

Année académique 2023 – 2024

N° d'ordre : 2024/MRI-009

MEMOIRE DE FIN DE FORMATION

pour l'obtention du diplôme de

LICENCE PROFESSIONNELLE

Option : Maintenance et Réseaux Informatiques

Thème :

Etude et mise en place d'un système de monitoring des infrastructures informatiques : cas de Dac Technologies

Présenté et soutenu publiquement le 21/10/2024 par

AGBODJAN Deuxévi Adjé Marcel

Devant le jury composé de :

Président du jury Dr. Ing. Barèrèm-Mêlgueba MAO, Ph. D
Enseignant-chercheur à l'Université de Lomé

Examinateur Mr. Mohamed Barate, Enseignant à l'Université de Lomé

Directeur de mémoire Dr. AGBESSI Akuété Pierre, Enseignant-chercheur à l'École Polytechnique de Lomé

Maître de stage Mr. ATSOU Sélom Kodjo, Administrateur système

Dédicaces

À ma mère,
voici la fleur de tes sacrifices, les fruits suivront

À mes grands-parents,
votre affection a toujours été mon moteur pour avancer

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à l'accomplissement de ce mémoire.

- Au **Pr. Kondo Hloindo Adjallah**, éminent Professeur et Directeur de l'École Polytechnique de Lomé (EPL), pour son engagement indéfectible envers cette institution et ses étudiants ;
- Au **Dr. Kossi ATCHONOUNGLO**, Maître de Conférences, ancien directeur du CIC et actuel directeur de la Direction des Ressources et Systèmes Informatiques (DRSI). Nous tenons à lui exprimer nos sincères remerciements pour sa guidance et son soutien inestimables tout au long de notre parcours ;
- Au **Pr. Akim Adékunlé SALAMI** : Professeur, Directeur Adjoint chargé des études. Nous tenons à le remercier pour les efforts qu'il déploie pour le bon déroulement des cours ;
- Au **Dr. Tchamye Tchaa-Essou BOROZE**, Maître de Conférences, ancien Directeur Adjoint chargé des études au CIC et actuel Directeur Adjoint à la vie estudiantine et aux partenariats externe de l'EPL. Nous tenons à le remercier non seulement pour son dévouement pour la cause des étudiants et sa culture de l'excellence mais aussi pour sa démarche d'implication des étudiants dans les prises de décisions. Nous sommes reconnaissants pour l'attention qui nous est accordée ;
- Au **Dr. AGBESSI Akuété Pierre**, Enseignant chercheur à l'EPL, par ailleurs mon directeur de mémoire, pour ses conseils, l'assistance et les conseils prodigués non seulement dans la rédaction de ce document, mais aussi durant les cours ;
- A tout le **corps enseignant** de l'EPL, pour leurs enseignements, explications et conseils qui nous ont été précieux ;
- A tout le personnel administratif et technique de l'EPL, pour votre disponibilité et les services rendus ;
- A **M. Fransisco AKUESON**, Directeur Général de Dac Technologies, pour sa confiance en nos capacités et de nous avoir accueillis au sein de l'équipe de Dac Technologies dans le cadre de notre stage de fin de formation. Sa générosité et son soutien ont grandement enrichi notre expérience professionnelle.
- A **M. Séлом K. ATSOU**, mon maître de stage, pour ses conseils avisés et les heures passées avec moi sur ce projet, de sa conception à sa réalisation, et jusqu'à la rédaction de ce document ;
- A Mes oncles **Michel** et **Innocent**, pour avoir suscité en moi l'envie de découvrir l'univers informatique ;
- A Mes frères **Richard** et **Roland**, pour votre soutien, vos conseils, et parfois même vos petites taquineries ;

- A **Ma famille**, pour votre amour et votre encouragement, c'est aussi grâce à vous que je suis arrivé à cette étape de ma vie ;
- A tous mes amis, camarades étudiants, aînés et collègues, pour leur soutien, leur aide et les moments partagés qui ont enrichi notre expérience ;

Résumé

Ce mémoire présente une étude et la mise en place d'un système de surveillance des infrastructures informatiques au sein de Dac Technologies, dans le cadre d'un stage de fin de formation. Le point de départ de ce projet est la problématique liée à l'insuffisance des outils de monitoring déjà intégrés aux systèmes informatiques de l'entreprise. Ces derniers ne permettaient pas une surveillance efficace et centralisée, avec des limites notables dans la visualisation des métriques critiques, la recherche dans les logs et l'émission d'alertes en cas de dysfonctionnements. Cette situation posait des risques quant à la disponibilité des services, la performance des systèmes, et la capacité de réponse rapide en cas d'incident.

Face à ces défis, l'objectif principal est d'implémenter une solution de surveillance basée sur Grafana, Prometheus et Loki, afin d'optimiser la disponibilité, la performance et la sécurité de ses systèmes critiques. Le projet s'appuie sur une méthodologie rigoureuse. Tout d'abord, une étude préliminaire des besoins spécifiques de Dac Technologies a été réalisée, révélant les attentes en termes de monitoring des infrastructures. À la suite de cette analyse, une recherche a été effectuée pour identifier les solutions les plus adaptées, ce qui a conduit à la sélection de trois outils complémentaires : Grafana, pour la visualisation des données, Prometheus, pour la collecte et le stockage des métriques, et Loki, pour l'agrégation et la gestion des journaux de logs. Ces outils ont été choisis pour leur capacité à répondre aux exigences techniques et fonctionnelles de Dac Technologies.

Les résultats obtenus démontrent l'efficacité de la solution déployée, notamment à travers la mise en place de tableaux de bord dans Grafana, permettant une visualisation claire et intuitive des métriques critiques, telles que l'utilisation du CPU, la mémoire et le stockage. Prometheus a été configuré pour collecter ces métriques essentielles, tandis que Loki a facilité la centralisation et la recherche dans les logs, renforçant la sécurité et la traçabilité des actions, ainsi que les capacités de diagnostic en cas d'incidents. Enfin, un système d'alertes automatisées a été mis en place pour notifier en temps réel toute anomalie ou saturation de ressources, permettant ainsi une réponse proactive et optimisant la réactivité de l'équipe.

L'évaluation du système a montré des résultats positifs, notamment une amélioration significative de la surveillance des infrastructures, avec une réduction notable des temps de réponse aux incidents. Le système mis en place offre une vue d'ensemble de l'état des infrastructures, garantissant ainsi une meilleure gestion des performances et de la sécurité des systèmes.

Cette solution permet à Dac Technologies de mieux surveiller et gérer ses infrastructures, contribuant ainsi à garantir une performance optimale et une gestion simplifiée des incidents. Les perspectives incluent l'amélioration de la sécurité au niveau des points de

Étude et la mise en place dun système de monitoring des infrastructure informatique

contact avec les utilisateurs et l'organisation de l'accès aux tableaux de bord, de sorte que chaque utilisateur ne visualise que les informations dont il a besoin.

Mots clés : système de monitoring, infrastructure informatique, Grafana, Loki, Prometheus

Abstract

This report presents a study and implementation of an IT infrastructure monitoring system at Dac Technologies as part of a final training internship. The project began by addressing a central issue : the limitations of existing monitoring tools integrated into the company's IT systems. These tools did not allow for effective and centralized monitoring, with notable shortcomings in critical metric visualization, log search capabilities, and alerting for potential malfunctions. This situation posed risks for service availability, system performance, and rapid incident response.

To tackle these challenges, the primary objective was to implement a monitoring solution using Grafana, Prometheus, and Loki, aiming to optimize the availability, performance, and security of critical systems. The project follows a rigorous methodology. First, a preliminary assessment of Dac Technologies' specific needs was conducted, revealing expectations for infrastructure monitoring. Following this analysis, research was carried out to identify the most suitable solutions, leading to the selection of three complementary tools : Grafana for data visualization, Prometheus for metric collection and storage, and Loki for log aggregation and management. These tools were selected for their ability to meet the technical and functional requirements of Dac Technologies.

The results demonstrate the effectiveness of the deployed solution, notably through the creation of Grafana dashboards, which provide a clear and intuitive view of critical metrics such as CPU usage, memory, and storage. Prometheus was configured to collect these essential metrics, while Loki facilitated centralized log management and search, enhancing security, traceability, and diagnostic capabilities in the event of incidents. Additionally, an automated alerting system was established to provide real-time notifications for anomalies or resource saturation, allowing a proactive approach and improving the team's responsiveness.

The system evaluation showed positive outcomes, including a significant improvement in infrastructure monitoring and a marked reduction in incident response times. The implemented system provides an overview of the infrastructure status, ensuring better performance and security management.

This solution enables Dac Technologies to more effectively monitor and manage its infrastructures, thus contributing to optimal performance and streamlined incident management. Future enhancements include strengthening security at user interaction points and organizing dashboard access so that each user only views the information relevant to their role.

Keywords : monitoring system, IT infrastructure, Grafana, Loki, Prometheus

Liste des Figures

1.1	Une vue du bâtiment du CIC	3
1.2	Une vue du bâtiment de l'EPL	4
1.3	Plan de présentation de l'EPL - Google Maps [2]	5
1.4	Organigramme administratif de l'EPL	5
1.5	Organigramme de Dac Technologies	8
3.1	Illustration simplifiée d'une infrastructure informatique [5]	15
3.2	Architecture de Prometheus [13]	26
3.3	Architecture interne de Loki[15]	28
4.1	Architecture simplifiée de la solution	33
4.2	Installation de Prometheus	34
4.3	Vérification des services Prometheus, Node-exporter et Alertmanager	35
4.4	Exposition des métriques par 'node-exporter'	40
4.5	Interface web de Prometheus	40
4.6	Exécution de la requête 'up'	41
4.7	Visualisation d'une alerte depuis l'interface web de Prometheus	41
4.8	Notification de l'alerte depuis Telegram	42
4.9	Notification de résolution de l'alerte depuis Telegram	44
4.10	Installation de Loki	45
4.11	Status du service Loki	45
4.12	Liste des fichiers journaux collecter	46
4.13	Connexion a l'interface de Grafana	47
4.14	Interface de Grafana	48
4.15	Panneau de configuration de Prometheus	49
4.16	Dashboard de node-exporter	50
4.17	Dashboard de Grafana	51
4.18	Menu déroulant pour sélectionner un serveur	51
4.19	Exploration de logs sur grafana	52
4.20	Lignes des fichier journaux contenant des erreurs	52
4.21	Listes des alertes configurées	53
4.22	Alertes envoyées aux administrateurs via Telegram	54
4.23	Vérification du module ngx_http_stub_status_module	62

Liste des Tableaux

2.1	Liste des serveurs et des services hébergés	10
3.1	Comparaison des fonctionnalités entre Nagios, ELK Stack et Prometheus-Grafana-Loki	24
4.1	Fiche technique du serveur de Prometheus	32
4.2	Fiche technique du serveur de Loki	32
4.3	Fiche technique du serveur de Grafana	32
4.4	Évaluation financière de la solution	50

Table des matières

Dédicaces	i
Remerciements	ii
Résumé	iv
Abstract	vi
Liste des figures	vii
Liste des Tableaux	viii
Table des matières	ix
Glossaire	xi
Introduction générale	1
1 Présentation des cadres de formation et de stage	2
1.1 Introduction	3
1.2 Présentation du cadre de formation	3
1.2.1 Présentation du CIC	3
1.2.2 Présentation de l'EPL	4
1.2.3 Objectifs de L'EPL	5
1.3 Présentation de l'entreprise d'accueil	7
1.3.1 Historique de l'entreprise	7
1.3.2 Mission et activités principales	7
1.3.3 Réalisations majeures	7
1.3.4 Organigramme	8
1.3.5 Adresse	8
1.4 Conclusion	8
2 Étude préalable du projet	9
2.1 Introduction	10
2.2 Étude de l'existant	10
2.2.1 Analyse de l'existant	10
2.3 Problématique	11
2.4 Intérêt du sujet	12
2.4.1 Objectif général	12
2.4.2 Objectifs spécifiques	12
2.5 Résultats Attendus	12

2.6 Méthodologie	13
2.7 Conclusion	13
3 Généralités	14
3.1 Introduction	15
3.2 Infrastructures informatiques	15
3.2.1 Les composants d'une infrastructure informatique	16
3.2.2 Les types d'infrastructures informatiques	16
3.2.3 Les facteurs à prendre en compte lors de la conception d'une infrastructure informatique	17
3.2.4 Les avantages d'une infrastructure informatique bien conçue	17
3.2.5 Gestion de l'infrastructure	17
3.3 Monitoring des Infrastructures Informatiques	18
3.3.1 Importance du Monitoring	18
3.3.2 Fonctionnement	19
3.3.3 Objectifs du Monitoring des Infrastructures Informatiques	21
3.4 Outils de Monitoring des Infrastructures Informatiques	21
3.4.1 Nagios	22
3.4.2 ELK Stack	22
3.4.3 Grafana - Prometheus - Loki	23
3.4.4 Tableau comparatif des solutions	23
3.4.5 Justification du choix de la solution	24
3.5 Présentation des outils choisis	25
3.5.1 Prometheus	25
3.5.2 Loki	26
3.5.3 Grafana	28
3.6 Conclusion	29
4 Mise en œuvre de la solution	30
4.1 Introduction	31
4.2 Préambule	31
4.2.1 Spécification du travail	31
4.2.2 Choix matériel	32
4.2.3 Architecture de la solution	33
4.3 Mise en œuvre	33
4.3.1 Environnement de test	33
4.3.2 Prometheus	34
4.3.3 Loki	45
4.3.4 Grafana	47
4.4 Évaluation financière du système	50
4.5 Bilan	50
4.6 Conclusion	54
Conclusion générale	55
Annexe	56
Références bibliographiques et webographiques	68

Glossaire

- **Agrégation** : désigne le mécanisme de capture, de normalisation et de consolidation de logs provenant de différentes sources au sein d'une plateforme centralisée en vue de la corrélation et de l'analyse des données
- **Dacbot-Dev** : Environnement de développement pour Dacbot
- **Dacbot-RCI** : Bot pour les tâches spécifiques
- **Downtime** : taux d'indisponibilité d'une machine, d'un appareil ou d'un service est le pourcentage de temps pendant lequel il a été hors-service ou inactif sur une période donnée
- **Fichiers journaux** : fichiers générés par des logiciels et contenant des informations sur les opérations, les activités et les modèles
- **Flask** : micro framework open-source de développement web en Python
- **GitLab** : Outil de gestion de code source et de CI/CD
- **HTTP** : Hypertext Transfer Protocol, protocole de communication client-serveur développé pour le World Wide Web
- **ICMP** : Internet Control Message Protocol, permet de transporter des messages de contrôle et d'erreur pour qu'une machine émettrice sache qu'il y a eu un incident de réseau
- **Licence GPL** : licence qui fixe les conditions légales de distribution d'un logiciel libre du projet GNU.
- **Localhost** : adresse correspondant à l'ordinateur sur lequel la requête actuelle est exécutée
- **Nextcloud** : Plateforme de stockage et de partage de fichiers
- **Nginx** : logiciel libre de serveur Web ainsi qu'un proxy inverse
- **Proxmox Backup Server** : logiciel de sauvegarde open source prenant en charge les machines virtuelles, les conteneurs et les hôtes physiques
- **Proxmox VE** : plateforme de virtualisation libre basée sur l'hyperviseur Linux KVM, et offre aussi des conteneurs avec LXC
- **Python** : langage de programmation interprété, multiparadigme et multiplate-formes

- **QoS** : Quality of service, désigne toute technologie capable de gérer la transmission des données, tout en réduisant la perte de paquets et la latence sur le réseau.
- **Rev-Proxy149 et Rev-Proxy51** : Reverse proxy pour la gestion du trafic HTTP/HTTPS
- **S.M.A.R.T.** : Self-Monitoring, Analysis, and Reporting Technology, système de surveillance du disque dur d'un ordinateur. Il permet de faire un diagnostic selon plusieurs indicateurs de fiabilité dans le but d'anticiper les erreurs sur le disque dur.
- **Shorewall** : outil de pare-feu open source pour Linux
- **SLI (Service Level Indicator)** : Un indicateur de niveau de service qui mesure un aspect spécifique de la performance d'un service, comme le taux de disponibilité ou le temps de réponse. Les SLIs permettent de quantifier la qualité de service perçue par les utilisateurs, servant de base aux objectifs de service.
- **SLO (Service Level Objective)** : Un objectif de niveau de service qui fixe une valeur cible pour un SLI donné, exprimant le niveau de performance attendu pour le service sur une période donnée (par exemple, une disponibilité de 99,9% sur un mois). Les SLOs sont utilisés pour évaluer si un service répond aux attentes de qualité définies par l'entreprise ou les utilisateurs.
- **TCP** : Transmission Control Protocol, norme de communication qui permet aux programmes applicatifs et aux dispositifs informatiques d'échanger des messages sur un réseau
- **Telegram Messenger** : application et service de messagerie instantanée multi-plateforme en freemium hébergé dans un cloud
- **TSDB** : Time Series Database, systèmes logiciels optimisés pour trier et organiser des informations mesurées de manière temporelle
- **UDP** : User Datagram Protocol, protocole de communication utilisé sur Internet pour les transmissions particulièrement sensibles au facteur temps
- **Oracle VM VirtualBox** : logiciel libre de virtualisation, aujourd'hui publié par Oracle
- **Virtualisation** : La virtualisation consiste, en informatique, à exécuter sur une machine hôte, dans un environnement isolé, des systèmes d'exploitation
- **VM** : virtual machine, virtualisation ou émulation d'un appareil informatique créée par un logiciel d'émulation ou instanciée sur un hyperviseur
- **VPN** : virtual private network, système permettant de créer un lien direct entre des ordinateurs distants, connectés à des réseaux locaux différents
- **Wireguard51** : Serveur VPN pour connexions sécurisées

Introduction générale

Dans le contexte actuel de la digitalisation des services, les entreprises dépendent de plus en plus de leurs infrastructures informatiques pour assurer la continuité et la qualité de leurs opérations. La fiabilité des infrastructures qui supporte cette digitalisation est devenue une condition sine qua non pour maintenir une compétitivité accrue et répondre aux exigences croissantes des clients. Pour garantir cette fiabilité, il est impératif d'avoir une vue d'ensemble précise et continue des systèmes informatiques, permettant des interventions rapides en cas de besoin. C'est dans cette optique que le monitoring des infrastructures informatiques joue un rôle crucial. Il offre une visibilité sur l'état des systèmes, permettant de détecter et de résoudre proactivement les problèmes avant qu'ils n'affectent les services[1].

Dac Technologies, une entreprise spécialisée dans le développement d'applications Web, est confrontée à cette nécessité. L'entreprise doit surveiller efficacement ses serveurs et applications pour assurer la disponibilité et la performance de ses services. Les outils de monitoring intégrés aux systèmes, comme ‘ps’, ‘top’, ou ‘free’,... et les scripts bien qu'utilent, se révèlent insuffisants pour répondre aux besoins spécifiques de Dac Technologies, notamment en termes de collecte historique des données, de visualisation en temps réel, et d'alertes automatiques en cas de dysfonctionnement.

Face à cette problématique, l'objectif principal de ce mémoire est de concevoir et de mettre en œuvre un système de surveillance avancé en utilisant des outils modernes tels que Grafana, Prometheus, et Loki. Ces outils offrent des capacités étendues de collecte, d'analyse et de visualisation des données (consommation des ressources, disponibilité applicative, les journaux des processus et services, ...) . permettant ainsi d'optimiser la gestion des infrastructures informatiques.

Ce mémoire se structure en quatre (4) chapitres. Tout d'abord, une présentation des cadres de formation et de stage permettra de situer le contexte académique et professionnel dans lequel ce travail a été réalisé. Ensuite, une étude préalable du projet abordera l'existant, ses limites, et définira la problématique ainsi que les objectifs du projet. La section suivante détaillera les généralités sur les infrastructures informatiques et l'importance du monitoring. Enfin, l'étude et la mise en place de la solution choisie seront exposées, suivies d'une discussion sur les perspectives futures et d'une conclusion récapitulative.

Ce travail vise non seulement à améliorer la surveillance des systèmes de Dac Technologies, mais aussi à contribuer à la réflexion globale sur les meilleures pratiques en matière de monitoring des infrastructures informatiques qui deviennent de plus en plus complexe.

Chapitre 1

Présentation des cadres de formation et de stage

1.1 Introduction

Il sera présenté dans ce chapitre, le cadre de formation et ensuite l'entreprise d'accueil. Le premier volet concerne l'École Polytechnique de Lomé (EPL) qui est le cadre de formation et le second concerne Dac Technologies qui est le cadre de stage.

