Ecole Nationale des
Sciences Appliquées Safi
Département Génie Informatique et
Télécommunications et Réseaux

N° d'ordre : …/19

Projet de Fin d'Etudes

En vue de l'obtention du diplôme
D'Ingénieur d'Etat en Génie Télécommunications et Réseaux

Monitoring à la demande pour la gestion de la QoS dans l'IoT

Effectué par

SAFRI Hamza

Soutenu le 26 Septembre 2019 Devant le jury :

M. ZYANE Abdellah	Encadrant	ENSA de Safi
M. BENLAMKADEM Abdellatif	Rapporteur	ENSA de Safi
Mme. MESTOURI Hind	Rapporteur	ENSA de Safi

M. Christophe CHASSOT, encadrant de LAAS-CNRS

Année Universitaire : 2018/2019

Dédicaces

Remerciements

Résumé

Les concepts d'objets connectés et d'Internet des objets (IoT) permettent aujourd'hui d'envisager le déploiement de nouvelles activités basées sur les capacités de communication d'objets communicants hétérogènes (capteurs, actionneurs, tags RFID, etc.).

Ce nouveau contexte repose le besoin en architectures et en protocoles de communication, en particulier au niveau des plateformes de service (niveau middleware) de l'IoT. Parallèlement émergent de nouveaux concepts et de nouvelles technologies liés d'une part au Cloud Computing et à ses dérivés (Fog et Edge) via la virtualisation de fonctions réseau (NFV) et d'autre part aux réseaux guidés par le logiciel (SDN). Ces technologies permettent d'une part de décolérer l'implantation des fonctions de communication des technologies propriétaires liées aux équipements sous-jacents, et d'autre part d'envisager le provisionnement dynamique d'infrastructures de communication guidé par le logiciel (SDCI, Software-Defined Communication Infrastructure).

Partant d'un travail mené au LAAS-CNRS sur la conception et le développement d'un système de gestion auto-adaptative de la qualité de service (QoS) pour les futures plateformes de service de l'IoT, le sujet du stage consiste à concevoir, développer et expérimenter des agents de monitoring et de surveillance en support du déploiement des futures SDCI.

Mots Clés : IoT, Monitoring Agent, NFV, SDN, VNF, ANF.

Abstract

The concepts of connected objects and Internet of Things (IoT) allow today to consider the deployment of new activities based on the communication capabilities of heterogeneous communicating objects (sensors, actuators, RFID tags, etc.).

These new contexts are based on the need for architectures and communication protocols, particularly at IoT's middleware level. At the same time, new concepts and technologies are emerging, related to Cloud Computing and its derivatives (Fog and Edge) via Network Function Virtualization (NFV) and Software-Driven Networks (SDN). These technologies make it possible, on the one hand, to decolorize the implementation of the communication functions of the proprietary technologies related to the underlying equipment, and on the other hand to consider the dynamic provisioning of software-driven communication infrastructures (SDCI, Software -Defined Communication Infrastructure).

In this context, starting from a work conducted at LAAS-CNRS on the design and development of a self-adaptive quality of service (QoS) management system for the future IoT service platforms, the subject of the internship consists in designing, developing and experimenting monitoring and surveillance agents in support of the deployment of future SDCIs

Keywords : IoT, Monitoring Agent, NFV, SDN, VNF, ANF.

Liste des figures

Liste des tableaux

Liste des abréviations

ANF	Applicative Network Function
API	Application Programming Interface
DB	Database
ETSI	European Telecommunications Standards Institute
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
ISG	Industry Specification Group
JVM	Java Virtual Machine
M2M	Machine-to-Machine
MM	Monitoring Manager
NFV	Network Function Virtualization
OS	Operating System
OSGI	Open Services Gateway initiative
NF	Network Function
QoS	Quality of Service
REST	Representational State Transfer
SARA	Services et Architectures pour Réseaux Avancés
SDCI	Software Defined Communication Infrastructure
SDN	Software Defined Networking
VM	Virtual Machine
VNF	Virtualized Network Function

Table des matières

Introduction générale

L'Internet de objets (Internet Of Thing - IoT) permet aujourd'hui l'échange d'informations provenant de dispositifs connectés présents dans le monde réel vers le réseau Internet. À un proche horizon, des milliards "d'objets connectés" seront ainsi reliés à l'Internet. Dans ce contexte, les réseaux actuels supports de l'Internet montrent des faiblesses notamment quant à leur capacité à supporter le trafic issu de ces objets.

Le concept de réseau défini par le logiciel (Software Defined Networking - SDN) est aujourd'hui très étudié pour introduire une plus forte dynamicité dans la configuration de ces réseaux SDN offre une flexibilité et une programmabilité dans le réseau sans troubler l'architecture sous-jacente de l'existant. Il permet la séparation des fonctions de contrôle et de transfert des données du réseau.

Parallèlement, l'évolution des technologies liées à la virtualisation ont conduit à l'émergence d'un nouveau concept, celui de virtualisation de fonctions réseau, par le biais notamment de la technologie NFV... Ce concept consiste à virtualiser les services réseaux à les déployer sur sous forme de logiciels sur des serveurs génériques, plutôt que sur des matériels dédiés. Ces logiciels portent le nom de *fonctions de réseau virtuelles* (VNF), déployées dans des machines virtuelles ou des conteneurs type Docker.NFV renforce d'une part l'agilité et la flexibilité du réseau aux côtés de SDN, et d'autre part réduit la maintenance et les coûts de matériels. Notons que le concept de fonction de réseau peut également se décliner sous d'autres formes que "virtuelle", notamment sous la forme de module applicatif intégrable dynamiquement dans des architectures logicielles basées composants, développées au moyen de technologies telles que OSGi. Dans la suite de ce mémoire, nous distinguerons ainsi deux types de fonctions de réseau ainsi "dématérialisées" : les VNF (*Virtual Network Function*) et les ANF (*Application Network Function*).

Quel que soit leur format (V- ou A-), le déploiement de fonction de réseau nécessite la disponibilité de capacités et de ressources sur les hôtes amenés à les héberger.

Un monitoring de ces ressources est ainsi nécessaire pour assurer le bon fonctionnement et la disponibilité des fonctions réseau (virtualisées ou applicatives)., Pour autant, les solutions disponibles en réponse à ce besoin sont encore insuffisantes, car non destinées à ces technologies, ou bien dépendant de l'infrastructure NFV (**NFVI**). Notons enfin que les

solutions propriétaires ne considèrent pas l'évolution du contexte et des besoins (Qosmos et Zenoss).

Face à ces limites, ce travail vise à présenter une approche de monitoring adaptée à la surveillance des VNFs et des ANFs.

Ce mémoire s'organise en quatre chapitres :

- Dans le premier chapitre, nous présentons tout d'abord le contexte et la problématique du stage. La deuxième partie du chapitre est dédiée aux objectifs de la solution visée et au cahier de charge du stage.

- Le deuxième chapitre est consacré à un état de l'art sur le concept de NFV. Nous décrivons tout d'abord les différentes architectures existantes. Ensuite, nous nous focalisons sur le monitoring et les solutions logicielles existantes. Enfin nous présentons la solution de monitoring proposée "Zabbix".

- La solution de monitoring à développer et sa conception sont exposées dans le chapitre 4. Celui-ci présente une approche de monitoring adaptée aux fonctions réseau dématérialisées (ANF/VNF) en commençant par une description générale du système de monitoring avant de détailler ses composants.

- Le chapitre 5 se focalise dans sa première partie aux principaux outils d'ingénierie logiciel utilisés pour la mise en œuvre de la solution. Sa deuxième partie est dédiée aux tests fonctionnels pour la validation de l'implémentation logicielle.

- Une conclusion générale sur le travail réalisé clôt finalement le mémoire.

CHAPITRE 1

CONTEXTE, PROBLÉMATIQUE ET CAHIER DE CHARGES

1. CONTEXTE GENERAL
2. CONTEXT SPÉCIFIQUE ET PROBLÉMATIQUE
3. OBJECTIF ET CAHIER DE CHARGE

Introduction

Ce chapitre présente une vue globale de la problématique générale de ce stage. Nous commençons par fournir une vue d'ensemble sur le cadre du projet avant de détailler la problématique de monitoring des ***fonctions réseau virtualisées*** **(VNF)** et ***fonctions réseau applicatives (ANF).***

La dernière partie de ce chapitre décrit le cahier des charges associé à la réalisation des travaux de ce stage.

I. CONTEXTE GENERAL

Le concept d'Internet des Objets (IoT : Internet of Things) a connu une évolution rapide durant ces dernières années. Ce concept est basé sur la capacité de communication de tous les objets qui nous entourent, étendant ainsi le modèle de l'Internet actuel ; dans ce contexte les réseaux actuels montrent encore davantage leur faiblesse car d'une part, ils ne pourront pas supporter le nombre des dispositifs connectés et la communication de nature hétérogène entre eux. D'autre part, la complexité du contrôle des données massives générées par ces derniers, alors ces limitations reposent encore le besoin des architectures et protocoles de communication adaptées à ces nouvelles activités.

Parallèlement à cette évolution, un ensemble de technologies et concepts sont aussi développés, liés notamment au cloud et ses dérivées (Fog, Edge) pour stocker et traiter les données générées par un dispositif connecté donné en temps réel et le plus proche possible de ce dernier, ce qui réduit la latence de transfert des données.

La virtualisation des fonctions réseau (NFV) est aussi un paradigme rapidement développé dans ces dernières années, il intègre les technologies de cloud et de virtualisation pour virtualiser les services réseaux et les déployer sous la forme d'une machine virtuelle (VM) ou d'un conteneur système (OS-CNT). Ces déploiements sont connus sous le nom des fonctions réseau virtuelles (Virtual Network Functions -VNF) ou dans des environnements modulaires sous forme d'un composant modulaire ; e.g. un bundle dans la plateforme OSGI (Open Services Gateway initiative ; portant le nom Fonctions Réseau Applicatives (Applicative Network Functions ANF).

La combinaison de ces technologies permet d'une part, de se débarrasser des matériels dédiés pour faire tourner des fonctions réseaux, et d'autre part, d'adresser les défis d'allocation automatique de ressources pour l'infrastructure de communication afin d'atteindre les niveaux attendus de la qualité de service (QoS).

Dans ce cadre, l'équipe SARA du département Réseau et Communication (RC) au sein du Laboratoire LAAS-CNRS (Annexe A) est l'une des équipes qui s'intéresse aux problématiques majeures traitant les différents aspects relatifs aux futures architectures de communication, a lancé le projet Software Defined Communication Infrastructure (SDCI) [1] où elle a proposé une approche consistant à concevoir, développer et expérimenter des modèles d'architectures

logicielles génériques pour une gestion auto adaptative de la QoS aux différents niveaux du système de communication, en :

- Tirant partie des opportunités technologiques liées au déploiement dynamique de fonctions de réseau, des réseaux programmables et de l'Autonomic Computing ;
- Tenant compte de l'hétérogénéité des solutions en cours de déploiement ;
- Assurant la cohérence des choix de configuration effectués aux différents niveaux Architecture M2M (Annex B) par le biais d'outils théoriques adéquats.

II. CONTEXTE SPÉCIFIQUE ET PROBLÉMATIQUE

Le Cloud et le NFV représentent les concepts clés pour provisionner les besoins de l'infrastructure de communication et redimensionner les réseaux afin de maintenir les niveaux satisfaisants de la QoS. Cependant, pour atteindre cet objectif, la surveillance du fonctionnement des fonctions réseau (Network Functions - FNs) sous ses différentes formes ANF/VNF est nécessaire pour avoir une vue d'ensemble sur les besoins de ces NFs.

A première vue, la mise en place d'une solution de monitoring de l'infrastructure de communication (ANF/VNF) est donc un volet prioritaire dans la disponibilité de ces services, et il ne fait pas de doute qu'il existe plusieurs solutions pour monitorer les VNFs mais ils ne sont pas adaptées aux ANFs, de plus, la plupart de ces solutions proposées sont propriétaires et exigent un ensemble de contraintes, et même les solutions libres (Open Source); dans la plupart des cas; sont déployables au niveau de l'infrastructure du cloud qui ne sera pas toujours accessible à tout le monde.

III. OBJECTIF ET CAHIER DE CHARGE

1. Objectif

Contribuant à un travail mené au LAAS-CNRS sur la conception et le développement d'un système de gestion auto-adaptative de la qualité de service au niveau des plateforme IoT, l'objectif du stage consiste à concevoir, développer un système de monitoring adapté à la fois pour les VNFs et ANFs afin de collecter un ensemble d'informations et les analyser pour répondre aux besoins des NFs en termes des ressources et pouvoir répondre aux exigences de la QoS.

2. Cahier Des Charges

Le cahier des charges de ce stage reposait sur les points suivants :

- Assimiler les concepts et les techniques liés au monitoring passif et actif, ainsi qu'aux standards ETSI-NFV [2] et l'OSGi,
- Concevoir et implémenter des agents logiciels adaptés au monitoring des fonctions réseau virtualisées,
- Concevoir et implémenter un Dashboard interactif exposant les différentes métriques collectées par les agents de monitoring,
- Intégrer au Dashboard la possibilité de supporter l'ajout dynamique de fonction de monitoring adaptée sur un hôte donné. Cet ajout dynamique consistera en :
    - Le choix automatique d'un agent adapté à l'hôte ciblé,
    - Le déploiement d'un agent.

Ces grandes parties ont été divisées en tâches sur la période de stage de la façon suivante :

Tableau 1. 1:Ordonnancement des taches

N°	Tâche	Début	Fin	Durée
1	Documentation	4 mars 2019	29 mars 2019	25 jours
2	Conception UML (cas d'utilisation, diagramme de composites, diagramme de séquences)	1 avril 2019	15 mai 2019	45 jours
3	Implémentation	16 mai 2019	9 aout 2019	84 jours
4	Tests	12 aout 2019	31 aout 2019	19 jours

IV. CONCLUSION

Dans ce chapitre, nous avons présenté le contexte général de stage qui combine les concepts généraux qui constituent le cadre du projet, puis nous avons énoncé la problématique de monitoring des VNFs et ANFs et finalement nous avons exposé l'objectif et le cahier des charges de ce stage. Le chapitre suivant présente un état de l'art sur la virtualisation des

fonctions réseau, les solutions de monitoring actuellement utilisées, ainsi qu'une analyse des métriques de décision adaptés aux VNFs et ANFs.

CHAPITRE 2
ETAT DE L'ART

1. **MOTIVATION**
2. **CONCEPT DE NFV**
3. **MONITORING**
4. **SOLUTION LOGICIELLES DE MONITORING**

Introduction

Une fois que nous avons défini le contexte de notre projet et cerné la problématique, intervient alors l'étape de l'état de l'art. Dans ce chapitre, nous présentons les différents aspects liés au concept de virtualisation des fonctions réseau (NFV) et le monitoring. La première section est consacrée à la présentation du concept de la virtualisation au sens large, et au paradigme de NFV. La deuxième section présente la notion de monitoring d'une manière globale. Cette section se décline en plusieurs sous sections portant sur : la définition du monitoring, le modèle générique du monitoring, les types de monitoring, les agents de monitoring associés à ces types, et finalement une présentation des solutions logicielles de monitoring.

I. LE CONCEPT DE LA VIRTUALISATION DE FONCTIONS DE RÉSEAU

1. Motivation

Le concept de NFV consiste à virtualiser les fonctions réseaux (telles que le routage, le firewall, etc.) pour se débarrasser des matériels dédiés et les déployer dans des serveurs standard [3]. Ces fonctions peuvent être instanciées, configurées, déplacées, etc. dans divers endroits du réseau selon les besoins des opérateurs, et sans nécessité d'installer de nouveaux équipements. La séparation des fonctions du matériel apporte de nombreux avantages pour l'opérateur réseau, parmi lesquels :

- ❏ Réduction de l'espace nécessaire aux équipements matériels du réseau.
- ❏ Réduction de la consommation électrique du réseau.
- ❏ Réduction des coûts de maintenance du réseau.
- ❏ Simplification des mises à niveau du réseau.
- ❏ Allongement du cycle de vie des équipements matériels du réseau.
- ❏ Réduction de la maintenance et des coûts matériels.

En référence à la problématique expliquée au Chapitre 1, un état de l'art de ce nouveau concept est donc nécessaire pour en connaître ses caractéristiques et ses composants pour bien concevoir une solution la mieux adaptée à cette problématique.

2. Définitions et Principes

2.1. Virtualisation

2.1.1. Principe

La virtualisation est une technologie qui permet de créer des services informatiques utiles à l'aide de ressources qui sont généralement liées au matériel. Elle permet d'exploiter toute la capacité d'une machine physique en la répartissant entre de nombreux utilisateurs ou environnements différents grâce à une couche d'abstraction qu'on l'appelle moniteur de machine virtuelle (VMM) ou hyperviseur.

2.1.2. Classification

On peut distinguer plusieurs formes de virtualisation :

- **Virtualisation complète (Full Virtualization) :** dans cette approche, VMM simule complètement le matériel sous-jacent, ce qui permet de faire fonctionner n'importe quel système d'exploitation en tant qu'invité dans une machine virtuelle.

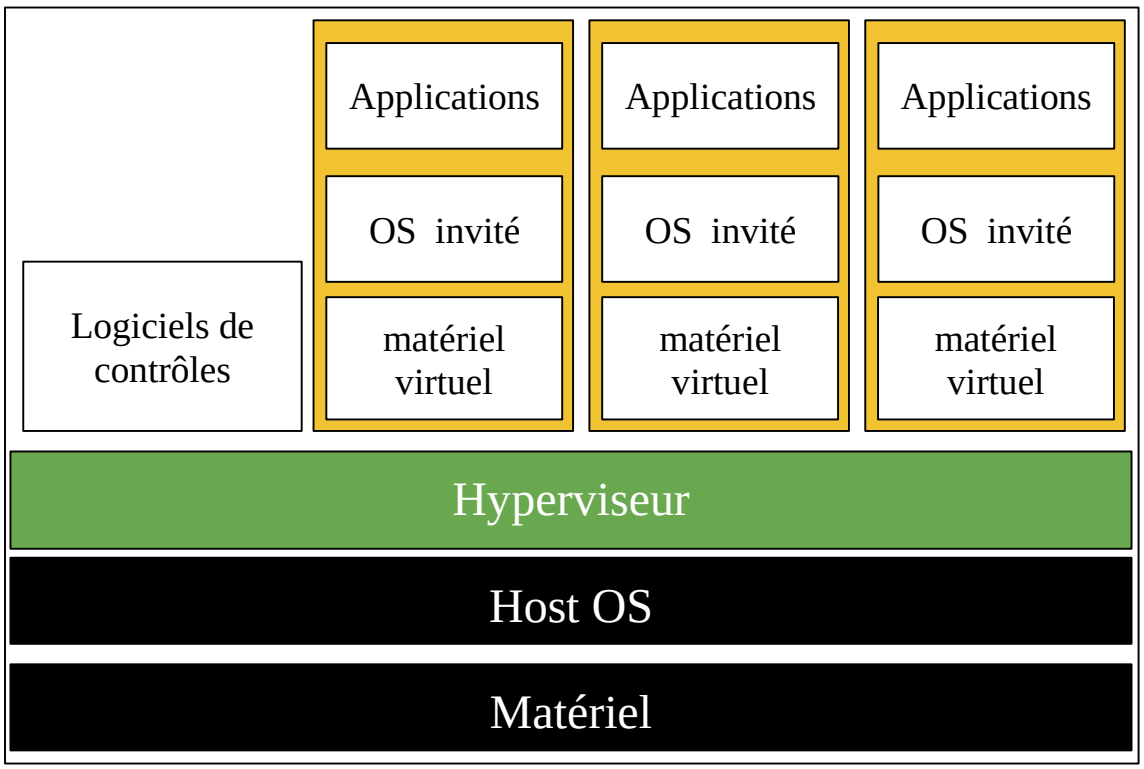


Figure 2. 1:Architecture de la virtualisation complète

- **Virtualisation au niveau du système d'exploitation**
- **(OS-Layer Virtualization) :** Aussi connue sous le nom de SKI (Single Kernel Image) ou de virtualisation par “conteneur” (Container – CNT), ce concept implémente la virtualisation en exécutant plusieurs instances du même système d'exploitation en parallèle. Cela signifie que ce n'est pas le matériel, mais le système d'exploitation de l’hôte qui est virtualisé. Cette approche est illustrée dans la Figure 2.2.

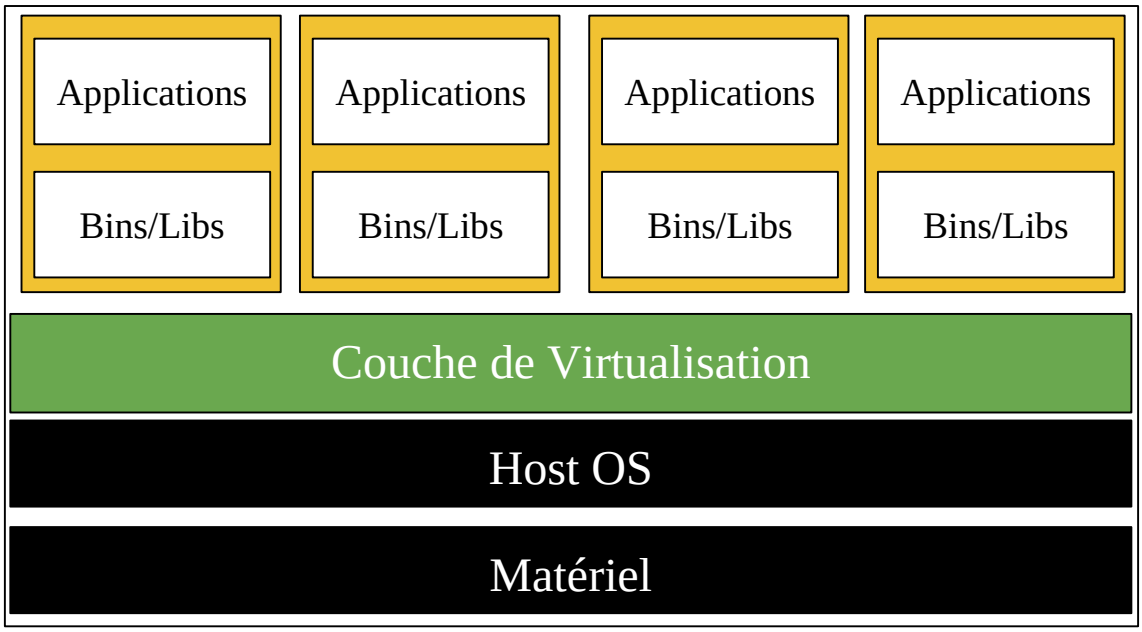


Figure 2. 2:Architecture de la virtualisation la OS-layer

- **Virtualisation de couche matérielle (Hardware-Layer Virtualization) :** parfois appelée virtualisation de plate-forme Ici, le VMM s'exécute sur une plate-forme matérielle compatible afin de cacher le matériel physique et de créer un environnement informatique simulé pour le logiciel invité, qu'il s'agisse d'applications utilisateur ou de systèmes d'exploitation complets. La Figure 2.3 décrit cette architecture.

