3.7 : Configuration du lien Elasticsearch

- configurer le nom d'utilisateur et le mot de passe des utilisateurs intégrés

  Les fonctionnalités de sécurité d'Elasticsearch étant activées, on accède à Kibana en se connectant avec un nom d'utilisateur et un mot de passe valide. Le nom d'utilisateur est configuré avec l'utilisateur intégré **kibana_system** et le mot de passe avec le mot de passe valide **password**.

  La figure 3.8 présente la configuration du nom d'utilisateur et du mot de passe.

```
# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
#elasticsearch.username: "kibana_system"
elasticsearch.username: "kibana_system"
elasticsearch.password: "password"
```

FIGURE 3.8 : Configuration du nom d'utilisateur et du mot de passe

Nous obtenons le mot de passe en exécutant la commande suivante : **elasticsearch-setup-passwords** dans le répertoire **/usr/share/elasticsearch/bin**.

```
$ ./elasticsearch-setup-passwords interactive
```

- configurer les paramètres d'alertes

  Dans le répertoire **/usr/share/kibana/bin** nous exécutons la commande **kibana-encryption-keys** pour générer des clés de chiffrement.

```
$ ./kibana-encryption-keys generate
```

  La figure 3.9 présente la génération des clés de chiffrement.

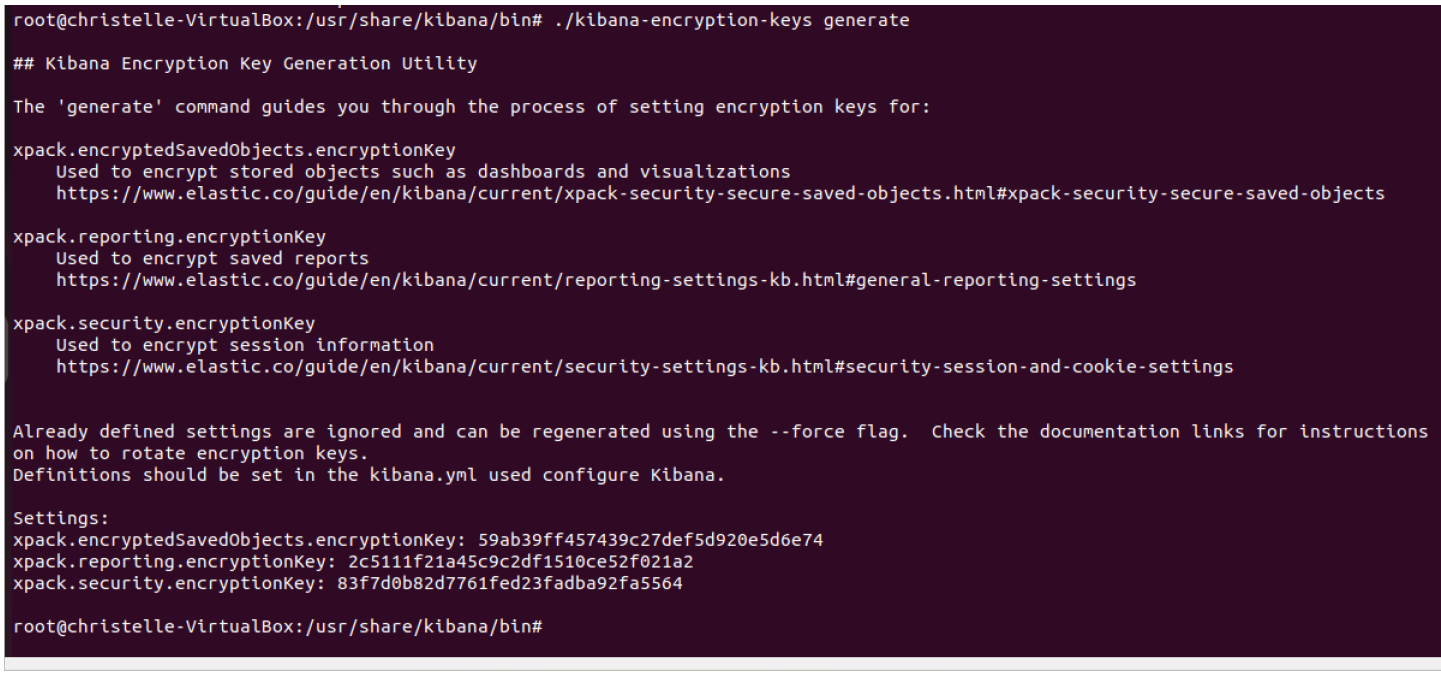


FIGURE 3.9 : Configuration de l'alerting

  Ces clés de chiffrement sont ensuite ajoutées dans fichier de configuration de Kibana **/etc/kibana/kibana.yml**.

Dans le répertoire de configuration de Kibana **/etc/kibana** nous allons :

- créer le magasin de clés de Kibana et y ajouter le mot de passe de l'utilisateur **kibana_system**.

  Dans le répertoire **/usr/share/kibana/bin** nous exécutons les commandes suivantes :

```
$ ./kibana-keystore create
$ ./kibana-keystore add elasticsearch.password
```

- activer Kibana et permettre son initialisation à chaque démarrage du système

```
$ sudo systemctl start kibana
$ sudo systemctl enable kibana
```

- vérifier l'état de Kibana

```
$ sudo systemctl status kibana
```

  La figure 3.10 présente l'état de Kibana.

```
root@christelle-VirtualBox:/etc/kibana# systemctl status kibana
● kibana.service - Kibana
     Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-03-02 06:10:54 WAT; 35min ago
       Docs: https://www.elastic.co
   Main PID: 6574 (node)
      Tasks: 15 (limit: 4599)
     Memory: 300.9M
        CPU: 1min 53.903s
     CGroup: /system.slice/kibana.service
             └─6574 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bin/../src/cli/dist --logging.dest=/var/log/kibana/kib

mars 02 06:10:54 christelle-VirtualBox systemd[1]: Started Kibana.
lines 1-12/12 (END)
```

FIGURE 3.10 : Kibana status

L'interface graphique de Kibana est accessible via un navigateur. Nous nous connectons avec l'utilisateur intégré **elastic** et le mot de passe précédemment configuré.

La figure 3.11 présente la page de connexion de Kibana.

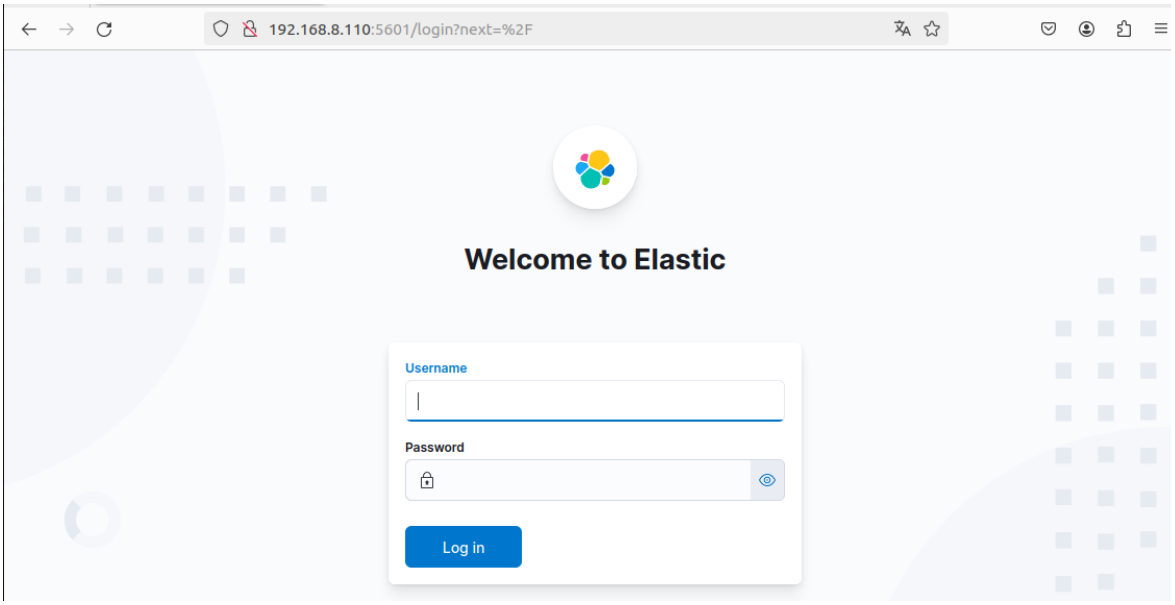


FIGURE 3.11 : Page d'authentification

2.1.3 Metricbeat

Nous installons Metricbeat sur le serveur Ubuntu

INSTALLATION

Nous allons d'abord télécharger les paquets Elastic puis installer Metricbeat [26]