1.2 Présentation du cadre de formation

L'École Polytechnique de Lomé (EPL) est le fruit de la fusion entre le CIC (Centre Informatique et de Calcul) et l'ENSI (École Nationale Supérieure d'Ingénieurs). L'École Nationale Supérieure d'ingénieurs (ENSI) comme son nom l'indique était une très grande école réputée dans la formation des ingénieurs dans différents domaines à savoir : le génie civil, mécanique et électrique. Le Centre Informatique et de Calcul (CIC) quant à lui s'était donné la mission de former ces étudiants aux métiers de l'informatique constitué en deux filières principales à savoir : le Génie Logiciel (GL) et la Maintenance et Réseaux Informatique (MRI). Cette formation a commencée au CIC (deux années) avant de se terminé à l'EPL

1.2.1 Présentation du CIC



FIG. 1.1 : Une vue du bâtiment du CIC

Le Centre Informatique et Calcul (CIC), devenu aujourd'hui La Direction des Ressources et Supports Informatiques (DRSI) était un établissement et un centre de ressources de l'Université de Lomé créé par arrêté N° 67/MENRS du 26 Septembre 1988. Il était situé dans l'enceinte de l'Université de Lomé (UL), non loin de la Faculté des Sciences (FDS) et de la Faculté des Sciences de la Santé (FSS). Sa mission était d'apporter un

appui logistique et technique en informatique aux établissements et à l'administration de l'Université de Lomé dans la mise en place de programmes informatiques, et d'assurer la diffusion de l'informatique. A ce titre, le CIC réalisait des applications informatiques au profit des institutions de la communauté universitaire et offrait un cadre pour la formation en informatique à l'UL. Le CIC dépendait de la présidence de l'UL et avait pour objectifs principaux de :

- Développer des applications informatiques ;
- Résoudre les problèmes technologiques particulièrement dans le domaine de la maintenance informatique à l'UL ;
- Former des étudiants compétents en informatique.

En ce qui concerne les offres de formations, on avait des licences et mastères professionnelles en Génie Logiciel (GL) et en Maintenance et Réseau Informatique (MRI).

1.2.2 Présentation de l'École Polytechnique de Lomé (EPL)



FIG. 1.2 : Une vue du bâtiment de l'EPL

L'École Polytechnique de Lomé (EPL) est le résultat de la fusion de l'École Nationale Supérieure d'Ingénieurs (ENSI) et de la branche enseignement du Centre Informatique et de Calcul (CIC). Crée par l'arrêté N° 090 bis/2021/MESR du 16 juillet 2021, l'EPL se positionne comme un pôle d'excellence dans le domaine des sciences de la technologie et du génie des technologies émergentes. Elle est située dans l'enceinte de l'Université de Lomé (UL), non loin de la Faculté Des Sciences (FDS) et de la Faculté des Sciences de la Santé (FSS). Son directeur actuel est le Professeur Kondo Hloindo ADJALLAH, nommé le 07 juin 2022.

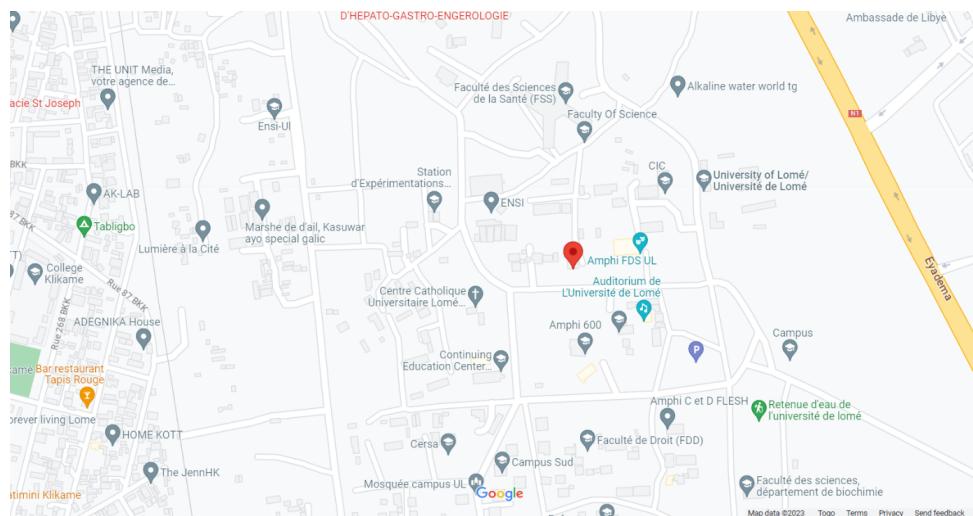


FIG. 1.3 : Plan de présentation de l'EPL - Google Maps [2]

Organisation administrative

Sur la figure 1.4 nous avons un aperçu de l'organisation administrative

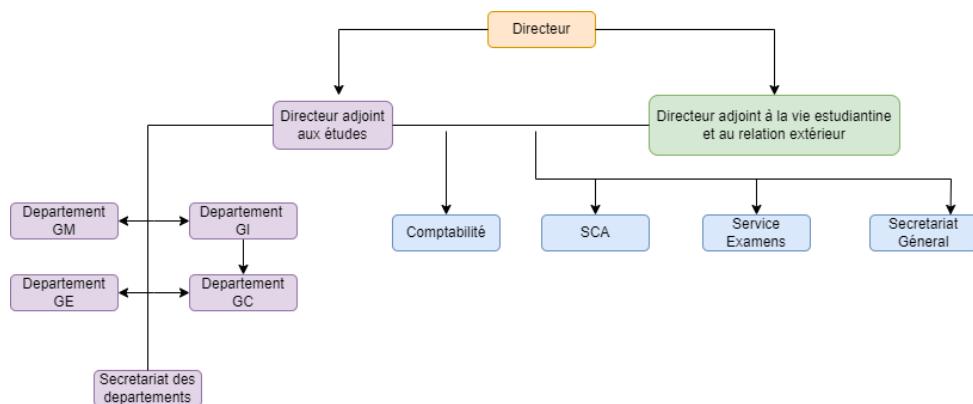


FIG. 1.4 : Organigramme administratif de l'EPL

1.2.3 Objectifs de L'École Polytechnique de Lomé (EPL)

L'École Polytechnique de Lomé (EPL) s'est fixé des objectifs clairs pour devenir un acteur majeur dans le domaine des sciences de la technologie et de l'ingénierie et de la recherche. Ses principaux objectifs comprennent [3] :

- **Formation intégrale** : Former des Techniciens Supérieurs, Ingénieurs et Docteurs, en offrant une variété de programmes académiques :
 - Licence Fondamentale en Sciences de l'Ingénieur :
 - Spécialités : Génie Civil, Génie Électrique, Génie Mécanique, Informatique et Systèmes, Intelligence Artificielle et Big Data, Logistique et Transport.
 - Licence Professionnelle en Sciences de l'Ingénieur :
 - Spécialités : Génie Civil, Génie Électrique, Génie Logiciel, Génie Mécanique, Systèmes et Réseaux Informatiques.

- Master Système et Informatique ;
- Master Intelligence Artificielle et Big Data.
- **Excellence Académique :** Devenir un pôle d'excellence académique et de recherche, en mettant l'accent sur les technologies émergentes.
- **Expertise et Conseil :** Offrir des services d'expertise et de conseil pour stimuler l'innovation industrielle.
- **Formation Continue :** Fournir des programmes de formation continue diplômants et certifiants pour les professionnels en activité.
- **Leadership dans les sciences de l'ingénierie :** Promouvoir l'excellence dans l'enseignement et la recherche en ingénierie et en technologie.
- **Contribuer au Développement :** Jouer un rôle clé dans le développement industriel du Togo et de la région.
- **Diplômes Reconnus :** Offrir des diplômes en co-diplômation, reconnus à la fois au Togo et en France.

En somme, l'École Polytechnique de Lomé (EPL) vise à former des techniciens, ingénieurs et docteurs en sciences de l'ingénieur y compris des doubles diplômes en partenariat avec l'Université de Technologie Belfort Montbéliard (UTBM). L'EPL en promouvant l'excellence académique et la recherche de pointe, elle offre des services d'expertise industrielle, propose des formations continues et contribue au développement industriel du Togo.

1.3 Présentation de l'entreprise d'accueil

1.3.1 Historique de l'entreprise

La société Dac Technologies est une entreprise togolaise créée en janvier 2020. Elle se spécialise dans le développement, l'administration et la conception des nouvelles technologies. Son objectif principal est de faciliter l'utilisation quotidienne de l'informatique en proposant des solutions sur mesure adaptées aux besoins spécifiques de ses clients. Dac Technologies s'engage à rendre les outils informatiques plus accessibles et plus faciles à utiliser.

1.3.2 Mission et activités principales

Dac Technologies a pour mission de fournir des services et des prestations dans le domaine de l'ingénierie informatique et des services associés. Ses activités principales incluent la conception et le déploiement d'applications publiques qui visent à réduire les tâches manuelles et à simplifier la vie quotidienne des utilisateurs.

Principales activités de Dac Technologies :

Développement de logiciels : Conception et développement d'applications personnalisées pour répondre aux besoins spécifiques de chaque client. Cela inclut l'analyse des besoins, la programmation, les tests et la mise en œuvre des solutions logicielles.

Administration de systèmes : Gestion complète des infrastructures informatiques des clients, y compris la maintenance des serveurs, la gestion des réseaux, la sécurité informatique et l'automatisation des processus de déploiement pour assurer une performance optimale et une disponibilité continue des systèmes.

Intégration de nouvelles technologies sur demande : Adaptation et intégration de technologies émergentes en fonction des demandes spécifiques des clients. Cela implique l'évaluation des nouvelles technologies, leur adaptation aux environnements existants, et la formation des utilisateurs pour garantir une adoption réussie et une amélioration des processus opérationnels.

1.3.3 Réalisations majeures

Dac Technologies a développé plusieurs solutions innovantes pour répondre aux besoins de ses clients. Parmi ses réalisations majeures, on compte :

- **DacPharma** : Un chatbot qui permet aux utilisateurs de consulter les pharmacies de garde, de localiser les pharmacies les plus proches, et de rechercher des informations spécifiques sur les pharmacies. DacPharma a référencé plus de 200 pharmacies et compte des centaines d'utilisateurs.
- **DacPharmaPro** : Une application destinée aux pharmaciens pour renseigner les informations nécessaires qui seront fournies aux utilisateurs. Cette application facilite également les échanges entre pharmaciens.

1.3.4 Organigramme

La figure 1.5 montre un aperçu de l'organisation administratif de Dac Technologies :

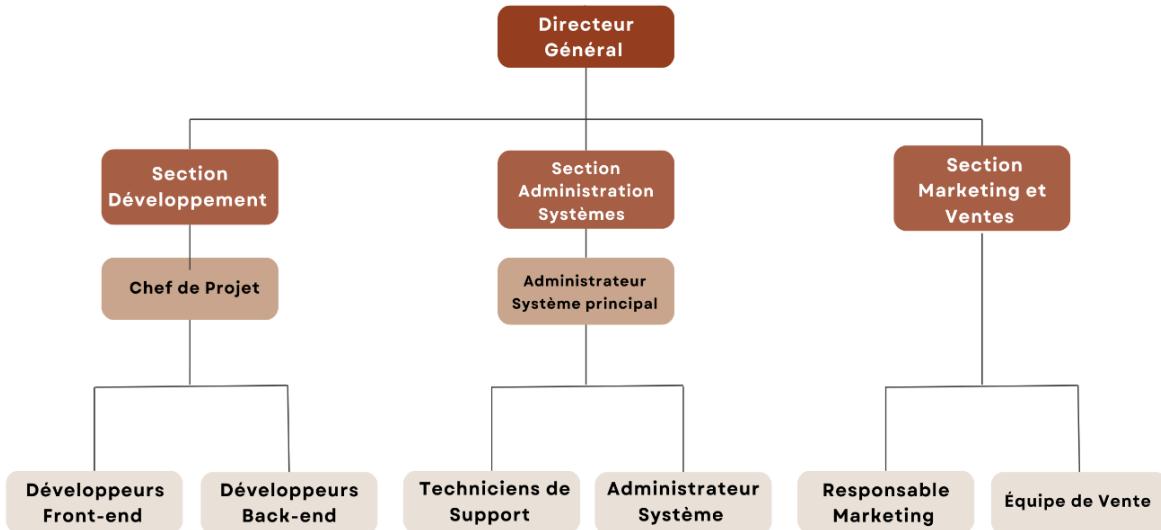


FIG. 1.5 : Organigramme de Dac Technologies

1.3.5 Adresse

DAC Technologies est basé au 14 Rue des callitris, Tokoin Solidarité, Lomé - Togo. Joignable au (+228) 22200299 ou au (+228) 92986644. Ou encore par email à l'adresse contact@dactechnologies.net ou visiter leur site web <https://www.dactechnologies.net>

1.4 Conclusion

Ce chapitre a mis en lumière les entités clés qui ont contribué à l'élaboration de ce document, notamment les cadres de formation et de stage. Le Centre Informatique et de Calcul de l'Université de Lomé (CIC - UL) devenu l'Ecole Polytechnique de Lomé a joué un rôle central dans l'acquisition des compétences théorique. Dac Technologies, en tant que structure hôte de notre projet, a été essentielle dans le développement pratique des connaissances acquises. Le prochain chapitre sera consacré à la présentation détaillée du thème de stage, de présenter sa problématique et enfin de comprendre l'intérêt d'apporter une solution à cette dernière.

Chapitre 2

Étude préalable du projet

2.1 Introduction

A mesure que les entreprises se digitalisent, leur infrastructure informatique devient de plus en plus complexe. le monitoring est devenu une priorité cruciale dans la gestion de l'infrastructure. Il est alors indispensable à Dac Technologies de disposer d'un système de monitoring adéquat. Dans ce chapitre, il sera question de faire une étude de l'existant de Dac Technologies, d'en ressortir les insuffisances, poser la problématique et ainsi dégager les objectifs de notre projet. Nous aurons également a spécifier les résultats attendus ainsi que la méthodologie adoptée dans le cadre de se projet.

2.2 Étude de l'existant

L'infrastructure informatique actuelle de Dac Technologies repose sur deux serveurs principaux, ‘cpufull’ et ‘diskfull’, chacun géré via Proxmox VE et sécurisé par le pare-feu Shorewall. Les serveurs hébergent plusieurs machines virtuelles (VMs) et offrent divers services essentiels à l'entreprise.

TAB. 2.1 : Liste des serveurs et des services hébergés

Serveur	Caractéristiques	Services hébergés
cpufull	CPU :32 RAM : 126 Go Espace disque : 8 To Hyperviseur : Proxmox VE Pare-feu : Shorewall	GitLab, Nextcloud, Rev-Proxy149, Proxmox Backup Server
diskfull	CPU : 8 RAM : 62,68 Go Espace disque : 6 To Hyperviseur : Proxmox VE Pare-feu : Shorewall	Dacbot-RCI, Dacbot-Dev, Rev-Proxy51, Wireguard51

2.2.1 Analyse de l'existant

Dans cette analyse, une évaluation de l'infrastructure actuelle a été réalisée pour ressortir ses points forts et ses limites.

2.2.1.1 Les mérites de l'existant

L'infrastructure actuelle de Dac Technologies offre une sécurité renforcée grâce au pare-feu Shorewall et au VPN Wireguard, protégeant efficacement contre les accès non autorisés. La virtualisation via Proxmox VE permet une gestion optimisée des ressources, isolant les services critiques sur des machines virtuelles dédiées. Enfin les outils natifs des systèmes d'exploitation permettent de faire un diagnostic en cas de dysfonctionnement de causes internes

2.2.1.2 Les limites de l'existant

Dans l'infrastructure actuelle de Dac Technologies, les outils natifs du système d'exploitation, et les fichiers journaux du système ainsi que les dashboards offert par l'interface de Proxmox VE jouent un rôle central dans la surveillance des serveurs et des applications. Cependant, ces outils présentent plusieurs limitations significatives :

- **Manque de centralisation et de corrélation des logs** : Les fichiers de logs sont généralement dispersés sur différents systèmes et services, ce qui complique leur centralisation et leur analyse. De plus, il est difficile de corrélérer les événements survenus dans différents logs pour diagnostiquer des problèmes complexes.
- **Absence de système d'alertes automatisées** : Les outils intégrés ne permettent pas de configurer des alertes en temps réel en fonction de seuils pré-définis. Les administrateurs doivent surveiller manuellement les systèmes ou créer des scripts personnalisés, ce qui n'est ni pratique ni fiable.
- **Visibilité limitée et absence de tableaux de bord** : Il n'existe pas de moyen intégré pour visualiser les données de performance sous forme de tableaux de bord dynamique. Cette absence de visualisation rend la compréhension de l'état global de l'infrastructure plus difficile et moins intuitive.
- **Scalabilité limitée** : Les outils actuels ne sont pas conçus pour évoluer facilement avec l'infrastructure. À mesure que Dac Technologies croît, la gestion et la surveillance des ressources deviennent de plus en plus complexes et nécessitent des solutions plus robustes et évolutives.

2.3 Problématique

Pour Dac Technologies, la disponibilité et les performances de ses applications sont essentielles. L'entreprise exprime le besoin d'une surveillance efficace de ses serveurs et applications pour répondre à plusieurs objectifs clés : obtenir un historique sur l'utilisation des ressources, connaître l'état actuel des applications, et recevoir des alertes en cas de dysfonctionnement.

Cependant, l'analyse des outils actuellement en place met en lumière plusieurs problèmes. Les outils embarqués, tels que les commandes ps, top ou free, offrent une visibilité limitée et ne fournissent ni historique détaillé de l'utilisation des ressources, ni système d'alerte en cas de défaillance. De plus, ces solutions présentent plusieurs lacunes importantes : un manque de centralisation et de corrélation des logs, l'absence de système d'alertes, une visibilité restreinte sur l'état global des systèmes, ainsi qu'une scalabilité limitée.

Face à ces défis, nous avons formulé les questions suivantes pour orienter notre étude :

- Comment centraliser et structurer les données de logs et de métriques pour obtenir une vision globale des infrastructures de Dac Technologies ?
- Comment mettre en place un système d'alerte réactif et automatisé afin de minimiser les temps de réponse en cas d'incidents ?
- Quel outil ou ensemble d'outils permettrait de répondre efficacement aux besoins spécifiques de l'entreprise en termes de visibilité, de centralisation et de scalabilité ?

Ce projet vise ainsi à étudier et implémenter une solution de monitoring adéquate pour renforcer la gestion des infrastructures de Dac Technologies et assurer une meilleure stabilité et performance de ses services.

2.4 Intérêt du sujet

L'intérêt de notre sujet se trouve dans les objectifs à atteindre.

2.4.1 Objectif général

Implémenter un système de surveillance des infrastructures informatiques de Dac Technologies afin d'optimiser la disponibilité, la performance et la sécurité des systèmes.

2.4.2 Objectifs spécifiques

- Identifier le périmètre de surveillance spécifiques de Dac Technologies.
- Configurer et déployer Grafana, Prometheus et Loki dans l'environnement informatique de Dac Technologies.
- Collecter et analyser les données pertinentes pour surveiller les infrastructures informatiques.
- Développer des tableaux de bord personnalisés pour visualiser l'état des ressources de l'infrastructure.
- Mettre en place des alertes et des notifications pour détecter les anomalies et les problèmes potentiels.

2.5 Résultats Attendus

À la fin de ce projet, nous visons à obtenir un système de surveillance des infrastructures informatiques fonctionnel, centralisé et optimisé, basé sur les outils Grafana, Prometheus, et Loki. Ce système devra fournir les livrables et fonctionnalités suivants :

- **Tableaux de bord et visualisation des métriques** : Un ensemble de tableaux de bord personnalisés dans Grafana, permettant une visualisation claire et en temps réel de l'état des ressources (CPU, RAM, stockage) et des performances historiques et actuelles des serveurs ainsi que des applications critiques de Dac Technologies.
- **Centralisation des logs et accès contrôlé** : Un accès centralisé aux journaux d'événements via Loki, disponible aux utilisateurs autorisés sans nécessiter un accès direct aux serveurs. Cette solution améliorera la sécurité et la traçabilité des actions utilisateur, avec un tableau listant les sessions actives et passées sur les machines et applications.
- **Système d'alerte et suivi des incidents** : Mise en place d'alertes automatiques pour signaler les services indisponibles ou en panne, et alerter proactivement en cas de saturation de ressources (mémoire, CPU, stockage).

Ces livrables contribueront à une gestion efficace des infrastructures de Dac Technologies, optimisant la surveillance et garantissant la disponibilité et la performance des systèmes critiques.

2.6 Méthodologie

Pour atteindre ces objectifs, nous suivrons les étapes suivantes :

- approfondir les besoins et les exigences en matière de surveillance de Dac Technologies ;
- rechercher et sélectionner les meilleures pratiques pour la mise en place de Grafana, Prometheus et Loki ;
- Déployer et configurer les outils nécessaires : Grafana, Prometheus, Loki, les agents de collecte de données ;
- tester et valider la solution dans un environnement de test avant le déploiement en production.