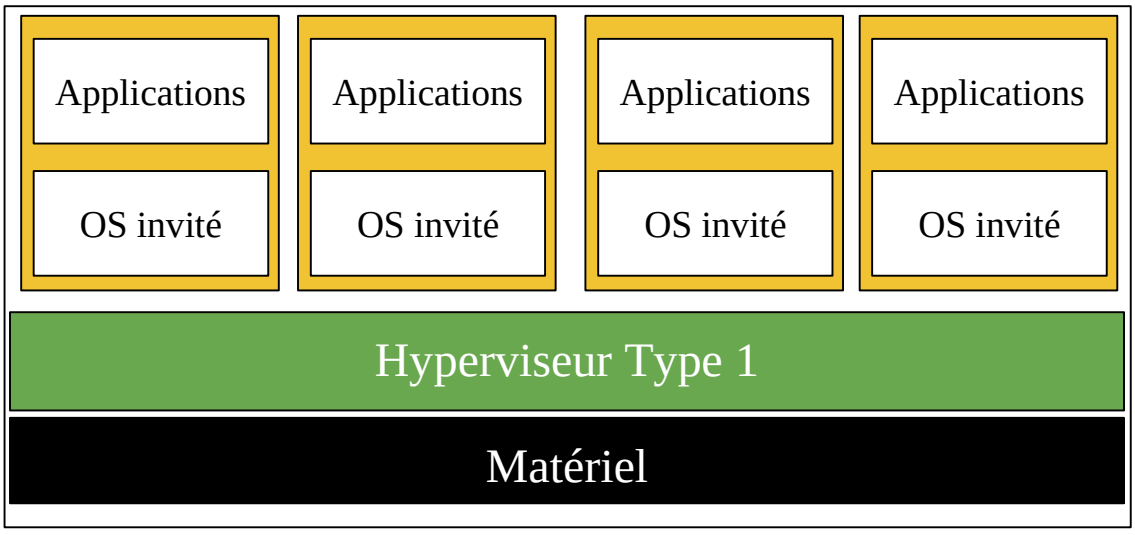


Figure 2. 3:Architecture de la virtualisation de la couche matérielle

- **Para-virtualisation:** La virtualisation en parallèle (ou para-virtualisation) fournit une simulation partielle du matériel sous-jacent. Dans la plupart des cas, les composants

matériels sont simulés. Il existe une interface logicielle entre le matériel hôte et le système d'exploitation invité modifié. Un point remarquable est que les machines invitées sont conscientes du fait qu'elles fonctionnent dans un environnement virtualisé [4]. Cette technique est utilisée par la plupart des hyperviseurs comme Proxmox et Xen [5].

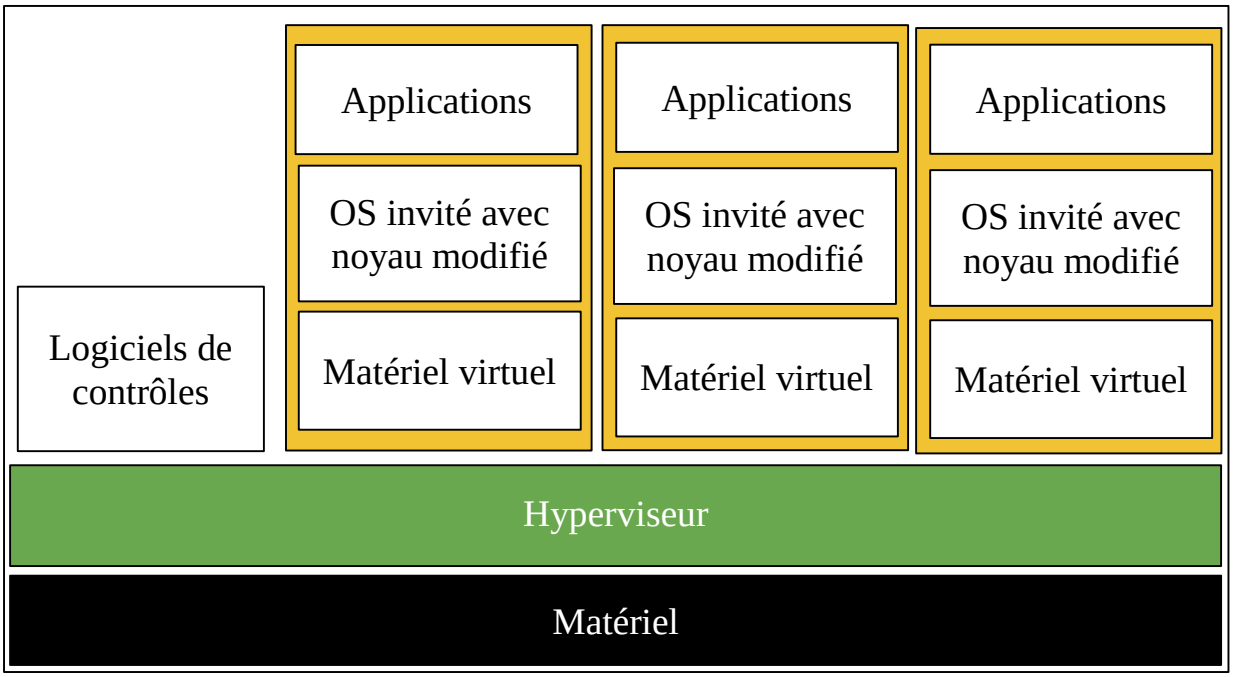


Figure 2. 4:Architecture de la para-virtualisation

- **Virtualisation d'applications (Application Virtualization) :** Dans la virtualisation d'applications, l'utilisateur peut exécuter une application serveur à l'aide des ressources locales sans avoir à recourir à la complexité nécessaire pour installer complètement cette application sur son ordinateur. Ces applications virtualisées sont conçues pour s'exécuter dans un petit environnement virtuel contenant les seules ressources nécessaires à l'exécution de l'application. Ainsi, dans la virtualisation d'applications, chaque utilisateur dispose virtuellement d'un environnement d'application isolé. Ce petit environnement virtuel isolé agit comme une couche entre l'application et le système d'exploitation hôte [4].

2.2. Service et Fonction Réseau

Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations. Ces équipements implémentent une logique pour exécuter une fonction bien

particulière. Cette logique est appelée “fonction réseau” ; la totalité des fonctions ainsi que la relation entre eux est appelée “service réseau”.

2.3. La virtualisation des fonctions de réseau (NFV)

2.3.1. Définition

La virtualisation des fonctions de réseau (NFV) est une approche apparue fin 2012 dans un Livre Blanc [6] cosigné par 13 opérateurs, dont l’objectif est d’extraire les fonctions réseaux des équipements dédiés et de les faire fonctionner dans un environnement virtualisé. Ces fonctions isolées portent le nom des fonctions réseau virtualisées ou Virtual Network Fonctions (VNFs) en anglais.

2.3.2. Les architectures proposées pour NFV

2.3.2.1. Architecture de ETSI-NFV

Le groupe ETSI ISG [10] a comme mission le développement des spécifications concernant le concept NFV. En 2013, il a proposé une architecture fonctionnelle pour le déploiement des fonctions réseau illustrées par la Figure 2.5. Cette architecture inclut principalement les blocs fonctionnels suivants (ETSI, 2014) :

- **Bloc Infrastructure NFV :** NFVI (Network Function Virtualisation Infrastructure) fournit les ressources matérielles (serveurs, COTS – Commercial Off The Sheld, cartes électroniques, …) et le logiciel de virtualisation. Le NFVI se compose donc :
  - ❑ D’une interface matérielle (stockage, réseau, calcul)
  - ❑ D’une interface virtuelle (stockage, réseau, calcul)
  - ❑ D’une couche de virtualisation entre le matériel et le logiciel
- **Bloc VNF (Virtualized Network Function**) : correspond aux fonctions réseaux indépendamment de leur langage de programmation et leur package.
- **NFV M&O (Management and Orchestration)** permettant de gérer les services réseaux de bout en bout. Il est composé de trois blocs
  - ❑ L’orchestrateur (NFV Orchestrator) : L’entité d’orchestration est responsable du cycle de vie des services réseau tant au niveau logiciel

que matériel sur plusieurs domaines en contrôlant les VIM de chaque domaine ;

- ❏ Un gestionnaire (VNFM) en charge du cycle de vie des VNFs ;
- ❏ Un gestionnaire (VIM) en charge de la gestion des ressources du NFVI (Allocation des ressources pour les VNFs, le cheminement de flux de donnés entres les différents blocs)

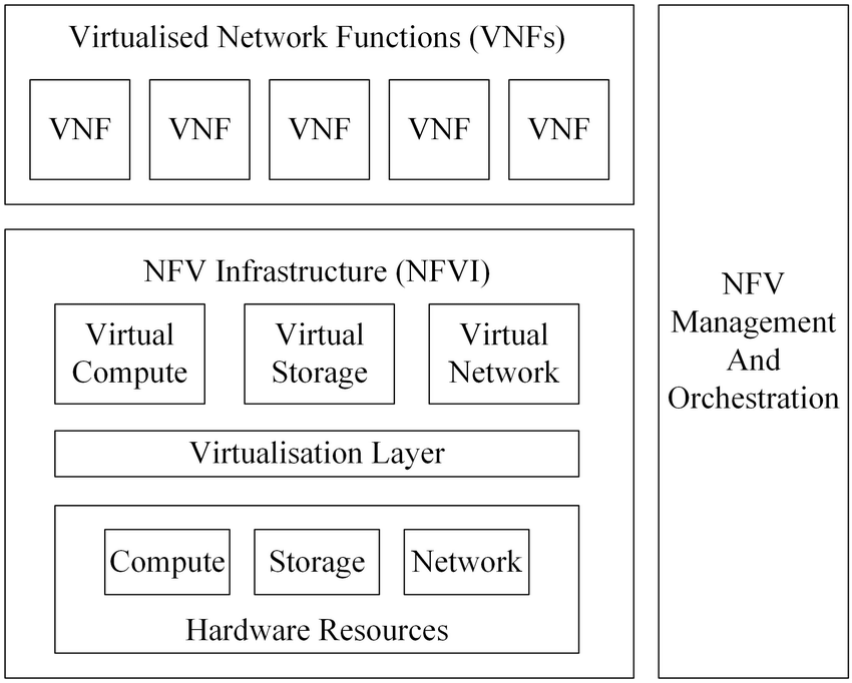


Figure 2. 5:Architecture fonctionnelle ETSI-NFV [9]

2.3.2.2. Architecture d'OpenNetVM

OpenNetVM [7] est une plate-forme NFV hautes performances de déploiement et d'exécution de fonctions réseaux virtualisées basée sur les conteneurs DPDK et Docker. OpenNetVM peut être compatible SDN, ce qui permet au contrôleur de réseau de fournir des règles qui dictent les fonctions réseau nécessaires pour traiter chaque flux de paquets.

Cette solution propose une architecture inclut 3 blocs principaux (cf. Figure 2.6) :

- **NF Manager :** Ce composant garde la trace des fonctions réseau actuellement exécutées dans les conteneurs et leur distribue les paquets dès leur arrivée.
- **SDN-enabled :** Le gestionnaire de NF est coordonné avec les contrôleurs SDN utilisant OpenFlow, ce qui permet au contrôleur de spécifier des chaînes de service composées de plusieurs NF qui doivent traiter un flux.

- **Mémoire partagée:** Les paquets sont DMA (en anglais **DMA** pour Direct **Memory** Access) i.e. directement dans une région de mémoire partagée qui permet au gestionnaire de NF d'accorder un accès direct aux NF aux paquets sans aucune copie supplémentaire **(Zero-Copy IO)**.

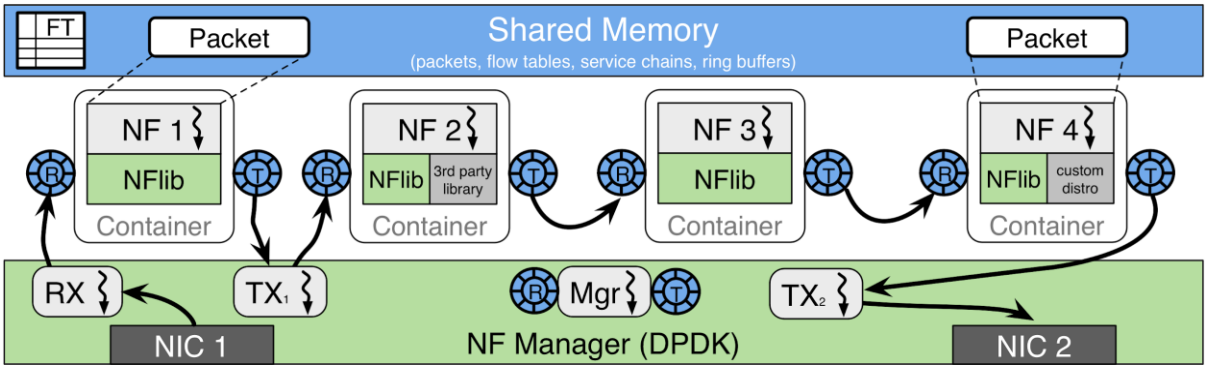


Figure 2. 6: Architecture fonctionnelle OpenNetVM [7]

2.3.3. Conteneurs de Virtualisations

2.3.3.1. Définition

Dans l'architecture NFV proposée par le groupe de travail ETSI-NFV (section 2.3.2.1), nous avons identifié qu'une fonction réseau virtualisée (VNF) est un code logiciel capable de fonctionner sur une infrastructure NFV (NFVI) et d'être orchestré par l'Orchestrateur NFV (NFVO) et le gestionnaire VNF (VNFM) [8]. Cependant, une fonction réseau ne peut pas fonctionner directement sur NFVI ; pour cela on l'héberge dans un conteneur qu'on l'appelle "**Conteneur de Virtualisation",** d'où l'origine de <<**V**>> dans l'acronyme <<**VNF**>>**.**

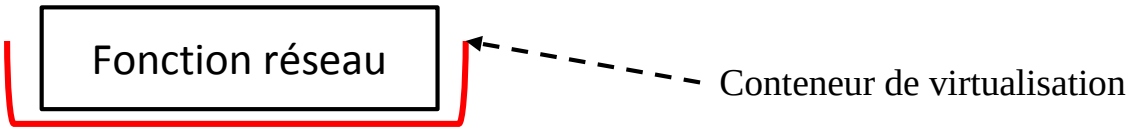


Figure 2. 7:architecture d'une VNF

Un Conteneur de Virtualisation est défini par le groupe ETSI [2] comme étant une partition d'un nœud de calcul qui fournit un environnement de calcul virtualisé isolé, De cela, nous pouvons déduire qu'une VNF est composée (Figure 2.7) d'une fonction réseau et d'un Conteneur de Virtualisation qui représente l'environnement virtuel d'exécution pour cette fonction.

2.3.3.2. Type des Conteneurs de Virtualisation

Le cadre architectural ETSI-NFV décrit dans la norme ETSI GS NFV 002 [9] identifie une couche de virtualisation, mais il ne se limite pas à l'utilisation d'une solution spécifique [11]. Dans ces conditions, il existe plusieurs techniques pour réaliser la virtualisation.

Dans cette optique, nous pouvons définir deux types de conteneurs de virtualisation :

- **Machine virtuelle (VM) :** une VM est un environnement de calcul virtualisé, qui simule un ordinateur/serveur physique avec tous ses composants (processeur, mémoire / stockage, interfaces / ports). Un hyperviseur (KVM par exemple) partitionne les ressources physiques sous-jacentes et les alloue à une machine virtuelle pour exécuter une VNF (cf. Figure 2.8).

- **Conteneur (CNT)** : un CTN est un environnement virtuel résultant d'une isolation d'un ensemble des processus et des ressources (CPU, mémoire, réseau, etc.) afin d'exécuter séparément une tâche bien spécifique. Ces conteneurs sont générés par "**Container Engine - CE "**, alors ces conteneurs peuvent héberger des fonctions réseau et les exécuter (Figure 2.8).

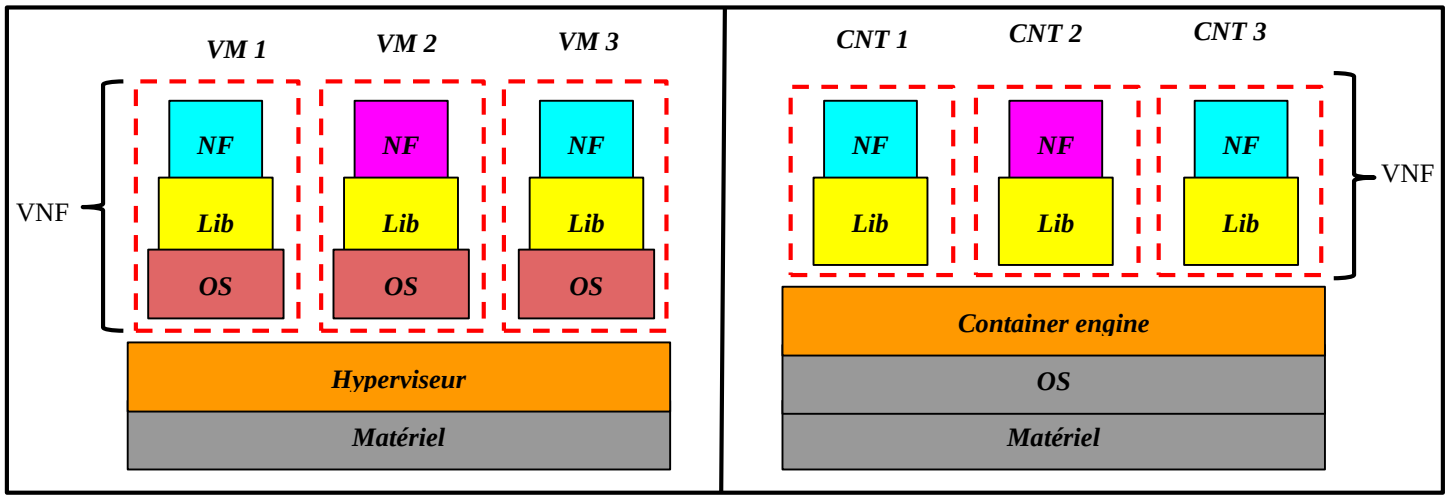


Figure 2. 8:Conteneur de virtualisation des VNFs

La fonction réseau est identique, que son "packaging" soit une machine virtuelle ou un conteneur, mais structurellement, ces deux environnements sont différents. Contrairement à la virtualisation complète, un conteneur n'embarque pas d'OS, il s'appuie directement sur le système d'exploitation (OS) sous-jacent, ce qui le rend plus léger qu'une VM qui intégrera un OS et qui exige plus de ressources (RAM, CPU et DISQUE).

2.3.4. Applicative Network Function (ANF)

Il est communément admis [11] qu'il existe deux types de fonctions réseaux, celles qui s'exécutent sur un matériel dédié (PNF pour Physical Network Functions), et celles qui s'exécutent dans des conteneurs de virtualisation (VNF pour Virtualized Network Functions). Cependant, ces deux types ne sont pas les seuls types de fonctions réseau envisageables, car il y en a une autre forme, plus légère car n'exigeant pas des ressources virtuelles, portant le nom "fonction réseau applicative" (ANF pour Applicative Network Function).

Ces ANFs sont des fonctions réseau déployables sous forme de logiciel sur des plateformes compatibles à une structure modulaire installée sur un matériel standard (Figure 2.9), ce qui étend la définition d'un conteneur de virtualisation. Un exemple consiste en le déploiement d'un mécanisme de gestion de la QoS (un différentiateur de paquets par exemple) dans un environnement non virtuel tel qu'une plateforme OSGi.

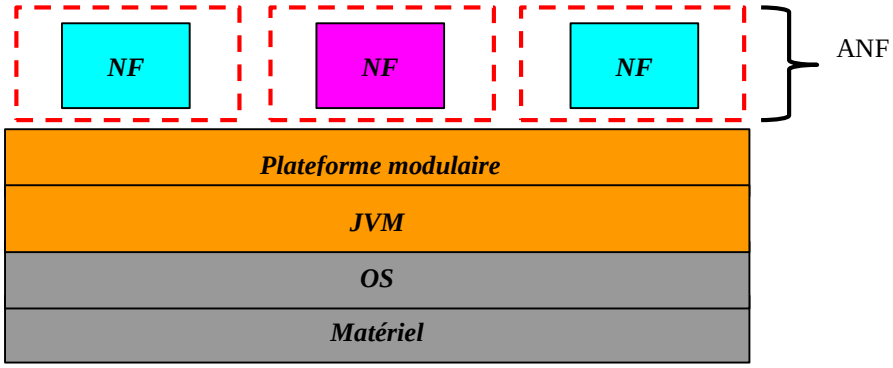


Figure 2. 9: conteneur virtualisation des ANFs

2.4. Monitoring

Le terme de "Monitoring", couramment utilisé, provient de l'anglais et signifie "supervision". Cependant une précision est à apporter, en effet ceci est un abus de langage, car en vérité le Monitoring est composé de deux disciplines : la métrologie et la supervision. Pour définir le monitoring, il est nécessaire de définir ces deux termes.

2.4.1. Définitions

2.4.1.1. Supervision

La supervision d'un système est une fonction qui consiste à récupérer l'état d'un système et de remonter une alerte sur la détection d'un comportement anormal sans connaissance de la cause [12] [13] [14].