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
    sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-
   keyring.gpg
$ sudo apt-get install apt-transport-https
$ echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.
   gpg] https://artifacts.elastic.co/packages/7.x/apt stable main
   " | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
$ sudo apt-get update && sudo apt-get install metricbeat
```

CONFIGURATION[27]

En premier lieu nous allons modifier le fichier de configuration de Metricbeat. Dans **/etc/metricbeat/metricbeat.yml** nous allons :

- activer le rechargement automatique de la configuration.

  L'activation du rechargement automatique permet à Metricbeat de surveiller périodiquement les fichiers de configuration, de détecter les changements et de recharger l'ensemble de la configuration.

  La figure 3.12 présente l'activation du rechargement automatique

```
# ========================== Modules configuration ============================

metricbeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml

  # Set to true to enable config reloading
  reload.enabled: true

  # Period on which files under path should be checked for changes
  #reload.period: 10s
```

FIGURE 3.12 : Activation du rechargement automatique

- configurer le point de terminaison de Kibana

  La figure 3.13 présente la configuration du point de terminaison kibana.

```
# ================================== Kibana ===================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "192.168.8.110:5601"
```

FIGURE 3.13 : Configuration du point de terminaison kibana

- configurer le point de sortie des données collectées

  Dans notre cas les données collectées sont envoyées à Elasticsearch. Nous allons fournir l'adresse de la machine sur laquelle Elasticseach est installé et les informations de connexion à Kibana.

  La figure 3.14 présente la configuration du point de sortie Elasticsearch.

```
# ------------------------------ Elasticsearch Output -------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.8.110:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "password"
```

FIGURE 3.14 : Configuration du point de sortie Elasticsearch

Nous vérifions ensuite l'état de Metricbeat

```
1    $ sudo systemctl status metricbeat
```

La figure 3.15 présente le status de Metricbeat

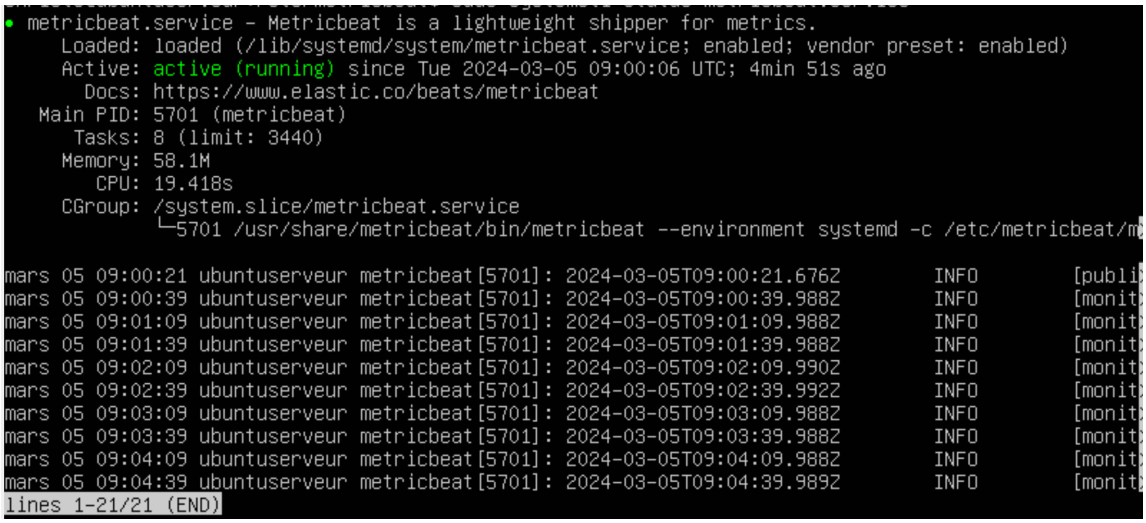


FIGURE 3.15 : Metricbeat status

Enfin nous allons activer le module **system** afin de surveiller et collecter les métriques systèmes du serveur telles que l'utilisation du processeur et de la mémoire, du trafic réseau, du disque. Nous activons le module **system** avec la commande suivante :