2.7 Conclusion

A travers l'analyse de l'existant, il est clair que la mise en place d'un système de monitoring est nécessaire pour que l'infrastructure de Dac Technologies soit aligné avec les objectifs de ce dernier. Maintenant que le besoin est réel, nous allons, dans le chapitre suivant, étudier ce qu'est le monitoring des infrastructure informatique.

Chapitre 3

Généralités sur le monitoring des infrastructures informatiques

3.1 Introduction

Le monitoring des infrastructures informatiques est essentiel pour assurer la disponibilité, la performance et la sécurité des systèmes informatiques. Ce processus implique l'utilisation de divers outils et techniques pour surveiller en temps réel les composants d'une infrastructure, y compris les serveurs, les bases de données, les réseaux et les applications. Dans ce chapitre, nous découvrirons ce qu'on entend par infrastructure informatique, pourquoi il faut la surveiller, ce qu'il faut surveiller et comment.

3.2 Infrastructures informatiques

L'infrastructure informatique est l'épine dorsale des entreprises modernes. Elle englobe le matériel, les logiciels et les réseaux qui permettent aux employés d'accéder aux applications et aux données dont ils ont besoin pour travailler. Il peut se définir comme étant l'ensemble des équipements matériels (postes de travail, serveurs, routeurs, firewall...) et des logiciels (serveurs web, ERP, applications métier,...) d'une entreprise. C'est un agencement fonctionnelle entre la puissance de calcul, les différentes applications, le service de stockage et le réseau d'entreprise [4]. Une infrastructure informatique bien conçue est essentielle pour la productivité, la collaboration et l'innovation. L'infrastructure a donc pour rôle de fournir aux collaborateurs d'une entreprise un accès , continu et sécurisé aux applications et aux données de l'entreprise. Pour cette raison, les équipements qui le composent supposent une installation et une maintenance correcte pour que l'entreprise puisse les exploiter au mieux.



FIG. 3.1 : Illustration simplifiée d'une infrastructure informatique [5]

3.2.1 Les composants d'une infrastructure informatique

On distingue trois (3) catégories de composants dans une infrastructure [6] :

- **matériels** : Les serveurs, les ordinateurs personnels, les périphériques de stockage, les routeurs et les commutateurs sont quelques exemples de matériel informatique. Cette catégorie regroupe tout ce qui est tangible ;
- **logiciels** : Le système d'exploitation, les applications de bureau, les bases de données et les logiciels de sécurité sont quelques exemples de logiciels d'infrastructure informatique. Cette catégorie est celle de tout ce qui est immatériel ;
- **réseaux** : Les réseaux informatiques relient les périphériques matériels et permettent aux utilisateurs de partager des données et des ressources.

En outre, l'infrastructure se caractérise par quatre (4) ressources essentielles qui font sa puissance :

- **le stockage** : Le stockage est utilisé pour garder les données de l'entreprise, telles que les fichiers, les bases de données et les applications. La quantité de stockage nécessaire dépend de la taille de l'entreprise et de la quantité de données qu'elle génère ;
- **le CPU** : Le CPU (Central Processing Unit) est le cerveau de l'ordinateur. Il traite les instructions et exécute les programmes. La vitesse et la puissance du CPU sont importantes pour garantir les performances de l'infrastructure informatique ;
- **la mémoire RAM** : La RAM (Random Access Memory) est la mémoire de travail de l'ordinateur. Elle stocke les données temporairement pendant que les programmes sont exécutés. La quantité de RAM nécessaire dépend du type de programmes utilisés et de la quantité de données qu'ils traitent ;
- **la bande passante** : La bande passante est la quantité de données qui peut être transmise sur un réseau en une seconde. Elle est importante pour garantir que les utilisateurs puissent accéder rapidement aux applications et aux données.

3.2.2 Les types d'infrastructures informatiques

Avec l'évolution du secteur informatique, de nombreux types d'infrastructures informatiques sont désormais disponibles, offrant diverses options d'utilisation et d'interaction [6].

Infrastructure classique : également appelé infrastructure sur site, l'infrastructure classique est hébergée dans les locaux de l'entreprise. Cette méthode n'utilise aucun service ou infrastructure d'autres entreprises. Bien que l'infrastructure classique présente certains avantages en matière de sécurité, elle est coûteuse à configurer et nécessite beaucoup d'entretien et d'espace physique.

Infrastructure cloud : dans ce cas, l'infrastructure est hébergée par un fournisseur de services cloud tiers. Le fournisseur de cloud achète et gère toute l'infrastructure informatique, et on y accède par la virtualisation. On peut utiliser toutes les ressources informatiques sous forme de services entièrement gérés. L'infrastructure cloud offre de la flexibilité, la capacité de mise à l'échelle et un haut degré de sécurité aux entreprises.

Infrastructure hybride : cette approche permet de disposer de ressources locales et de le renforcer par des services cloud.

3.2.3 Les facteurs à prendre en compte lors de la conception d'une infrastructure informatique

Pour garantir l'efficacité, les facteurs suivants doivent être pris en compte lors de la mise en place d'une infrastructure

- **disponibilité** : L'infrastructure informatique doit être disponible conformément aux politiques de l'entreprise ;
- **performances** : L'infrastructure informatique doit disposer de suffisamment de ressource pour répondre aux besoins des utilisateurs ;
- **sécurité** : L'infrastructure informatique doit être protégée contre les menaces externes et internes ;
- **accès** : Les utilisateurs doivent pouvoir accéder à l'infrastructure informatique depuis n'importe quel endroit conformément aux politiques de l'entreprise ;
- **évolutivité** : L'infrastructure informatique doit pouvoir évoluer pour répondre aux besoins croissants de l'entreprise.

3.2.4 Les avantages d'une infrastructure informatique bien conçue

En tenant compte des facteurs sus-cités, on garantit :

- **une productivité accrue** : Une infrastructure informatique bien conçue peut aider les employés à être plus productifs en leur fournissant un accès rapide et fiable aux applications et aux données dont ils ont besoin ;
- **une collaboration améliorée** : Une infrastructure informatique peut faciliter la collaboration entre les employés en leur permettant de partager facilement des fichiers et de communiquer entre eux ;
- **une innovation accrue** : Une infrastructure informatique peut favoriser l'innovation en fournissant aux employés les outils dont ils ont besoin pour développer de nouvelles idées et les tester ;
- **une sécurité renforcée** : Une infrastructure informatique bien conçue peut aider à protéger les données sensibles contre les menaces de sécurité ;
- **une réduction des coûts** : Une infrastructure informatique peut aider à réduire les coûts en optimisant l'utilisation des ressources et en réduisant les besoins en maintenance.

3.2.5 Gestion de l'infrastructure

Pour garantir le bon fonctionnement de l'infrastructure, l'administrateur prend des dispositions pour minimiser les incidents. Parmi ces dispositions, on peut citer :

Les sauvegardes régulières : c'est l'opération qui consiste à dupliquer et à mettre en sécurité les données contenues dans un système informatique. Cette opération doit être programmée et réalisée régulièrement afin de pouvoir restaurer le système informatique dans un état de fonctionnement à la suite d'un incident.

Les mises à jour : cette opération consiste à installer la dernière version d'une application ou d'un système d'exploitation. Une mise à jour sert à améliorer le rendement, l'efficacité ou la prestation d'un service ou d'un produit, et parfois à corriger les anomalies d'un programme donné

La maintenance : elle consiste à assurer le bon fonctionnement d'un système d'information et d'un parc informatique d'une entreprise.

Le plan de reprise d'activité : abrégé en **PRA**, c'est un ensemble de procédures (techniques, organisationnelles, sécurité) qui permettent à une entreprise de prévoir par anticipation, les mécanismes pour reconstruire et remettre en route un système d'information en cas de sinistre important ou d'incident critique.

Le monitoring : encore appelé le monitorage, c'est une activité de surveillance et de mesure d'une activité informatique. On l'emploie pour avoir une vue globale sur l'infrastructure

L'infrastructure informatique est un élément essentiel de toute entreprise. Elle doit être conçue et gérée avec soin pour répondre aux besoins de l'entreprise et garantir des performances optimales. Le monitoring des ressources est une partie importante de la gestion de l'infrastructure informatique.

3.3 Monitoring des Infrastructures Informatiques

Le monitoring informatique, également connu sous le nom de supervision informatique, est le processus de collecte et d'analyse des données sur la performance et la santé d'une infrastructure informatique complexe et hybride. Cela inclut les services cloud, les serveurs sur site, les réseaux, les conteneurs orchestrés, les applications et les périphériques de stockage. Les données collectées peuvent être utilisées pour identifier les problèmes potentiels, résoudre les problèmes existants et optimiser les performances globales du système. [7]

3.3.1 Importance du Monitoring

En adoptant une approche proactive du monitoring, les entreprises peuvent récolter de nombreux avantages, dont les suivants [8] :

- **Prévention des pannes et downtime :** En surveillant les composants critiques tels que les serveurs, les réseaux et les bases de données, le monitoring permet d'identifier les signes précurseurs de défaillance et de prendre des mesures correctives avant qu'une panne ne se produise, réduisant ainsi les interruptions de service.

- **Optimisation des Performances** : En analysant les métriques de performance telles que l'utilisation du CPU, la consommation de mémoire et le trafic réseau, l'administrateur peut identifier les goulots d'étranglement et les inefficacités, ce qui permet d'optimiser les performances des systèmes.
- **Réactivité et Temps de Résolution Réduit** : Grâce à la surveillance en temps réel et à l'alerte proactive, l'administrateur peut intervenir rapidement pour résoudre les problèmes, minimisant ainsi l'impact sur les utilisateurs finaux et améliorant la satisfaction client.
- **Planification des Capacités** : En recueillant des données sur l'utilisation des ressources et les tendances de charge, le monitoring permet aux organisations de planifier efficacement les besoins en capacité et de prendre des décisions éclairées concernant les investissements futurs en infrastructure.

3.3.2 Fonctionnement

Pour répondre au besoin, celui de fournir des informations permettant aux administrateurs d'agir, un système de monitoring doit pouvoir collecter, parser, stocker, restituer les données et alerter en cas de dysfonctionnement. Pour se faire, plusieurs composants entre en jeu a divers étape. [9]

1. La collecte des Données

La première étape dans le processus de monitoring est la collecte des données. Les données sont variées et de sources diverses. On peut collecter les données sur :

- l'état physique d'une machine : température, disques (S.M.A.R.T.) ;
- la charge d'une machine : nombre d'utilisateurs, de requêtes, le processeur, débit réseau ;
- la disponibilité applicative : présence de processus et leur réponse par exemple ;
- les fichiers journaux concernant une application ou un composant système ;
- les performances du réseau : débit, latence, taux d'erreur, QoS ;
- la nature des protocoles d'un réseau et leur taux relatif : UDP, TCP, ICMP...

Ces données peuvent être regroupées en trois (3) types :

- **booléen** : ces données représentent un état binaire, tel qu'un indicateur de présence ou d'absence d'un événement. Par exemple, le service Nginx est-il disponible ou non ;
- **quantitative** : ces données mesurent des quantités et permettent une analyse précise en termes de chiffres. Par exemple, l'utilisation du processeur (en pourcentage) ;
- **qualitative** : ces données décrivent des états ou des caractéristiques qui ne sont pas directement mesurables en chiffres, mais qui sont essentielles pour l'analyse qualitative du système. Par exemple les types de protocoles réseau utilisés (UDP, TCP).

Cette collecte peut se faire de différentes manières en fonction des sources de données et des types de métriques à surveiller :

- **Agents** : Ce sont des programmes légers installés sur les hôtes surveillés (serveurs, équipements réseaux, etc.). Les agents collectent des données sur l'utilisation du CPU, la mémoire, le disque, le réseau, et d'autres paramètres de performance pour ensuite les transmettre au système.
- **Exporters** : Ce sont des composants qui exposent les métriques de systèmes ou d'applications sous forme de points de terminaison HTTP. Ils sont souvent utilisés dans des environnements de micro services.
- **Protocoles de Surveillance** : SNMP (Simple Network Management Protocol), WMI (Windows Management Instrumentation), et d'autres protocoles peuvent être utilisés pour interroger les appareils et collecter des données sans nécessiter l'installation d'agents.

2. Transport des Données

Une fois collectées, les données doivent être transportées vers un système centralisé pour le stockage et l'analyse. Ce transport peut se faire de plusieurs manières :

- **Push** : Les agents envoient activement les données vers le serveur de monitoring ;
- **Pull** : Le serveur de monitoring interroge le point de terminaison des agents pour récupérer les données.

3. Stockage des Données

Le stockage des données de monitoring est crucial pour leur analyse et visualisation ultérieure. Les systèmes de monitoring utilisent différentes bases de données en fonction du type de données :

- **Bases de Données de Séries Temporelles (TSDB)** : Conçues pour stocker des séries temporelles de données, où chaque point de données est associé à un horodatage. Exemples : Prometheus, InfluxDB ;
- **Bases de Données Relationnelles** : Utilisées pour stocker des données de monitoring structurées dans des tables. Exemples : MySQL, PostgreSQL ;
- **Indexation et Recherche** : Pour les fichiers journaux, des moteurs de recherche comme Elasticsearch sont utilisés pour indexer et rechercher des événements de manière efficace.

4. Analyse et Traitement

Les données stockées doivent être analysées pour extraire des informations utiles et détecter des anomalies. Les techniques d'analyse peuvent inclure :

- **seuils statistiques** : Comparaison des métriques collectées avec des valeurs seuils prédéfinies pour détecter les dépassemens ;
- **algorithmes d'apprentissage automatique** : Utilisation de l'IA et du machine learning pour détecter des anomalies complexes et des tendances ;
- **corrélation d'évènements** : Analyse des relations entre différents événements pour identifier des causes probables de problèmes.

5. Alerting

Le système de monitoring doit être capable de générer des alertes en cas de détection de problèmes. Les alertes peuvent être configurées pour différents canaux :

- notifications par E-mail : Envoi d'e-mails aux administrateurs ou aux équipes responsables ;
- intégrations avec des Systèmes de Gestion d'Incidents : Envoi d'alertes vers des outils comme PagerDuty, Opsgenie ou ServiceNow ;
- messages sur les plateformes de collaboration : Envoi de notifications vers des outils de collaboration comme Slack ou Microsoft Teams ou encore Rocketchat.

6. Visualisation

La visualisation des données est essentielle pour comprendre les tendances et identifier les problèmes rapidement. Les outils de visualisation permettent de créer des tableaux de bord dynamiques et des graphiques :

- **tableaux de Bord** : Affichage des métriques clés et des logs en temps réel ou sur des périodes spécifiques ;
- **graphiques et diagrammes** : Utilisation de divers types de graphiques (lignes, barres, pie charts, etc.) pour représenter visuellement les données ;
- **exploration de données** : Fonctionnalités permettant de zoomer, filtrer et analyser en profondeur les données collectées.

3.3.3 Objectifs du Monitoring des Infrastructures Informatiques

Les objectifs principaux visés en mettant en place un système de monitoring des infrastructures informatiques sont :

- **surveillance en temps réel** : Assurer une surveillance proactive des composants critiques pour détecter les problèmes dès qu'ils surviennent ;
- **collecte et analyse de données** : Recueillir des données sur les performances, la disponibilité et l'utilisation des ressources, et les analyser pour obtenir des informations exploitables ;
- **notification et alertes** : Configurer des seuils d'alerte pour être informé immédiatement en cas de dépassement de limites prédéfinies ou de comportements anormaux ;
- **reporting et visualisation** : Générer des rapports réguliers et utiliser des tableaux de bord pour visualiser les tendances, les anomalies et les performances globales du système.

3.4 Outils de Monitoring des Infrastructures Informatiques

Vu l'importance que revêt le monitoring, beaucoup d'entreprise se sont spécialisées dans le développement d'application de monitoring. On retrouve sur le marché aussi bien des

applications propriétaire que des applications open-source. Nous allons étudier quelques unes de ses applications, présenter les avantages et inconvénients, faire une comparaison pour au final choisir la solution adéquate pour l'infrastructure de Dac Technologies.

3.4.1 Nagios

Nagios (anciennement appelé Netsaint) est une application permettant la surveillance système et réseau. Elle surveille les hôtes et services spécifiés, alertant lorsque les systèmes ont des dysfonctionnements et quand ils repassent en fonctionnement normal. C'est un logiciel libre sous licence GPL. [10]

- Avantages
 - solution open-source mature et largement utilisée ;
 - personnalisable et flexible ;
 - large communauté et documentation extensive ;
 - coût total de possession (TCO) inférieur aux solutions SaaS.
- Inconvénients
 - interface utilisateur moins intuitive que les solutions modernes ;
 - nécessite une expertise technique pour la configuration et la maintenance ;
 - peut être difficile à gérer pour les grandes infrastructures.

3.4.2 ELK Stack

Composée d'Elasticsearch, de Kibana, de Beats et de Logstash (parfois appelés la Suite ELK) et de bien d'autres, cette suite fiable et sécurisée permet de collecter n'importe quel format de données depuis n'importe quelle source, puis d'interroger, d'analyser et de visualiser les données. [11]

- Avantages
 - solution open-source puissante pour l'analyse des logs et la visualisation des données ;
 - offre une grande flexibilité et une capacité de personnalisation ;
 - permet d'analyser de gros volumes de données ;
- Inconvénients
 - nécessite une expertise technique pour la configuration et la maintenance ;
 - peut être complexe à mettre en œuvre et à gérer ;
 - moins adapté au monitoring d'infrastructure en temps réel.

3.4.3 Grafana - Prometheus - Loki

Grafana, Prometheus, et Loki sont trois outils open-source largement utilisés pour la surveillance, la gestion des logs, et la visualisation des données dans les infrastructures informatiques modernes. Chacun de ces outils joue un rôle spécifique dans un écosystème de monitoring, souvent déployé ensemble pour fournir une solution complète de gestion de la performance et des événements.

- Avantages

- solution gratuite, open-source et légère a déployer ;
- basée sur des métriques et adaptée au monitoring d'infrastructure en temps réel ;
- scalable : Prometheus est hautement évolutif et peut être utilisé pour surveiller de grands systèmes distribués ;
- flexible : Prometheus est un système flexible qui peut être personnalisé pour répondre aux besoins spécifiques de chaque utilisateur ;
- puissant : Grafana offre un large éventail de fonctionnalités pour la visualisation de données, l'analyse de données et la configuration d'alertes.

- Inconvénients

- nécessite une expertise technique pour la configuration et la maintenance ;
- moins mature que d'autres solutions et communauté moins importante ;
- fonctionnalités d'analyse des logs moins avancées que l'ELK Stack.

3.4.4 Tableau comparatif des solutions

Le tableau 3.1 présente une étude comparatif des fonctionnalités de divers solution de monitoring disponible sur le marché.

TAB. 3.1 : Comparaison des fonctionnalités entre Nagios, ELK Stack et Prometheus-Grafana-Loki

Fonctionnalités	Nagios	ELK Stack	Prometheus-Grafana-Loki
Type de Monitoring	serveurs, services, réseaux	Monitoring basé sur les logs	Monitoring des métriques et des logs
Mise à l'Échelle	Scalable, mais nécessite plus de configuration	Scalable via Elasticsearch	Scalable (Prometheus + Grafana supportent le clustering)
Type de Données Gérés	État des services, temps de réponse, disponibilité	Logs, événements	Séries temporelles (métriques), logs (avec Loki)
Visualisation	Interface Web basique	Kibana (visualisations riches pour les logs)	Grafana (visualisations avancées des métriques et logs)
Alerting	Basé sur les états des services	Alertes basées sur les logs	Alertmanager
Gestion des Logs	Support limité (intégration possible avec syslog)	Excellent pour l'analyse et l'agrégation des logs	Loki (solution dédiée pour la gestion des logs)
Facilité de Configuration	Configuration manuelle des fichiers, peut être complexe	Nécessite des configurations pour ingestion, analyse et stockage	Relativement simple, basé sur des fichiers de configuration YAML
Extensibilité	Très extensible via plugins	Extensible via des plugins et intégrations Kibana	Extensible via des dashboards et des plugins Grafana
Intégrations Tierces	Large gamme de plugins, compatible avec de nombreux outils	Intégrations avec différents systèmes de logs et data sources	Intégrations riches avec de nombreux systèmes de métriques et logs
Écosystème DevOps	Intégration partielle (CI/CD, Infrastructure as Code possible)	Très utilisé pour les pipelines CI/CD via les logs	Très utilisé dans les pipelines DevOps pour le monitoring
Gestion de la Redondance et de la Haute Disponibilité	Configuration de redondance possible, mais complexe	Prise en charge de la redondance avec Elasticsearch clusters	Prise en charge native avec Thanos (pour Prometheus) et Grafana

3.4.5 Justification du choix de la solution

Malgré le fait que Dac Technologies ait manifesté le désir que soit utilisé Grafana, Prometheus et Loki pour la mise en place de son système de monitoring, un détour a été fait pour explorer d'autres solutions. Parmi ces options, le trio Grafana, Prometheus, et

Loki s'est démarqué par sa légèreté, sa flexibilité, et son adaptabilité aux besoins de Dac Technologies. Cette solution offre une surveillance en temps réel, une visualisation puissante des métriques, et une gestion centralisée des logs, répondant ainsi aux principales exigences identifiées dans l'infrastructure de l'entreprise.