2.4.1.2. Métrologie

La métrologie est le processus consistant à “historiser” les données collectées sous forme numérique et à les soumettre à divers types de traitement avant de les visualiser sous forme des graphes ou autres [12] [13] [14].

2.4.1.3. Monitoring

Le monitoring est un processus consistant à superviser un système et à obtenir les informations d’une manière permanente des différents éléments de ce système, et à les consolider pour les analyser [15] et les tracer. Les informations collectées sont diverses, cependant nous pouvons les classifier en 4 grands groupes [15] :

- **Informations de performance :** ces informations numériques couvrent toutes les couches matérielles comme l’utilisation des ressources (CPU/RAM/DISK) et les performances réseaux (débits, taux d’erreurs etc.).

- **Informations d’état :** ces informations décrivent le niveau de disponibilité du système, car un système à tout moment peut être allumé et fonctionne correctement, comme il ne peut pas être allumé ou être allumé mais ne fonctionne pas correctement.

- **Informations d’erreur :** ces informations incluent des informations sur les erreurs et le fonctionnement incorrect d’un élément dans le système. Les informations d'erreur peuvent inclure des codes d'erreur ainsi que des informations sur le nombre d'erreurs rencontrées au cours d'un intervalle de temps précédent.

- **Informations de configuration :** ces informations décrivent les modifications de configuration non approuvées dans un environnement. Par exemple, surveillance de l'appartenance à un groupe, surveillance du compte de l'ordinateur serveur, surveillance de la sécurité de la base de données SQL, etc.

2.4.2. Modèle générique de monitoring

Le modèle générique de monitoring illustré Figure 2.10 présente les fonctions d’un système de monitoring implémentées dans la plupart des solutions. En premier lieu, le composant de collecte de données extrait les données brutes des divers dispositifs et composants logiciels constituant le système surveillé, et les soumet à divers types de traitement en temps réel avant

de les pré-stocker dans une base de données (DB). Ensuite, un traitement des données pré-stockées est mis en œuvre pour réduire le volume de données afin qu'il soit gérable. De plus, une conversion de format à une donnée **stockable** dans la base de données est également effectuée. La base de données stocke ces données pour les analyser ultérieurement, en utilisant les différents types d'algorithmes de traitement de données, ce qui génère un rapport sur l'état général du système prêt à être présenté à l'administrateur.

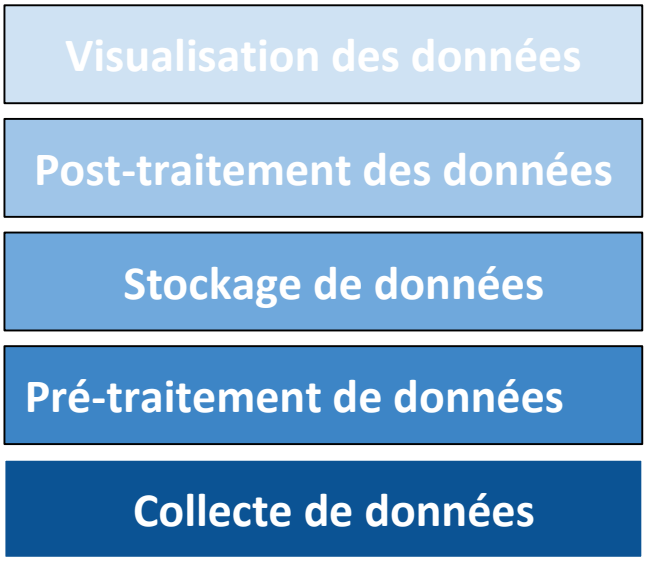


Figure 2. 10:Modèle générique de monitoring

2.4.3. Les types de Monitoring

Le Monitoring est le meilleur moyen de s'assurer du bon fonctionnement d'un système ; c'est également l'une des tâches indispensables pour la gestion de la QoS.
Bien entendu, chaque système a ses propres besoins de supervision, pour cela le choix de type de monitoring est un élément d'influence sur l'efficacité de la supervision. Dans ce but, nous distinguons deux types de monitoring :

- **Monitoring par agent (Agent-based Monitoring) :** ce type consiste à installer des agents (cf. section 2.2.4) logiciels personnalisés dans les éléments surveillés à partir desquels des données doivent être collectées.

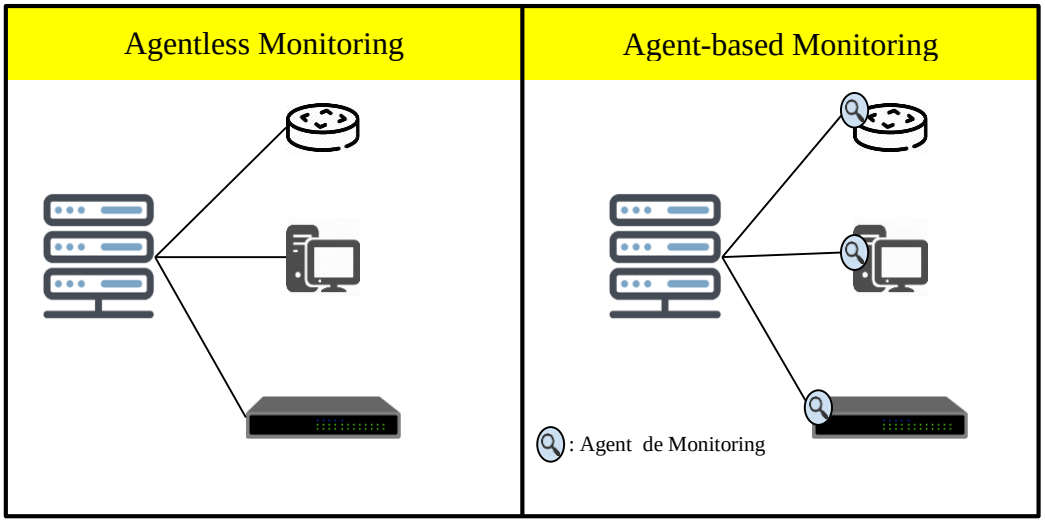


Figure 2. 11:Type de Monitoring

- **Monitoring sans agent (Agentless Monitoring): ce** type de modèle fournit un service centralisé qui interroge en temps réel plusieurs éléments surveillés directement et collecte les données souhaitées sans avoir à installer l'agent logiciel sur ces éléments.

2.4.4. Agent de Monitoring

Un agent de Monitoring est une application résidant dans un périphérique et chargée de collecter un ensemble d'informations sur des métriques (section suivante) prédéfinies.

2.4.5. Type d'agent de Monitoring

Une solution de monitoring utilise différentes méthodes pour récupérer les informations nécessaires sur un élément supervisé ; cependant, pour un monitoring basé sur les agents, il existe deux modes possibles :

- Mode passif : un agent qui envoie à intervalles réguliers (ou non) les métriques et messages vers le serveur de supervision.

- Mode actif : le serveur de supervision qui interroge à intervalles réguliers un agent qui réside au niveau du périphérique surveillé.

Cependant, dans la suite de ce mémoire nous définissons ces deux modes comme suit :

- Mode passif : un agent de monitoring qui envoie à intervalles **réguliers** les métriques et messages d'avertissements vers le serveur de supervision.

- Mode actif : un élément externe (serveur de supervision ou autre) qui interroge les agents pour envoyer les métriques.

2.4.6. La notion de métrique

2.4.6.1. Définition

Une métrique est un attribut mesuré sur un système qui permet de mieux le décrire, et de mieux le caractériser.

2.4.6.2. Classifications

Un système informatique se compose de nombreux composants et appareils, caractérisables par de nombreuses métriques. Nous distinguons trois catégories des métriques :

- Métriques de système : ces métriques sont liées directement à l'infrastructure qui se compose de différentes ressources matérielles (CPU, RAM, DISK).

- Métriques applicatives : ces métriques mesurent l'expérience utilisateur ainsi que l'état global du service (latence, quantité de réponse par unité de temps).

- Les événements : cette catégorie comporte les fichiers journaux (fichiers "log") permettant de diagnostiquer certains problèmes.

2.4.6.3. Les métriques de base

Une tâche cruciale lors de la définition de l'approche de monitoring est l'identification des métriques, lesquelles doivent être collectées à partir de l'infrastructure virtualisée. Bien que la liste des métriques puisse être assez longue, il est nécessaire, pour des raisons d'évolutivité et d'efficacité, de citer l'ensemble des métriques de monitoring collectables pour les ANFs et les VNFs. Le tableau 1 ci-dessous récapitule une liste de ces mesures, qui sont « génériques » en ce sens qu'elles ne sont pas spécifiques à un type donné. Cette liste est destinée à être mise à jour en permanence tout au long du projet afin de s'aligner sur les capacités techniques et les exigences des composants en développement et des cas d'utilisation mis en œuvre.

Tableau 2. 1:Les métriques de Base

	Métriques	unité	Description
VNF	CPU	%	la quantité de CPU utilisée activement en pourcentage du nombre total de CPU disponibles dans une VM/CNT
	RAM	%	la quantité de RAM utilisée activement en pourcentage du nombre total de RAM disponibles dans une VM/CNT
	DISK usage	%	le pourcentage d'espace disque utilisé du nombre total de disque disponibles dans une VM/CNT
	NETWORK I/O	KB/s	le volume de trafic sur une interface réseau spécifique d'une VM/CNT, y compris le trafic de données externe et interne.
	rx_packets	paquets/s	Paquets reçus par VM/CNT
	tx_packets	paquets/s	Paquets envoyés par VM/CNT
	rx_bytes	%	Octets reçus par VM/CNT
	tx_bytes	%	Octets envoyés par VM/CNT
	io_services_bytes_read	B/s	Nombre d'octets transférés à partir du disque
	io_services_bytes_write	B/s	Nombre d'octets transférés vers disque
	Disk I/O	KB/s	La vitesse de lecture / écriture des périphériques de stockage (tels que les disques durs, les SSD) et des partitions (partitions de disque)
	Connections		les connections ouvertes par port

ANF*	CPU	%	la quantité de CPU utilisée activement par la machine virtuelle Java(JVM) en pourcentage du nombre total de CPU disponibles dans le système et prédéfinie pour la JVM.
	RAM	%	la quantité de RAM utilisée activement par la machine virtuelle Java(JVM) en pourcentage du nombre total de RAM disponibles dans le système et prédéfinie pour la JVM.
	Threads	------	le nombre de threads actifs dans la machine virtuelle Java.

* : Dans ce rapport on se concentre sur les fonctions réseau applicatives qui s'exécutent dans un environnement OSGi. Ce choix a été fait pour être en adéquation avec les mécanismes de gestion de QoS développés dans le cadre d'un projet interne au LAAS suivant la même technologie

2.5. Solutions logicielles de Monitoring

2.5.1. Solutions logicielles de Monitoring

Pour adapter les fonctions réseau à l'évolution de l'environnement d'exécution et garantir que les exigences de qualité de service continuent d'être satisfaites, il est nécessaire d'utiliser un système de surveillance complet capable de traiter l'ensemble des exigences relatives à différents niveaux, notamment l'infrastructure, les réseaux et les applications. Dans cet objectif, le Tableau 2 présente les principales solutions qui sont utilisées pour le monitoring des infrastructures (os, réseaux, applications et matériels), des applications Java, VNFs et les ANFs.

Tableau 2. 2:Tableau Comparatif des solutions existantes

Outils De Monitoring	***Nature***		***Type d'agent de Monitoring***		***Environnements***				
	Outil	Technologie	Actif	Passive	Infrastructure	Java	NFV		Conteneur
							VNF	ANF	
Nagios	X		X	X	X	X			
Zabbix	X		X	X	X	X	X		X
Cacti	X		X	X	X				
Qosmos	X		X	X			X		
Zenoss	X		X	X	X	X	X		
JMX		X				X			
VisualVM	X		X			X			
Jprofiler	X		X			X			
Jconsole	X		X			X			
YourKit	X		X			X			
T-nova	X						X		
Doctor	X						X		
Prediction	X						X		
scout	X		X	X	X	X			X

Cadvisor	X								X

Solution Libre

Dans le tableau précédent, nous avons exposé des solutions et des projets actuellement utilisés pour la surveillance d'un ensemble d'environnements. Parmi ces solutions se trouvent celles qui ont pu surveiller les environnements NFV tel que Qosmos, Zenoss et Zabbix. Cependant, les deux premiers sont propriétaire, alors que Zabbix est la seule solution libre la plus complète proposant une approche de monitoring pour NFV (section 2.5.2).
Bien entendu, nous avons pu, à partir de cet état d'art, trouver des solutions de monitoring pour les VNFs, mais ils ne couvrent pas les ANFs, ce qui montre le besoin d'une solution de monitoring adaptée pour les différents types des fonctions réseau.

2.5.2. ZABBIX : étude d'une solution de monitoring des NFV

Zabbix est une solution de monitoring open source destinée à divers composants informatique notamment infrastructure, services cloud. Cette solution se base sur un ensemble de modules (Template) développés par zabbix et sont fournis avec elle, tandis que d'autre sont construits par les utilisateurs zabbix, par exemple le module de monitoring pour le container Docker [16]. La possibilité d'importer des modules personnalisés lui a permis d'être le premier choix non seulement pour les administrateurs mais aussi pour d'autre projet comme "**Open Baton Zabbix Plugin".**

Open Baton est une plate-forme open source fournissant une implémentation complète de la spécification ETSI NFV Management et Orchestration (MANO) [8]; elle vise à implémenter un plugin pour intégrer ce MANO à Zabbix Server (Figure 2.12).

Bien que, **Open Baton** a développé un module (plugin) zabbix pour le monitoring des NFV, mais cette solution est insuffisante pour des simples raisons:

- la solution de monitoring est déployée au niveau de l'infrastructure qui ne sera pas toujours accessible à tout le monde comme le montre la figure.
- zabbix est connue comme une solution efficace mais la configuration est difficile.
- zabbix ne prend pas en considération le monitoring des ANF.

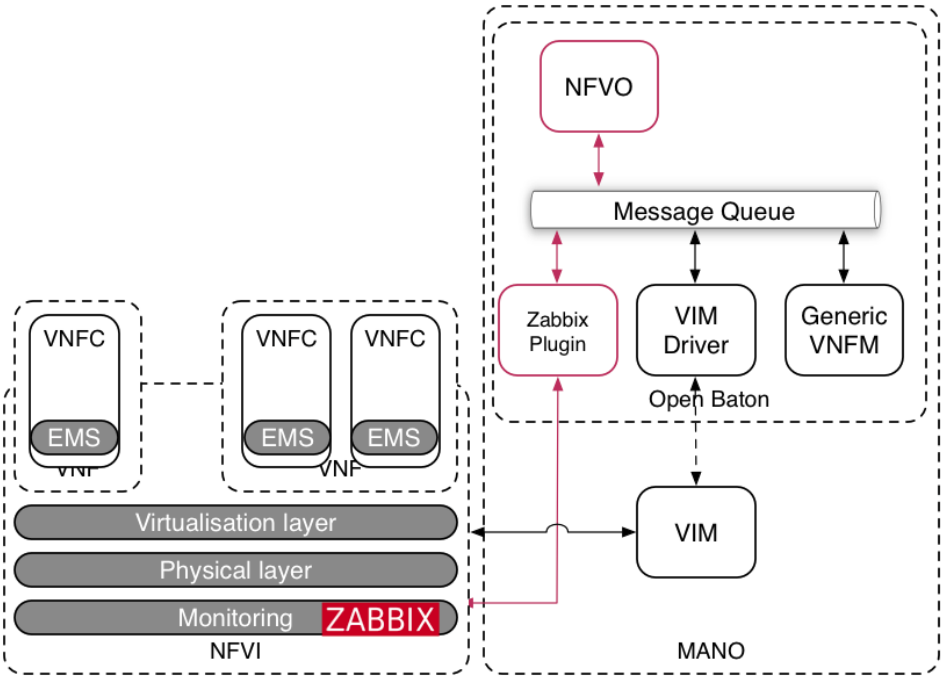


Figure 2. 12: Plugin de monitoring Zabbix

II. CONCLUSION

Les outils de monitoring présentent plusieurs fonctionnalités. Certains intègrent le monitoring de NFV, cependant il n'y a pas de solution pour surveiller les ANF, car la plupart des solutions se focalisent sur les VNF. De plus, même si ces solutions sont efficaces, il y a une complexité au niveau de leur utilisation. C'est cela qui a motivé la réalisation d'un outil de monitoring applicatif permettant d'intégrer toutes ces spécificités : un outil qui soit simple d'utilisation, avec une interface flexible, conviviale, pour le déploiement des agents de monitoring adaptées à la fois pour les ANF et les VNFs. Le chapitre suivant a pour objectif de proposer une conception d'un système de monitoring pour la surveillance des fonctions réseaux applicatives et virtuelle.

CHAPITRE 3
CONCEPTION DU SYSTÈME DE MONITORING

1. SPÉCIFICATION GÉNÉRALE DU SYSTÈME MONITORING
2. SPÉCIFICATION AGENT DE MONITORING
3. SPÉCIFICATION ET CONCEPTION DU MONITORING MANAGER (MM)
4. CONCLUSION

Introduction

Dans ce chapitre, nous présentons la conception de la solution de monitoring proposée. Tout d'abord Nous commençons par une présentation générale de l'approche de monitoring adoptée, puis nous en décrivons les composants et les fonctionnalités. Ensuite nous détaillons les métriques de performance, avant de détailler le fonctionnement des différents types d'agents de monitoring proposés par le système. Nous consacrons la dernière section à la description du composant central de ce système et son interface web.

I. Approche de Monitoring des fonctions réseau

1. Spécification générale du système Monitoring

1.1.Description générale

La solution de monitoring décrite dans ce rapport est une application qui surveille des fonctions réseau virtualisées et applicatives. Cette solution supporte à la fois l'interrogation (Polling) et la réception (Trapping) de données. Tous les rapports et les configurations sont accessibles via une interface web simple à utiliser afin d'éliminer les difficultés de déploiement et résume tous les agents déployés selon le type de conteneur de virtualisation.
Le système fournit une interface programmable qui permet d'exécuter des manipulations directement sur ce système et d'assurer une facilité d'intégration dans des différents environnements.

1.2.Fonctionnement

Contrairement aux solutions logicielles présentées dans l'état d'art (cf. section 2.5) qui surveillent les fonctions réseau à partir des infrastructures de virtualisation, notre approche de monitoring consiste à être le plus proche possible de la fonction réseau mais le plus indépendamment possible de l'infrastructure. Dans cette optique, nous avons proposé une approche de monitoring basée agent qui consiste à insérer un agent (cf. section 2.5) de monitoring au niveau du conteneur de virtualisation (cf. section 2.3.3). Ceci permet de collecter un ensemble de métriques influençant le fonctionnement et/ou les performances des fonctions réseau qui y sont hébergées (Figure 3.1).

1.3.Composant de la solution de monitoring

La solution de monitoring est constituée de plusieurs composants majeurs :

- **Monitoring Manager** : c'est le composant central du monitoring qui est responsable de créer des agents de monitoring à partir d'un descripteur (cf. section 3.3) et de les configurer. Ce composant gère aussi le déploiement et la modification de la configuration des agents de monitoring.

- **Interface WEB** : cette interface est fournie pour permettre la configuration et le déploiement des agents. Elle organise les agents déployés selon le type de conteneur de virtualisation ; de plus elle est chargée par la visualisation des données collectées

par chaque agent. L'interface interagit directement avec le Monitoring Manager ; pour cela elle fonctionne généralement (mais pas nécessairement) sur la même machine que celle qui exécute le Monitoring Manager.

- **Monitoring Manager database**: c'est une base de données accessible par le monitoring manager, les agents et l'interface web. Toutes les données collectées par les agents sont stockées dans cette base.

- **Agents de monitoring** : ils sont déployés sur des hôtes surveillés pour superviser les ressources locales et les applications, et pour envoyer les données collectées à la base de données.

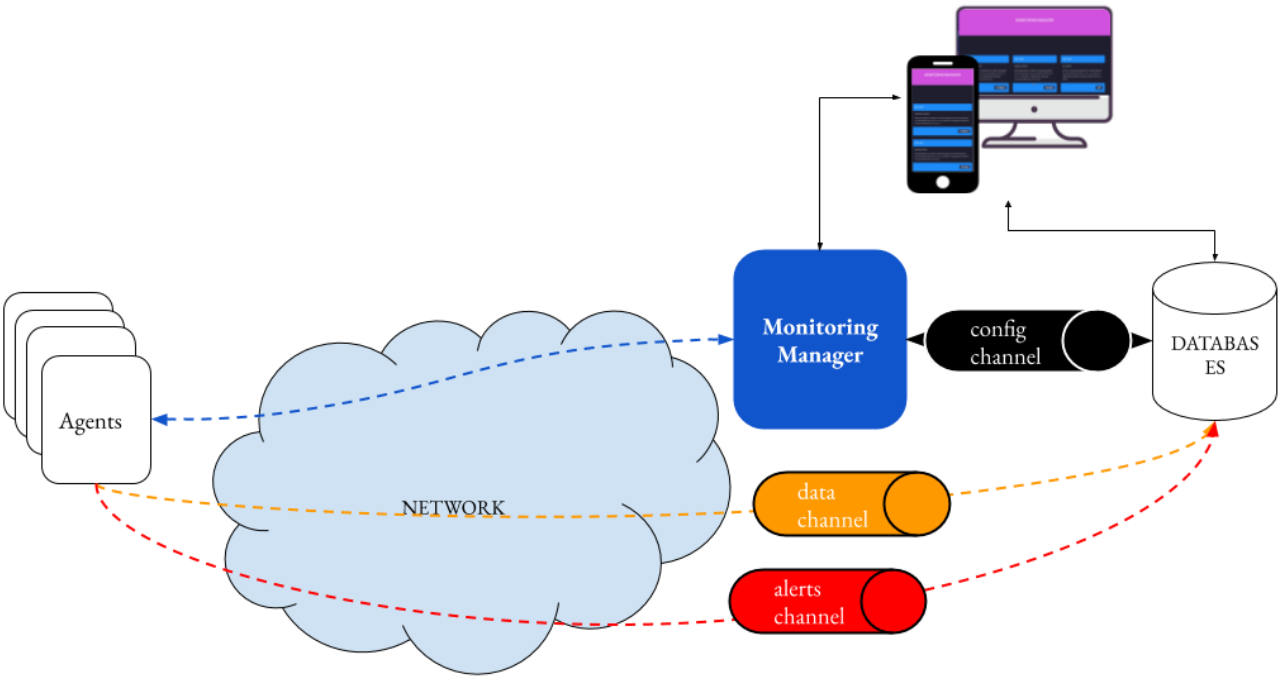


Figure 3. 1:Architecture générale de système de monitoring

1.4.Fonctionnalités de système de monitoring

Le système de monitoring offre une multitude de fonctionnalités de base résumées dans le tableau 3.1 :

Tableau 3. 1: Fonctionnalités de système de monitoring

Acteur	Fonctionnalités	Description
Agent	Collecte De Données	l'agent est responsable de collecter les métriques définies dans sa configuration et les stocker au niveau de la base de données.
	Notification De Seuil	la capacité d'un agent de comportement passive d'envoyer des notifications au cas de dépassement d'un seuil de référence prédéfinie au niveau de la configuration pour une métrique.
Monitoring Manager	Ajouter Agent	la capacité de créer un agent adapté à l'hôte cible en se basant sur un descripteur (section 3.3) et le déployer.
	Modifier Agent	la capacité de modifier la configuration des agents déjà déployés et l'appliquer.
	Supprimer Agent	la capacité d'arrêter et de supprimer définitivement un agent déjà déployé.
	Arrêter Agent	la capacité d'arrêter un agent de monitoring sans le supprimer.
	Démarrer Agent	la capacité de démarrer un agent de monitoring.
Interface Web	Visualisation Des Données	la capacité d'organiser les données collectées sur un hôte et les visualiser sous forme de graphes, tableaux et texte.