```
1    $ sudo metricbeat modules enable system
```

Sur Kibana nous avons une représentation des données collectées par Metricbeat.
La figure 3.16 présente le tableau de bord de Kibana.

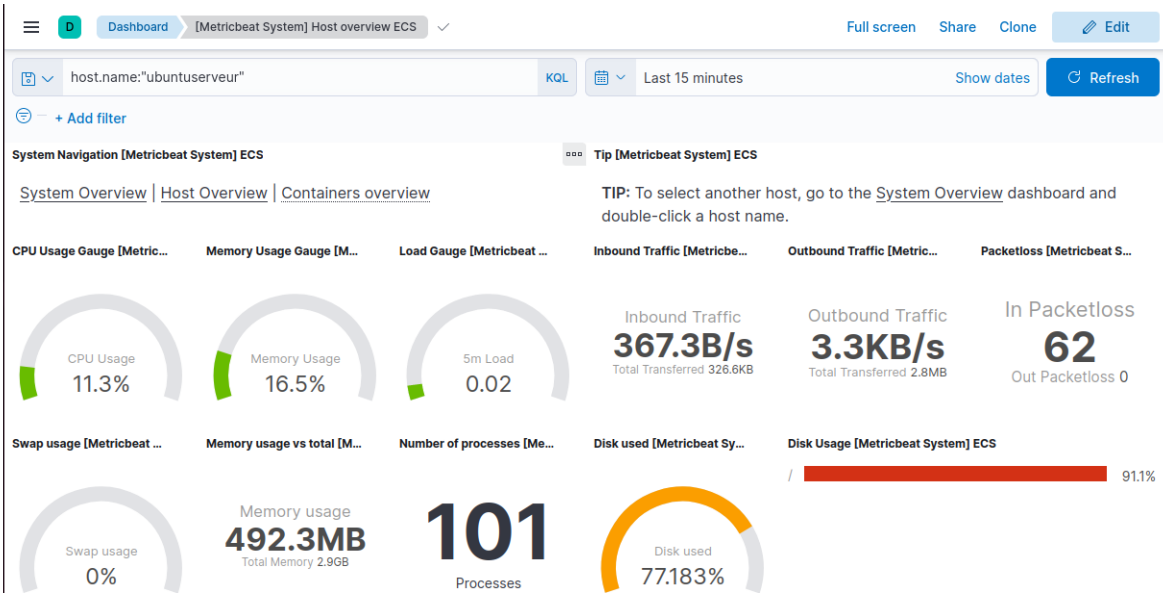


FIGURE 3.16 : Tableau de bord de Kibana

2.2 Slack

Dans **Ubuntu Software** nous recherchons et installons le logiciel Slack.
La figure 3.17 présente l'installation de Slack.

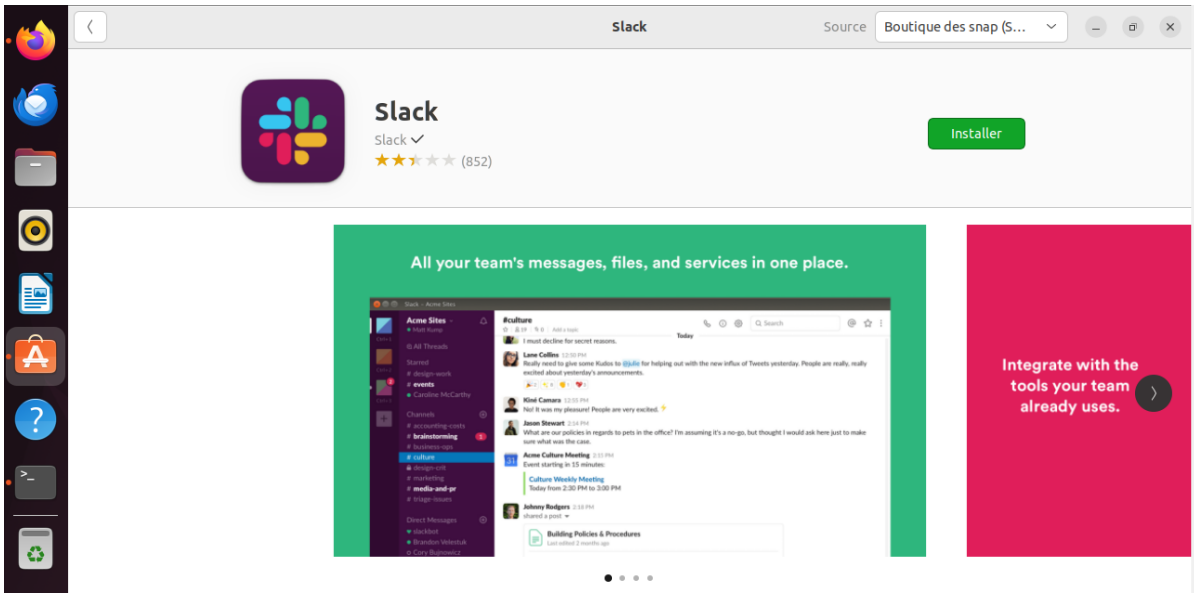


FIGURE 3.17 : Installation de Slack

Dans slack nous allons créer un espace de travail, une équipe puis y ajouter des membres. La figure 3.18 présente l'interface de Slack.

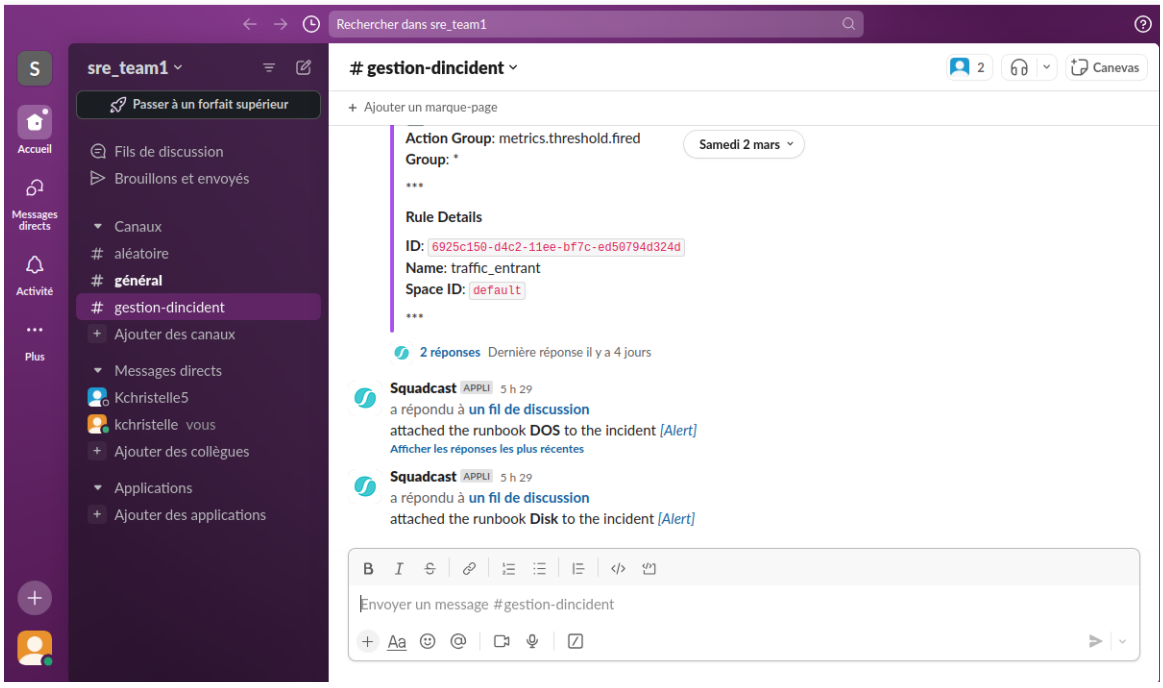


FIGURE 3.18 : Interface de Slack

2.3 Squadcast

INSTALLATION

Squadcast est un logiciel basé sur le cloud, nous y accédons via un navigateur web avec l'URL **https ://www.squadcast.com**. Pour utiliser Squadcast nous créons un compte. Une fois notre compte créé nous accédons à Squadcast en nous connectant.

La figure 3.19 présente la page de connexion de Squadcast.

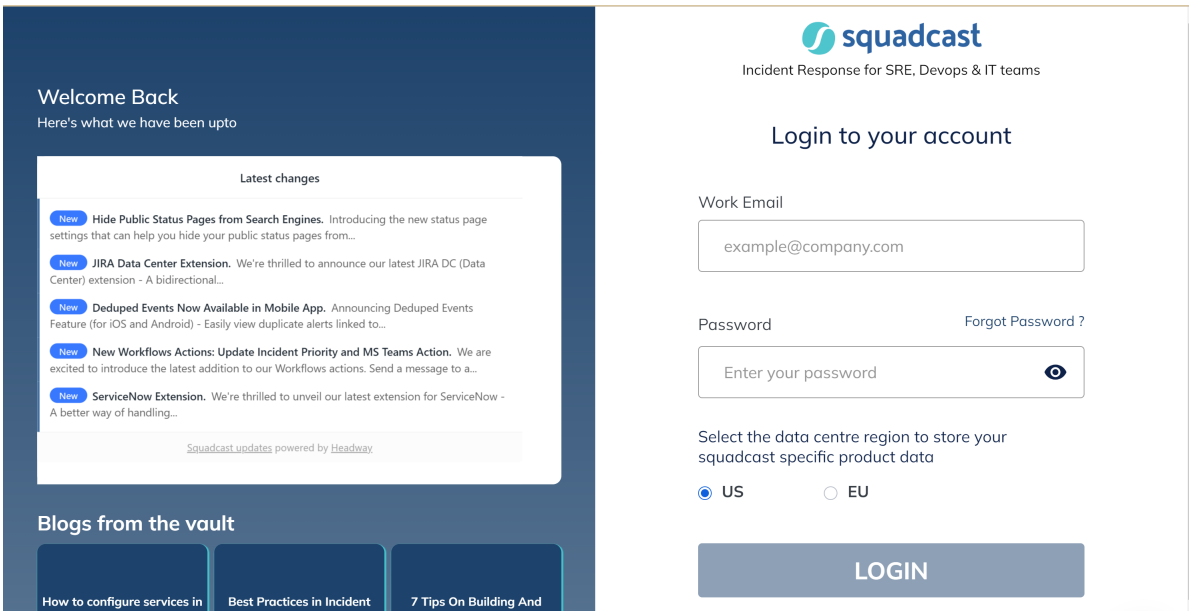


FIGURE 3.19 : Page de connexion de Squadcast

CONFIGURATION[28]

Nous allons intégrer Kibana à Squadcast et Squadcast à Slack.

- Intégration de Kibana à Squadcast

  Nous allons d'abord créer une politique d'escalade.

  Une politique d'escalade permet d'informer la bonne équipe au bon moment lorsqu'une alerte survient.

  Dans **Escalation policies** nous allons ajouter une nouvelle politique d'escalade.

  Pour créer une politique de remontée nous donnons un **nom à la politique d'escalalde**, attribuons un **propriétaire** , ajoutons un **User** et créons une règle.

  La figure 3.20 présente la définition du nom et du propriétaire de la politique d'escalade.



FIGURE 3.20 : Nom et propriétaire de la politique d'escalade

La figure 3.21 présente la règle que la politique suivra.

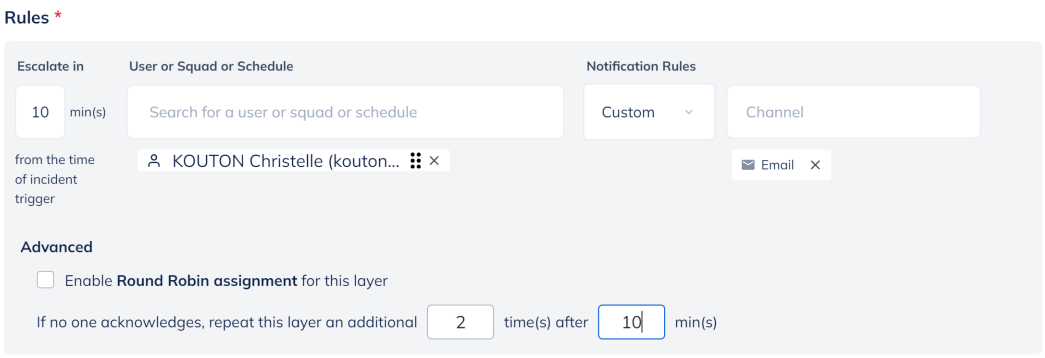


FIGURE 3.21 : Régle de la politique

Ensuite nous allons créer un service. Un service fait référence à un composant, un système ou une application au sein d'une infrastructure informatique qui nécessite une surveillance et une gestion des incidents.