3.5 Présentation des outils choisis

Examinons maintenant plus en détail les outils sélectionnés pour notre système.

3.5.1 Prometheus

Prometheus est un système de surveillance open-source avec un modèle de données multidimensionnel, un langage de requête flexible (PromQL), une base de données de séries temporelles efficace et une approche d'alerte moderne. Il est conçu pour collecter et surveiller les métriques des systèmes et applications distribués. Prometheus est bien adapté à la surveillance des microservices et des architectures orientées services hautement dynamiques. Il récupère des métriques à partir des cibles configurées à des intervalles donnés, évalue l'expression des règles, affiche le résultat et peut afficher des alertes si une condition est remplie. [12]

3.5.1.1 Fonctionnalités

Les fonctionnalités les plus remarquables de Prometheus par rapport à d'autres systèmes de surveillance sont :

- les cibles sont découvertes via la découverte de services ou la configuration statique ;
- collecte de données en séries temporelles : Prometheus collecte les métriques à intervalle régulier à partir des cibles exposant des points de terminaison HTTP ;
- modèle de données multidimensionnel : Prometheus utilise un modèle de données multidimensionnel pour stocker les métriques par le nom de la métrique et l'ensemble de dimensions clé/valeur. Cela permet des requêtes et des analyses puissantes ;
- base de données de séries temporelles efficace : Prometheus utilise une base de données de séries temporelles efficace pour stocker les métriques. Cela permet de stocker et de requêter de grandes quantités de données de séries temporelles ;
- langage de requête flexible : Prometheus dispose d'un langage de requête flexible appelé PromQL qui permet d'exploiter cette multidimensionnalité des données ;
- approche d'alerte moderne : Prometheus propose une approche d'alerte moderne basée sur des règles. Cela permet de définir des règles pour générer des alertes lorsqu'il y a des problèmes avec les systèmes ou les applications.

3.5.1.2 Composants

L'écosystème Prometheus se compose de plusieurs composants, dont les plus essentiels sont :

- le serveur Prometheus qui récupère et stocke les données ;
- les bibliothèques clientes pour l'instrumentation du code d'application ;
- les exporters à usage spécifique pour des services tels que nginx, mysqlserver, les systèmes linux... ;
- Alertmanager pour gérer les alertes.

3.5.1.3 Architecture

Le diagramme 3.2 illustre l'architecture de Prometheus et certains de ses composants d'écosystème :

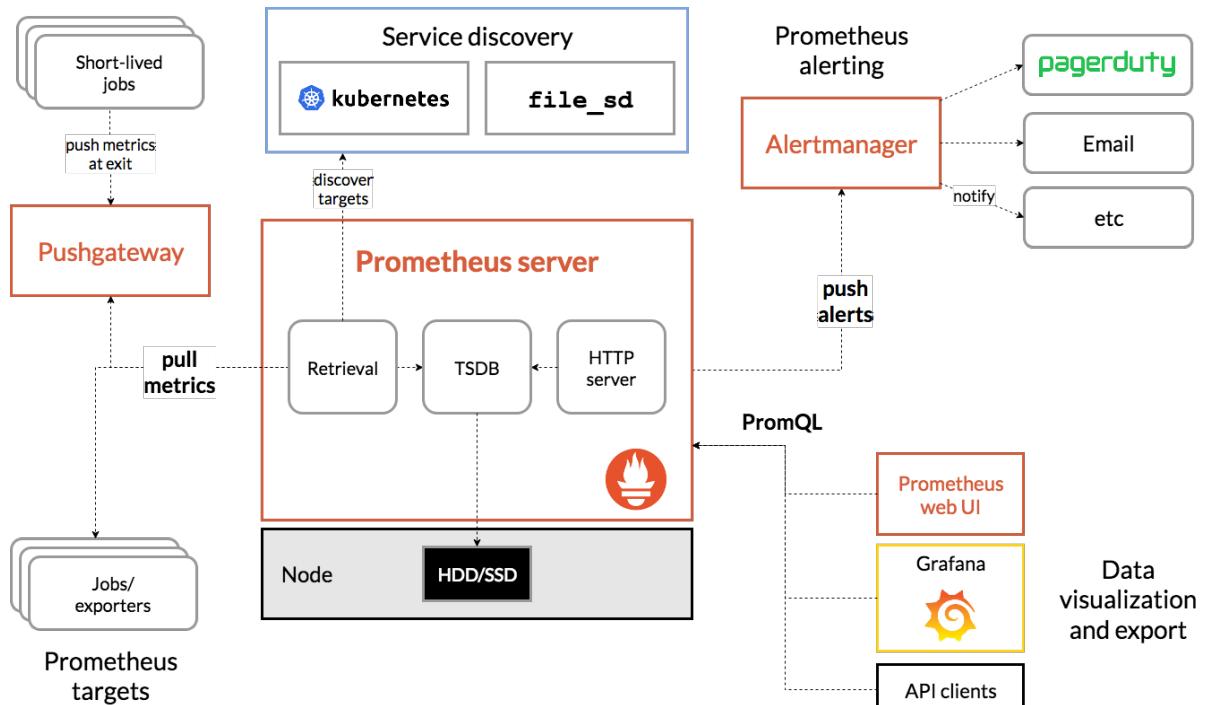


FIG. 3.2 : Architecture de Prometheus [13]

Prometheus collecte les métriques des services configures, soit directement, soit par l'intermédiaire d'une passerelle push. Il stocke localement toutes les données recueillies et applique des règles sur ces données pour créer et enregistrer de nouvelles le TSDB à partir des données existantes ou pour déclencher des alertes. Les données collectées peuvent ensuite être visualisées à l'aide de Grafana ou d'autres consommateurs d'API.

3.5.2 Loki

Loki est un système d'agrégation de fichiers journaux multi-locataire conçu pour collecter, stocker et gérer de grands volumes de données. Il est conçu pour être évolutif, hautement

disponible et rentable. Loki s'intègre aux autres produits Grafana, tels que Grafana et Tempo, pour une solution d'observabilité complète. [14]

3.5.2.1 Fonctionnalités

Voici quelques-unes des principales fonctionnalités de Loki :

- agrégation de logs horizontale : Loki peut gérer de grands volumes de données de logs en agrégeant les logs de plusieurs sources ;
- stockage de logs compressé : Loki stocke les logs dans des segments compressés pour réduire l'espace de stockage ;
- indexation des logs : Loki crée un index des logs pour les rendre facilement consultables ;
- multi-location : Loki prend en charge le multi-location, ce qui signifie que plusieurs utilisateurs peuvent partager une seule instance Loki ;
- intégration tierce : Loki s'intègre à des agents et outils tiers, tels que Promtail et Grafana ;
- déploiement flexible : Loki peut être déployé sur site ou dans le cloud.

3.5.2.2 Composants

Loki est un système modulaire qui contient de nombreux composants dont les plus importants sont :

- Distributor : Le service de distribution est responsable de la gestion des demandes push entrantes des clients. Il s'agit de la première étape du chemin d'écriture des données de journal ;
- Ingester : Le service d'ingestion est responsable de la conservation des données et de leur envoi vers un stockage à long terme ;
- Querier : Le service de requête est responsable de l'exécution des requêtes LogQL (Log Query Language) ;
- Ruler : Le service Ruler gère et évalue les expressions de règle et/ou d'alerte fournies dans une configuration de règle.

3.5.2.3 Architecture

Le schéma 3.3 représente l'architecture interne de Loki :

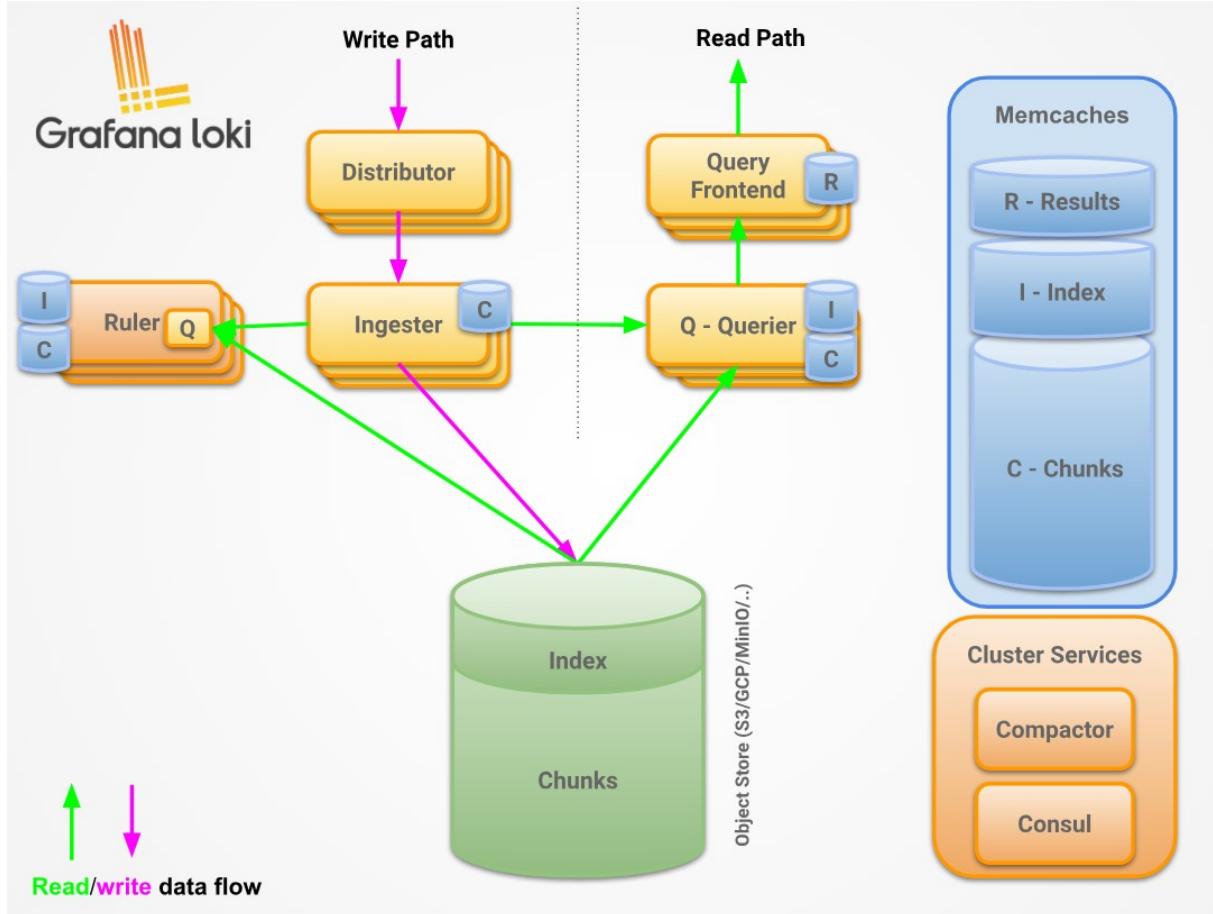


FIG. 3.3 : Architecture interne de Loki[15]

Le chemin d'écriture dans Loki commence par la réception des journaux par le distributeur, qui les achemine vers les ingesteurs en fonction d'un anneau de hachage cohérent. Les ingesteurs créent ou ajoutent des blocs pour stocker les journaux, puis accusent réception. Le distributeur renvoie un succès si un quorum d'ingesteurs confirme l'écriture. Le chemin de lecture, quant à lui, divise les requêtes en sous-requêtes traitées par des ingesteurs et des magasins de sauvegarde. Les résultats sont ensuite dédupliqués, fusionnés, et renvoyés au client par le frontend de requête.

3.5.3 Grafana

Grafana est un logiciel open-source de visualisation de données et de gestion d'alertes. Il est conçu pour collecter, agréger et afficher des données de séries temporelles provenant de diverses sources, telles que Prometheus, InfluxDB et Elasticsearch. Grafana offre une variété de fonctionnalités pour la création de tableaux de bord personnalisés, l'analyse de données et la configuration d'alertes. [16]

3.5.3.1 Fonctionnalités

Voici quelques-unes des principales fonctionnalités de Grafana :

- visualisation de données : Grafana offre une large gamme de types de graphiques et de tableaux pour visualiser des données de séries temporelles ;
- requêtage de données : Grafana utilise un langage de requête puissant appelé PromQL pour interroger et agréger des données de séries temporelles ;
- alertes : Grafana peut être configuré pour envoyer des alertes par e-mail, SMS ou d'autres canaux lorsqu'il y a des problèmes avec vos systèmes ou applications ;
- dashboards : Grafana permet de créer des tableaux de bord personnalisés pour visualiser et analyser des données ;
- plugins : Grafana prend en charge un large éventail de plugins pour étendre ses fonctionnalités ;
- intégration : Grafana s'intègre à d'autres outils de surveillance et d'observabilité, tels que Prometheus, Alertmanager et Tempo.

3.6 Conclusion

L'état de l'art a permis de poser les bases théoriques et pratiques nécessaires pour comprendre l'importance du monitoring des infrastructures informatiques dans un environnement d'entreprise moderne. Les concepts fondamentaux liés aux infrastructures, aux outils de monitoring, ainsi que leurs avantages ont été explorés en détail. Plusieurs solutions telles que, Nagios, ELK Stack et la combinaison Grafana-Prometheus-Loki, ont été analysées en termes de capacités, de flexibilité, et de facilité d'intégration. Fort de cette analyse, il est désormais temps de passer à l'étape cruciale de la mise en œuvre. Dans la prochaine partie, nous détaillerons les étapes pratiques de l'installation, de la configuration et de l'intégration de Prometheus-Loki-Grafana en nous appuyant sur les concepts abordés dans ce travail pour construire une solution de monitoring performante et évolutive.

Chapitre 4

Mise en œuvre de la solution

4.1 Introduction

Dans ce chapitre, nous allons découvrir plus en profondeur chacun des outils qui constituent notre système de monitoring : Prometheus, Loki, et Grafana. Nous allons également explorer leur mise en place pour répondre aux besoins spécifiques de Dac Technologies, en particulier pour l'amélioration de la surveillance des applications et la centralisation des logs.

4.2 Préambule

Avant de détailler les spécificités de chaque outil et leur intégration, il est essentiel de définir les exigences qui orienteront l'architecture de notre solution de monitoring. Ce préambule présente les critères et contraintes définis pour garantir que la solution soit en adéquation avec les attentes de Dac Technologies.

4.2.1 Spécification du travail

Les spécifications du travail sont établies pour assurer que le système de monitoring proposé réponde aux objectifs de performance, de sécurité et de fiabilité. Ces spécifications guideront le choix des outils et les configurations à adopter pour l'infrastructure de surveillance.

4.2.1.1 Contraintes

Pour garantir la performance optimale des applications, minimiser les coûts et assurer la fiabilité de la solution, les contraintes suivantes ont été imposées :

- l'agrégation des logs ne doit en aucun cas perturber le fonctionnement des systèmes ;
- la collecte des logs doit être exhaustive, avec un taux de perte minimal ;
- l'utilisation de l'espace de stockage doit être optimisée pour éviter toute surcharge ;
- la sécurité de la solution est primordiale, garantissant la confidentialité et l'intégrité des données ;
- un accès multi-utilisateurs est requis, mais chaque utilisateur ne doit avoir que les permissions nécessaires pour accomplir ses tâches spécifiques.

Les contraintes définies ci-dessus orienteront les décisions architecturales et technologiques prises dans le développement de ce système de monitoring."

4.2.1.2 Périmètre de travail

Le périmètre de travail couvre l'entièreté de l'infrastructure de Dac Technologies étudiée précédemment [2.2](#). Dans ce contexte, le système de monitoring devra surveiller plusieurs aspects des machines virtuelles, notamment :

- l'espace disque disponible,
- les processus actifs et l'utilisation des ressources,
- l'utilisation de la RAM,
- les fichier journaux de tous les processus, garantissant une traçabilité complète des événements au sein des machines virtuelles.

Ce périmètre de travail englobe donc la surveillance complète des serveurs virtuels et des applications métiers critiques hébergées sur l'infrastructure cloud de Dac Technologies.

4.2.2 Choix matériel

Pour des raison de scalabilité, nous avons choisi de déployer chaque solution sur des serveurs différents. Voici un aperçu des caractéristiques clés de chaque serveur :

- Serveur Prometheus

TAB. 4.1 : Fiche technique du serveur de Prometheus

Caractéristiques	Valeurs
Nbre CPU et fréquence	4 CPUs, 2.70 GHz
Mémoire RAM	2 Go
Stockage	32 Go
Système d'exploitation	Debian 12

- Serveur Loki

TAB. 4.2 : Fiche technique du serveur de Loki

Caractéristiques	Valeurs
Nbre CPU et fréquence	4CPUs, 2.70GHz
Mémoire RAM	2 Go
Stockage	92 Go
Système d'exploitation	Debian 12

- Serveur Grafana

TAB. 4.3 : Fiche technique du serveur de Grafana

Caractéristiques	Valeurs
Nbre CPU et fréquence	2 CPUs, 2.70GHz
Mémoire RAM	1.5 Go
Stockage	32 Go
Système d'exploitation	Debian 12

4.2.3 Architecture de la solution

Le schéma 4.1 montre l'architecture de notre solution finale

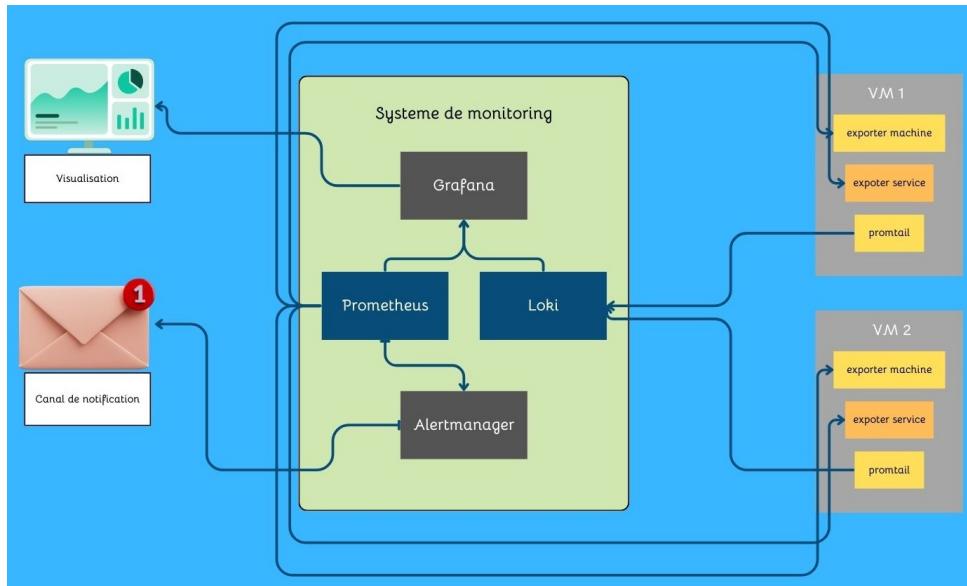


FIG. 4.1 : Architecture simplifiée de la solution

Ce système de monitoring fonctionne de manière collaborative pour assurer une surveillance continue de l'infrastructure informatique. Les exporteurs déployés sur les machines collectent des métriques et des logs, qui sont ensuite envoyés respectivement à Prometheus et Loki. Prometheus stocke les métriques et déclenche des alertes en cas d'anomalies, tandis que Loki conserve les logs pour une analyse plus approfondie. Alertmanager reçoit ces alertes et les transmet aux canaux de notification appropriés. Enfin, Grafana offre une interface visuelle intuitive pour créer des tableaux de bord personnalisés, permettant ainsi aux administrateurs de visualiser en temps réel l'état de leur système et d'identifier rapidement les problèmes potentiels.

L'intérêt principal de ce système réside dans sa flexibilité, sa scalabilité et sa capacité à personnaliser les visualisations. Il est particulièrement adapté pour les environnements complexes et dynamiques.

4.3 Mise en œuvre

La mise en œuvre présentée dans ce document est faite dans un environnement de test avec des VMs virtualisés avec Oracle VM VirtualBox

4.3.1 Environnement de test

Notre environnement de test est composé de trois (3) VMs, avec les caractéristiques suivantes :

- monitoring** : Cette VM est dédiée à la surveillance des applications et des systèmes. Elle exécute Prometheus pour la collecte des métriques, Loki pour l'agrégation des fichiers journaux ainsi que Grafana pour la visualisation. Elle est équipée de 2 CPU, 4 Go de RAM et 20 Go de stockage.