1.5.Les métriques de performances

Chaque évaluation d'un système demande un ensemble de paramètres qu'on appelle des métriques de performances. Cependant ces métriques varient d'un environnement à un autre. Pour définir des métriques de performance standards, nous avons résumé dans la section 2.4.6.3 les différentes métriques collectables pour les différents environnements de déploiement des fonctions réseau pris en charge dans ce rapport. A partir de cette section, nous avons choisi le CPU et la RAM comme métriques de performance principales car la surcharge d'une de ces métriques peut influencer négativement le fonctionnement des fonctions réseaux. Les autres métriques sont collectées mais elles jouent un rôle moins important dans la prise des décisions pour le déclenchement d'une action.

2. Spécification agent de monitoring

2.1.Description générale de l'agent

L'agent de monitoring est un élément indispensable dans notre solution ; ce composant a la capacité de s'exécuter sur des périphériques et des environnements avec des ressources limitées et de collecter les métriques préconfigurées. De plus, il est basé sur l'architecture REST (representational State Transfert), ce qui nous permet d'exposer un ensemble des URIs (Uniform Resource Identifier) (section 2.5), consommables par le monitoring manager, l'administrateur ou même par l'environnement d'hôte.

2.2.Fonctionnement des agents de monitoring

Les agents de monitoring créés par le monitoring manager (cf. section 3) ont la capacité d'interagir directement avec l'environnement qui héberge la fonction réseau pour collecter les informations et les envoyer à une base de données ou des notifications vers le monitoring manager sans utiliser aucun protocole de monitoring (exemple : SNMP). Chaque agent utilise un fichier de configuration qui décrit son comportement (actif, passif), les seuils, les métriques à surveiller, les informations sur le Monitoring Manager, etc.

2.3.Agent de monitoring pour les machines virtuelles (VM)

2.3.1. Description

Cet agent récupère les informations en se basant sur la lecture des fichiers système et l'utilisation d'un ensemble de commandes système (Figure 3.2)

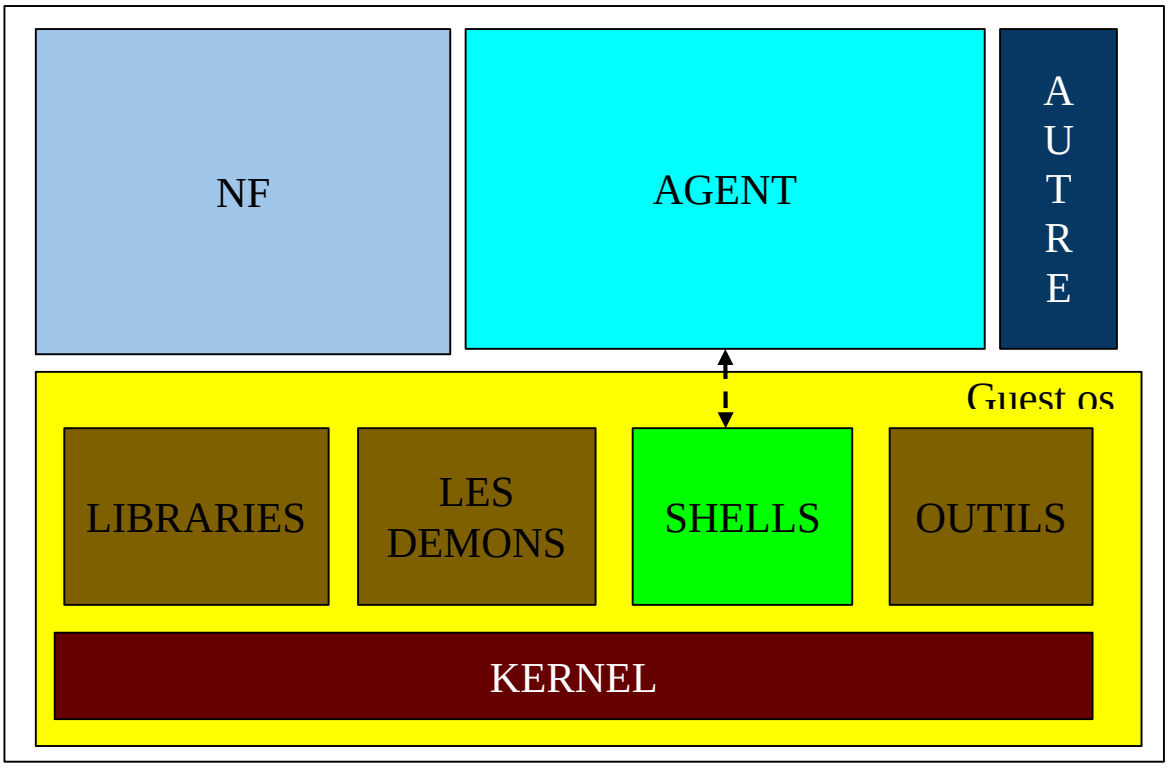


Figure 3. 2: Fonctionnement l'agent des VM

2.3.2. Calcule de métrique

L'environnement d'exécution expose un ensemble d'informations qu'on peut récupérer avec une multitude de méthodes et les manipuler pour extraire une métrique de performance, pour cela, dans cette section, nous présentons un tableau résumant la méthode de récupération de chaque métrique.

Tableau 3. 2:les métriques monitorées par l'agent VM

Métriques	méthode de calcul	Description
CPU	$\frac{(UT_t - UT_{t-1}) - (idle_t - idle_{t-1})}{UT_t - UT_{t-1}}$ t: un instant donnée t-1: instant précédent UT: temps d'utilisation de CPU idle : temps d'inactif de CPU	ces informations sont collectées à travers le fichier de system /proc/stat.

RAM	$\frac{(100-(libre-cache-tmp))}{total} \cdot 100$ libre : la mémoire vive non utilisé cache : mémoire dans la cache sans swap cache tmp:mémoire dans le cache du tampon total : RAM utilisable totale	ces informations sont collectées à travers le fichier de system /proc/meminfo.
NETWORK		les informations concernant une interface réseau sont collectées en utilisant le fichier /proc/net/dev qui contient les statistiques de transmission et de réception, en plus de l'utilisation de la commande ifstat pour récupérer la bande passante du réseau. les informations décrivant les connections ouvertes dans un système sont accessibles en utilisant la commande lsof avec l'option "i"
DISK		les information sur la consommation du disque est collectable via l'utilisation de la commande df .

2.3.3. Agent de monitoring pour les conteneurs2

2.3.3.1.Cgroup

Cgroup - groupes de contrôle - ont pour but de contrôler les ressources systèmes utilisées par un ou plusieurs processus. Les processus sous contrôle sont affectés dans des groupes sur

lesquels agissent des contrôleurs de ressources. Chaque contrôleur gère un type de ressources [17]:

- ❏ cpuset : allocation CPU (numéro CPU/core CPU).
- ❏ cpuacct : consommation CPU (nombre de cycles).
- ❏ memory : contrôle de la mémoire vive et de la mémoire swap.
- ❏ devices : contrôle l'accès aux périphériques.
- ❏ blkio : contrôle l'accès aux périphériques de type bloc (ex: disque dur).
- ❏ net_cls : contrôle l'accès aux périphériques réseau

2.3.3.2.Namespace

Il existe différents types d'espace de noms (ou namespaces) chaque type d'espace de noms s'applique à une ressource spécifique afin de créer des barrières entre les processus.

Docker utilise quelques types d'espace de noms afin d'isoler ses différents conteneurs, nous pouvons citer :

- ❏ pid : isole les processus (pid : Process ID).
- ❏ net : permet la gestion des interfaces réseaux (net : networking)
- ❏ ipc : gère les accès inter-processus (ipc : InterProcess Communication)
- ❏ mnt : gestion des points de montage (mnt : mount)
- ❏ uts : utilisé pour isoler le noyau et identificateurs de version (UTS : Unix Timesharing System)

2.3.3.3.Description d'agent

Les conteneurs sont caractérisés par la légèreté due à l'absence d'OS (cf. chapitre 2 section 2.1.2). Ces conteneurs utilisent des fonctionnalités natives au noyau Linux, comme les **cgroups** ou les **namespaces**, mais offrent aussi les outils pour le faire de manière simplifiée. Le monitoring manager propose un agent capable d'être intégré au niveau du conteneur et d'interagir avec les fichiers **de contrôle correspondant à ce conteneur** afin de récupérer les métriques de performance prédéfinies dans la configuration de l'agent (Figure 3.4).

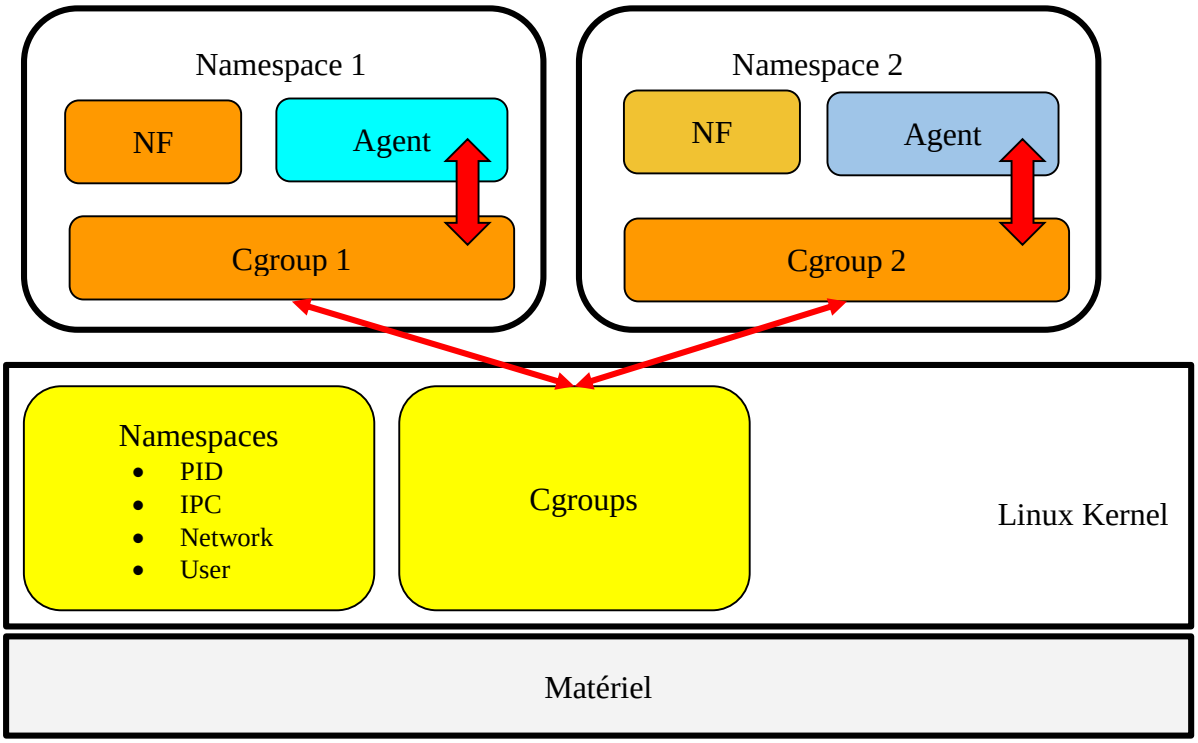


Figure 3. 3: Fonctionnement des agents des conteneurs

2.3.3.4.Calcule de métrique

Dans cette section nous présentons un tableau résumant la méthode de récupération de chaque métrique de l'agent de monitoring pour les conteneurs.

\

Tableau 3. 3:les métriques monitorées par l'agent des conteneurs

Métriques	méthode de calcul	Description
CPU	$\frac{(UT_t - UT_{t-1}) - (idle_t - idle_{t-1})}{UT_t - UT_{t-1}}$ t: un instant donnée t-1: instant précédent UT: temps d'utilisation de CPU idle : temps d'inactif de CPU	ces informations sont collectées à travers le fichier de system **/proc/stat** et **/sys/fs/cgroup/cpuacct/.**
RAM	$\frac{(100 - (libre - cache - tmp))}{total} \cdot 100$ libre : la mémoire vive non utilisé cache : mémoire dans la cache sans swap cache tmp: mémoire dans le cache du tampon total : RAM utilisable totale	ces informations sont collectées à travers le fichier de system **/sys/fs/cgroup/memory**.
NETWORK		les informations concernant une interface réseau sont collectées en utilisant le fichier **/sys/class/net/nom_interface /statistics/** qui contient les statistiques de transmission et de réception, en plus de l'utilisation de la commande ifstat pour récupérer la bande passante du réseau. les informations décrivant les connections ouvertes dans un système sont accessibles en utilisant la commande lsof

		avec l'option "i"
DISK		les informations sur la consommation du disque sont collectables via l'utilisation de la commande df .

2.3.4. Agent de monitoring pour les ANF

2.3.4.1. Framework OSGI

OSGi (Open Service Gateway Initiative) est un framework Java permettant de développer et de déployer des programmes logiciels modulaires (bundle) et des bibliothèques.

2.3.4.2. Bundle

Un bundle OSGi est un fichier d'archive Java contenant du code Java, des ressources et un manifeste décrivant le bundle et ses dépendances [18].

2.3.4.3. Description

Les fonctions réseau applicatives s'exécutent dans la plateforme modulaire OSGi (section 2.4.6.3), sous forme de bundles, alors dans cette optique nous proposons un agent de monitoring déployable dans cet environnement sous la même forme des ANFs. Il est capable d'interagir et récupérer les données à partir d'un agent JMX (Java Management Extensions) (cf. chapitre 4 section 2) pré-déployé au niveau du java virtuelle machine (JVM) (Figure 3.4).

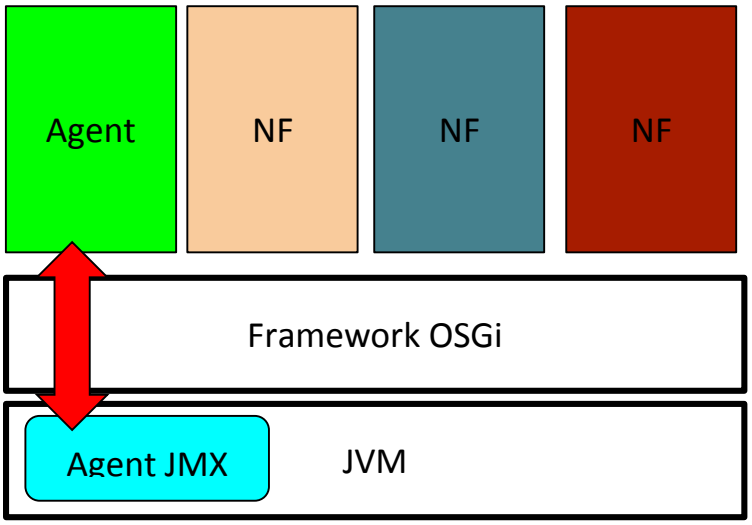


Figure 3. 4: Fonctionnement des agents JVM

2.3.4.4. Calcule de métrique

Dans cette section, nous présentons un tableau résumant la méthode de récupération de chaque métrique de l'agent de monitoring pour les conteneurs ANF

Tableau 3. 4:les métriques monitorées par l'agent des JVM

Métriques	méthode de calcul	Description
CPU	$\frac{\sum_{i=1}^{n}(thread_i)_t}{processorsCount}$ $(thread_i)_t$: Consommation cpu du thread i à l'instant t. $processorsCount$: Nombre de core de CPU utilisé par la JVM. n : Thread avec l'id n	La consommation de CPU pour les threads et le nombre de core de CPU utilisés par la JVM sont récupérés à partir de l'agent JMX

RAM	$\frac{total - libre}{total}.100$ total : quantité totale de mémoire de la machine virtuelle Java libre: quantité de mémoire disponible dans la machine virtuelle Java	cette métrique représente la consommation de la mémoire de la jvm par rapport à la quantité de mémoire total définie pour cette dernière.
	$\frac{max - total + libre}{sysram}.100$ total : quantité totale de mémoire de la machine virtuelle Java libre: quantité de mémoire disponible dans la machine virtuelle Java max: la quantité maximale de mémoire de la machine virtuelle Java sysram: la quantité totale de mémoire physique	cette métrique représente la consommation de la mémoire de la JVM par rapport à la quantité de mémoire total du système.
THREADS		ces informations sont directement collectables à partir du l'agent JMX mais l'agent les formatent dans une forme compréhensible.

2.4.Configuration des agents de monitoring

Le fonctionnement de l'agent est déterminé par un fichier de configuration, appelé *conf.conf* dans le cas des VNFs et *anf_agent.properties* dans le cas des ANFs. Ce fichier est situé dans le répertoire d'installation d'agent afin de décrire un ensemble des paramètres utilisées pour récupérer, et stocker les données collectées.

Les fichiers de configuration se composent de cinq sections :

- **host** : contient les différents paramètres pour accéder à l'hôte surveillé.

- **agent**: contient les différents paramètres décrivant l'agent.
- **server**: contient les différents paramètres sur le serveur de monitoring (Monitoring Manager).
- **DB**: contient les différents paramètres pour accéder à la base de données.
- **contrainte :** contient les informations sur les seuils de référence pour les métriques**.**

De plus, chaque section contient un ensemble des paramètres résumés dans le tableau 3.5.

Tableau 3. 5:Paramètres de fichier de configuration

Paramètre	Section	Description	Champs obligatoires
ip	host	L'adresse ip ou le nom de l'hôte du machine hébergeant la NF	X
port	host	Le numéro de port pour les communications distantes avec l'agent de monitoring.	X
type	host	Le type du conteneur de la NF (OSGi,VM,CNT)	X
username	host	nom utilisateur à utiliser pour accéder au conteneur	X
password	host	mot de passe à utiliser pour accéder au conteneur	obligatoire pour une authentification ssh avec mot de passe sinon l'authentification par clé publique pré-installée dans le hôte
metrics	agent	les métriques que l'agent doit envoyer à la base de données.	X
type	agent	le type de comportement d'agent	X

		(passif ou actif)	
activated	agent	décrit l'état de l'agent au moment de déploiement s'il est démarré ou non	X
refresh_peiode	agent	définit la période de temps qui sépare deux collectes de métriques successives.	obligatoire dans le cas d'agent passif, par défaut 30 secondes
server_ip	server	L'adresse ip ou le nom de l'hôte du serveur de monitoring	X
server_port	server	Le numéro de port pour les communications distantes avec les serveurs de monitoring	X
alerts_url	DB	L'adresse ip ou le nom de l'hôte de l'agent d'acheminement des données pour stocker les notifications	obligatoire dans le cas d'agent passif.
alerts_port	DB	Le numéro de port pour les communications distantes avec l'agent d'acheminement des données pour stocker les notifications	obligatoire dans le cas d'agent passif.
data_url	DB	L'adresse ip ou le nom de l'hôte du agent d'acheminement des données pour stocker les métriques collectées	X
data_port	DB	Le numéro de port pour les communications distantes avec l'agent d'acheminement des données pour stocker les métriques collectées	X
hdfs_url	DB	L'adresse ip ou le nom de l'hôte de la base de données	X

hdfs_port	DB	Le numéro de port pour les communications distantes avec la base de données	X
mem	contrainte	Le seuil de référence pour la consommation de la ram	obligatoire dans le cas d'agent passif.
cpu	contrainte	Le seuil de référence pour la consommation de la ram	obligatoire dans le cas d'agent passif.

2.5. Les endpoints des agents

Les agents exposent trois ressources utilisables par un client REST. Le tableau 3.6 les décrit les ressources exposées par les agents.

Tableau 3. 6:Ressources exposées par les agents

Resource (URI)	Description
<hôte_ip>:<port>/api/cpu	récupérer la consommation de CPU.
<hôte_ip>:<port>/api/mem	récupérer la consommation de RAM.
<hôte_ip>:<port>/api/all	récupérer les différentes informations sur l'agent est son environnement.

3. Spécification et conception du Monitoring Manager (MM)

3.1. Description générale du MM

Le gestionnaire de monitoring est le noyau du système de monitoring, il est caractérisé par une facilité de déploiement et une intégration logicielle fluide, grâce à son architecture basée REST. Il permet aussi d'exposer des ressources que nous pouvons les classer dans trois grandes fonctionnalités :

- Fonctionnalités de gestion et de configuration : sont les différentes opérations exécutables sur un agent (création, modification et suppression de la configuration).

- Fonctionnalités stockage des données : sont les opérations du stockage, de la récupération, de la modification ou de la suppression des données dans la base de données.

- Fonctionnalités de visualisation : la partie responsable de la récupération et la visualisation des données collectées sous forme graphique ou des tableaux et texte.