Dans **Service** nous ajoutons un nouveau service.

La création d'un service se fait en deux étapes. La première étape consiste à définir le **nom du service**, choisir une **politique d'escalade**, choisir un **propriétaire** et à entrer des **balises**

La figure 3.22 présente la définition des informations du service.

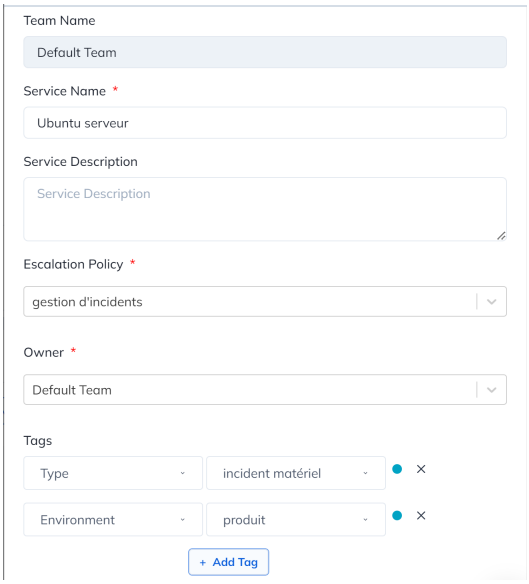


FIGURE 3.22 : Définition du service

La deuxième étape consiste à ajouter une source d'alerte. Nous choisissons Kibana.

Nous allons copier le Webhook URL, ajouter la source et sauvegarder.

La figure 3.24 présente l'ajout d'une source d'alerte.

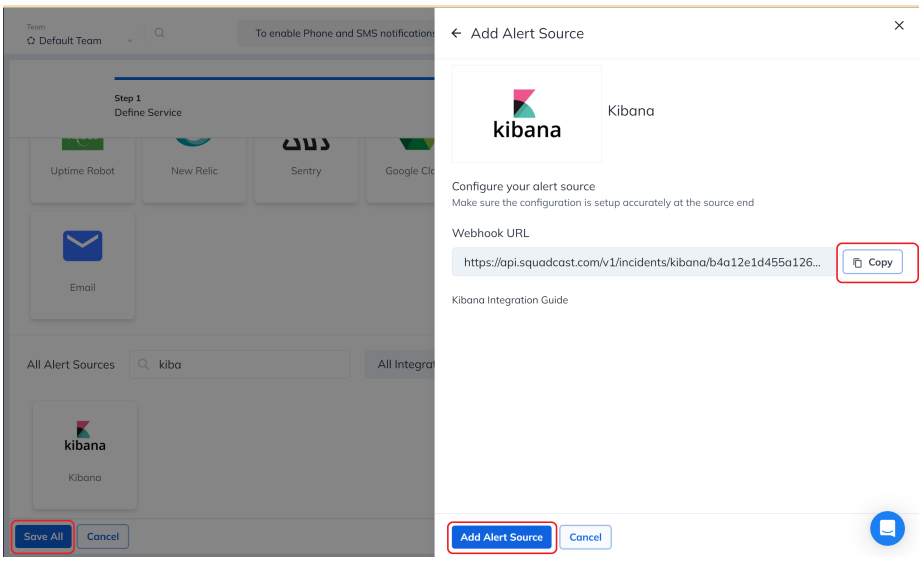


FIGURE 3.23 : Ajout d'une source d'alerte

Enfin nous allons créer un connecteur webhook.

Dans Kibana nous allons coller l'URL précédemment copiée.

La figure 3.24 présente la création du connecteur webhook.

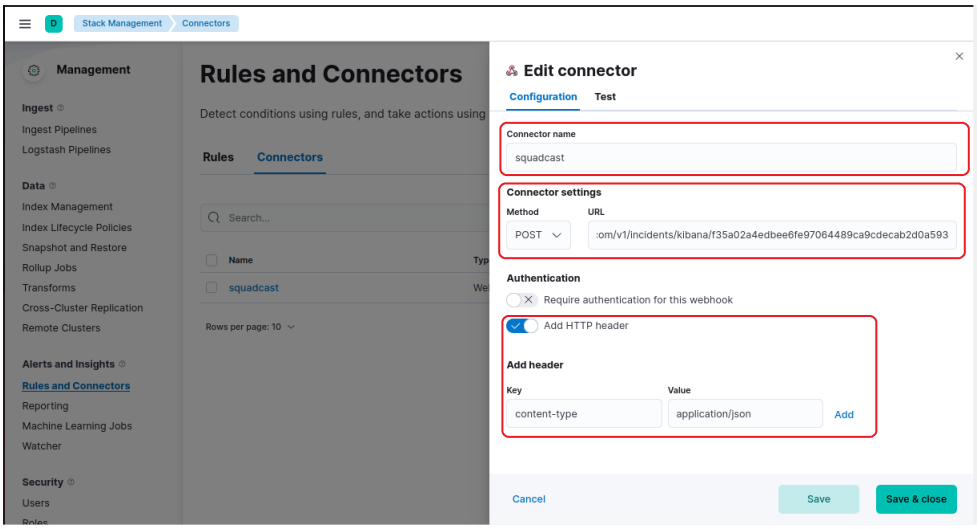


FIGURE 3.24 : Création d'un connecteur webhook

- Intégration de Squadcast à Slack

  Dans **Setting** puis **Extension** nous allons intégrer Squadcast à Slack en autorisant Squadcast à accéder à Slack puis en choisissant le canal Slack dans lequel les alertes Squadcast seront envoyées.

  La figure 3.25 présente l'intégration de Squadcast à Slack.

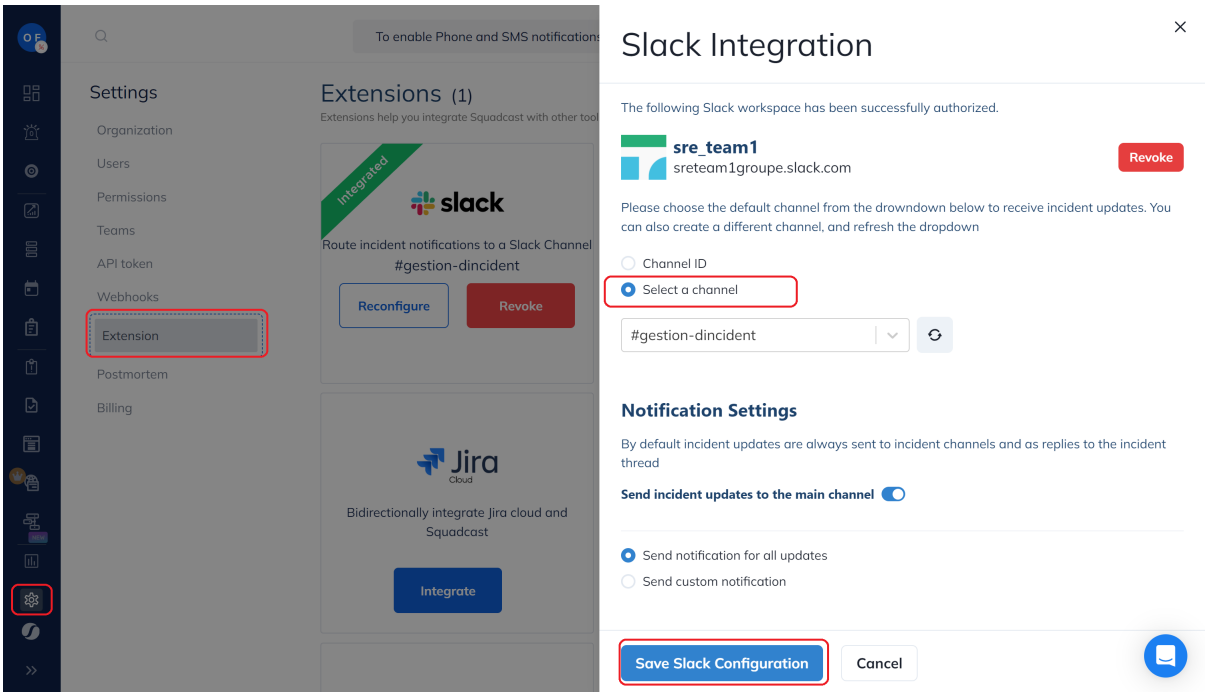


FIGURE 3.25 : Intégration de Squadcast à Slack

Ensuite nous autorisons la redirection des incidents du service **Ubuntu serveur** vers la chaîne Slack intégrée .

La figure 3.26 présente l'ajout d'une chaîne Slack à un service.

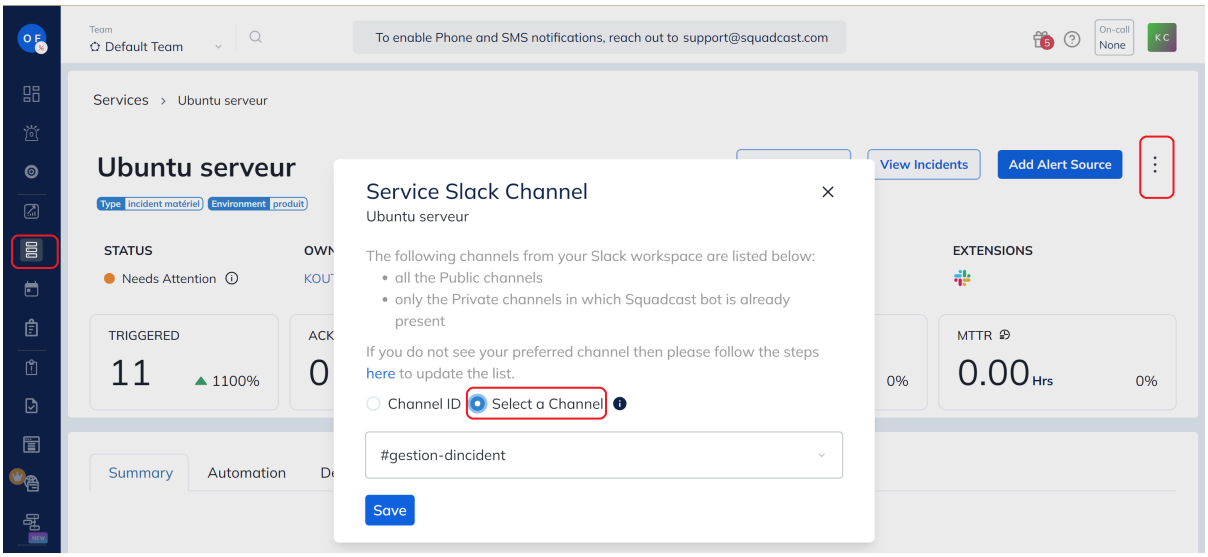


FIGURE 3.26 : Configuration de Slack à un service

2.4 Terraform

INSTALLATION [29]

Pour installer Terraform nous exécutons les commandes suivantes :

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --
    dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.
   gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" |
    sudo tee /etc/apt/sources.list.d/hashicorp.list
$ sudo apt update && sudo apt install terraform
```