2. **server 1** : Ce serveur est utilisé pour une application web. Il fonctionne avec un serveur web Nginx et une application PHP. Les ressources allouées à cette VM incluent 1 CPU, 2 Go de RAM et 25 Go de stockage.
3. **server 2** : Ce serveur est dédié un système de visioconférence. Il permet aux utilisateurs de l'application web de communiquer. Cette VM est équipée de 4 CPU, 8 Go de RAM et 60 Go de stockage.

4.3.2 Prometheus

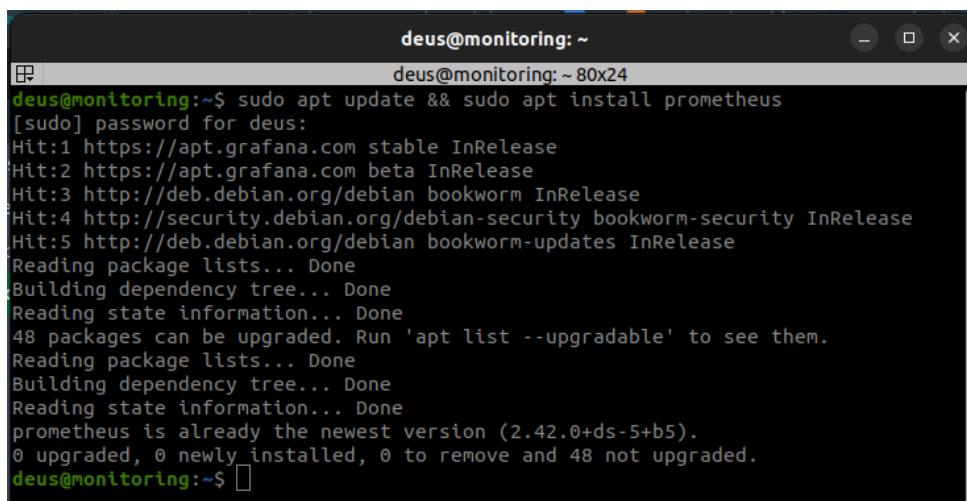
Procédons maintenant au déploiement de Prometheus et de ses agents pour la collecte des métriques.

4.3.2.1 Installation

Prometheus peut être installé de diverse manière : via docker, via package linux, via le code source... Dans notre cas, nous avons choisi de faire l'installation par paquet linux. Pour se faire nous procéderons d'abord une mise à jour du système puis l'installation du paquet Prometheus :

```
sudo apt update && sudo apt install prometheus.
```

La figure 4.2 montre l'installation de Prometheus



```
deus@monitoring:~$ sudo apt update && sudo apt install prometheus
[sudo] password for deus:
Hit:1 https://apt.grafana.com stable InRelease
Hit:2 https://apt.grafana.com beta InRelease
Hit:3 http://deb.debian.org/debian bookworm InRelease
Hit:4 http://security.debian.org/debian-security bookworm-security InRelease
Hit:5 http://deb.debian.org/debian bookworm-updates InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
48 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
prometheus is already the newest version (2.42.0+ds-5+b5).
0 upgraded, 0 newly installed, 0 to remove and 48 not upgraded.
deus@monitoring:~$
```

FIG. 4.2 : Installation de Prometheus

L'installation de Prometheus via le paquet installe également alertmanager et node-exporter. Avec la commande suivante, on vérifie si les services sont installés et ont correctement démarré :

```
systemctl status prometheus prometheus-node-exporter alertmanager
```

```
deus@monitoring: ~
deus@monitoring: ~ 204x55

● alertmanager.service - AlertManager
   Loaded: loaded (/etc/systemd/system/alertmanager.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-06-14 17:55:07 BST; 10min ago
     Main PID: 601 (alertmanager)
        Tasks: 1 (limit: 4645)
       Memory: 36.0MiB
          CPU: 3.035s
         CGroup: /system.slice/alertmanager.service
                   └─664 /usr/local/bin/alertmanager --config.file /etc/alertmanager/alertmanager.yml --web.external.url=http://192.168.56.104:9093

Warning: some journal files were not opened due to insufficient permissions.

● prometheus.service - Monitoring system and time series database
   Loaded: loaded (/lib/systemd/system/prometheus.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-06-14 17:55:03 BST; 10min ago
     Docs: https://prometheus.io/docs/introduction/overview/
     Main PID: 501 (prometheus)
        Tasks: 1 (limit: 4645)
       Memory: 123.3MiB
          CPU: 9.332s
         CGroup: /system.slice/prometheus.service
                   └─540 /usr/bin/prometheus

Warning: some journal files were not opened due to insufficient permissions.

● prometheus-node-exporter.service - Prometheus exporter for machine metrics
   Loaded: loaded (/lib/systemd/system/prometheus-node-exporter.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-06-14 17:55:03 BST; 10min ago
     Docs: https://github.com/prometheus/node_exporter
     Main PID: 539 (prometheus-node)
        Tasks: 1 (limit: 4645)
       Memory: 31.2MiB
          CPU: 9.868s
         CGroup: /system.slice/prometheus-node-exporter.service
                   └─539 /usr/bin/prometheus-node-exporter

Warning: some journal files were not opened due to insufficient permissions.
-
```

FIG. 4.3 : Vérification des services Prometheus, Node-exporter et Alertmanager

Sur la figure 4.3 on voit bien que tous les services ont démarré correctement

4.3.2.2 Instrumentation des cibles

Tout d'abord, il faut installer les exporters nécessaires sur les cibles. Dans notre cas, nous aurons besoins de :

- node exporter : il expose une grande variété de métriques liées au matériel et au noyau de l'hôte ;
- nginx exporter : il permet d'exposer des métriques nginx pour surveiller les performances ;
- module python prometheus-client : ce module permet d'exposer des metriques provenant d'application développée avec python.

Les exporters sont installes avec la commande :

```
sudo apt install prometheus-nginx-exporter prometheus-node-exporter
```

et le module python avec la commande¹ :

```
pip install prometheus-client
```

¹Les détails sur ce module et l'application web sont donnees dans l'annexe

On configure ensuite le serveur prometheus pour pouvoir collecter les métriques. Le fichier de configuration se trouve dans `/etc/prometheus/prometheus.yml`²

```
1 scrape_configs:
2   - job_name: 'prometheus'
3     scrape_interval: 5s
4     scrape_timeout: 5s
5     static_configs:
6       - targets: ['localhost:9090']
7
8   - job_name: node
9     static_configs:
10      - targets:
11        - moodle.local:9100
12        - bbb.local:9100
13
14   - job_name: web_application
15     static_configs:
16       - targets:
17         - web_application.local:5000
```

Listing 4.1: Fichier de configuration de Prometheus

Dans cette configuration, `job_name` représente un groupe de cibles à surveiller :

- `prometheus` : ce groupe surveille Prometheus lui-même, via l'URL `localhost:9090` ;
- `node` : ce groupe rassemble deux instances de node-exporter, accessibles sur les hôtes `moodle.local:9100` et `bbb.local:9100`. Ces cibles sont des serveurs qui exposent des métriques système ;
- `web_application` : ce groupe représente une application web accessible via `web_application.local:5000`. Prometheus va récupérer des métriques depuis cette application.

4.3.2.3 Alerting

Alertmanager est un composant de l'écosystème qui se charge d'envoyer les notifications en cas de problème. D'abord on va définir les règles d'alerte dans le répertoire `/etc/prometheus/rules` séparées dans les fichiers `nodes_rules.yml` et `web_application_rules.yml`. Une règle se présente comme suit :

²fichier de configuration complet dans l'annexe

```

1 groups:
2 - name: NodeExporter
3   rules:
4
5     # Alert for any instance that is unreachable for >5 minutes.
6     - alert: InstanceDown
7       expr: up == 0
8       for: 1m
9       labels:
10      severity: critical
11      annotations:
12        summary: "Instance {{ $labels.instance }} down"
13        description: "{{ $labels.instance }} of job {{ $labels.job
}} has been down for more than 5 minutes."

```

Cette configuration définit une règle d'alerte Prometheus nommée InstanceDown dans le groupe NodeExporter. Elle se déclenche lorsque l'état up d'une instance est égal à 0 (indiquant une instance hors ligne) pendant plus d'une minute. L'alerte est marquée avec une sévérité de type "critical", signifiant une urgence. Des annotations sont ajoutées pour fournir un résumé et une description détaillée de l'alerte, mentionnant l'instance et le job affectés, afin de faciliter l'identification du problème.

Quelques alertes importantes mise en place :

1. **ServiceBreakdown** : Cette alerte vise à détecter un dysfonctionnement critique sur un service. En se basant sur l'expression $probe_success < 1$, elle se déclenche lorsque le service ne répond plus correctement, indiquant une panne. Cela permet aux administrateurs d'intervenir rapidement pour résoudre le problème et limiter l'impact sur les utilisateurs ;
2. **ServiceUnreachable** : Cette alerte a pour objectif d'identifier un serveur devenu injoignable. L'expression $up < 1$ est utilisée pour signaler qu'un serveur n'est plus accessible, ce qui peut indiquer une défaillance majeure. Grâce à cette alerte, les administrateurs peuvent être avertis immédiatement et intervenir pour rétablir le service ;
3. **HostOutOfMemory** : Cette alerte vise à détecter une saturation de la mémoire RAM disponible sur un hôte. L'expression $(node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10)$ est utilisée pour déclencher l'alerte lorsque la mémoire disponible descend en dessous de 10%. Cela permet d'intervenir avant que l'hôte ne subisse un ralentissement des performances ou un crash complet ;
4. **HostHighCpuLoad** : Cette alerte vise à détecter une charge CPU élevée sur un hôte. En utilisant l'expression $sum by (instance) (avg by (mode, instance) (rate(node_cpu_seconds_total{mode!="idle"}[2m]))) > 0.8$, l'alerte se déclenche lorsque la charge CPU dépasse 80% pendant une minute. Classée avec une sévérité de type "warning", elle informe les administrateurs d'une utilisation élevée des ressources CPU, leur permettant ainsi d'anticiper et de résoudre des problèmes potentiels avant qu'ils n'affectent les performances du système ;

5. **HostMemoryIsUnderutilized** : Cette alerte identifie un sous-utilisation prolongée de la mémoire sur un hôte. L'expression $100 - (\text{avg_over_time}(\text{node_memory_MemAvailable_bytes}[30m]) / \text{node_memory_MemTotal_bytes} * 100) < 20$ se déclenche si moins de 20% de la mémoire est utilisée pendant une semaine. Classée avec une sévérité "info", elle suggère de considérer la réduction de l'espace mémoire, indiquant un possible surdimensionnement des ressources sur l'hôte. Il est conseillé d'augmenter l'intervalle de répétition de l'alerte à un rythme quotidien ou hebdomadaire.

Ensuite on déclare le fichier qui contient les règles dans la configuration de Prometheus

```
1 # Alertmanager configuration
2 alerting:
3   alertmanagers:
4     - static_configs:
5       - targets:
6         - localhost:9093
7
8 # Load rules once and periodically evaluate them according to the
9 # global 'evaluation_interval'.
10 rule_files:
11   - /etc/prometheus/rules/nodes_rules.yml
12   - /etc/prometheus/rules/web_application_rules.yml
```

Et au final on configure le canal sur lequel les alertes seront envoyées dans le fichier **/etc/alertmanager/alertmanager.yml**

```
1 global:
2   resolve_timeout: 2m
3   smtp_require_tls: false
4
5 route:
6   group_by: [ 'alernename' ]
7   # Send all notifications to me.
8   group_wait: 30s
9   group_interval: 1m
10  repeat_interval: 5s
11  receiver: 'notif-me'
12
13 receivers:
14 - name: 'notif-me'
15   telegram_configs:
16   - bot_token: #####
17     api_url: https://api.telegram.org
18     chat_id: xxxxxxxxxx
19     parse_mode: ''
```

Alertmanager a été configuré pour envoyer les notifications via Telegram

4.3.2.4 Tests

Exposition des métriques : Chaque exporter écoute un port spécifique afin de rendre ses métriques disponibles. Le ‘node-exporter’ par exemple, écoute sur le port 9100. Pour afficher toutes les métriques exposées par ‘node-exporter’, on peut utiliser la commande ‘curl <cible_ip_adresse>:9100/metrics‘ ou entrer directement l’adresse ‘<cible_ip_adresse>:9100/metrics‘ dans un navigateur.

Étude et la mise en place dun système de monitoring des infrastructure informatique

```

< > C ⚠ Not secure monitoring.local:9100/metrics

# HELP apt.autoremove pending Apt packages pending autoremoval.
# TYPE apt.autoremove pending gauge
apt.autoremove pending 0
# HELP apt.upgrades held Apt packages pending updates but held back.
# TYPE apt.upgrades held gauge
apt.upgrades.held[arch="","origin=""] 0
# HELP apt.upgrades.pending Apt packages pending updates by origin.
# TYPE apt.upgrades.pending gauge
apt.upgrades.pending[arch="all","origin="Debian:bookworm-security/stable-security"] 4
apt.upgrades.pending[arch="all","origin="Debian:bookworm/stable"] 2
apt.upgrades.pending[arch="arm64","origin=". beta:beta/beta"] 1
apt.upgrades.pending[arch="arm64","origin=". stable:stable/stable"] 2
apt.upgrades.pending[arch="amd64","origin="Debian:bookworm-security/stable-security"] 23
apt.upgrades.pending[arch="amd64","origin="Debian:bookworm/stable"] 16
# HELP go_gc.duration.seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc.duration.seconds summary
go_gc.duration.seconds.quantile("0") 2.1727e-05
go_gc.duration.seconds.quantile("0.25") 2.8644e-05
go_gc.duration.seconds.quantile("0.5") 3.5085e-05
go_gc.duration.seconds.quantile("0.75") 5.0103e-05
go_gc.duration.seconds.quantile("1") 8.000311288
go_gc.duration.seconds.sum 0.015148721
go_gc.duration.seconds.count 349
# HELP go goroutines Number of goroutines that currently exist.
# TYPE go.goroutines gauge
go.goroutines 9
# HELP go.info Information about the Go environment.
# TYPE go.info gauge
go.info(version="gol.19.8") 1
# HELP go.memstats.alloc.bytes Number of bytes allocated and still in use.
# TYPE go.memstats.alloc.bytes gauge
go.memstats.alloc.bytes 2.165672e+06
# HELP go.memstats.alloc.bytes.total Total number of bytes allocated, even if freed.
# TYPE go.memstats.alloc.bytes.total counter
go.memstats.alloc.bytes.total 6.85623456e+08
# HELP go.memstats.buck.hash.sys.bytes Number of bytes used by the profiling bucket hash table.
# TYPE go.memstats.buck.hash.sys.bytes gauge
go.memstats.buck.hash.sys.bytes 1.07305e+06
# HELP go.memstats.frees Total number of frees.
# TYPE go.memstats.frees.total counter
go.memstats.frees.total 1.621833e+07
# HELP go.memstats.gc.sys.bytes Number of bytes used for garbage collection system metadata.
# TYPE go.memstats.gc.sys.bytes gauge
go.memstats.gc.sys.bytes 9.397256e+06
# HELP go.memstats.heap.alloc.bytes Number of heap bytes allocated and still in use.
# TYPE go.memstats.heap.alloc.bytes gauge
go.memstats.heap.alloc.bytes 1.65567e+06
# HELP go.memstats.heap.idle.bytes Number of heap bytes waiting to be used.
# TYPE go.memstats.heap.idle.bytes gauge
go.memstats.heap.idle.bytes 3.366912e+06
# HELP go.memstats.heap.inuse.bytes Number of heap bytes that are in use.
# TYPE go.memstats.heap.inuse.bytes gauge
go.memstats.heap.inuse.bytes 4.562944e+06
# HELP go.memstats.heap.objects Number of allocated objects.
# TYPE go.memstats.heap.objects gauge
go.memstats.heap.objects 19298
# HELP go.memstats.heap.released.bytes Number of heap bytes released to OS.
# TYPE go.memstats.heap.released.bytes gauge
go.memstats.heap.released.bytes 2.8672e+06
# HELP go.memstats.heap.sys.bytes Number of heap bytes obtained from system.
# TYPE go.memstats.heap.sys.bytes gauge
go.memstats.heap.sys.bytes 9.29256e+06
# HELP go.memstats.last_gc_time.seconds Number of seconds since 1970 of last garbage collection.
# TYPE go.memstats.last_gc_time.seconds gauge

```

FIG. 4.4 : Exposition des métriques par ‘node-exporter’

Prometheus dispose également d’une interface web qui permet d’interagir avec le serveur de métriques et de visualiser celles-ci en temps réel. On peut y accéder via un navigateur en entrant l’adresse suivante : ‘<prometheus_server_ip_address> :9090’. Cette interface offre des fonctionnalités de recherche et de requête pour explorer les données collectées par Prometheus.

Element	Value
no data	

FIG. 4.5 : Interface web de Prometheus

Sur l’interface web, une barre de recherche permet d’exécuter des requêtes PromQL pour interroger les métriques. Par exemple, pour vérifier l’état d’une instance, on peut effectuer une requête sur la métrique ‘up’. La figure ci-dessous montre l’exécution de cette requête, où la valeur de ‘up’ à 1 signifie que le serveur est actif et fonctionnel.

Étude et la mise en place dun système de monitoring des infrastructure informatique

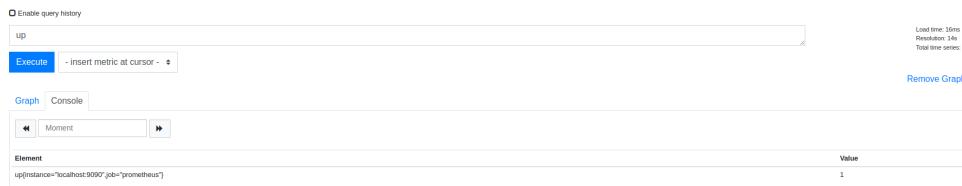


FIG. 4.6 : Exécution de la requête ‘up’

Alerting : Pour tester le système d’alerte, nous avons simulé un problème en arrêtant une machine virtuelle (VM). Après quelques instants, une alerte se déclenche automatiquement sur l’interface web de Prometheus, indiquant l’incident détecté.

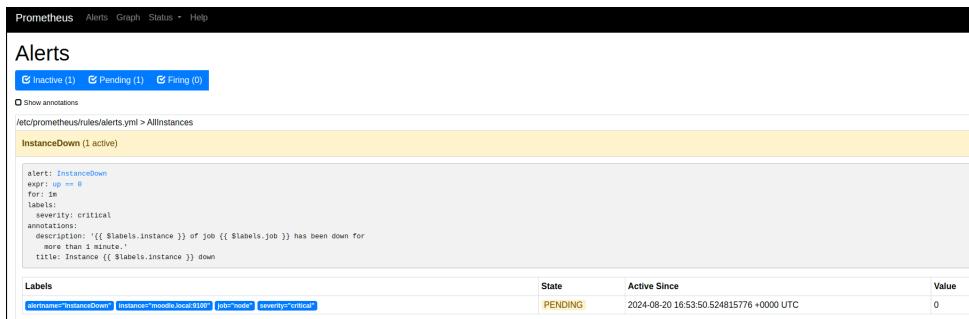


FIG. 4.7 : Visualisation d’une alerte depuis l’interface web de Prometheus

En parallèle, nous avons configuré une intégration avec Telegram pour recevoir une notification instantanée lorsqu’une alerte est déclenchée. La figure ci-dessous montre un exemple de notification envoyée via Telegram suite à l’incident détecté.