3.2. Structure Interne du MM

Le MM est conçu pour être l'élément central dans tout le système de monitoring, à partir de la création des agents jusqu'à la visualisation des données (section 3.1). Ces fonctionnalités sont dues à une combinaison entre les composants logicielle interne de cet élément. La Figure 3.6 présente l'architecture interne du MM :

- **Request Handler** : est le composant responsable de la gestion des requêtes reçues par le MM. Il lit les requêtes, traite les en-têtes, et fournit le contenu au composant "Filter"

- **Filter** : Ce composant traite le contenu fourni afin de choisir l'opération à exécuter sur l'agent (créer, modifier, supprimer). Par la suite, il extrait la configuration de l'agent, que nous appelons descripteur d'agent (section 3.3) et il l'envoie à l'**agent configurator**.

- **Agent Configurator**: ce composant est constitué de 3 sous composant :

    - ❑ Agent Templates : il définit les templates de base que le composant Agent builder utilise pour construire un agent.

    - ❑ Agent builder: c'est l'élément responsable de créer des agents personnalisés à partir des templates d'agent dans le composant en se basant sur le descripteur (section 3.3).

    - ❑ Compressor : est un élément capable de compresser l'agent généré avant de l'envoyer à un hôte.

- **Agent Deployer** : le Monitoring Manager assure le déploiement et la modification d'agent à distance, ce composant s'en charge de réaliser ces différentes opérations en utilisant une communication sécurisée grâce au protocole SSH.

- **DB** : c'est une base de données qui centralise le stockage des différentes données (configuration, stockage des métriques). Elle est accessible en lecture et écriture pour les composants de systèmes de monitoring.

- Interface WEB : est une interface permettant la visualisation des données collectées par chaque agent et la (re-)configuration des nouveaux agents.

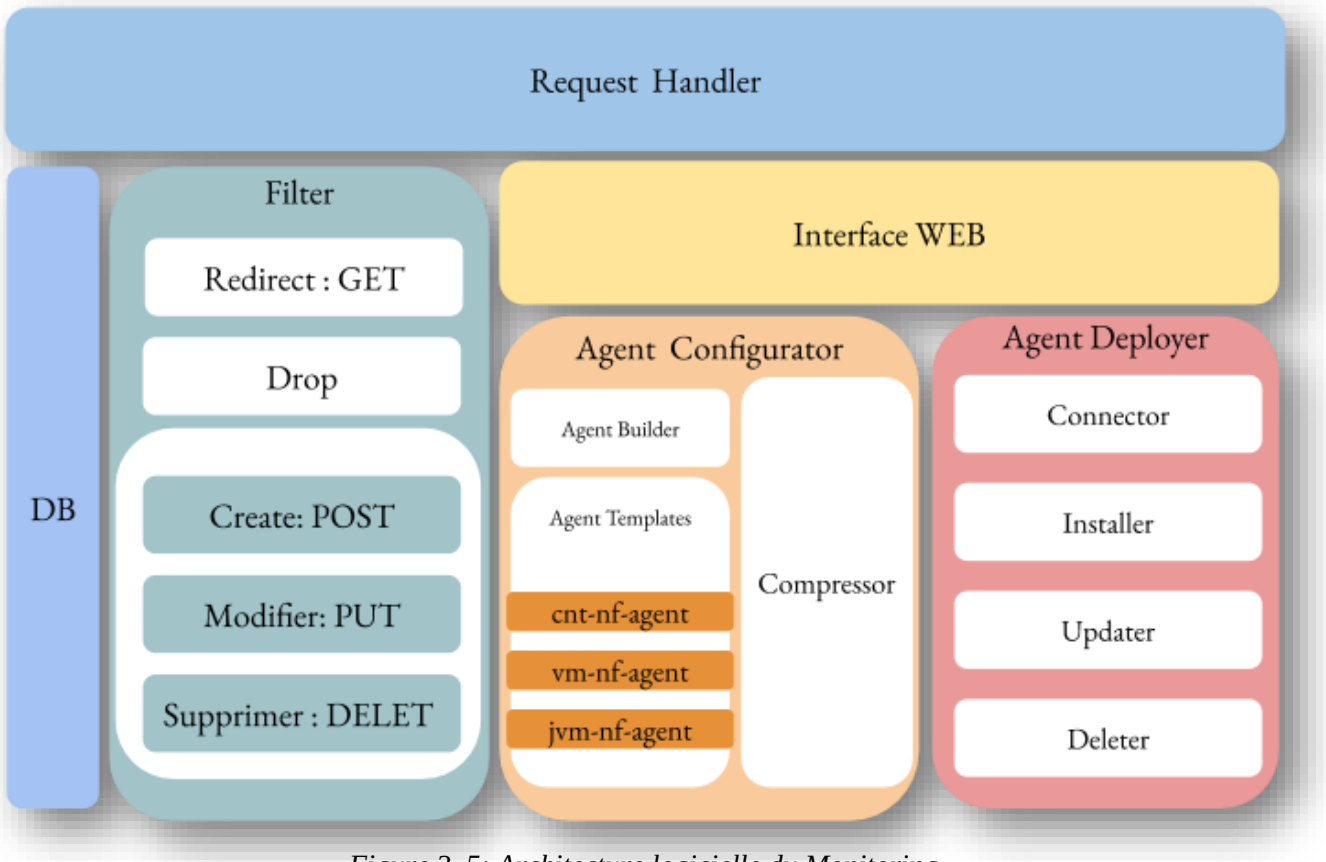


Figure 3. 5: Architecture logicielle du Monitoring

3.3. Description haut niveau du Monitoring Manger

Dans la section précédente, nous avons présenté le rôle de chaque composant du Monitoring Manger alors que cette section se focalise sur les points de communication entre les composants interne et les différents acteurs externes. Le MM expose 4 interfaces de communications (Figure 3.7) :

- **HTTP REQUEST :** permet de traiter les différentes requêtes issues des clients REST ou de l'interface web du système.

- **send data:** permet la communication entre l'agent et le monitoring manager afin de récupérer des métriques et des informations sur l'agent et son environnement. La communication se base sur des méthodes Http (GET, POST, PUT, DELETE)

- **write data**: permet la communication entre l'agent et la base de données interne du monitoring manager pour stocker les métriques collectées ainsi que les notifications.

- **send agent :** permet la communication du monitoring manager et les hôtes en utilisant le protocole SSH aussi bien que les méthodes HTTP.

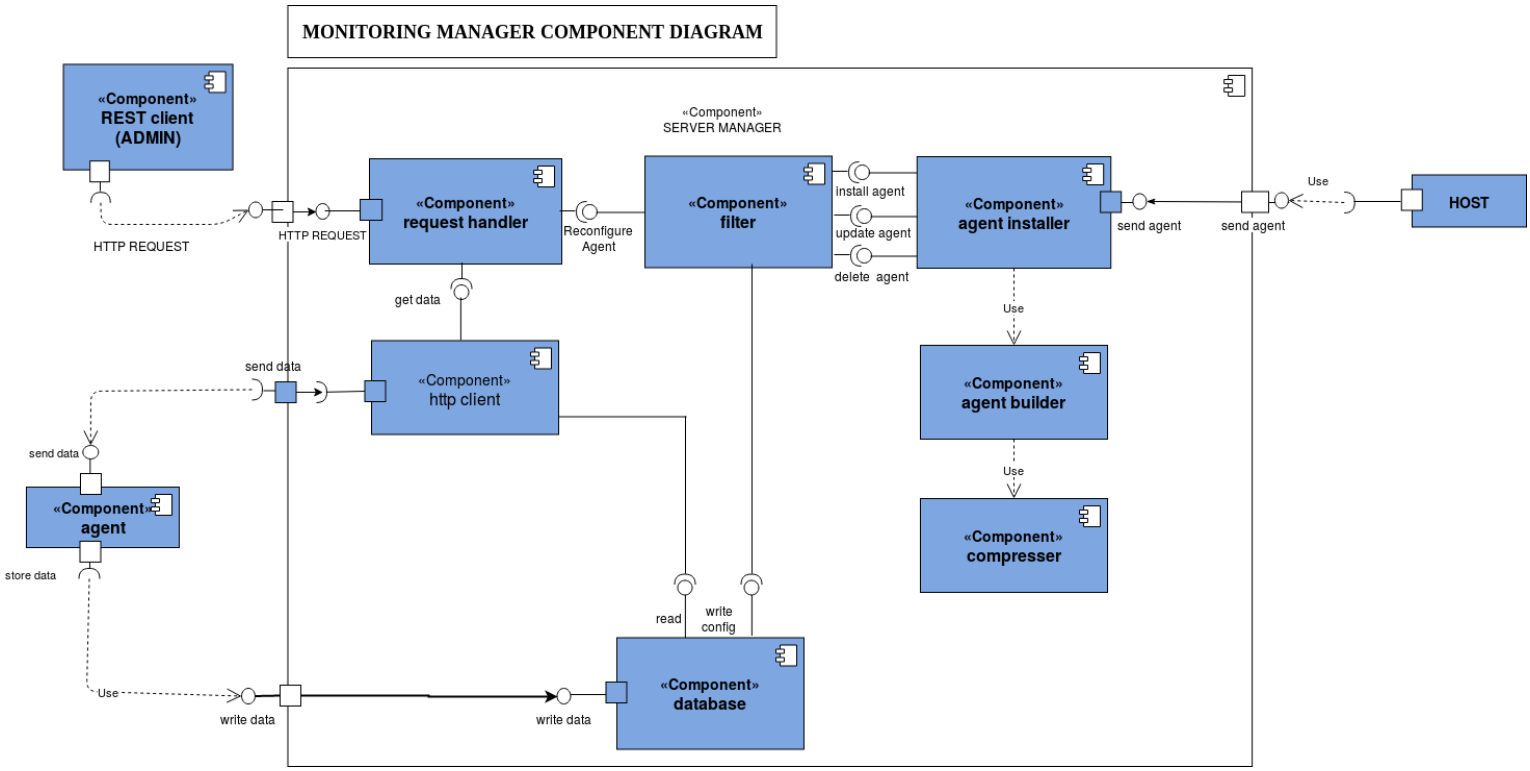


Figure 3. 6: Diagramme de composites du Monitoring Manager

3.4.Modèle informationnel

Dans cette section nous présentons le modèle qui décrit les entrées pour chaque opération fournit par le Monitoring Manager. Ce modèle contient les informations descriptives qu'un système externe doit fournir au MM pour valider la demande et exécuter une opération.

3.4.1. Descripteur pour ajouter ou modifier un agent

Tableau 3. 7:Descripteur de configuration des agents

{	
"access_host": {	Les différentes informations sur les hôtes.
"username": "string",	Nom utilisateur à utiliser pour accéder au conteneur
"host port": "8080",	Le numéro de port pour les communications distantes avec l'agent de monitoring.
"server ip": "127.0.0.1",	L'adresse ip ou le nom de l'hôte du serveur de monitoring
"host ip": "127.0.0.1",	L'adresse ip ou le nom de l'hôte du machine hébergeant la NF
"password": "string",	Mot de passe à utiliser pour accéder au conteneur, vide en cas d'authentification par clés.
"type": "string",	Le type du conteneur de la NF (OSGi,VM,CNT)
"server port": "5000"	Le numéro de port pour les communications distantes avec les serveurs de monitoring
},	
"metrics": {	les métriques que l'agent doit envoyer à la base de données.
"mem": true,	Activation du RAM comme métrique
"cpu": true	Activation du CPU comme métrique

},	
"agent contrainte": {	Le seuil de référence pour la consommation des métriques
"mem": 0,	Le seuil de référence pour la consommation de la RAM
"cpu": 0	Le seuil de référence pour la consommation du CPU
},	
"agent": {	Contient les différents paramètres décrivant l'agent.
"refresh_period": 10,	Période du temps pour envoyer les données à la base de données (cas passif)
"activated": true,	Ce champ donne le choix de démarrer ou non le monitoring après le déploiement
"type": "passive/active"	Le type d'agent de monitoring.
},	
"database": {	Contient les différents paramètres pour accéder la base de données.
"hdfs": {	Les informations pour accéder à Hadoop Distributed File System et récupérer les données
"hdfs_port": "string",	Le numéro de port pour les communications distantes avec la base de données
"hdfs_url": "string"	L'adresse ip ou le nom de l'hôte de la base de données

},	
"ALERTS": {	Les informations nécessaires pour stocker les notifications
"port": "string",	Le numéro de port pour les communications distantes avec l'agent d'acheminement des données pour stocker les notifications.
"flume_url": "string"	L'adresse IP ou le nom de l'hôte de l'agent d'acheminement des données pour stocker les notifications
},	
"data": {	les informations nécessaires pour stocker les métriques
"flume_port": "string",	Le numéro de port pour les communications distantes avec l'agent d'acheminement des données pour stocker les métriques collectées
"port": "string"	L'adresse IP ou le nom de l'hôte de l'agent d'acheminement des données pour stocker les métriques collectées
}	
}	
}	

Remarque : Les informations fournies lors de la création ou la modification d'un agent seront utilisées pour la création du fichier de configuration d'agent décrit dans la section 2.3

3.4.2. Descripteur pour supprimer, arrêter et démarrer un agent

Tableau 3. 8:descripteur de gestion des agents

{	
"username": "string",	Nom d'utilisateur à utiliser pour accéder au conteneur
"host port": "8080",	Le numéro de port pour les communications distantes avec l'agent de monitoring en cas de ANFs.
"host ip": "127.0.0.1",	L'adresse ip ou le nom de l'hôte du machine hébergeant la NF.
"password": "string",	Mot de passe à utiliser pour accéder au conteneur.
}	

3.5. Opérations fournis par le monitoring manager

Cette section décrit les cas d'utilisation textuels illustrant les opérations fournis aux acteurs externes du MM. Ces cas d'utilisation sont synthétisés sur la Figure 3.7.

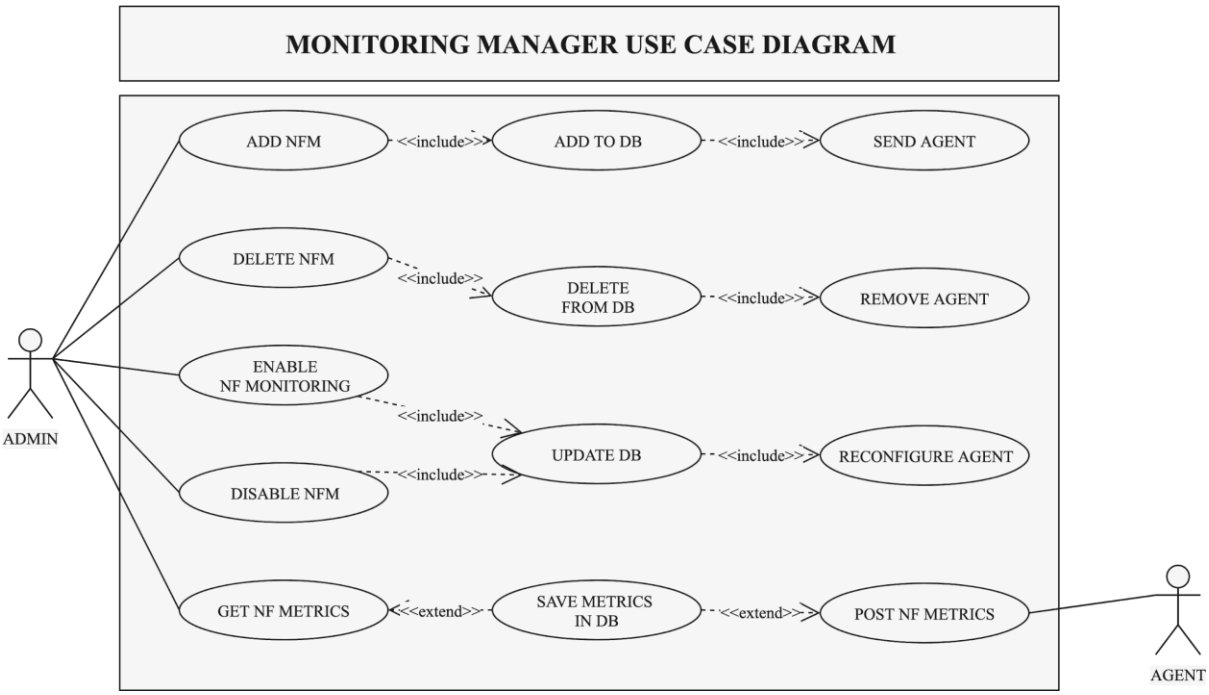


Figure 3. 7: Les opérations fournis par le monitoring manager

ID	1
Nom :	ADD NFM
Description:	Ajouter un nouvel agent de monitoring
Acteurs primaires:	- Administrateur - Orchestrateur ANF - Orchestrateur VNF

Enchaînement principal:

1. L'acteur envoie le descripteur d'agent.
2. Le MM reçoit le descripteur et le vérifie.
3. Si le descripteur est valide:
   - 3.1. le MM Ajoute la configuration à la base de données.
   - 3.2. Le MM crée un agent, l'envoie à l'hôte et le déploie si l'acteur à valider le champ dans le

descripteur

Enchaînement alternatif:	Si le descripteur n'est pas validé, l'opération est annulée et un message d'erreur est envoyé à l'utilisateur.

ID	2
Nom :	DELETE NFM
Description:	suppression définitive d'un agent de monitoring
Acteurs primaires:	- Administrateur - Orchestrateur ANF - Orchestrateur VNF

Enchaînement principal:

1. Acteur envoie une demande de suppression au MM.
2. Le MM reçoit la demande et la vérifie.
3. Si la demande est valide :
   - 3.1. le MM se connecte à l'hôte
   - 3.2. le MM arrête l'agent puis exécute une commande de suppression.

Enchaînement alternatif:	Si le descripteur n'est pas validé l'opération est annulée et un message d'erreur est envoyé à l'utilisateur.

ID	3
Nom :	DISABLE NFM
Description:	Arrêter agent de monitoring ce qui implique que l'hôte n'est plus monitoré
Acteurs primaires:	- Administrateur - Orchestrateur ANF - Orchestrateur VNF

Enchaînement principal:

1. Acteur envoie une demande d'arrêt d'agent au MM.
2. Le MM reçoit la demande et la vérifie.
3. Si la demande est valide :
   - 3.1. le MM se connecte à l'hôte.
   - 3.2. le MM arrête l'agent.

Enchaînement alternatif: Si le descripteur n'est pas validé, l'opération est annulée et un message d'erreur est envoyé à l'utilisateur.

ID 4

Nom : ENABLE NFM

Description: Démarrer l'agent de monitoring d'un agent déjà arrêté.

Acteurs primaires:

- Administrateur
- Orchestrateur ANF
- Orchestrateur VNF

Enchaînement principal:

1. Acteur envoie une demande de démarrage d'agent au MM.
2. Le MM reçoit la demande et la vérifie.
3. Si la demande est valide:
   - 3.1. le MM se connecte à l'hôte.
   - 3.2. le MM démarre l'agent.

Enchaînement alternatif: Si le descripteur n'est pas validé, l'opération est annulée et un message d'erreur est envoyé à l'utilisateur.

ID 5

Nom : GET NF METRICS

Description:	Récupérer les métriques collectées dans la base de données sur une NF
Acteurs primaires:	- Administrateur - Orchestrateur ANF - Orchestrateur VNF
Enchaînement principal:	1. Acteur envoie une demande de données au MM . 2. Le MM reçoit la demande et la vérifie. 3. Si la demande est valide: 3.1. le MM se connecte à NF et récupère les informations. 3.2. le MM envoie les données à l'acteur.
Enchaînement principal:	Si le descripteur n'est pas validé, l'opération est annulée et un message d'erreur est envoyé à l'utilisateur.

ID	6
Nom :	DELETE NFM
Description:	suppression définitive d'un agent de monitoring
Acteurs primaires:	- Agent
Enchaînement principal:	1. L'agent collecte les métriques à partir de la machine hôte. 2. l'agent envoie les données collectées à la base des données à une période de temps régulière ou non s'il y a une notification.

3.6.Les interactions du Monitoring Manager et les acteurs

Dans cette section, nous illustrons les interactions entre le monitoring manager et les acteurs afin de fournir les différentes opérations. Nous nous contentons de présenter quelques interactions à titre illustratif.

3.6.1. Ajouter un nouvel agent

L'objectif principal de cette opération est la création d'un agent personnalisé, alors comme illustré par la Figure 3.9, l'administrateur ou l'orchestrateur ANF/VNF procède comme suit pour créer et déployer un agent :

- L'acteur commence par la création d'une configuration qui respecte le descripteur décrit dans la section 3.4.1, ensuite, il l'envoie au monitoring manager pour les vérifications.

- Une fois le monitoring manager valide le descripteur, il crée un agent qui correspond à la configuration et il l'envoie à l'hôte.

- Si l'agent est bien déployé, alors le Monitoring Manager enregistre la configuration dans la base de données et envoie un message de réussite.