Vérifions que Terraform a bien été installé :

```
christelle@christelle-VirtualBox:~$ terraform --help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
  version       Show the current Terraform version
  workspace     Workspace management
```

FIGURE 3.27 : Vérification de l'état du service Terraform

2.5 Tests de fonctionnalité

Nous allons effectuer deux tests avec des incidents survenus au Port Autonome de Cotonou (PAC).

2.6 Premier test

Pour ce premier test nous allons simuler une **surcharge de la bande passante** qui peut être un problème à l'origine du ralentissement des logiciels. Pour ce faire nous allons augmenter le trafic des données entrant dans le Ubuntu serveur.

Dans un premier temps nous allons créer la règles d'alerte Kibana qui permet de détecter automatiquement l'incident lorsque le trafic de donnée entrant dans le serveur augmente de manière anormale.
La figure 3.28 présente la création de la règle de détection de l'augmentation du trafic réseau entrant.

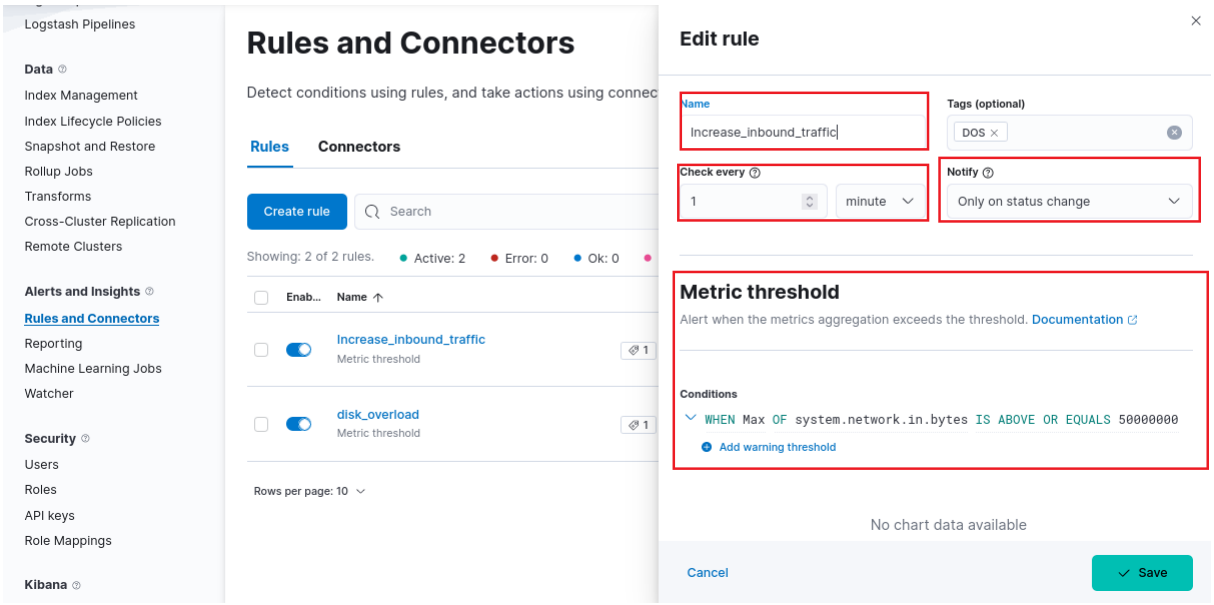


FIGURE 3.28 : Alerte de la surcharge de la bande passante

Ensuite nous allons réaliser un **ping flood**. Le ping flood est une forme d'attaque par déni de service qui repose sur le protocole ICMP (Internet Control Message Protocol). Il provoque la surcharge de l'ordinateur cible par l'envoie en vague de paquets « Echo Request » ICMP.