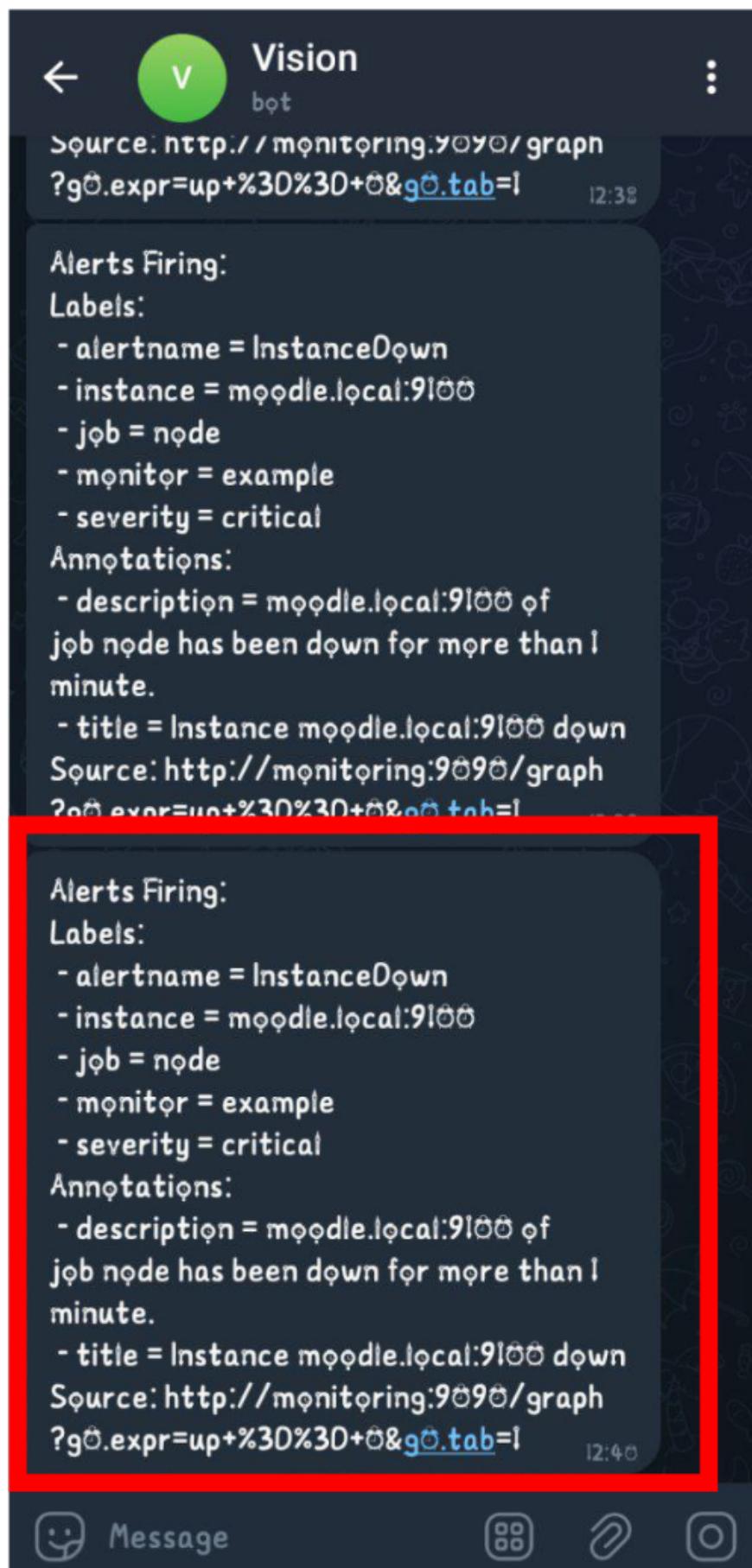


FIG. 4.8 : Notification de l'alerte depuis Telegram

L'alerte InstanceDown signale une interruption critique de service sur l'instance moodle.local au port 9100, associée au job node. Cette alerte indique que l'instance moodle.local :9100 est injoignable depuis plus d'une minute, ce qui pourrait affecter la surveillance et les métriques collectées de cet hôte. La sévérité critique implique une intervention rapide pour rétablir l'accèsibilité. Les détails de l'alerte et l'état actuel de l'instance sont consultables sur l'interface Prometheus via le lien suivant : <http://monitoring:9090/graph?g0.expr=up+&g0.tab=1>.

Lorsque le problème est résolu, Prometheus détecte automatiquement la reprise normale du service, et une nouvelle notification est envoyée sur Telegram pour indiquer la résolution de l'alerte, permettant ainsi un suivi complet des incidents et de leurs résolutions.

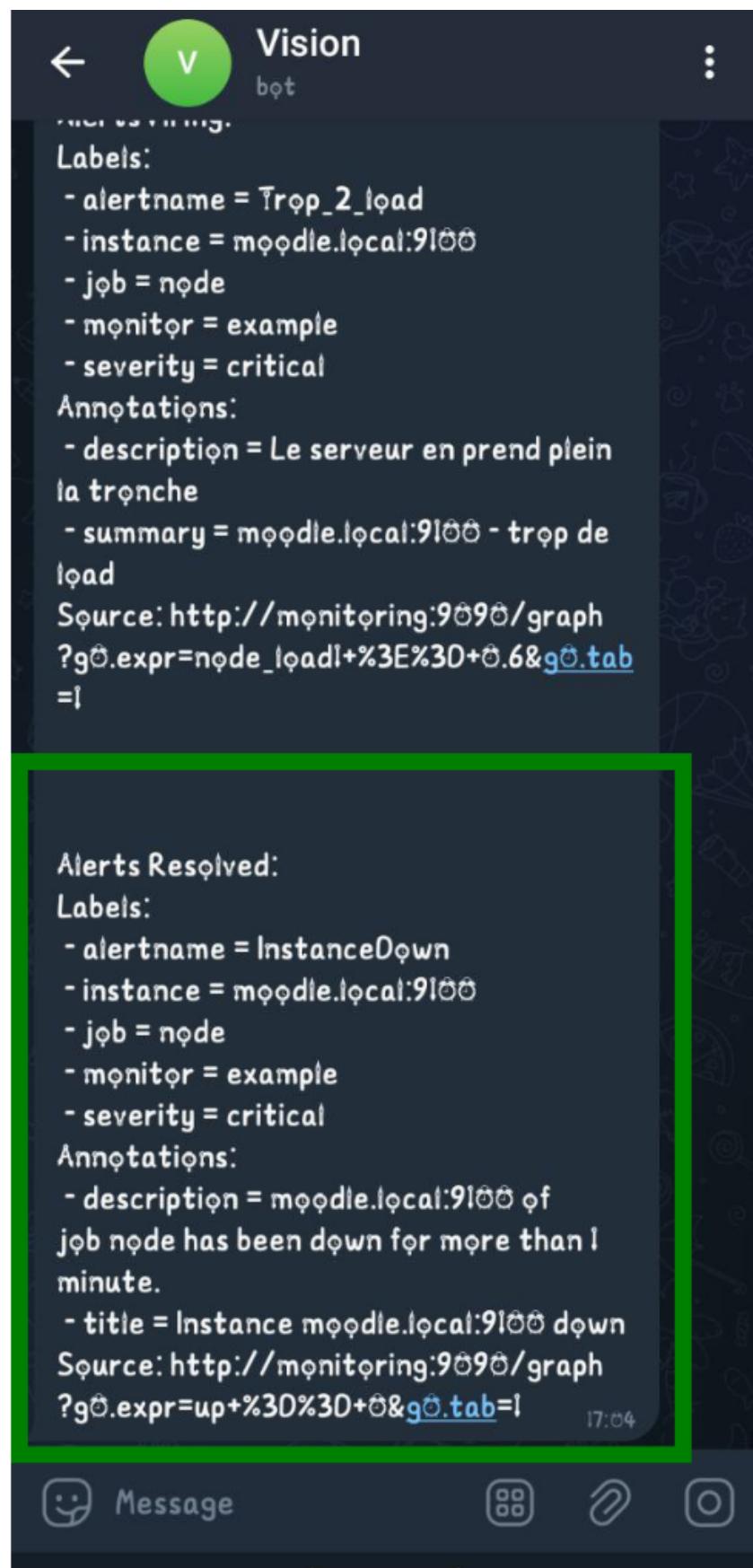


FIG. 4.9 : Notification de résolution de l'alerte depuis Telegram

Sur la figure 4.9, notre système envoie un message pour notifier que l'alerte signalée précédemment a été résolue.

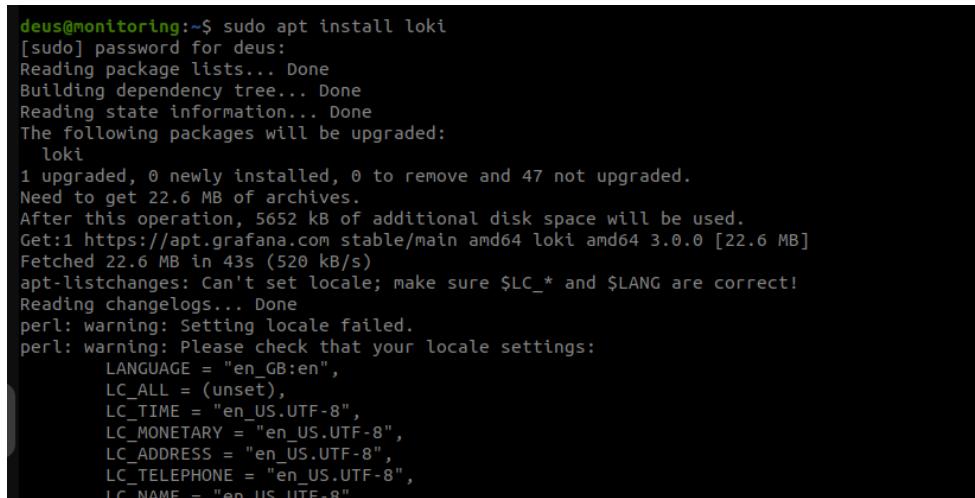
4.3.3 Loki

Nous allons maintenant déployer Loki et Promtail pour lagrégation des logs.

4.3.3.1 Installation

Sur le serveur, on installe Loki avec la commande³ :

```
sudo apt install loki
```

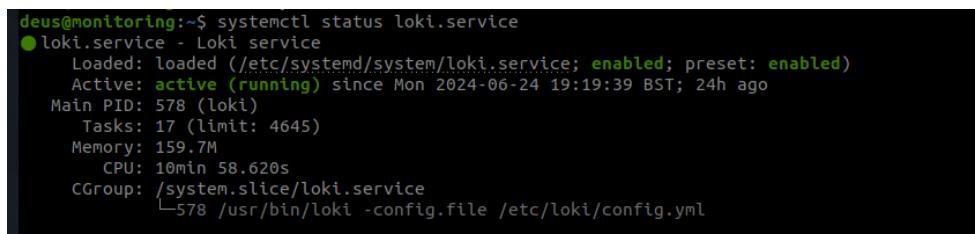


```
deus@monitoring:~$ sudo apt install loki
[sudo] password for deus:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  loki
1 upgraded, 0 newly installed, 0 to remove and 47 not upgraded.
Need to get 22.6 MB of archives.
After this operation, 5652 kB of additional disk space will be used.
Get:1 https://apt.grafana.com stable/main amd64 loki amd64 3.0.0 [22.6 MB]
Fetched 22.6 MB in 43s (520 kB/s)
apt-listchanges: Can't set locale; make sure $LC_* and $LANG are correct!
Reading changelogs... Done
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
  LANGUAGE = "en_GB:en",
  LC_ALL = (unset),
  LC_TIME = "en_US.UTF-8",
  LC_MONETARY = "en_US.UTF-8",
  LC_ADDRESS = "en_US.UTF-8",
  LC_TELEPHONE = "en_US.UTF-8",
  LC_NAME = "en_US.UTF-8"
```

FIG. 4.10 : Installation de Loki

On vérifie si le service a démarrer correctement avec la commande :

```
systemctl status loki.service
```



```
deus@monitoring:~$ systemctl status loki.service
● loki.service - Loki service
   Loaded: loaded (/etc/systemd/system/loki.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-06-24 19:19:39 BST; 24h ago
     Main PID: 578 (loki)
        Tasks: 17 (limit: 4645)
       Memory: 159.7M
          CPU: 10min 58.620s
        CGroup: /system.slice/loki.service
                  └─578 /usr/bin/loki -config.file /etc/loki/config.yml
```

FIG. 4.11 : Status du service Loki

4.3.3.2 Instrumentation des cibles

Sur le serveur a monitroer on installe l'agent Promtail avec la commande :

```
sudo apt install promtail
```

³Cette installation nécessite de suivre les étapes décrites a la section 4.3.4.1

Après l'installation de Promtail, on le configure pour qu'il pusse envoyer les logs au serveur Loki. Le fichier de configuration se trouve a **/etc/promtail/config.yml**

```

1   server:
2     http_listen_port: 9080
3     grpc_listen_port: 0
4
5   positions:
6     filename: /tmp/positions.yaml
7
8   clients:
9     - url: http://localhost:3100/loki/api/v1/push
10
11 scrape_configs:
12   - job_name: system
13     static_configs:
14       - targets:
15         - localhost
16       labels:
17         job: varlogs
18         #NOTE: Need to be modified to scrape any additional logs of
19         #      the system.
20         --path--: /var/log/*log

```

Dans la section clients, on renseigne l'adresse du serveur Loki et dans le scrape_configs on renseigne le chemin des fichiers journaux a importer

4.3.3.3 Tests

Pour tester, on saisi dans un navigateur, l'adresse du serveur sur le port 9080 (<http://monitoring.local:9080/targets>)

Type	Ready	Labels	Details	
			Path	Position
File	TRUE	job=varlogs	/var/log/README	1041
			/var/log/alternatives.log	0
			/var/log/alternatives.log.1	13390
			/var/log/auth.log	17682
			/var/log/auth.log.1	18310
			/var/log/auth.log.2.gz	535
			/var/log/auth.log.3.gz	1696
			/var/log/auth.log.4.gz	13277
			/var/log/cron.log	14797
			/var/log/cron.log.1	13692
			/var/log/cron.log.2.gz	788
			/var/log/cron.log.3.gz	1423
			/var/log/cron.log.4.gz	5936
			/var/log/dpkg.log	474
			/var/log/dpkg.log.1	4314
			/var/log/dpkg.log.2.gz	23300
			/var/log/faillog	0
			/var/log/kern.log	64273
			/var/log/kern.log.1	192703

FIG. 4.12 : Liste des fichiers journaux collecter

On voit sur l'image 4.12 la liste des fichier scraper par promtail

4.3.4 Grafana

Procémons maintenant au déploiement de Grafana et à la configuration des sources de données avec Prometheus et Loki afin de créer des tableaux de bord

4.3.4.1 Installation

L'installation de grafana se fait en 4 etapes

1. Installation des packets pre-requis :

```
sudo apt-get install -y apt-transport-https software-properties-common  
wget
```

2. Import des clés GPG

```
sudo mkdir -p /etc/apt/keyrings/ \\  
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | \\  
sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null
```

3. ajout d'un référentiel pour les versions stables

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] \\  
https://apt.grafana.com stable main" | \\  
sudo tee -a /etc/apt/sources.list.d/grafana.list
```

4. Installation de Grafana