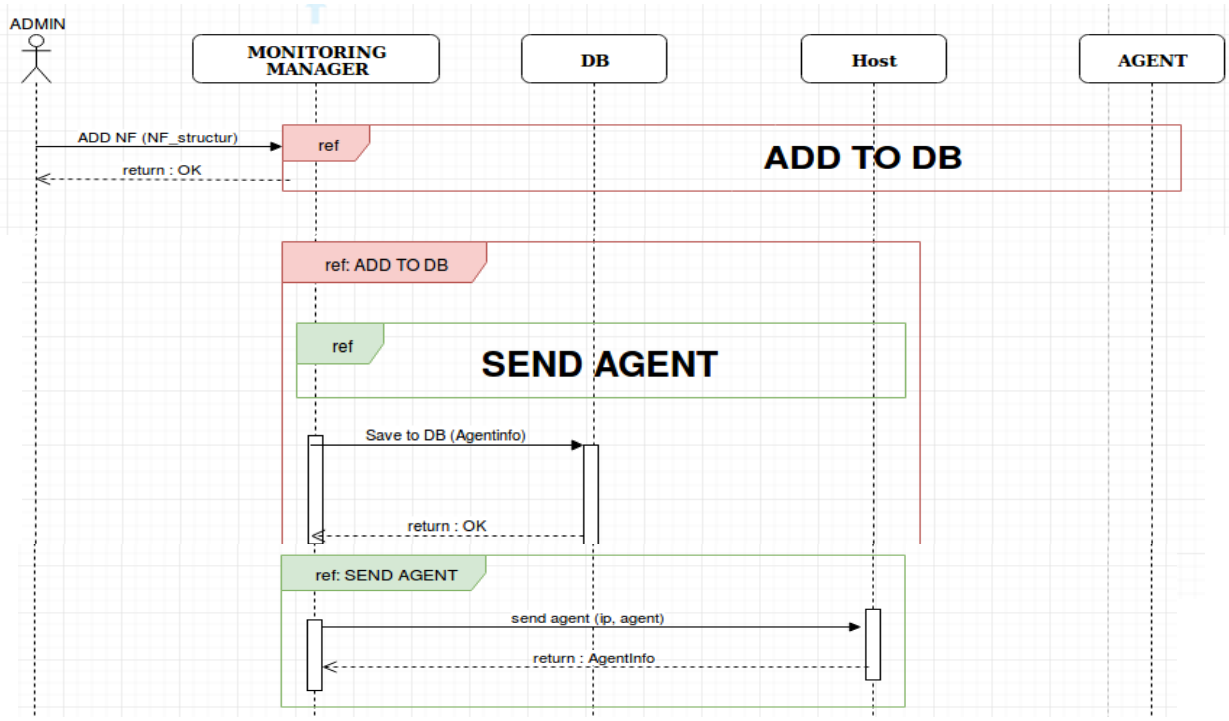


Figure 3. 8: Diagramme de séquence pour ajouter un agent

3.6.2. Suppression d'un agent

L'objectif principal de cette opération est d'arrêter le monitoring et supprimer l'agent responsable de la collecte des métriques, comme illustré par la Figure 3.10, l'administrateur ou l'orchestrateur ANF/VNF procède comme suit pour créer et déployer un agent :

- L'acteur envoie l'adresse IP au monitoring manager et un numéro de port en cas de besoin ;
- Une fois le monitoring manager se connecte à l'hôte, il arrête l'agent et le supprime ;
- Si l'agent est bien supprimé, le monitoring manger envoie un message de réussite, après la suppression de la base de données.

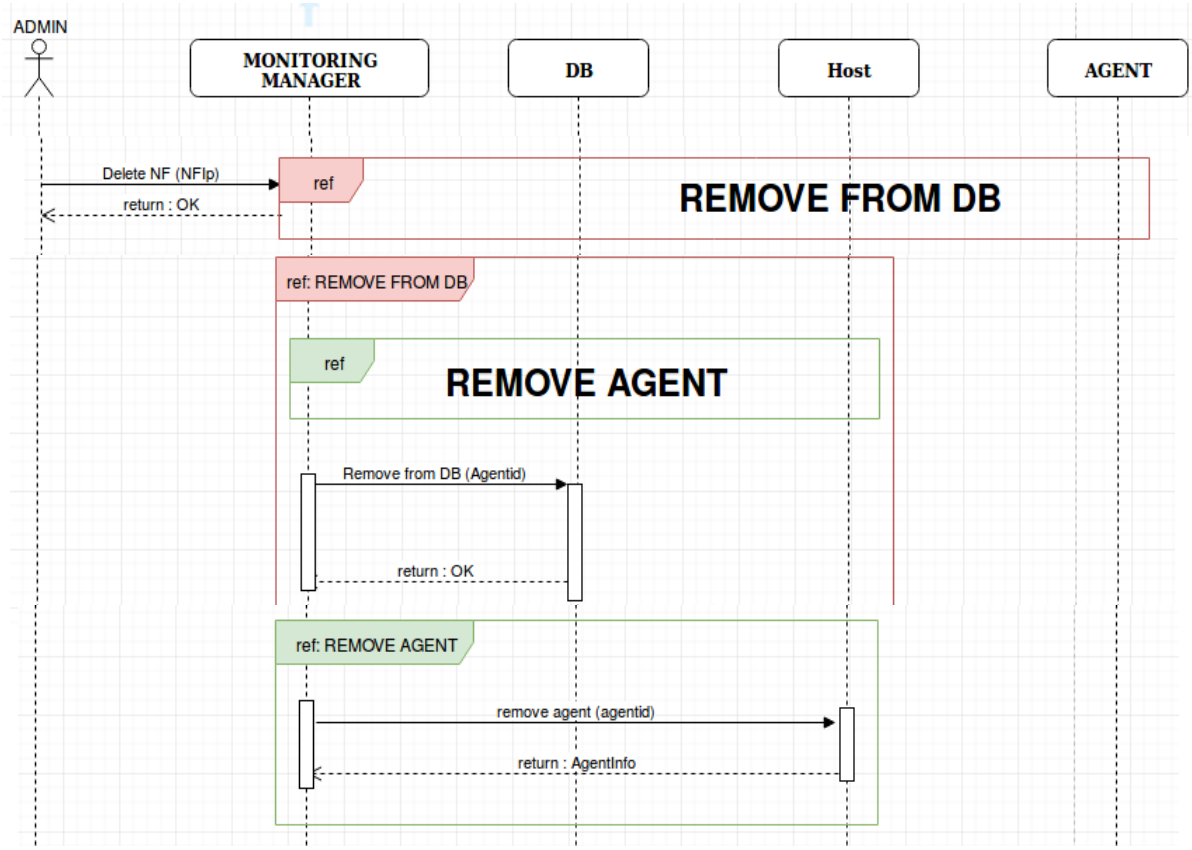


Figure 3. 9: Diagramme de séquence pour la supprimer un agent

3.7.Les endpoints de Monitoring Manager

Les Monitoring Manager exposent 3 types de ressources utilisable par un client REST. Le tableau 3.9 les décrit :

Tableau 3. 9:Les ressource exposées par le monitoring manager

	Resource (URI)	Description
VNF	<hôte_ip>:<port>/manager/VNFManager/add/vnf	ajouter un agent pour monitorer les vnfs
	<hôte_ip>:<port>/manager/VNFManager/delete	supprimer un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/VNFManager/enable	démarrer un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/VNFManager/disable	arrêter un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/VNFManager/update/vnf	modifier la configuration d'un agent de monitoring pour les vnfs
ANF	<hôte_ip>:<port>/manager/ANFManager/add/	ajouter un agent pour monitorer les vnfs
	<hôte_ip>:<port>/manager/ANFManager/delete	supprimer un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/ANFManager/enable	démarrer un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/ANFManager/disable	arrêter un agent de monitoring pour les vnfs
	<hôte_ip>:<port>/manager/ANFManager/update/	modifier la configuration d'un agent de monitoring pour les vnfs
Monitoring Manager	<hôte_ip>:<port>/server/configuration	configuration des bases des données pour le monitoring manager

4. Interface web du Monitoring Manager (MM)

L'interface Web est fournie pour un accès facile au système de monitoring de n'importe où et de n'importe quelle plate-forme. Elle est constituée de quatre sections :

- Section **Home** : est la page d'accueil de l'interface web du système de monitoring. Elle résume les différentes sections de cette interface Figure 3.11.

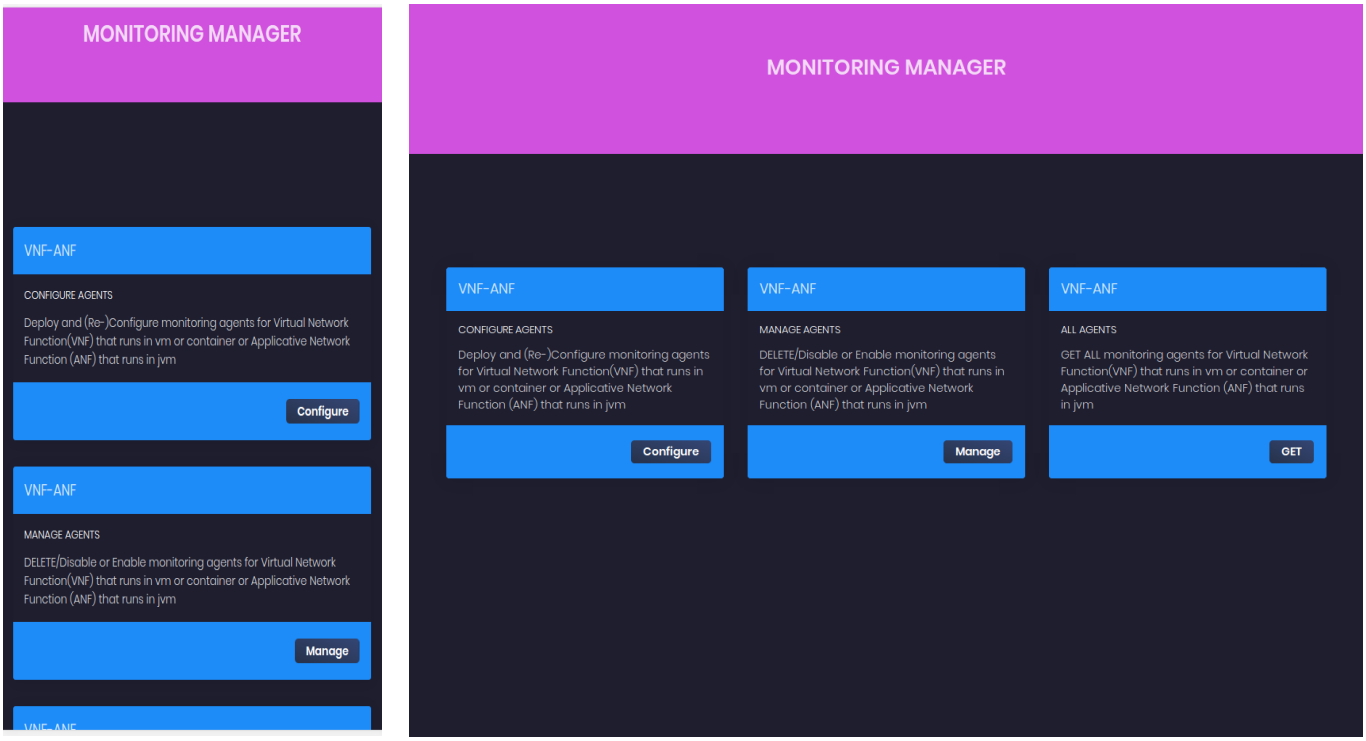


Figure 3. 10: Page d'accueil de l'interface web du monitoring manager

- Section **agent opérations** : cette section mets à disposition de l'utilisateur deux interfaces ; une pour la configuration de l'agent (ajout et la modification) Figure 3.12,

et une autre pour la gestion des agents déjà déployés (démarrer, arrêter et supprimer) Figure 3.11

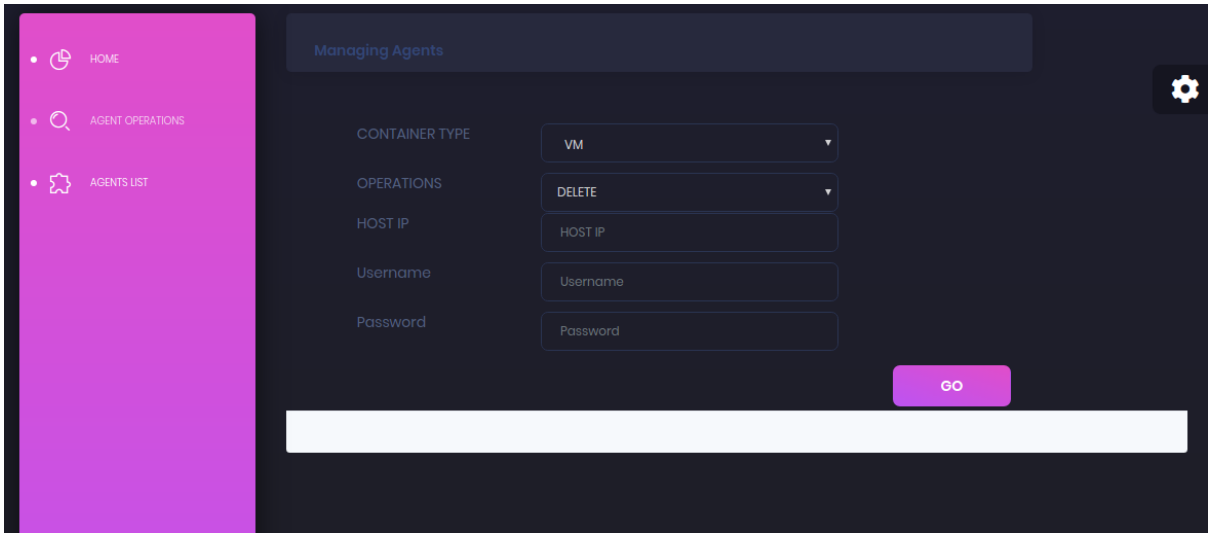


Figure 3. 12: page de gestion des agents de monitoring

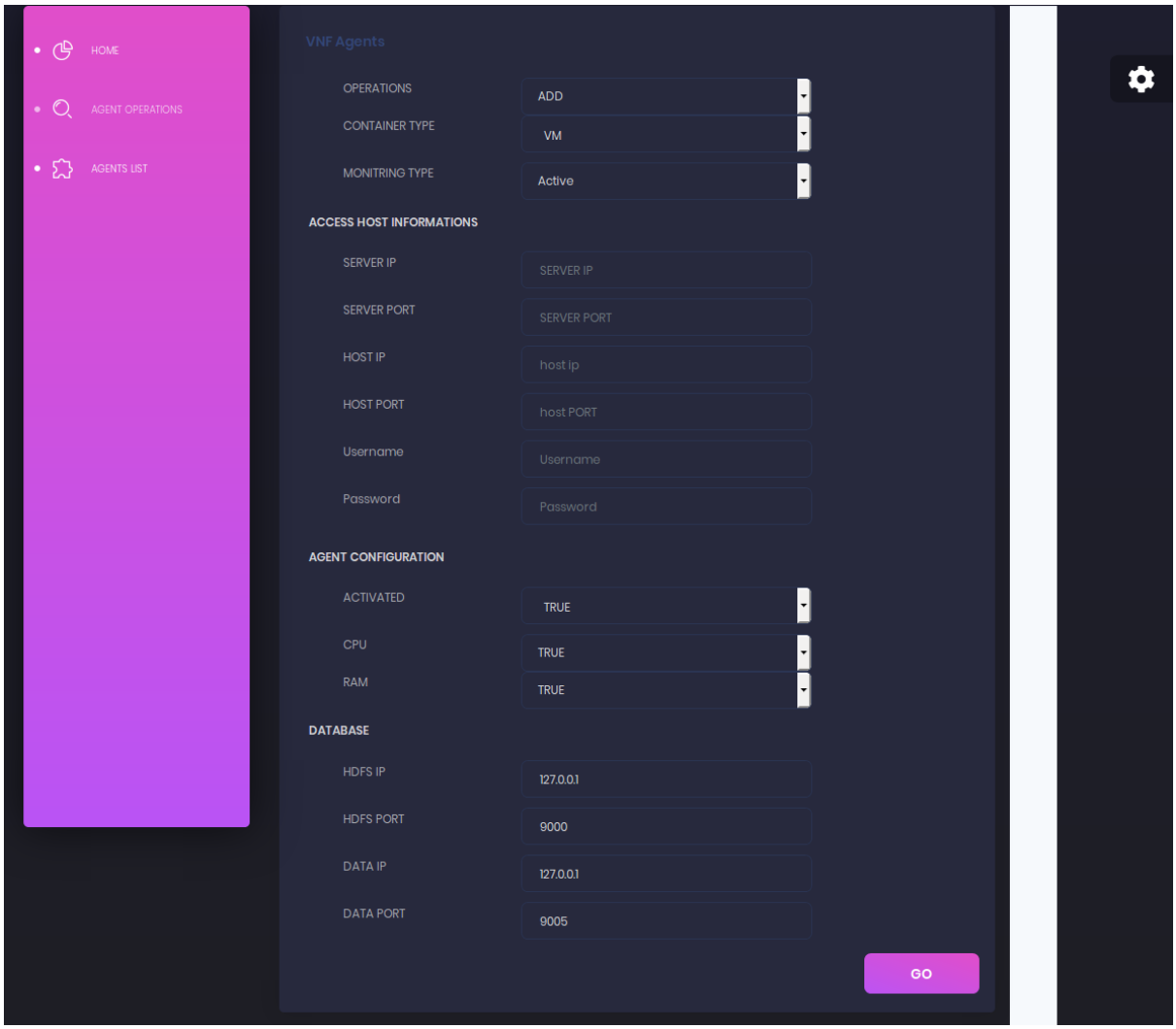


Figure 3. 11:page de configuration des agents de monitoring

- Section **list agent** : liste tous les agents déjà déployés selon le type de conteneur de virtualisation et envoie les données à la base de données. Cette liste donne l'accès au Dashboard de chaque agent Figure (3.13).

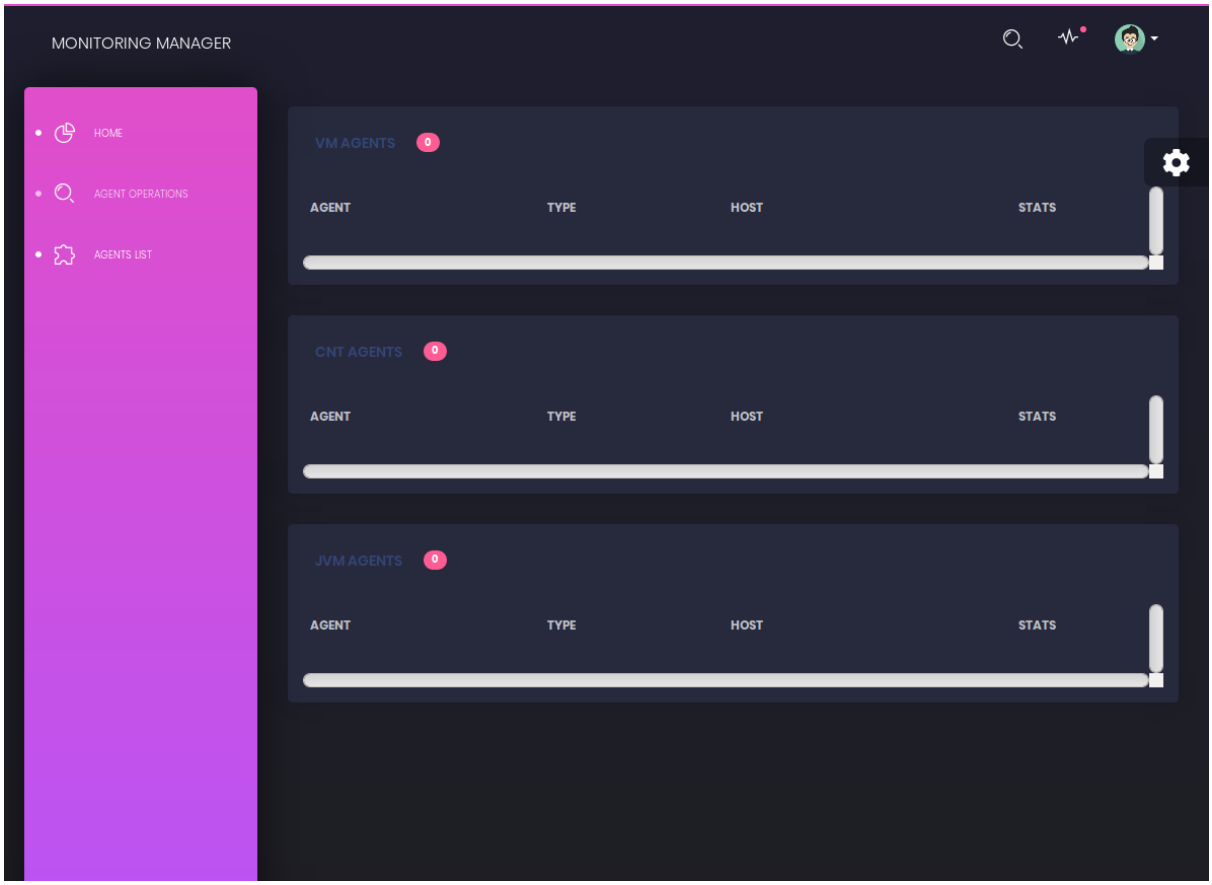


Figure 3. 13: page des agents déployés

II. Conclusion

Ce chapitre a présenté la conception de notre solution de monitoring. Nous avons commencé par la spécification du système de monitoring, puis de ses composants principaux à savoir, les agents et le monitoring manager. La conception a été présentée sous la forme de modèles UML. Afin d'implémenter ce travail, nous avons utilisé un ensemble d'outils d'ingénierie logicielle que nous décrivons dans le prochain chapitre, avant de valider les tests fonctionnels et non-fonctionnels de notre solution.

CHAPITRE 4
IMPLÉMENTATION ET TESTS

1. ARCHITECTURE
2. IMPLÉMENTATION
3. STOCKAGE DES DONNÉES
4. TESTS FONCTIONNELS
5. CONCLUSION

Introduction

Ce chapitre se focalise dans sa première partie sur la présentation des différents outils d'ingénierie utilisés pour la mise en œuvre de la solution de monitoring. La deuxième partie est dédiée aux tests fonctionnels et non-fonctionnels pour la validation de l'implémentation logicielle.

I. Implémentation de la solution

L'implémentation de la solution proposée a reposé sur un certain nombre d'outils que nous pouvons classer en trois groupes : architecture, implémentation et stockage des données.

1. Architecture

Le Monitoring Manager et les agents de monitoring sont des api restful, i.e. des applications qui respectent les contraintes de l'architecture REST. REST est un style architectural basé sur les normes Web et le protocole HTTP. Ce style a été initialement décrit par Roy Fielding en 2000 [19]. Dans une architecture basée sur REST, tout est une ressource. Une ressource est accessible via une interface commune basée sur la méthode standard HTTP. Ces méthodes sont utilisées dans ce travail comme décrit dans le tableau ci-dessous.

Tableau 4. 1: Utilisation des Méthodes HTTP par le MM

Méthode HTTP	DESCRIPTION
GET	Cette méthode est utilisée par le MM et les acteurs pour récupérer des informations à partir de la base de données ou des agents
POST	Cette méthode est utilisée par le MM pour ajouter un nouvel agent de monitoring sur un hôte
PUT	Cette méthode est utilisée par le MM pour modifier la configuration d'un agent de monitoring. Elle est aussi utilisée pour démarrer et arrêter un agent.
DELETE	Cette méthode est utilisée par le MM pour supprimer un agent de monitoring.