Nous réalisons le ping flood avec l'outil **hping3**.

La commande suivante envoie un ping le plus rapidement possible à l'adresse de la machine cible 192.168.8.110 sur le port 80 et rejette les paquets Echo Reply entrant.

```
1    $ hping3 --icmp --flood --rand-source -p 80 192.168.8.110
```

Lorsque le trafic des données entrants dépasse le seuil configuré nous recevons une alerte dans Squadcast et un message d'alerte dans Slack. Dans le message nous retrouvons plusieurs informations telles que le nom de l'incident, la source d'alerte, le service, la priorité. Ici l'incident est détecté automatiquement et est assigné à l'équipe adéquate. Les informations contenue dans le message d'alerte font référence au phase d'identification de l'incident, d'analyse, d'hiérarchisation et d'affectation du processus de gestion des incidents qui ont été réalisé automatique.

La figure 3.29 présente le message d'alerte Slack provenant de Squadcast.

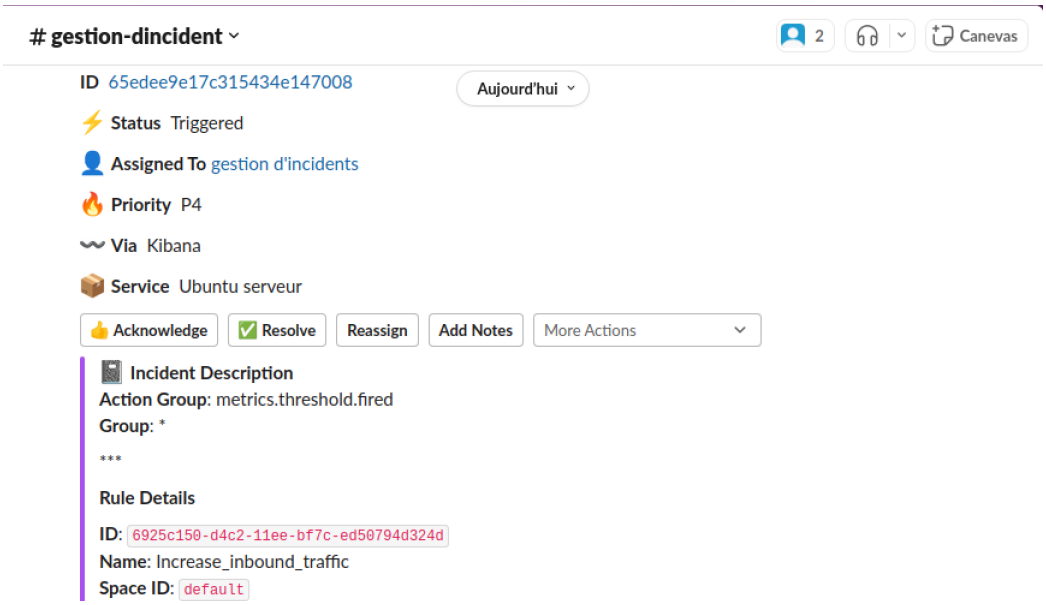


FIGURE 3.29 : Alerte de la surcharge du disque

La figure 3.30 présente l'alerte sur le tableau de bord de Squadcast.

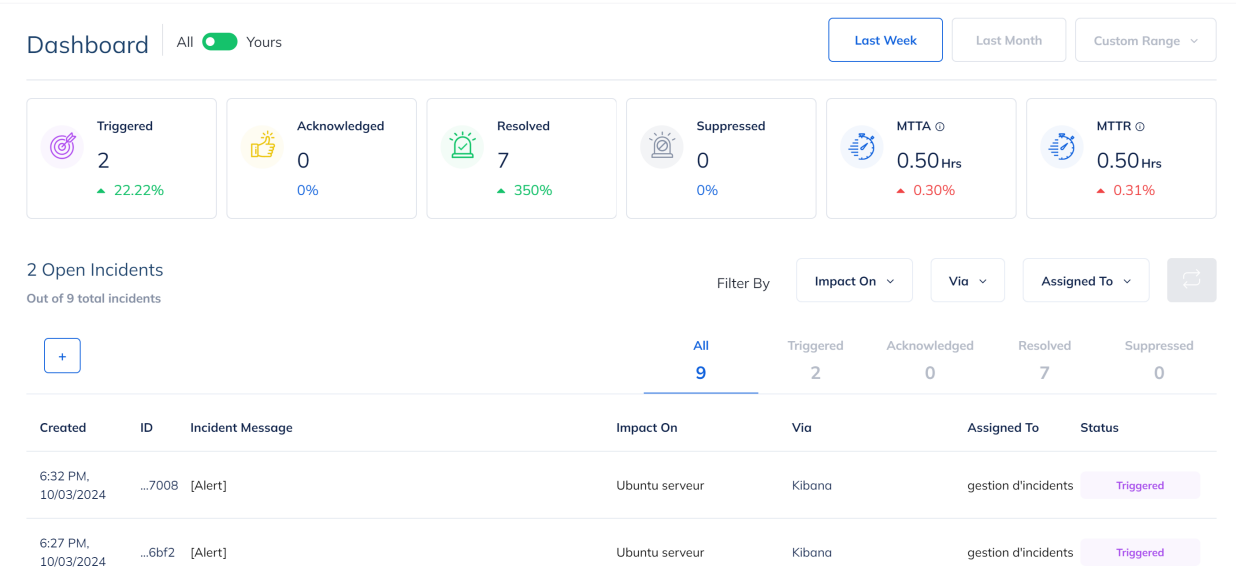


FIGURE 3.30 : Alerte Squadcast

Nous résolvons l'incident à l'aide de Terraform. Pour résoudre l'incident nous allons créer un fichier de configuration Terraform dans lequel nous allons d'abord établir une connexion automatique au serveur par SSH puis exécuter un script bash de règles iptables. Ces règles iptables limitent le taux de nouvelles connexions à vingt par minute.

La figure 3.31 présente le fichier de configuration Terraform pour la résolution de l'attaque DOS.

```
resource "null_resource" "dos" {

  connection {
    type        = "ssh"
    user        = "christe"

    password    = "${var.pw}"
    host        = "192.168.8.110
    agent       = false
    timeout     = "5m"
  }
  provisioner "remote-exec" {
    inline = [

      "echo ${var.pw} | sudo -S chmod u+x s_dos.sh",
      "./dos.sh",
    ]
  }
}
```

FIGURE 3.31 : Script pour la résolution de l'attaque DOS

Nous exécutons notre fichier avec les commandes suivante :

```
$ sudo terraform init
$ sudo terraform plan
$ sudo terraform apply
```

2.7 Deuxième test

Pour ce deuxième test nous allons simuler une **surcharge du disque** sur notre Ubuntu serveur

Dans un premier temps nous allons créer la règles d'alerte Kibana qui permet de détecter automatiquement l'incident.
La figure 3.32 présente la création de la règle de détection pour la surcharge du disque dur.