```
sudo apt update && sudo apt install grafana
```

Après l'installation on accède à l'interface de Grafana via le web sur le port **3000**

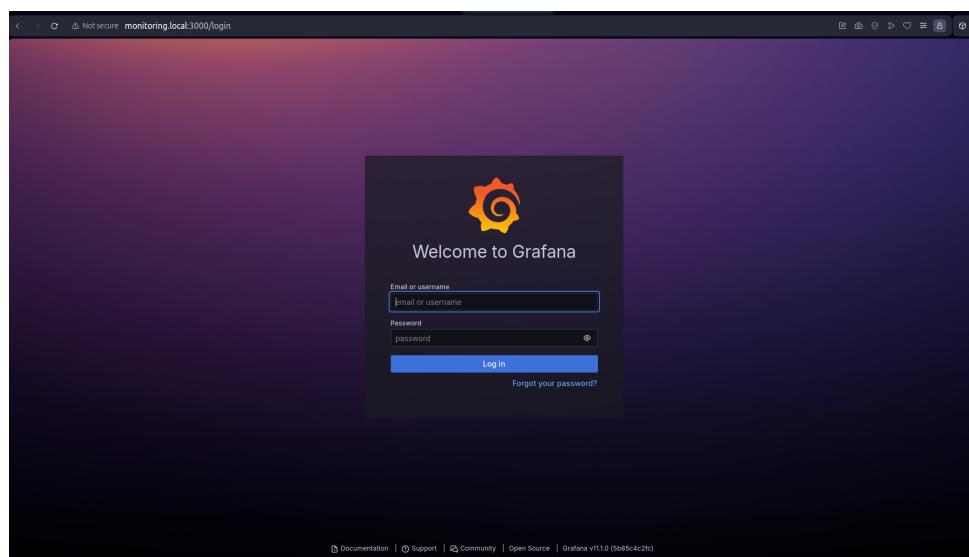


FIG. 4.13 : Connexion a l'interface de Grafana

Étude et la mise en place dun système de monitoring des infrastructure informatique

Les identifiants par defaut sont **admin :admin**. On est appelle a changer le mot de passe a la premiere connexion

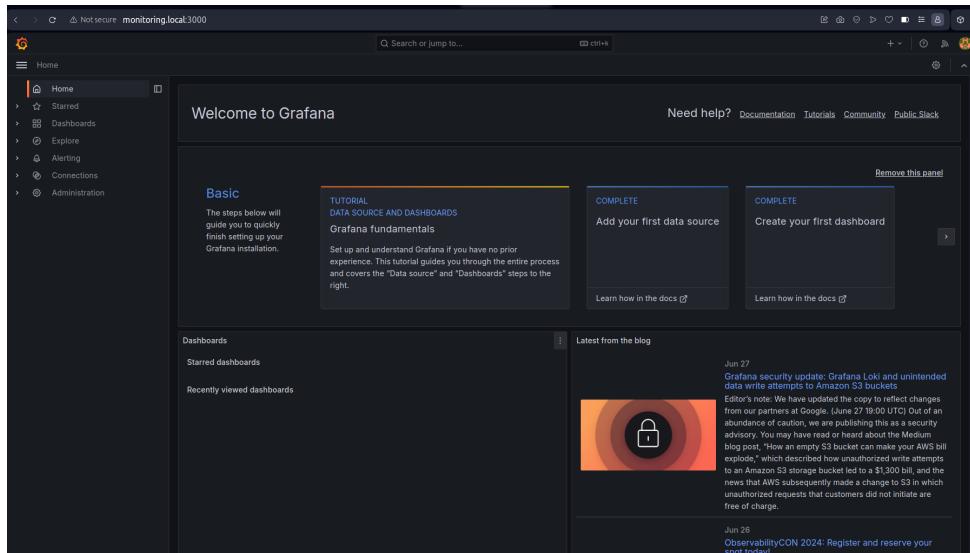


FIG. 4.14 : Interface de Grafana

4.3.4.2 Les sources de données

Grafana est livré avec un support intégré pour de nombreuses sources de données. Chaque source de données est fournie avec un éditeur de requêtes , qui formule des requêtes personnalisées en fonction de la structure de la source. Après avoir ajouté et configuré une source de données, on peut alors effectuer les opérations suivantes :

- interrogez les données avec Explore,
- visualisez les données en panneaux,
- créer des règles pour les alertes.

Pour ajouter Prometheus et Loki qui sont nos sources de données, on suit les étapes suivantes :

1. cliquer sur Connexions dans le menu de gauche ;
2. sous Connexions , cliquer sur Ajouter une nouvelle connexion ;
3. entrer Prometheus ou Loki dans la barre de recherche ;
4. sélectionner le template approprié ;
5. cliquer sur Ajouter une nouvelle source de données en haut à droite.

On est ensuite redirigé vers l'onglet Paramètres pour configurer Prometheus ou Loki

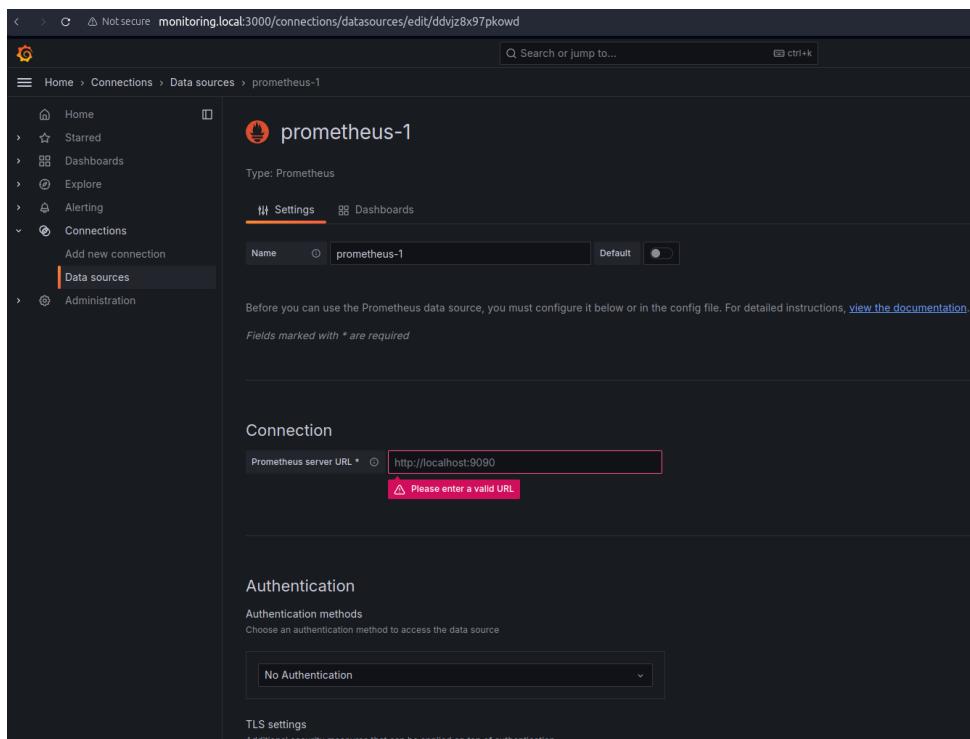


FIG. 4.15 : Panneau de configuration de Prometheus

Les informations requise sont :

- **Name** : c'est le nom de la source de données. Ici nous allons mettre **prometheus**
- **Prometheus server URL** : L'URL de votre serveur Prometheus. Pour notre cas, le serveur est en local donc a l'adresse : *http://localhost:9090*

4.3.4.3 les tableaux de bord

Un dashboard Grafana est un regroupement de panels (graphiques, tableaux, jauge, etc.) qui permettent de visualiser et analyser les données de manière unifiée. Les dashboards offrent une vue d'ensemble de vos métriques provenant de diverses sources comme Prometheus et Loki

Pour créer un nouveau dashboard vide, on clique sur le bouton New Dashboard dans le menu de gauche. Grafana prépare alors un dashboard vide avec :

- un champ pour nommer le dashboard,
- un nouveau panel par défaut,
- un bouton pour ajouter d'autres panels.

Chaque panel possède des options de configuration dans différents onglets :

- Queries : pour définir les requêtes qui récupèrent les données à afficher ;
- Visualization : pour choisir le type de visualisation et le personnaliser ;
- General : pour donner un titre au panel et d'autres options générales ;

- Alert : pour définir des alertes sur le panel.

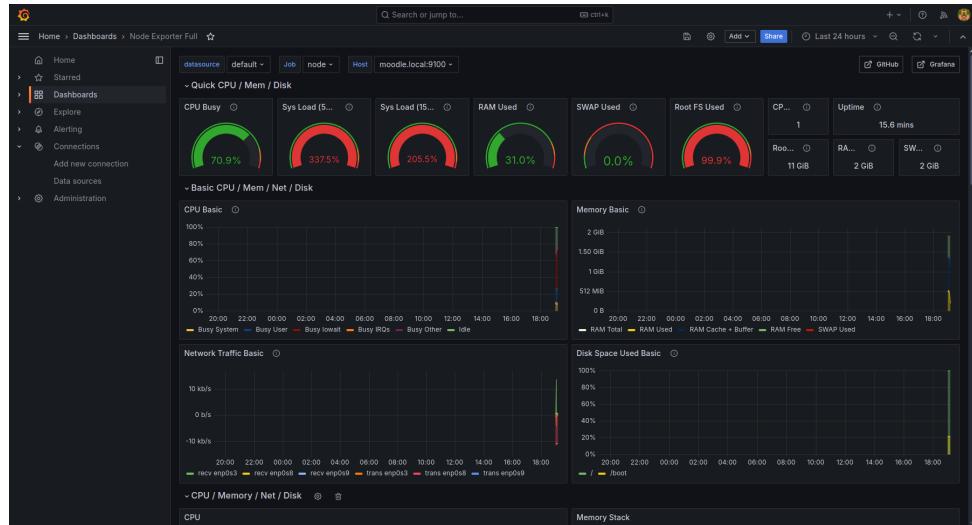


FIG. 4.16 : Dashboard de node-exporter

4.4 Évaluation financière du système

Le but de cette section est de réaliser une estimation du coût total du système de monitoring. La tableau 4.4 résume les coups liés à la réalisation.

TAB. 4.4 : Évaluation financière de la solution

	Prix unitaire	Quantité	Prix total
Coûts Matériels	43 000 FCFA/mois ⁴	12	516 000 FCFA
Coût de l'implémentation	2 500 FCFA/heure	220 heures ⁵	550 000 FCFA
Coûts de Maintenance et Support	2 000 FCFA/heure	1440 heures ⁶	2 880 000 FCFA
Coût total	3 946 000 FCFA		

Seul le coût de l'implémentation est une charge temporaire. Hormis celui-ci, l'exploitation du systeme revient a 3 396 000 FCFA par an

4.5 Bilan

Après le déploiement de notre solution, il est désormais beaucoup plus facile d'explorer et de gérer l'infrastructure informatique de Dac Technologies. Auparavant, il était nécessaire

¹abonnement starter chez OVHcloud

²5 heures par jour durant 44 jours

³4 heures par jour durant 1 an

Étude et la mise en place d'un système de monitoring des infrastructures informatiques

de se connecter individuellement à chaque serveur pour collecter des informations sur l'utilisation des ressources, ce qui rendait le processus long et complexe. Désormais, grâce au système de monitoring, tout cela peut être fait à partir d'un seul endroit : le dashboard de Grafana.

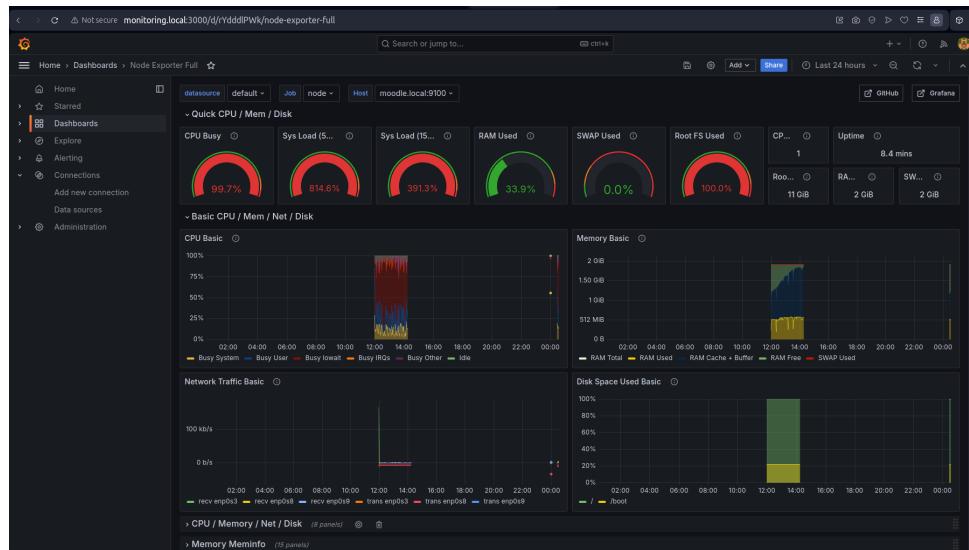


FIG. 4.17 : Dashboard de Grafana

Sur ce dashboard (figure 4.17), nous avons un aperçu global de la consommation de la RAM, du CPU, du stockage, et bien d'autres métriques essentielles. Il est possible de passer facilement d'un serveur à un autre à l'aide du menu déroulant Host, tout en ayant une vision d'ensemble en temps réel des performances de l'infrastructure (figure 4.18).

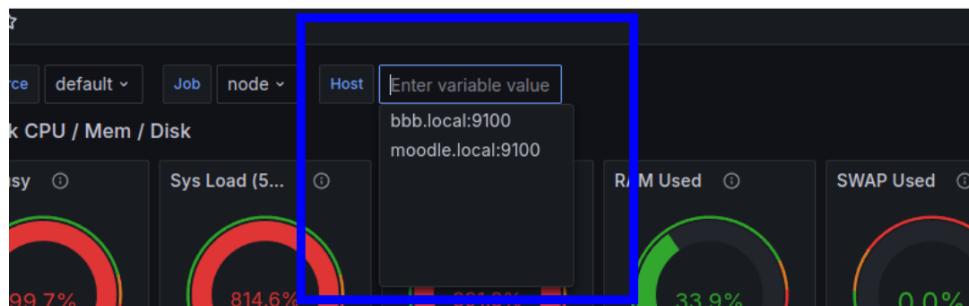


FIG. 4.18 : Menu déroulant pour sélectionner un serveur

L'accès aux fichiers journaux a également été simplifié. Auparavant, consulter les logs en cas de problème pouvait être compliqué, surtout lorsqu'il était difficile d'identifier immédiatement quel composant du système était en cause. Désormais, grâce à l'intégration de Loki avec Grafana, tous les logs sont centralisés et peuvent être consultés depuis une interface unique. De plus, la possibilité de filtrer les logs à l'aide de requêtes LogQL permet d'accélérer considérablement le processus de diagnostic (figure 4.19).

Étude et la mise en place dun système de monitoring des infrastructure informatique

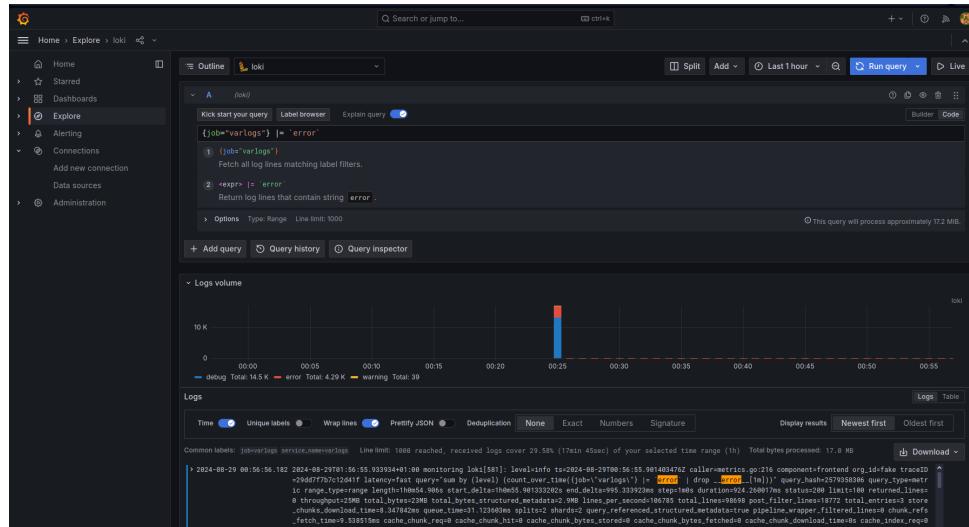


FIG. 4.19 : Exploration de logs sur grafana

La requete `job="varlogs" |= 'error'` par exemple nous permet on a toutes les lignes qui contiennent "error" (figure 4.20)

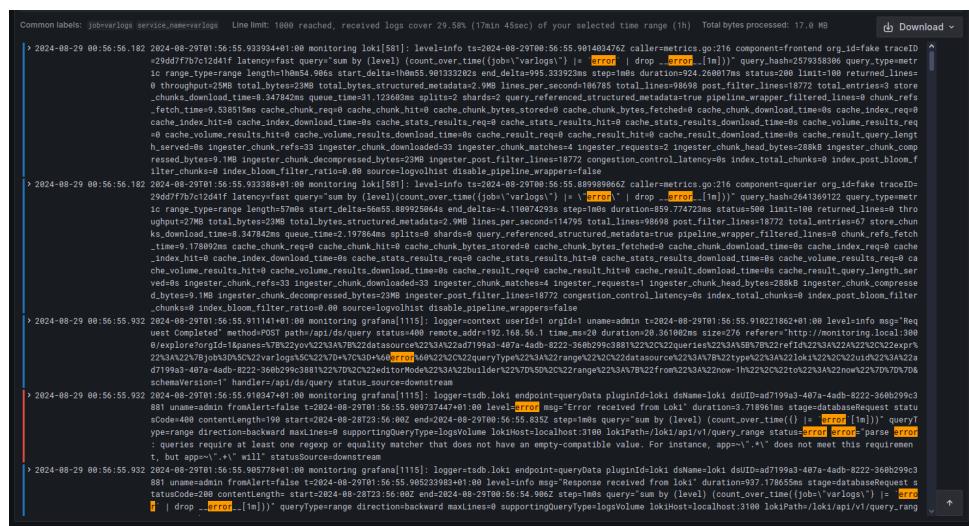


FIG. 4.20 : Lignes des fichiers journaux contenant des erreurs

L'un des changements les plus significatifs est la détection proactive des pannes. Au lieu d'attendre qu'un utilisateur signale un problème, le système d'alerte configuré avec Prometheus envoie automatiquement des notifications aux administrateurs dès qu'un seuil critique est atteint. Ces notifications incluent des détails sur la panne, ce qui permet une prise en charge rapide et efficace.

Sur la figure 4.21 nous avons la liste des métriques à surveiller

The screenshot shows the Prometheus 'Alerts' page. At the top, there is a navigation bar with links for 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, the word 'Alerts' is displayed in a large, bold, dark font. Underneath 'Alerts', there is a blue header bar containing three status indicators: 'Inactive (9)', 'Pending (1)', and 'Firing (0)'. To the right of these indicators is a checkbox labeled 'Show annotations'. Below the header, there are two main sections, each preceded by a file path: '/etc/prometheus/rules/alerts.yml > AllInstances' and '/etc/prometheus/rules/alerts.yml > test'. The 'AllInstances' section contains a yellow header row with the text 'InstanceDown (4 active)' in bold. Below this, there are eight green rows, each representing a different alert rule: 'HostCpusUnderutilized (0 active)', 'HostDiskWillFillIn24Hours (0 active)', 'HostHighCpuLoad (0 active)', 'HostMemoryIsUnderutilized (0 active)', 'HostOutOfDiskSpace (0 active)', 'HostOutOfMemory (0 active)', 'HostSystemdServiceCrashed (0 active)', and 'ServiceBreakdown (0 active)'. The 'test' section has a single green row with the text 'Trop_2_load (0 active)'.

FIG. 4.21 : Listes des alertes configurées

La figure 4.22 nous montre les alertes qui ont été envoyé par Alertmanager à notre canal de notification

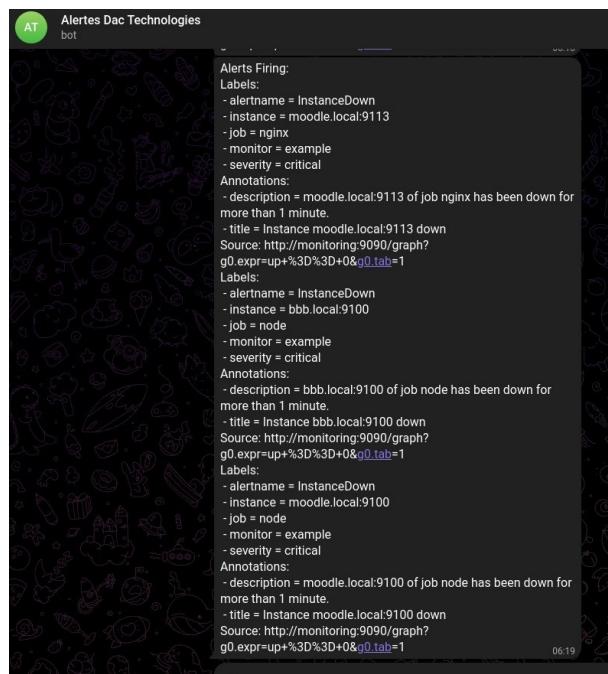


FIG. 4.22 : Alertes envoyées aux administrateurs via Telegram

Tout ceci contribue à rendre l'infrastructure informatique de Dac Technologies hautement disponible, plus réactive et mieux surveillée, assurant ainsi la continuité des services et une meilleure expérience utilisateur pour les clients et les employés de l'entreprise.

4.6 Conclusion

En conclusion de ce chapitre sur la mise en uvre, nous avons exploré les étapes cruciales ayant conduit à létablissement dun système de monitoring robuste pour Dac Technologies. À travers l'analyse approfondie des besoins spécifiques de l'entreprise et le déploiement d'outils adaptés tels que Grafana, Prometheus et Loki, nous avons réussi à créer un système capable de surveiller efficacement les performances et la disponibilité des infrastructures informatiques.

La mise en place de ce système a permis non seulement daméliorer la visibilité sur létat des infrastructures, mais aussi de centraliser les données, facilitant ainsi la prise de décision proactive. Les alertes automatiques générées en cas de dysfonctionnements garantissent une réactivité accrue, contribuant à la stabilité et à la sécurité des services.

Conclusion générale

Le présent mémoire a permis de répondre à un besoin critique pour Dac Technologies en implémentant un système de monitoring des infrastructures informatiques, adapté à l'environnement de l'entreprise. Ce travail s'est articulé autour de plusieurs étapes, allant de l'analyse des solutions existantes à la mise en œuvre d'un système de surveillance basé sur Grafana, Prometheus et Loki.

Dans un contexte de digitalisation croissante des services, où la surveillance proactive des systèmes informatiques est essentielle, ce projet a apporté une amélioration significative à la disponibilité, la performance et la sécurité des infrastructures de Dac Technologies. L'intégration d'outils modernes de visualisation et de collecte de métriques a permis de surmonter les limites des outils natifs, notamment en centralisant les données, en historisant les métriques et en générant des alertes automatiques.

Bien que le système mis en place réponde efficacement aux besoins actuels de l'entreprise, plusieurs perspectives permettent d'envisager des améliorations. Tout d'abord, sécuriser les points de contact de notre système avec les cibles, garantissant ainsi la confidentialité des échanges. Enfin, garantir que chaque tableau de bord soit accessible uniquement aux utilisateurs concernés. Cela inclut une gestion affinée des accès pour que chaque utilisateur ait une vue personnalisée des informations pertinentes à son rôle.

Ce travail constitue ainsi non seulement laboutissement de ce projet, mais aussi une base solide pour l'évolution future de la surveillance des systèmes informatiques au sein de Dac Technologies, contribuant à la performance et à la pérennité de ses services.

Annexe

Fichier de configuration de Prometheus

```
1 # Sample config for Prometheus.
2
3 global:
4     scrape_interval:      15s # Set the scrape interval to every 15
5         seconds. Default is every 1 minute.
6     evaluation_interval: 15s # Evaluate rules every 15 seconds. The
7         default is every 1 minute.
8     # scrape_timeout is set to the global default (10s).
9
10    # Attach these labels to any time series or alerts when
11        communicating with
12    # external systems (federation, remote storage, Alertmanager).
13    external_labels:
14        monitor: 'example'
15
16 # Alertmanager configuration
17 alerting:
18     alertmanagers:
19         - static_configs:
20             - targets:
21                 - localhost:9093
22
23 # Load rules once and periodically evaluate them according to the
24     # global 'evaluation_interval'.
25 rule_files:
26     # - "first_rules.yml"
27     # - "second_rules.yml"
28     - /etc/prometheus/rules/alerts.yml
29     - /etc/prometheus/rules/nginx.yml
30
31
32 # A scrape configuration containing exactly one endpoint to
33     # scrape:
34 # Here it's Prometheus itself.
35 scrape_configs:
36     # The job name is added as a label 'job=<job_name>' to any
37         # timeseries scraped from this config.
38     - job_name: 'prometheus'
```

```
34 # Override the global default and scrape targets from this
35   # job every 5 seconds.
36   scrape_interval: 5s
37   scrape_timeout: 5s
38
39   # metrics_path defaults to '/metrics'
40   # scheme defaults to 'http'.
41
42   static_configs:
43     - targets: ['localhost:9090']
44
45   - job_name: 'loki'
46
47   # Override the global default and scrape targets from this
48   # job every 5 seconds.
49   scrape_interval: 5s
50   scrape_timeout: 5s
51
52   # metrics_path defaults to '/metrics'
53   # scheme defaults to 'http'.
54
55   static_configs:
56     - targets: ['localhost:3100']
57
58
59
60   - job_name: node
61     # If prometheus-node-exporter is installed, grab stats about
62       # the local
63       # machine by default.
64     static_configs:
65       - targets:
66         - moodle.local:9100
67         - bbb.local:9100
68
69   - job_name: nginx
70     static_configs:
71       - targets:
72         - moodle.local:9113
73
74   - job_name: web_app
75     static_configs:
76       - targets:
77         - localhost:5000
```

Fichier de configuration des alertes

```
1 groups:
2   - name: test
3     rules:
4       - alert: Trop_2_load
5         expr: node_load1 >= 0.6
6         for: 10s
7         labels:
8           severity: critical
9         annotations:
10           summary: "{{\$labels.instance}} - Charge trop élevée"
11           description: "Le serveur subit une charge importante, ce
12             qui peut affecter ses performances."
13
14   - name: AllInstances
15     rules:
16       - alert: InstanceDown
17         expr: up == 0
18         for: 1m
19         annotations:
20           title: 'Instance {{ \$labels.instance }} hors service'
21           description: "L'instance {{ \$labels.instance }} du job {{ \$labels.job }} est hors service depuis plus d'une minute."
22
23       - alert: HostHighCpuLoad
24         expr: (sum by (instance) (avg by (mode, instance) (rate(
25           node_cpu_seconds_total{mode!="idle"}[2m]))) > 0.8) * on(
26             instance) group_left (nodename) node_uname_info{nodename =~ ".+"}
27         for: 1m
28         labels:
29           severity: warning
30         annotations:
31           summary: "Charge CPU élevée sur l'hôte (instance {{ \$labels.instance }})"
32           description: "La charge CPU est supérieure à 80%\n            VALEUR = {{ \$value }}\n            ÉTIQUETTES = {{ \$labels }}"
33
34       - alert: HostOutOfMemory
35         expr: (node_memory_MemAvailable_bytes /
36           node_memory_MemTotal_bytes * 100 < 10) * on(instance)
37           group_left (nodename) node_uname_info{nodename=~".+"}
38         for: 1m
39         labels:
40           severity: warning
41         annotations:
42           summary: "Mémoire insuffisante sur l'hôte (instance {{ \$labels.instance }})"
```