2. Implémentation

Les principaux outils utilisés pour implémenter le Monitoring Manager et les agents sont le Framework web de Python Flask-RESTPlus, le Framework Java Jersey et l'api Java Management Extensions (JMX).

- **Flask-RESTPlus**: est une extension pour Flask qui prend en charge la création rapide d'API REST. Flask-RESTPlus encourage les meilleures pratiques avec une

configuration minimale. Il fournit une collection cohérente de décorateurs et d'outils pour décrire une API et exposer correctement sa documentation à l'aide de Swagger. [20].

- **JMX :** est l'acronyme de Java Management Extensions. C'est une spécification qui définit une architecture, une API et des services pour permettre de surveiller et de gérer des ressources en Java. JMX permet de mettre en place, en utilisant un standard, un système de surveillance et de gestion d'une application, d'un service ou d'une ressource sans avoir à fournir beaucoup d'efforts. [21]

- **Jersey** : La structure de services Web RESTful de Jersey est une source ouverte, de qualité de production, destinée au développement de services Web RESTful en Java, qui prend en charge les API JAX-RS et sert d'implémentation de référence JAX-RS (JSR 311 et JSR 339). La structure Jersey est plus que l'implémentation de référence JAX-RS. Jersey fournit sa propre API qui étend la boîte à outils JAX-RS avec des fonctionnalités et des utilitaires supplémentaires pour simplifier davantage le développement du service et du client RESTful [22].

3. Stockage des données

Les agents génèrent une quantité massive de données que nous devons stocker afin de les analyser par la suite. Dans cette optique, nous avons choisi flume apache comme un bus de données vues sa fiabilité, et hadoop comme une base de données en raison de sa capacité à stocker et traiter de vastes quantités de données rapidement,

- apache flume : Flume est un service distribué, fiable et disponible pour la collecte, l'agrégation et le transfert efficaces de grandes quantités de données de journal. Son architecture est simple et flexible, basée sur la transmission en continu des flux de données. Il est robuste et tolérant aux pannes avec des mécanismes de fiabilité ajustables et de nombreux mécanismes de basculement et de récupération. Il utilise un modèle de données extensible simple qui permet une application analytique en ligne. [23]

- hadoop : La bibliothèque de logiciels Apache Hadoop est une structure qui permet le traitement distribué de grands ensembles de données sur des grappes d'ordinateurs à l'aide de modèles de programmation simples. Il est conçu pour passer de serveurs

uniques à des milliers de machines, chacune offrant un calcul et un stockage locaux. Plutôt que de compter sur du matériel pour offrir une haute disponibilité, la bibliothèque elle-même est conçue pour détecter et gérer les défaillances au niveau de la couche application, fournissant ainsi un service hautement disponible sur une grappe d'ordinateurs, chacun pouvant être sujet aux défaillances [24].

II. Tests Fonctionnels

L'objectif de cette section est de montrer les capacités du monitoring manager à configurer et à gérer des agents dans les trois différents environnements (VM, CNT et OSGI). Pour cela, nous proposons l'architecture illustrée dans la Figure 4.2. Cette architecture est proposée dans la thèse à laquelle ce travail se contribue, elle est constituée de plusieurs composants mais les VNFs, les ANF et le Monitoring Manager représentent les éléments centraux de cette section.

Dans tous les scénarios**,** nous supposons que :

- Le Monitoring Manager a été déployé et est capable de communiquer avec les VNFs et les ANFs.
- Les VNFs et les ANFs sont capables d'accueillir les agents de monitoring.
- Les agents de flume et la base de données sont activés.
- Un administrateur va déclencher les actions.

Remarque : Les différents composants de l'architecture de la figure 4.1 sont définis dans la section terminologie.

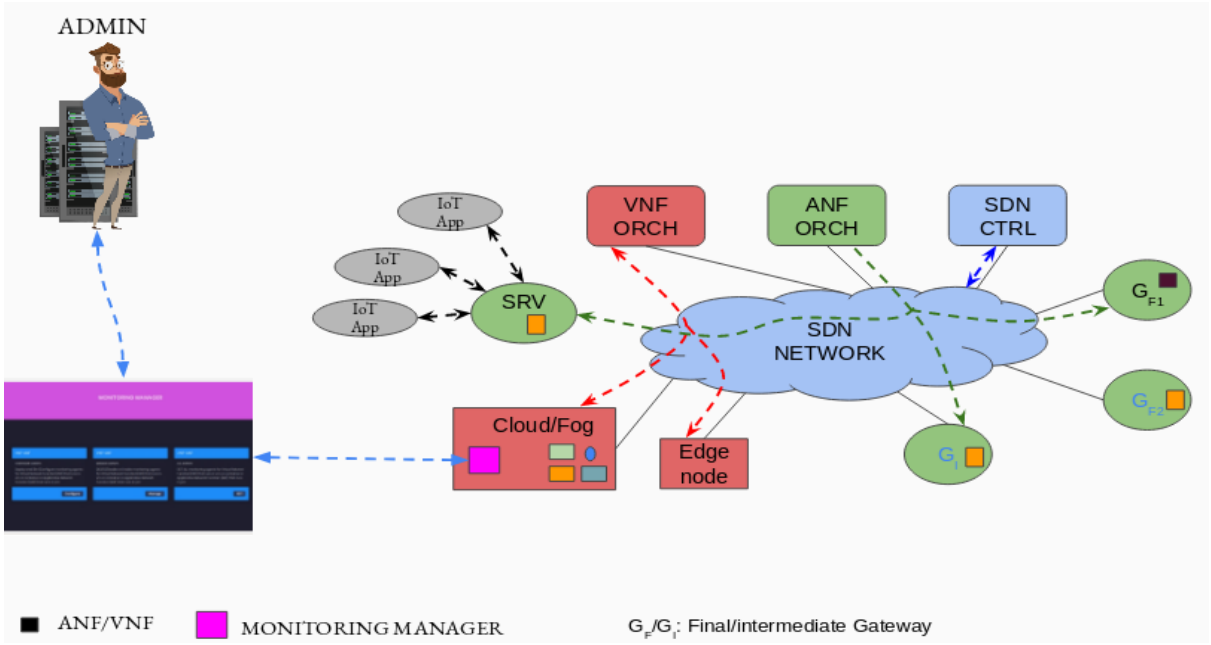


Figure 4. 1: architecture générale de déploiement de monitoring manager et les agents

1. Les environnements de Test

AGENT	ENVIRONNEMENT
VM	- Système : Ubuntu - RAM : 16 Go - CPU : 4
Docker	- Image : Ubuntu - RAM : 5 Gib
OSGI	- IDE : Eclipse - Environnement : Equinoxe

2. Scénario n°1: Déploiement d'un agent dans une VM

- L'administrateur va remplir les champs pour décrire l'agent dont il a besoin avant de déclencher la configuration de l'agent.

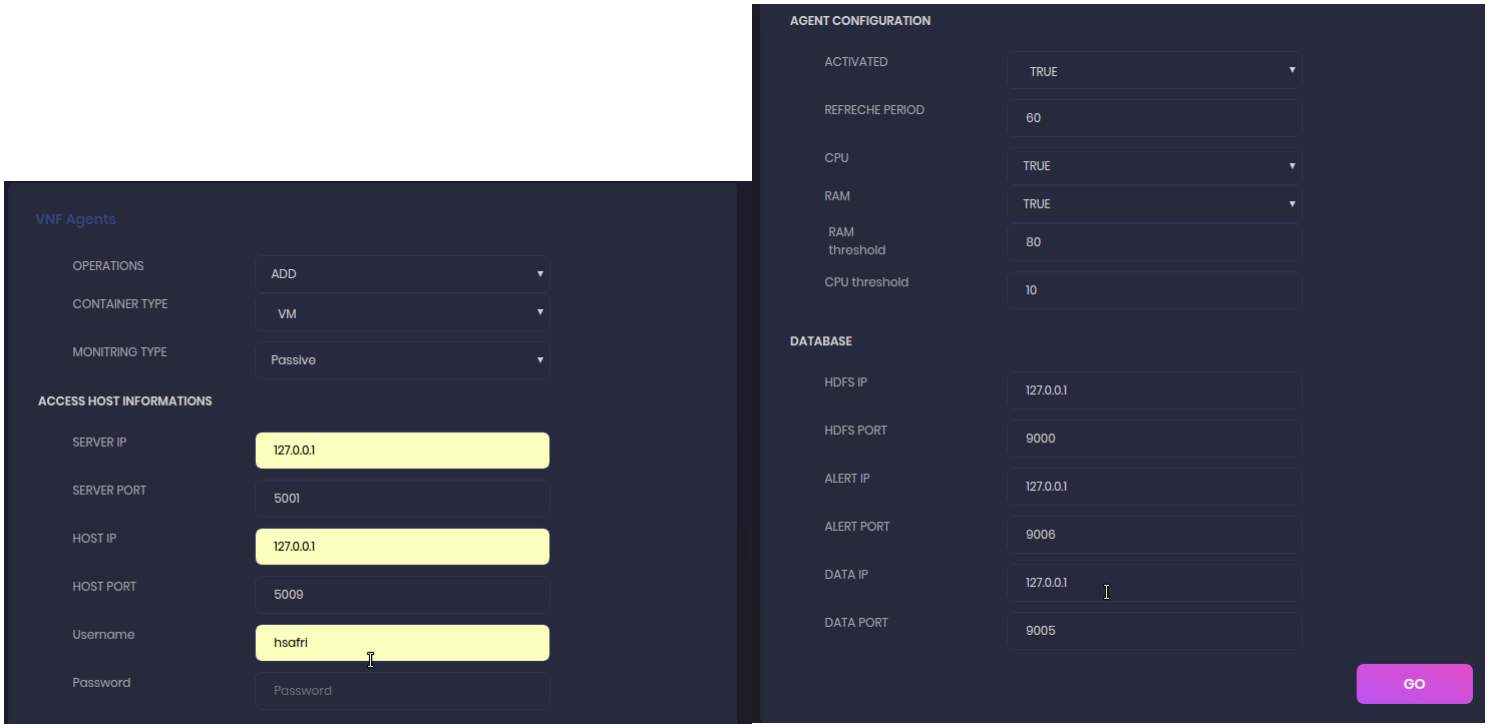


Figure 4. 2: Configuration d'un agent

- Une fois que l'agent est bien déployé, un message en haut de la page s'affiche. Cet agent n'apparaît dans la liste des agents que lorsque le Monitoring Manager détecte ses données dans la BD.

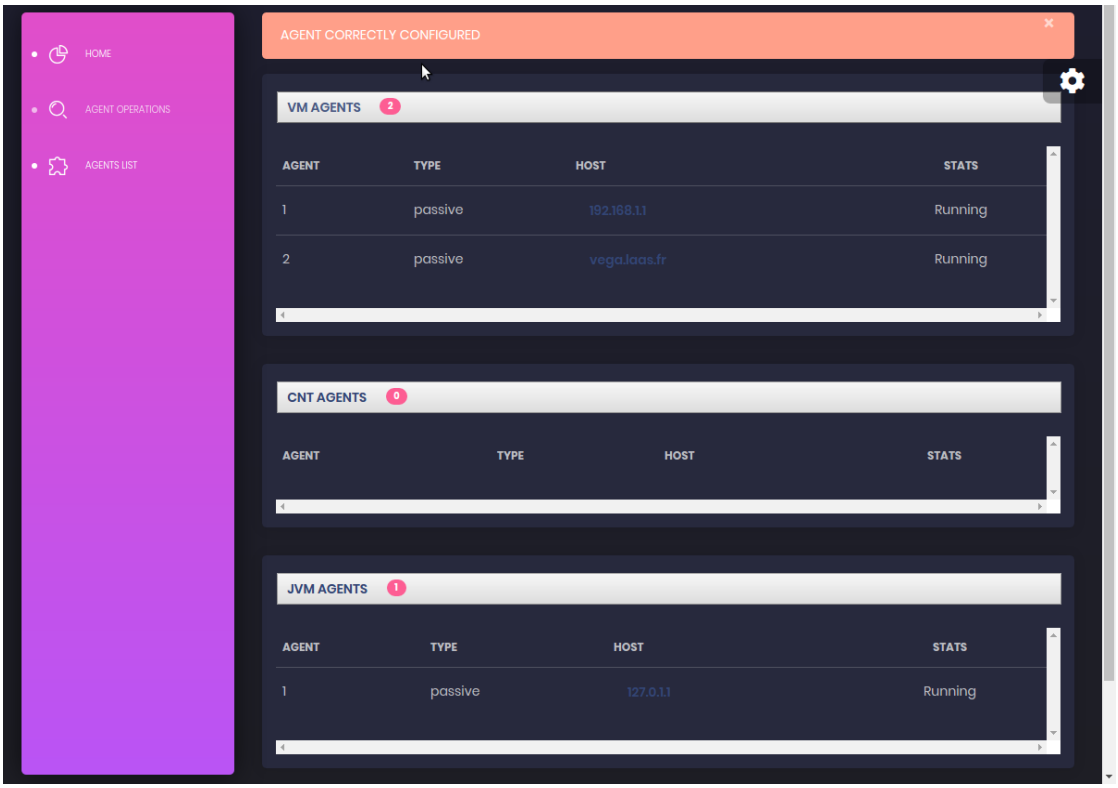


Figure 4. 3:Message du déploiement réussi d'agents

- Après l'ajout d'agent au niveau de la table des agents, nous pouvons récupérer les informations sur l'agent et son environnement en cliquant sur son adresse.

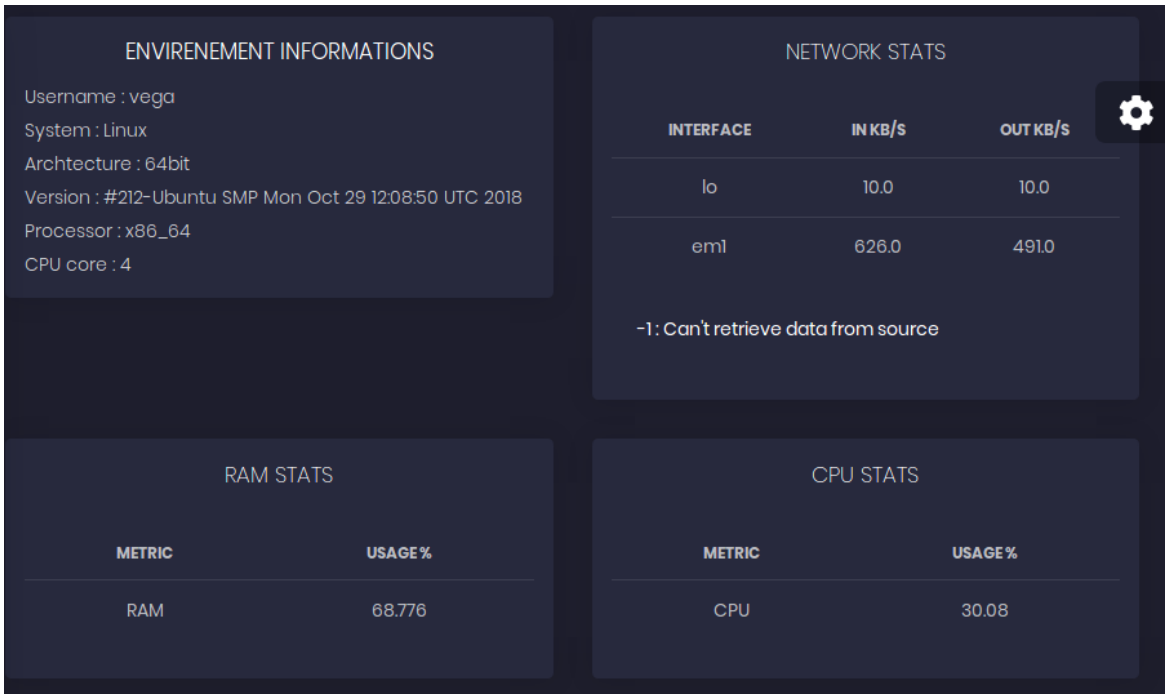


Figure 4. 5: les données sur l'environnement du l'agent

FILE SYSTEM	USAGE %	READ KB/S	WRITE KB/S
udev	1	0	0
tmpfs	1	0	0
/dev/dm-0	4	27,04	7,99
/dev/sda1	48	0,09	0,00
perdita:/vol/vol1/sara/hsafri	78	0	0
pongo:/vol/vol1/users8/tguerout	84	0	0
pongo:/vol/vol1/users6/smedjiah	84	0	0
pongo:/vol/vol1/sara/jlaguila	84	0	0
perdita:/vol/vol1/sara/couedrao	78	0	0

Figure 4. 4:les données sur le disque de stockage de l'environnement

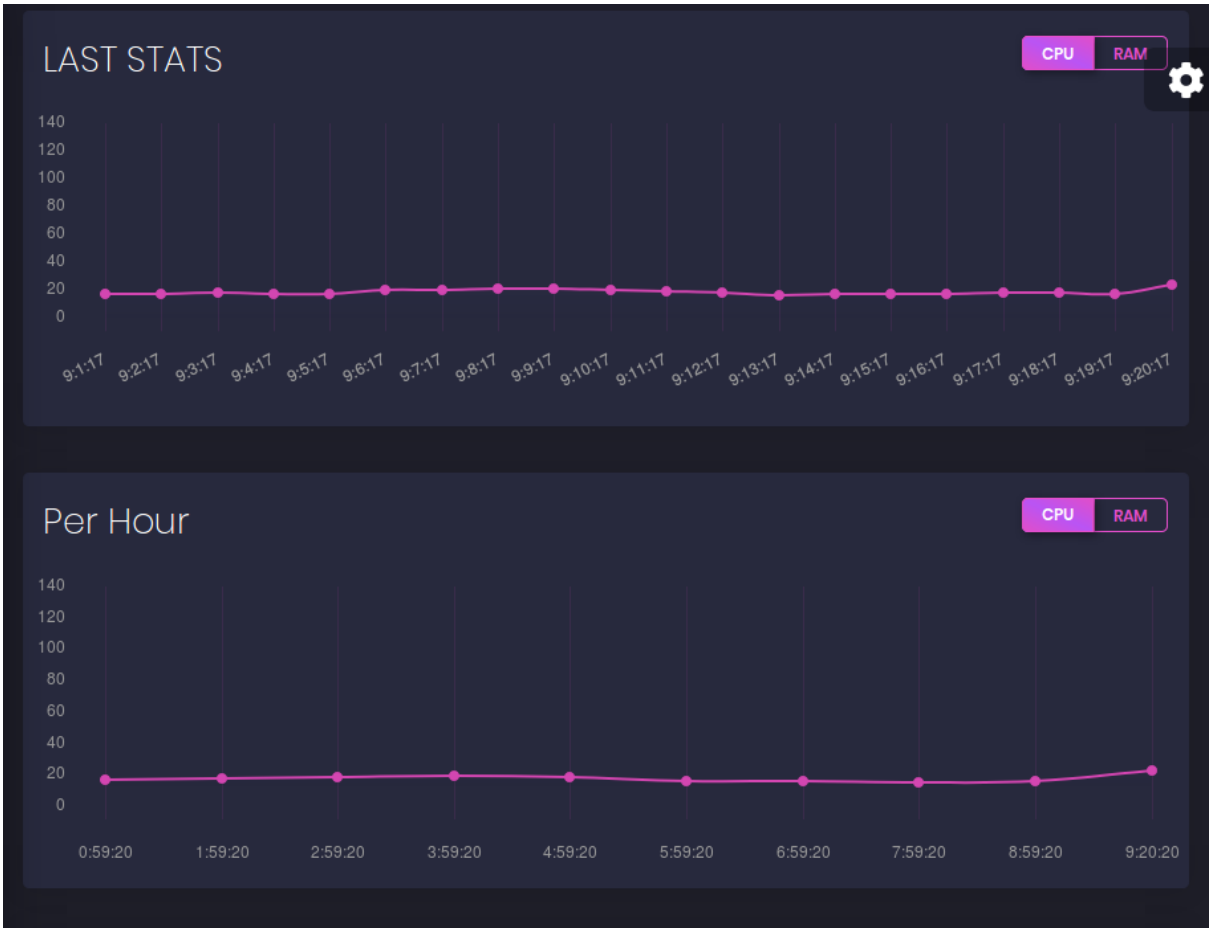


Figure 4. 6:Visualisation la consommation de cpu dans (l') les dernières heures

CPU USAGE	RAM USAGE	TIME
17.4813793103	69.316	2019-September-4-Wednesday 9:1:17
17.3268965517	69.3700344828	2019-September-4-Wednesday 9:2:17
18.0920689655	69.419862069	2019-September-4-Wednesday 9:3:17
17.8375862069	69.2719655172	2019-September-4-Wednesday 9:4:17
17.8079310345	69.2891724138	2019-September-4-Wednesday 9:5:17
20.7268965517	69.3834827586	2019-September-4-Wednesday 9:6:17
20.4906896552	69.4285172414	2019-September-4-Wednesday 9:7:17
21.0403448276	69.5200689655	2019-September-4-Wednesday 9:8:17

Figure 4. 7:Les données collectées dans les dernières minutes

La configuration est la même pour les trois agents. Pour cela, dans le reste des scénarios nous allons présenter les Dashboard des autres types des conteneurs de virtualisation.