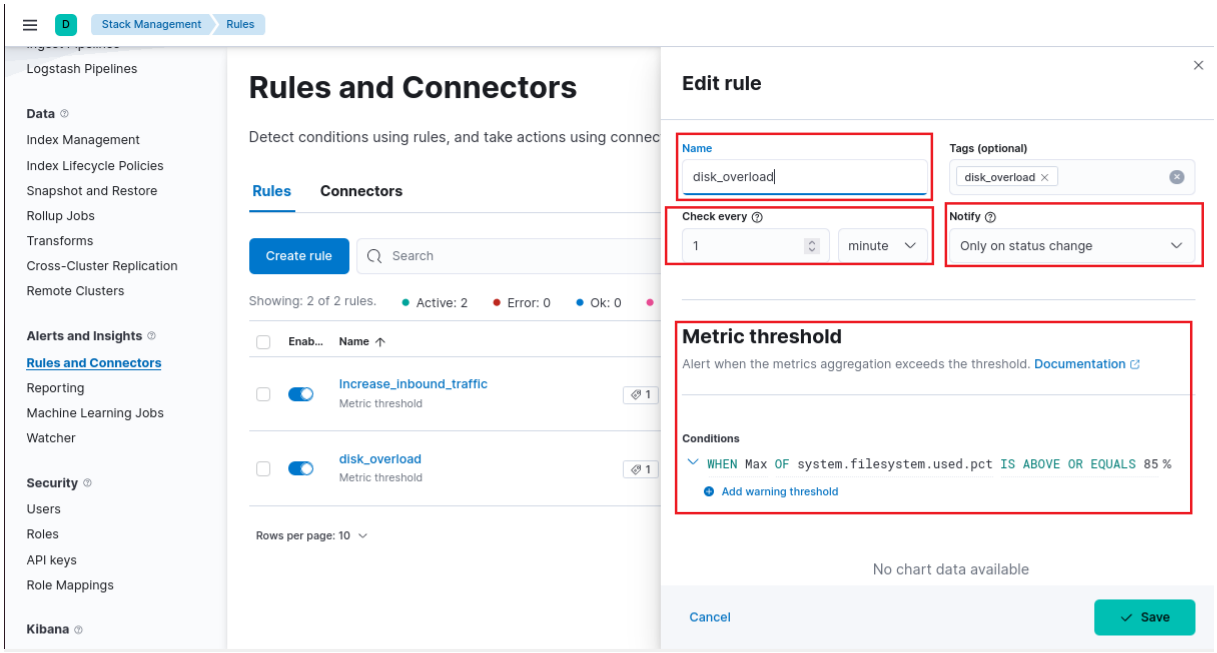


FIGURE 3.32 : Alerte de la surcharge du disque

Lorsque nous atteignons ou dépassons le seuil de la capacité du disque dur configuré l'équipe assignée sera informée en temps réel par un message d'alerte Slack. Nous allons utiliser la commande **wget** pour télécharger le fichier.

```
$ wget https://releases.ubuntu.com/focal/ubuntu-20.04.6-live-server-amd64.iso
```

Après le téléchargement nous reçevons quelques secondes après une notification de Slack.

Dans le message d'alerte nous retrouvons plusieurs informations telles que le nom de l'incident, la source d'alerte, le service, la priorité. Les informations comprises dans le message d'alerte indique les phases d'identification de l'incident, d'analyse et d'hiérarchisation, d'affectation du processus de gestion des incidents.

La figure 3.33 présente un message d'alerte Slack provenant de Squadcast.

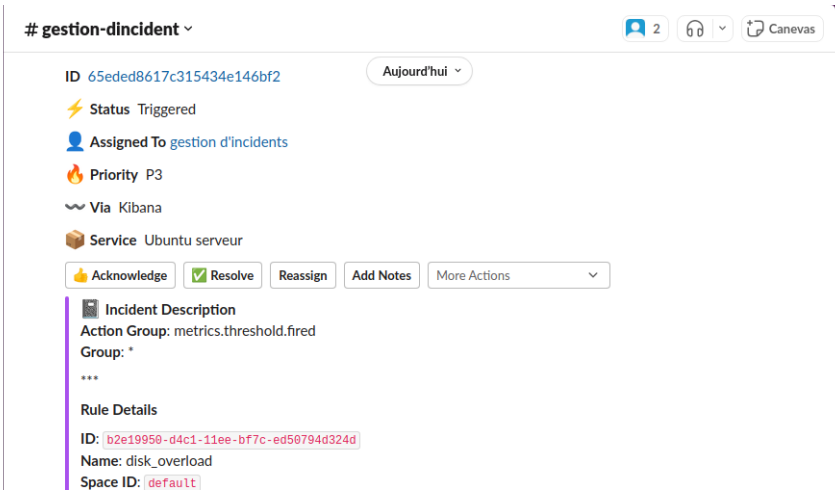


FIGURE 3.33 : Alerte de l'augmentation du trafic entrant

Nous résolvons l'incident à l'aide de Terraform. Pour résoudre l'incident nous allons créer un fichier de configuration Terraform dans lequel nous établissons d'abord une connexion automatique au serveur par SSH. Nous exécutons ensuite des commandes de nettoyage du cache, des paquets périmés, des dépendance inutile.

La figure 3.34 présente le fichier de configuration Terraform pour la résolution de la surcharge du disque dur.

```
resource "null_resource" "disk" {

  connection {
    type        = "ssh"
    user        = "christe"
    password    = "${var.pw}"
    host        = "192.168.8.109"
  }
  provisioner "remote-exec" {
    inline = [
      "./disk.sh",
      "hostname",
      "whoami",
# Supprimer le cache des paquets périmés
      "echo ${var.pw} | sudo -S apt-get autoclean",
# Supprimer tout le cache
      "sudo apt-get clean",
# Supprimer les paquets installés automatiquement comme dépendances et devenus inutiles
      "sudo apt-get autoemove",
      "sudo rm -r ~/ubuntu* ",
    ]
  }
}
```

FIGURE 3.34 : Script pour la résolution de la surcharge du disque

Nous exécutons notre fichier avec les commandes suivante :

```
$ sudo terraform init
$ sudo terraform plan
```

```
3    $ sudo terraform apply
```

Il faut noter que ces tests ne sont que deux parmi tant d'autres et ne montre pas toutes les fonctionnalités de notre solution.

Conclusion

Dans ce chapitre, nous avons décrit en détail le processus de mise en œuvre de notre projet, présenté le fonctionnement de notre solution et décrit les résultats obtenus.

Conclusion Générale et Perspectives

Dans le présent mémoire, nous avons abordé la problématique liée à la gestion des incidents informatiques. L'implémentation d'un système de gestion des incidents représente une étape cruciale dans l'amélioration de la stabilité, de la fiabilité et de la réactivité des services informatiques au sein d'une organisation. À travers ce mémoire, nous avons exploré les principes fondamentaux et rôles de l'ingénierie de fiabilité des sites, les étapes du processus de gestion des incidents. Nous avons examiné en détail les différentes phases du processus de gestion des incidents, de la détection initiale à la résolution finale, en soulignant l'importance de la documentation, de l'analyse post-incident et de l'amélioration continue. Nous avons également exploré les outils et les technologies disponibles pour soutenir la gestion des incidents, en mettant en évidence leur rôle dans l'automatisation des tâches répétitives et la réduction des temps de résolution. Toutefois, il est important de reconnaître que la gestion des incidents est un processus continu qui nécessite un engagement constant envers l'amélioration et l'innovation. En continuant à surveiller, évaluer et ajuster nos processus de gestion des incidents, nous pouvons garantir une prestation de services informatiques toujours plus fiable et réactive, répondant ainsi aux besoins évolutifs de notre organisation et de ses utilisateurs. Ce mémoire a permis d'approfondir notre compréhension de la gestion des incidents.

En perspective nous envisageons de fournir des recommandations automatisées pour la résolution d'incidents similaires passés, d'automatiser la résolution proprement dite de certains incidents par exemple le redémarrages de services ou le nettoyage des caches.

Webographie

[1] Site reliability engineering (sre) what is it, what are the advantages, and how do i implement it ? https://www.linkedin.com/pulse/site-reliability-engineering-sre-what-advantages-how-do-dan-martines, consulté le 05 juin 2024.