```
38     description: "La mémoire disponible est inférieure à 10%\n      n  VALEUR = {{ $value }}\n      ÉTIQUETTES = {{ $labels }}"\n\n39\n40 - alert: HostMemoryIsUnderutilized\n  expr: (100 - (avg_over_time(node_memory_MemAvailable_bytes\n    [30m]) / node_memory_MemTotal_bytes * 100) < 20) * on(\n    instance) group_left (nodename) node_uname_info{nodename\n    =~ ".+"}\n  for: 1w\n  labels:\n    severity: info\n  annotations:\n    summary: "Mémoire de l'hôte sous-utilisée (instance {{\n      $labels.instance }})"\n    description: "La mémoire de l'hôte est sous-utilisée (<\n      20%) pendant une semaine. Envisagez de réduire l'espace\n      mémoire.\n      n  VALEUR = {{ $value }}\n      ÉTIQUETTES = {{\n        $labels }}"\n\n46\n47\n48 - alert: HostOutOfDiskSpace\n  expr: ((node_filesystem_avail_bytes * 100) /\n    node_filesystem_size_bytes < 10 and ON (instance, device,\n    mountpoint) node_filesystem_readonly == 0) * on(instance\n    ) group_left (nodename) node_uname_info{nodename=~".+"}\n  for: 2m\n  labels:\n    severity: warning\n  annotations:\n    summary: "Espace disque insuffisant sur l'hôte (instance\n      {{ $labels.instance }})"\n    description: "Le disque est presque plein (< 10% d'espace\n      disponible)\n      n  VALEUR = {{ $value }}\n      ÉTIQUETTES =\n      {{ $labels }}"\n\n55\n56\n57 - alert: HostDiskWillFillIn24Hours\n  expr: ((node_filesystem_avail_bytes * 100) /\n    node_filesystem_size_bytes < 10 and ON (instance, device,\n    mountpoint) predict_linear(node_filesystem_avail_bytes{\n      fstype!~ "tmpfs"}[1h], 24 * 3600) < 0 and ON (instance,\n    device, mountpoint) node_filesystem_readonly == 0) * on(\n    instance) group_left (nodename) node_uname_info{nodename\n    =~ ".+"}\n  for: 2m\n  labels:\n    severity: warning\n  annotations:\n    summary: "Le disque de l'hôte sera plein dans 24 heures (\n      instance {{ $labels.instance }})"\n    description: "Le système de fichiers est sur le point d'\n      être plein dans les prochaines 24 heures à la vitesse\n      actuelle d'écriture.\n      n  VALEUR = {{ $value }}\n      ÉTIQUETTES = {{\n        $labels }}"
```

```
65          ÉTIQUETTES = {{ $labels }}"
```

```
66
67      - alert: HostCpuIsUnderutilized
68          expr: (100 - (rate(node_cpu_seconds_total{mode="idle"})[30m
69              ] * 100) < 20) * on(instance) group_left (nodename)
70              node_uname_info{nodename=~".+"}
71          for: 1w
72          labels:
73              severity: info
74          annotations:
75              summary: "CPU de l'hôte sous-utilisé (instance {{ $labels
76                  .instance }})"
77              description: "Le CPU est sous-utilisé (< 20%) pendant une
78                  semaine. Envisagez de réduire le nombre de CPU.\n
79                  VALEUR = {{ $value }}\n    ÉTIQUETTES = {{ $labels }}"
80
81
82      - alert: HostSystemdServiceCrashed
83          expr: (node_systemd_unit_state{state="failed"} == 1) * on(
84              instance) group_left (nodename) node_uname_info{nodename
85                  =~ ".+"}
86          for: 0m
87          labels:
88              severity: warning
89          annotations:
90              summary: "Le service systemd de l'hôte a planté (instance
91                  {{ $labels.instance }})"
92              description: "Un service systemd a planté.\n    VALEUR = {{
93                  $value }}\n    ÉTIQUETTES = {{ $labels }}"
94
95
96      - alert: ServiceBreakdown
97          expr: probe_success < 1
98          for: 2m
99          labels:
100              severity: critical
101              application: part_of
102          annotations:
103              summary: "Dysfonctionnement sur un service (instance {{
104                  $labels.instance }})"
105              description: "Un service ne répond plus ou renvoie des
106                  erreurs.\n    VALEUR = {{ $value }}\n    ÉTIQUETTES = {{
107                  $labels }}"
108
```

Module python prometheus_client

```
1  from flask import Flask, request, jsonify, render_template
2  from prometheus_client import Counter, generate_latest,
3      CONTENT_TYPE_LATEST
4  import logging
5
6  app = Flask(__name__)
```

```

6      # Configuration du fichier journal
7      logging.basicConfig(filename='app.log', level=logging.INFO,
8                          format='%(asctime)s %(levelname)s: %(message)s')
9
10
11     # Création des compteurs pour Prometheus
12     USER_COUNTER = Counter('connected_users', 'Number of connected
13                               users')
14     BUTTON_COUNTER = Counter('button_presses', 'Number of button
15                               presses')
16
17
18     @app.route('/')
19     def index():
20         global connected_users
21         connected_users += 1
22         USER_COUNTER.inc()
23
24         # Journaliser la connexion de l'utilisateur
25         user_ip = request.remote_addr
26         logging.info(f'New connection from {user_ip}')
27
28         return render_template('index.html')
29
30     @app.route('/press_button', methods=['POST'])
31     def press_button():
32         BUTTON_COUNTER.inc()
33         return "Button pressed!"
34
35     @app.route('/metrics')
36     def metrics():
37         return generate_latest(), 200, {'Content-Type':
38                                         CONTENT_TYPE_LATEST}
39
40     @app.errorhandler(Exception)
41     def handle_exception(e):
42         # Journaliser les erreurs
43         logging.error(f"Error: {str(e)}")
44         return jsonify({"error": str(e)}), 500
45
46     if __name__ == '__main__':
47         app.run(host='0.0.0.0', port=5000)

```

Cette application est une application web développée avec Flask, qui permet de suivre les connexions des utilisateurs ainsi que les interactions avec un bouton via une interface simple. À chaque nouvelle connexion d'un utilisateur sur la page d'accueil, l'application enregistre cette connexion en incrémentant un compteur et en consignant l'adresse IP de l'utilisateur dans un fichier de journalisation ('app.log'). De plus, l'application expose

un endpoint qui permet de suivre les clics sur un bouton. Chaque fois que le bouton est pressé, un compteur est incrémenté pour suivre le nombre total de clics. Ces interactions sont également visibles via une interface frontend basique.

L'application utilise le module `prometheus_client` pour la gestion des métriques. Le module `prometheus_client` permet d'intégrer la collecte de metrics dans l'application Python. Dans ce contexte, deux compteurs Prometheus ont été créés : `USER_COUNTER` pour suivre le nombre total d'utilisateurs qui se connectent à l'application, et `BUTTON_COUNTER` pour suivre le nombre de fois que le bouton est pressé. Les compteurs sont des métriques cumulatives qui ne diminuent jamais, ce qui les rend idéales pour suivre des événements tels que des connexions ou des interactions avec un bouton.

Chaque fois qu'un utilisateur accède à la page d'accueil, le compteur `USER_COUNTER` est incrémenté, et chaque fois que le bouton est pressé, `BUTTON_COUNTER` est incrémenté. L'application expose ensuite ces métriques via un endpoint `/metrics`, où les valeurs des compteurs sont générées sous un format compatible avec Prometheus. Ce format permet à Prometheus de récupérer périodiquement ces métriques pour surveiller l'état de l'application. Cela offre une visibilité sur le nombre de connexions et d'interactions dans l'application au fil du temps.

Installation de `prometheus-nginx-exporter`

`prometheus-nginx-exporter` est un exporter Prometheus qui expose les métriques de Nginx. Après son installation [4.3.2.2](#), il faut également configurer Nginx pour compléter l'installation. D'abord on vérifie si `ngx_http_stub_status_module` est bien disponible sur notre serveur. Ce module permet d'accéder aux informations de base sur l'état de Nginx via une «page d'état». Il affiche des informations telles que le nombre total de connexions client actives, celles acceptées et celles traitées, le nombre total de requêtes et le nombre de connexions en lecture, en écriture et en attente[\[12\]](#). Pour faire sa vérification, on utilise la commande : `nginx -V 2>&1 | grep -o avec-http_stub_status_module`

```
deus@deus-HP-ZBook-17-G3:~$ nginx -V 2>&1 | grep -o with-http_stub_status_module
with-http_stub_status_module
deus@deus-HP-ZBook-17-G3:~$
```

FIG. 4.23 : Vérification du module `ngx_http_stub_status_module`

La commande retourne le nom du module, ce qui signifie qu'il est disponible. Après avoir vérifié le module, on l'active dans le fichier de configuration `/etc/nginx/nginx.conf` pour configurer une URL accessible localement à l'adresse `http://localhost/nginx_status` qui est la page d'état

```
location /status {
    stub_status;
    allow 127.0.0.1;
    deny all;
}
```

Pour sécuriser l'accès à `stub_status`, il est recommandé de le restreindre à une adresse IP spécifique ou à un réseau local, `localhost` dans notre cas

Fichier de configuration des alertes

```
1 groups:
2   - name: test
3     rules:
4       - alert: Trop_2_load
5         expr: node_load1 >= 0.6
6         for: 10s
7         labels:
8           severity: critical
9         annotations:
10           summary: "{{labels.instance}} - trop de load"
11           description: "Le serveur en prend plein la tronche"
12
13   - name: AllInstances
14     rules:
15       - alert: InstanceDown
16         # Condition for alerting
17         expr: up == 0
18         for: 1m
19         # Annotation - additional informational labels to store
20         # more information
21         annotations:
22           title: 'Instance {{ $labels.instance }} down'
23           description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1 minute.'
24         # Labels - additional labels to be attached to the alert
25         labels:
26           severity: 'critical'
27
28       - alert: HostHighCpuLoad
29         expr: (sum by (instance) (avg by (mode, instance) (rate(
30           node_cpu_seconds_total{mode!="idle"}[2m]))) > 0.8) * on(
31             instance) group_left (nodename) node_uname_info{nodename=~".+"}
32         for: 1m
33         labels:
34           severity: warning
35         annotations:
36           summary: Host high CPU load (instance {{ $labels.instance }})
37           description: "CPU load is > 80%\n  VALUE = {{ $value }}\n  LABELS = {{ $labels }}"
38
39       - alert: HostOutOfMemory
40         expr: (node_memory_MemAvailable_bytes /
41             node_memory_MemTotal_bytes * 100 < 10) * on(instance)
42             group_left (nodename) node_uname_info{nodename=~".+"}
43         for: 1m
44         labels:
45           severity: warning
```

```
41 annotations:
42     summary: Host out of memory (instance {{ $labels.instance
43         }})
44     description: "Node memory is filling up (< 10% left)\n
45         VALUE = {{ $value }}\n    LABELS = {{ $labels }}"
46
47 # You may want to increase the alert manager 'repeat_interval'
48 # for this type of alert to daily or weekly
49 - alert: HostMemoryIsUnderutilized
50   expr: (100 - (avg_over_time(node_memory_MemAvailable_bytes[30
51     m]) / node_memory_MemTotal_bytes * 100) < 20) * on(instance)
52     ) group_left (nodename) node_uname_info{nodename=~".+"}
53   for: 1w
54   labels:
55     severity: info
56   annotations:
57     summary: Host Memory is underutilized (instance {{ $labels.
58         instance }})
59     description: "Node memory is < 20% for 1 week. Consider
60         reducing memory space. (instance {{ $labels.instance }})\n
61         VALUE = {{ $value }}\n    LABELS = {{ $labels }}"
62
63 # Please add ignored mountpoints in node_exporter parameters
64 # like
65 # "--collector.filesystem.ignored-mount-points=^(sys|proc|dev|
66 #     run)($/)".
67 # Same rule using "node_filesystem_free_bytes" will fire when
68 #     disk fills for non-root users.
69 - alert: HostOutOfDiskSpace
70   expr: ((node_filesystem_avail_bytes * 100) /
71     node_filesystem_size_bytes < 10 and ON (instance, device,
72     mountpoint) node_filesystem_READONLY == 0) * on(instance)
73     group_left (nodename) node_uname_info{>
74   for: 2m
75   labels:
76     severity: warning
77   annotations:
78     summary: Host out of disk space (instance {{ $labels.
79         instance }})
80     description: "Disk is almost full (< 10% left)\n    VALUE =
81         {{ $value }}\n    LABELS = {{ $labels }}"
82
83 # Please add ignored mountpoints in node_exporter parameters
84 # like
85 # "--collector.filesystem.ignored-mount-points=^(sys|proc|dev|
86 #     run)($/)".
87 # Same rule using "node_filesystem_free_bytes" will fire when
88 #     disk fills for non-root users.
89 - alert: HostDiskWillFillIn24Hours
```

```
73     expr: ((node_filesystem_avail_bytes * 100) /
74             node_filesystem_size_bytes < 10 and ON (instance, device,
75             mountpoint) predict_linear(node_filesystem_avail_bytes{
76               fstype!~"tmpfs"}[1h], 24 * 3600) < 0 and >
77     for: 2m
78     labels:
79       severity: warning
80     annotations:
81       summary: Host disk will fill in 24 hours (instance {{ $labels.instance }})
82       description: "Filesystem is predicted to run out of space
83                     within the next 24 hours at current write rate\n  VALUE =
84                     {{ $value }}\n  LABELS = {{ $labels }}"
85
86 # You may want to increase the alert manager 'repeat_interval'
87 # for this type of alert to daily or weekly
88 - alert: HostCpuIsUnderutilized
89   expr: (100 - (rate(node_cpu_seconds_total{mode="idle"}[30m])
90             * 100) < 20) * on(instance) group_left (nodename)
91             node_uname_info{nodename=~".+"}
92   for: 1w
93   labels:
94     severity: info
95   annotations:
96     summary: Host CPU is underutilized (instance {{ $labels.instance }})
97     description: "CPU load is < 20% for 1 week. Consider
98                   reducing the number of CPUs.\n  VALUE = {{ $value }}\n  LABELS = {{ $labels }}"
99
100 - alert: HostSystemdServiceCrashed
101   expr: (node_systemd_unit_state{state="failed"} == 1) * on(
102     instance) group_left (nodename) node_uname_info{nodename=~".+"}
103   for: 0m
104   labels:
105     severity: warning
106   annotations:
107     summary: Host systemd service crashed (instance {{ $labels.instance }})
108     description: "systemd service crashed\n  VALUE = {{ $value }}\n  LABELS = {{ $labels }}"
109
110 - alert: ServiceBreakdown
111   expr: probe_success < 1
112   for: 2m
113   labels:
114     severity: critical
115     application: part_of
116   annotations:
```

```
108     summary: "Dysfonctionnement sur un service (instance
109           instance {{ $labels.instance }})"
110   description: "Un service r  pond avec erreur\n  VALUE = {{
111       $value }}\n  LABELS = {{ $labels }}"
112 
```

Fichier de configuration Loki

```
1 auth_enabled: false
2
3 server:
4   http_listen_port: 3100
5   grpc_listen_port: 9096
6
7 common:
8   instance_addr: 127.0.0.1
9   path_prefix: /tmp/loki
10  storage:
11    filesystem:
12      chunks_directory: /tmp/loki/chunks
13      rules_directory: /tmp/loki/rules
14  replication_factor: 1
15  ring:
16    kvstore:
17      store: inmemory
18
19 query_range:
20   results_cache:
21     cache:
22       embedded_cache:
23         enabled: true
24         max_size_mb: 100
25
26 schema_config:
27   configs:
28     - from: 2020-10-24
29       store: tsdb
30       object_store: filesystem
31       schema: v13
32       index:
33         prefix: index_
34         period: 24h
35
36 ruler:
37   alertmanager_url: http://localhost:9093
```

Fichier de configuration Promtail

```
1 clients:
2 - url: http://localhost:3100/loki/api/v1/push
3
4 scrape_configs:
5 - job_name: system
6   static_configs:
7     - targets:
8       - localhost
9     labels:
10    job: varlogs
11    #NOTE: Need to be modified to scrape any additional logs of
12    #       the system.
13    __path__: /var/log/*
14 - job_name: application
15   static_configs:
16     - targets:
17       - localhost
18     labels:
19       job: web
       __path__: ~/prometheus_exporter/app.log
```

Références bibliographiques et webographiques

- [1] Inside, «Améliorer votre infrastructure informatique avec la surveillance ou monitoring IT », consulté le 14 Août 2024. [En ligne] Available : <https://inside-group.fr/expertises/ops-infrastructure/monitoring/>
- [2] Google Maps, «Plan de présentation de l'EPL - Google Maps », consulté le 11 Août 2024. [En ligne] Available : <https://maps.app.goo.gl/HiKpRNAHm631yFEDA>
- [3] logoti, «Lecole polytechnique de lomé ouvre ses portes dès cette rentrée universitaire.»,2022, consulté le 11 Août 2024. [En ligne] Available : <https://logoti.net/actualite/lecole-polytechnique-de-lome-ouvre-ces-portes-des-cette-rentree-universitaire/>.
- [4] BlueBearsIT, «Qu'est-ce que l'infrastructure informatique d'une entreprise ? », consulté le 17 Août 2024. [En ligne] Available : <https://bluebear-sit.com/infrastructure-informatique/>.
- [5] Almeria, «Infrastructures et réseaux », consulté le 17 Août 2024. [En ligne] Available : <https://www.almeria.fr/infrastructure-informatique/>
- [6] Amazon Web Services, Inc, «Qu'est-ce que l'infrastructure informatique ? », 2022, consulté le 16 Août 2024. [En ligne] Available : <https://aws.amazon.com/fr/what-is/it-infrastructure/>.
- [7] New Relic, «Quest-ce que le monitoring dinfrastructure ? Bonnes pratiques et cas d'utilisation », consulté le 20 Août 2024. [En ligne] Available : <https://newrelic.com/fr/blog/best-practices/what-is-infrastructure-monitoring>
- [8] J. Gabès, «Intérêt de la supervision et de la métrologie », dans *Nagios 3 pour la supervision et la métrologie : Déploiement, configuration et optimisation*, Éditions Eyrolles, 2009, chapitre 1.
- [9] J. Gabès, «Grandes lignes de létude et de la mise en place d'une solution de supervision », dans *Nagios 3 pour la supervision et la métrologie : Déploiement, configuration et optimisation*, Éditions Eyrolles, 2009, chapitre 2.
- [10] J. Gabès, *Choix d'une solution de supervision : atouts et fonctionnement de Nagios*, dans *Nagios 3 pour la supervision et la métrologie : Déploiement, configuration et optimisation*, Éditions Eyrolles, 2009, chapitre 3.
- [11] elastic, «Elastic Stack features », consulté le 20 Août 2024. [En ligne] Available : <https://www.elastic.co/elastic-stack/features>

- [12] Prometheus Authors, «*Documentation officielle de Prometheus* », 2022, consulté le 16 Août 2024. [En ligne] Available : <https://prometheus.io/docs/introduction/overview/>.
- [13] Prometheus Authors, «*Prometheus Architecture* », consulté le 20 Août 2024. [En ligne] Available : <https://prometheus.io/docs/introduction/overview/#architecture>
- [14] Grafana Labs, «*Documentation officielle de Loki* », 2024, consulté le 16 Août 2024. [En ligne] Available : <https://grafana.com/docs/loki/latest/>.
- [15] Grafana labs, «*Loki architecture* », consulté le 20 Août 2024. [En ligne] Available : <https://grafana.com/docs/loki/latest/get-started/architecture/>
- [16] Grafana Labs, «*Documentation officielle de Grafana* », 2024, consulté le 16 Août 2024. [En ligne] Available : <https://grafana.com/docs/grafana/latest/>.
- [17] Oracle Corporation VirtualBox, «*Download VirtualBox for Linux Hosts* », consulté le 26 Août 2024. [En ligne] Available : https://www.virtualbox.org/wiki/Linux_Downloads.
- [18] Q.-H. Tran, «*Architecture pour les analyses distribuées et parallèles de traces logicielles* », Thèse de doctorat, Polytechnique Montréal, 2022. consulté le 16 Août 2024. [En ligne] Available : https://publications.polymtl.ca/10481/1/2022_QuocHaoTran.pdf