[2] How to be an effective incident manager https://www.linkedin.com/pulse/how-effective-incident-manager-hassan-el-hajj, consulté le 04 juillet 2024.

[3] Qu'est-ce qu'un sre (site reliability engineer) ? https://www.gologic.ca/definition-sre/, consulté le 05 juin 2024.

[4] L'ingénierie de la fiabilité des sites, qu'est-ce que c'est ? https://www.redhat.com/fr/topics/devops/what-is-sre, consulté le 05 juin 2024.

[5] Embracing risk https://sre.google/sre-book/embracing-risk/, consulté le 05 juin 2024.

[6] Principes sre : les 7 règles fondamentales https://www.dotcom-monitor.com/blog/fr/principes-sre-les-7-regles-fondamentales/#:~:text=Le%20r%C3%B4le%20de%20SRE%20se,les%20objectifs%20de%20l'entreprise., consulté le 05 juin 2024.

[7] Sre principles and practices for your project https://anywhere.epam.com/en/blog/why-sre-principles-practices, consulté le 05 juin 2024.

[8] Quel est l'objectif principal du sre ? https://medium.com/@EmilyVancampp/whats-the-main-objective-of-sre-429faf47ed41#:~:text=The%20main%20objective%20of%20Site,scale%2C%20highly%20reliable%20software%20systems., consulté le 05 juin 2024.

[9] Devops https://fr.wikipedia.org/wiki/Devops, consulté le 05 juin 2024.

[10] Sre vs. devops : Key differences and how they work together https://www.atlassian.com/devops/frameworks/sre-vs-devops#:~:text=SRE%20focuses%20on%20delivery%20and,they%20can%20complement%20each%20other., consulté le 05 juin 2024.

[11] La gestion des incidents informatiques https://www.itaia.fr/tout-sur-le-cloud-computing/continuite-et-reprise-dactivite-draas/incidents-informatiques/, consulté le 05 juin 2024.

[12] Gestion des incidents pour les équipes haute vélocité https://www.atlassian.com/fr/incident-management#the-importance-of-incident-management, consulté le 05 juin 2024.

[13] Gestion des incidents : Pourquoi et comment gérer et suivre les incidents ? https://blog-gestion-de-projet.com/gestion-incidents/, consulté le 05 juin 2024.

[14] Top 12 site reliability engineering (sre) tools https://lightrun.com/site-reliability-engineering-tools/, consulté le 05 juin 2024.

[15] Datadog vs kibana vs new relic https://stackshare.io/stackups/datadog-vs-kibana-vs-new-relic, consulté le 05 juin 2024.

[16] Compare squadcast vs pagerduty vs opsgenie https://crozdesk.com/compare/squadcast-vs-pagerduty-vs-opsgenie, consulté le 05 juin 2024.

[17] Teams vs slack : Que choisir ? https://www.mozzaik365.com/fr/blog-posts/teams-vs-slack#:~:text=Slack%20se%20distingue%20par%20une,nombreux%20onglets%20verticaux%20et%20horizontaux., consulté le 05 juin 2024.

[18] Slack et telegram https://www.getapp.fr/compare/91563/114023/slack/vs/telegram, consulté le 05 juin 2024.

[19] Ansible et terraform : quelles sont les différences ? https://www.redhat.com/fr/topics/automation/ansible-vs-terraform, consulté le 05 juin 2024.

[20] Comparatif entre saltstack, ansible et puppet http://www.ageekslab.com/art2/, consulté le 05 juin 2024.

[21] Install elasticsearch with debian package https://www.elastic.co/guide/en/elasticsearch/reference/7.17/deb.html, consulté le 04 mars 2024.

[22] Configuring elasticsearch https://www.elastic.co/guide/en/elasticsearch/reference/7.17/settings.html, consulté le 02 mars 2024.

[23] Set up minimal security for elasticsearch https://www.elastic.co/guide/en/elasticsearch/reference/current/security-minimal-setup.html, consulté le 04 mars 2024.

[24] Install kibana with debian package https://www.elastic.co/guide/en/kibana/7.17/deb.html, consulté le 08 mars 2024.

[25] Installation et configuration de la stack elk https://devopssec.fr/article/installation-configuration-elk, consulté le 10,mars 2024.

[26] Metricbeat quick start : installation and configuration https://www.elastic.co/guide/en/beats/metricbeat/7.17/metricbeat-installation-configuration.html#metricbeat-installation-configuration, consulté le 13 mars 2024.

[27] Configure modules https://www.elastic.co/guide/en/beats/metricbeat/7.17/configuration-metricbeat.html#config-variants, consulté le 14 mars 2024.

[28] Kibana https://support.squadcast.com/integrations/alert-source-integrations-native/kibana, consulté le 14 mars 2024.

[29] Install terraform https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli, consulté le 14 mars 2024.

Table des matières

Dédicace **ii**

Remerciements **iii**

Résumé **iv**

iv

Abstract **v**

v

Liste des figures **vi**

Liste des tableaux **vii**

Liste des acronymes **viii**

Introduction **1**

1 Problématique . . . 1
2 Contexte . . . 1
3 Objectif général . . . 2
4 Objectifs spécifiques . . . 2
5 Les contributions des travaux . . . 2
6 Organisation du mémoire . . . 2

1 Revue de littérature **3**

Introduction . . . 3
1 Framework SRE . . . 3
1.1 Définition . . . 3
1.2 Les principes fondamentaux de l'ingénierie de fiabilité de site (SRE) . . . 3
1.2.1 Acceptation du risque . . . 3
1.2.2 Les objectifs de niveaux de service (SLI, SLO, SLA) . . . 4
1.2.3 Eliminer le labeur . . . 4
1.2.4 Surveillance des systèmes distribués . . . 4
1.2.5 Automatisation . . . 4
1.2.6 Ingénierie des versions . . . 5
1.2.7 Simplicité . . . 5
1.3 Les principaux objectifs du framework SRE . . . 6
2 SRE un cadre de travail ITSM . . . 6
2.1 ITSM . . . 6

2.1.1 Définition . . . 6
2.1.2 Les principaux objectifs de l'ITSM . . . 6
2.1.3 SRE un cadre ITSM . . . 6
3 SRE et DevOps . . . 7
3.1 DevOps . . . 7
3.2 Relation entre SRE et DevOps . . . 7
4 La gestion des incidents . . . 8
4.1 Incident informatique . . . 8
4.1.1 Définition . . . 8
4.1.2 Les catégories d'incidents informatiques . . . 8
4.2 La gestion des incidents . . . 8
4.2.1 Définiton . . . 8
4.2.2 Processus de gestion des incidents . . . 8
Conclusion . . . 10

2 Présentation et choix des outils du système de gestion automatisé des incidents 11
Introduction . . . 11
1 Les outils SRE . . . 11
1.1 APM (Application Performance Management) ou outils de surveillance . . . 11
1.2 Système automatisé de réponse aux incidents . . . 12
1.3 Outils de communication à temps réel . . . 12
1.4 Outils de configuration et d'automatisation . . . 13
2 Présentation des solutions choisies . . . 14
2.1 Kibana . . . 14
2.2 Squadcast . . . 14
2.3 Slack . . . 14
2.4 Terraform . . . 14
2.5 Brève description de la solution . . . 15
Conclusion . . . 15

3 Implémentation du système de gestion des incidents et test de fonctionnalité 16
Introduction . . . 16
1 Mode de fonctionnement du système de gestion d'incident . . . 16
2 Installation et configuration des logiciels . . . 17
2.1 Suite elastic . . . 17
2.1.1 Elasticsearch : . . . 17
2.1.2 Kibana . . . 20
2.1.3 Metricbeat . . . 22
2.2 Slack . . . 24
2.3 Squadcast . . . 25
2.4 Terraform . . . 29
2.5 Tests de fonctionnalité . . . 30
2.6 Premier test . . . 30
2.7 Deuxième test . . . 33
Conclusion . . . 35

Conclusion 36

Webographie 37

Table des matières 40