3. Scénario n°2 : Déploiement d'un agent dans une environnement OSGI

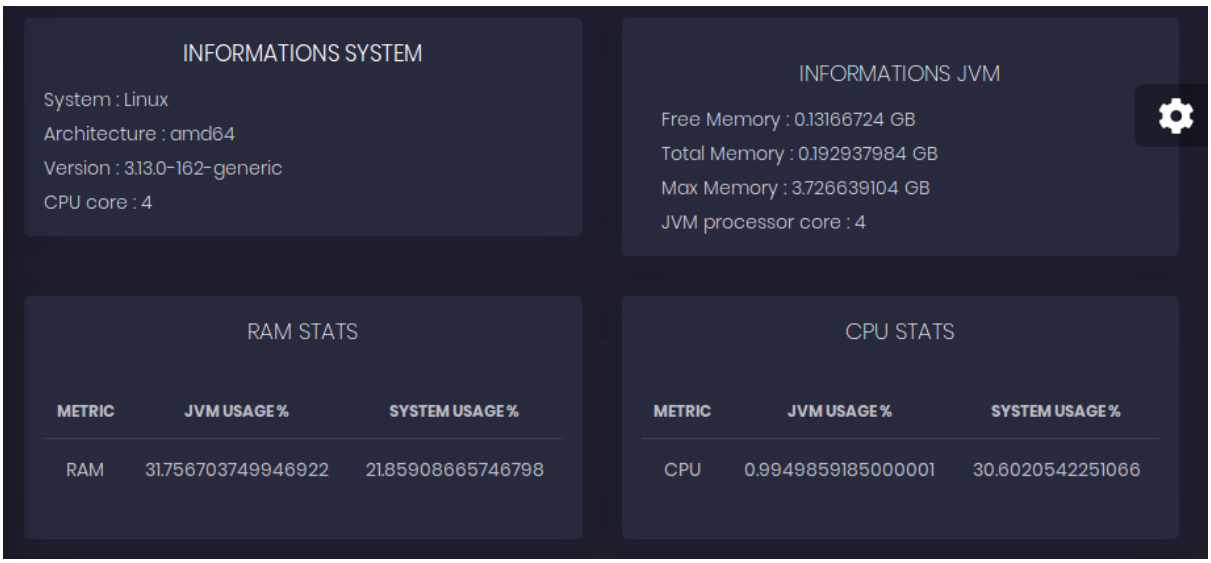


Figure 4. 9:Informations sur l'environnement agent

Figure 4. 8:Informations sur la consommation de la Heap mémoire de la JVM

NON HEAP MEMORY STATS

MEMORY	INIT GB	USED GB	COMMITTED GB	MAX GB
Code Cache	0.002555904	0.007744256	0.007744256	0.25165824
Compressed Class Space	0	0.002507032	0.002507032	1.073741824
Metaspace	0	0.021335752	0.021335752	-1
Summary	0.002555904	0.03158704	0.03407872	-1

-1: Can't retrieve data from source

Figure 4. 11:Informations sur la consommation de la non Heap mémoire de la JVM

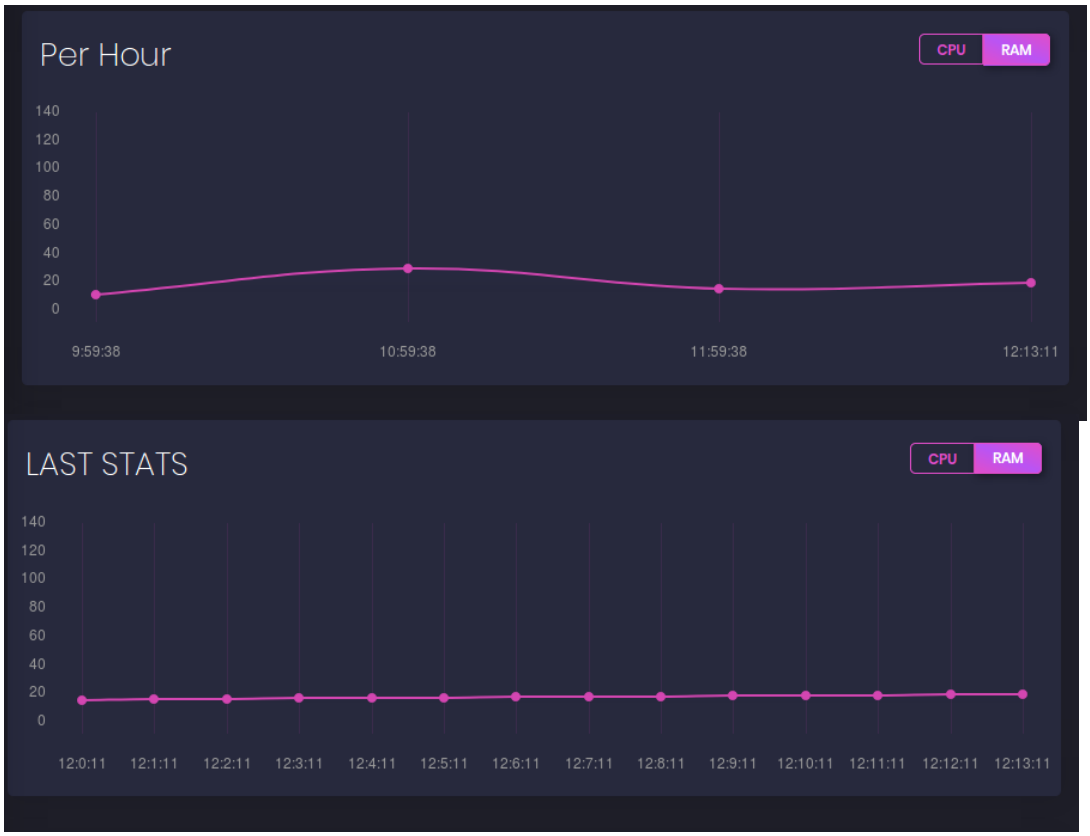
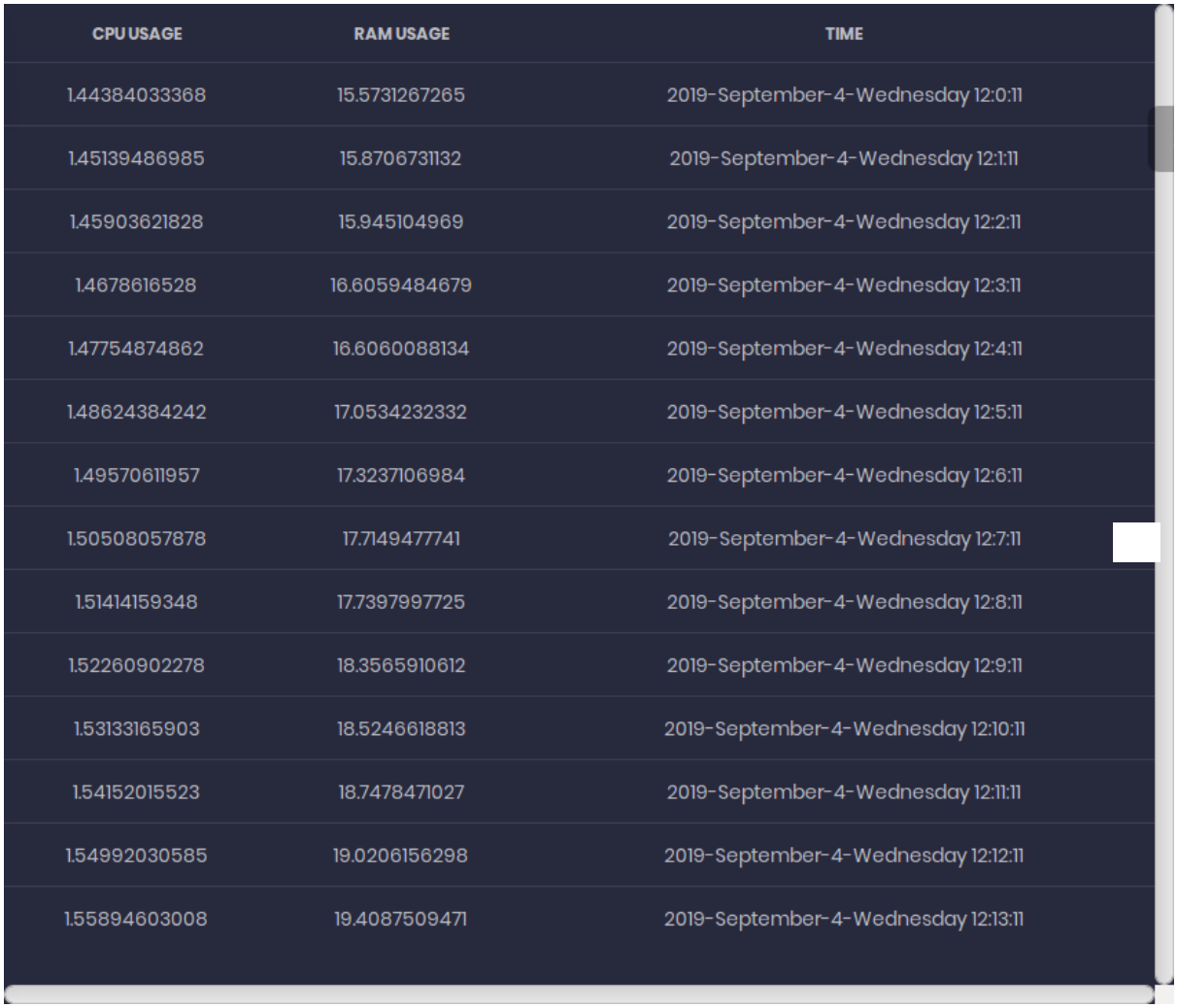


Figure 4. 10:Visualisation la consommation de la ram dans (l') les dernières heures



CPU USAGE	RAM USAGE	TIME
1.44384033368	15.5731267265	2019-September-4-Wednesday 12:0:11
1.45139486985	15.8706731132	2019-September-4-Wednesday 12:1:11
1.45903621828	15.945104969	2019-September-4-Wednesday 12:2:11
1.4678616528	16.6059484679	2019-September-4-Wednesday 12:3:11
1.47754874862	16.6060088134	2019-September-4-Wednesday 12:4:11
1.48624384242	17.0534232332	2019-September-4-Wednesday 12:5:11
1.49570611957	17.3237106984	2019-September-4-Wednesday 12:6:11
1.50508057878	17.7149477741	2019-September-4-Wednesday 12:7:11
1.51414159348	17.7397997725	2019-September-4-Wednesday 12:8:11
1.52260902278	18.3565910612	2019-September-4-Wednesday 12:9:11
1.53133165903	18.5246618813	2019-September-4-Wednesday 12:10:11
1.54152015523	18.7478471027	2019-September-4-Wednesday 12:11:11
1.54992030585	19.0206156298	2019-September-4-Wednesday 12:12:11
1.55894603008	19.4087509471	2019-September-4-Wednesday 12:13:11

Figure 4.13 : Les données collectées dans les dernières minutes

4. Scénario n°1: Déploiement d'un agent dans un conteneur Docker

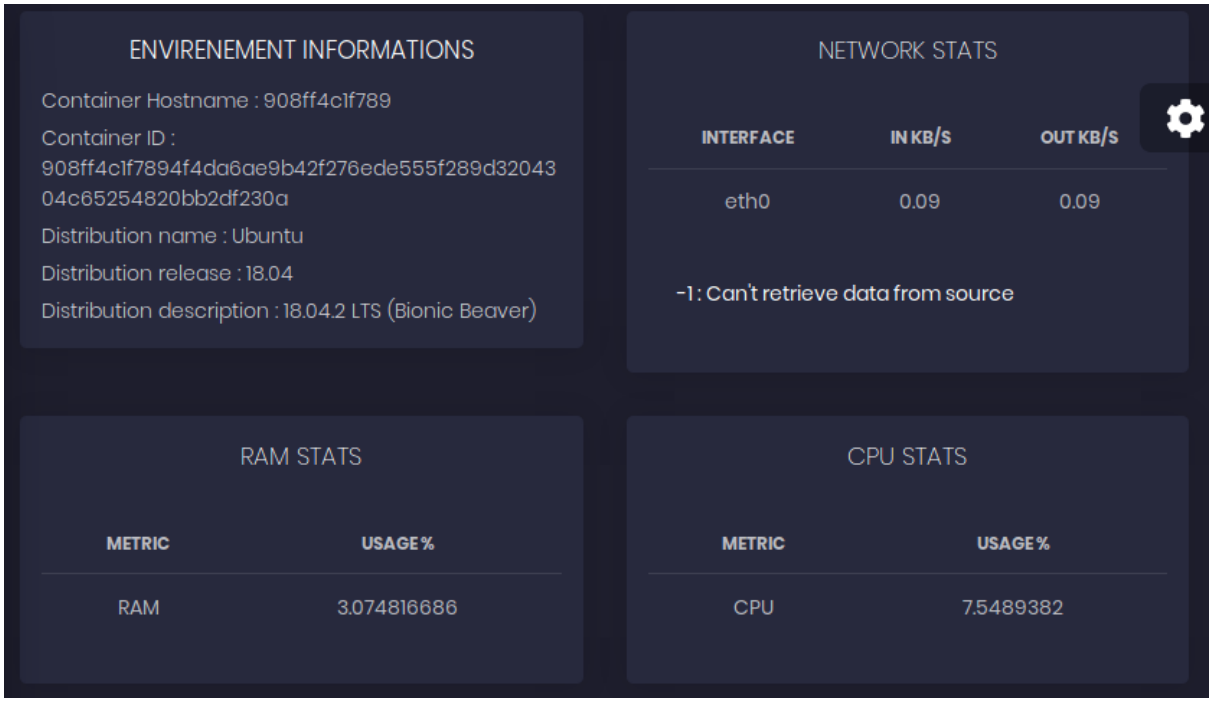


Figure 4.14: Information sur L'environnement du l'agent (Docker)

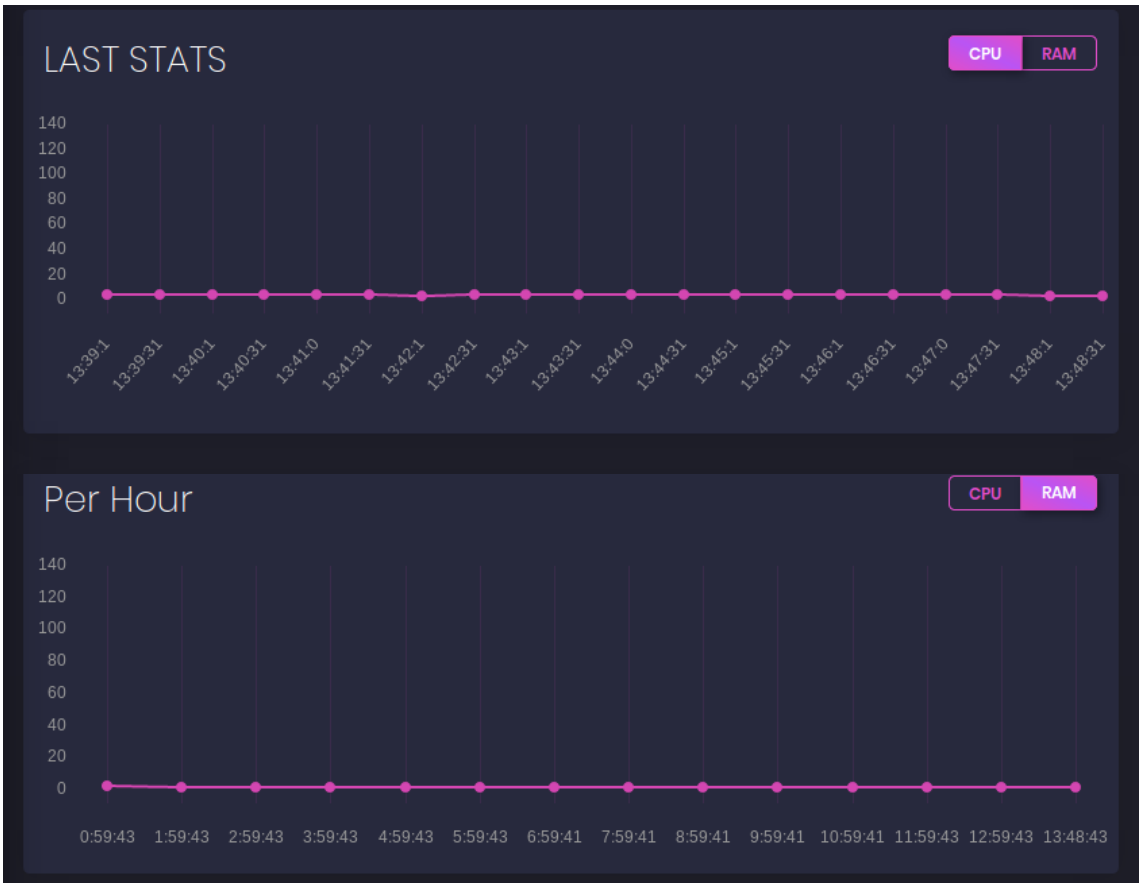
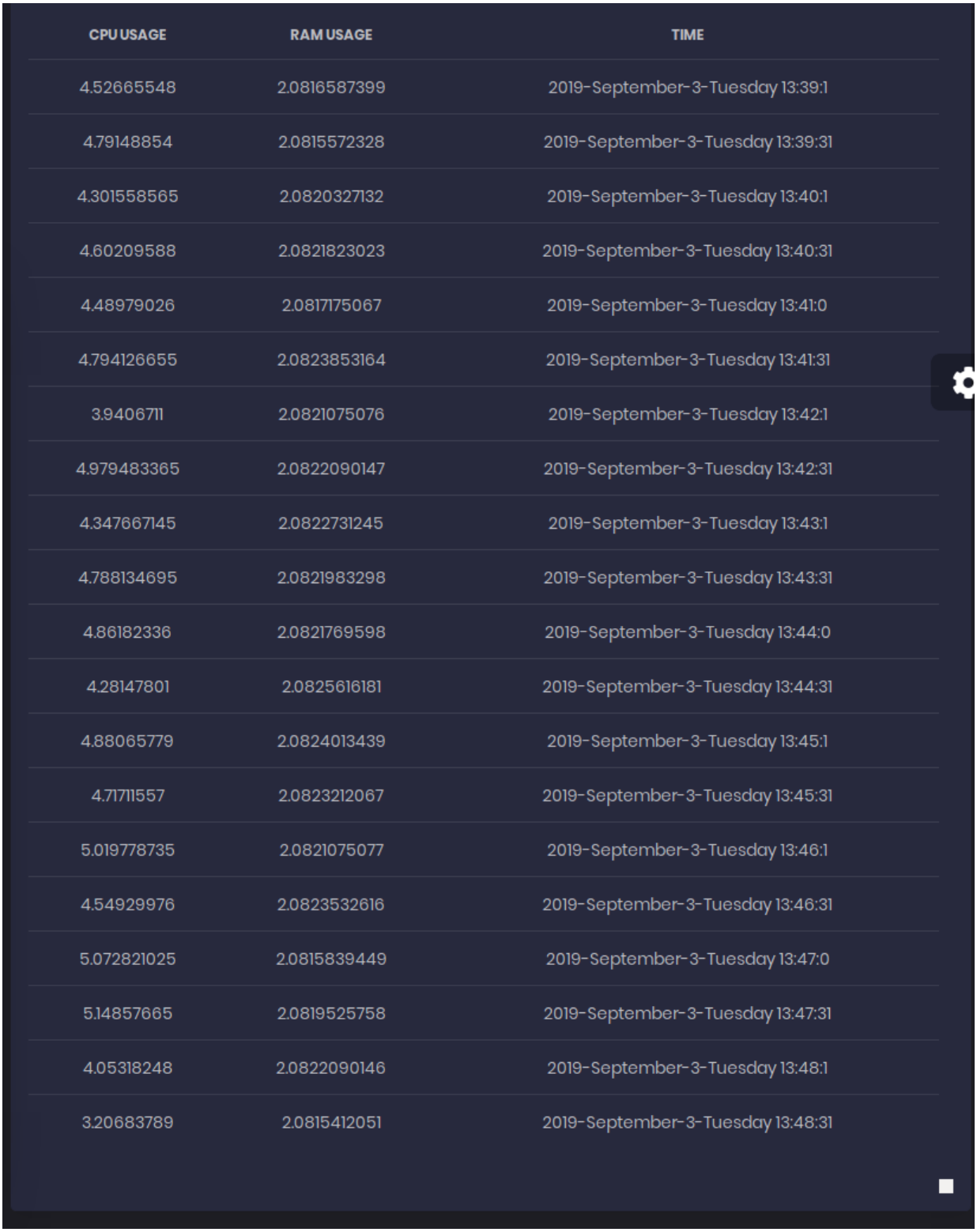


Figure 4.15 : Visualisation des données collectées



CPU USAGE	RAM USAGE	TIME
4.52665548	2.0816587399	2019-September-3-Tuesday 13:39:1
4.79148854	2.0815572328	2019-September-3-Tuesday 13:39:31
4.301558565	2.0820327132	2019-September-3-Tuesday 13:40:1
4.60209588	2.0821823023	2019-September-3-Tuesday 13:40:31
4.48979026	2.0817175067	2019-September-3-Tuesday 13:41:0
4.794126655	2.0823853164	2019-September-3-Tuesday 13:41:31
3.9406711	2.0821075076	2019-September-3-Tuesday 13:42:1
4.979483365	2.0822090147	2019-September-3-Tuesday 13:42:31
4.347667145	2.0822731245	2019-September-3-Tuesday 13:43:1
4.788134695	2.0821983298	2019-September-3-Tuesday 13:43:31
4.86182336	2.0821769598	2019-September-3-Tuesday 13:44:0
4.28147801	2.0825616181	2019-September-3-Tuesday 13:44:31
4.88065779	2.0824013439	2019-September-3-Tuesday 13:45:1
4.71711557	2.0823212067	2019-September-3-Tuesday 13:45:31
5.019778735	2.0821075077	2019-September-3-Tuesday 13:46:1
4.54929976	2.0823532616	2019-September-3-Tuesday 13:46:31
5.072821025	2.0815839449	2019-September-3-Tuesday 13:47:0
5.14857665	2.0819525758	2019-September-3-Tuesday 13:47:31
4.05318248	2.0822090146	2019-September-3-Tuesday 13:48:1
3.20683789	2.0815412051	2019-September-3-Tuesday 13:48:31

Figure 4.16: les données collectées sur la consommation de la CPU et la RAM

III. Conclusion

Ce chapitre s'est focalisé sur l'architecture REST adoptée pour l'implémentation es composants de monitoring. Il a présenté en premier lieu, les principaux outils utilisés pour implémenter la solution et traiter ses données, et en deuxième lieu, les tests de validation de déploiement et la visualisation des données, et cela dans les trois environnements.

Les tests non fonctionnels n'ont pas été présentés à cette étape de développement de la solution proposée. Ils constituent une perspective directe de ces travaux dans l'optique (notamment) d'évaluer l'impact de notre solution de monitoring sur les performances du système.

Conclusion Générale

Ce mémoire a été réalisé dans le cadre de mon projet de fin d'étude ingénieur réalisé au LAAS du CNRS, dans le contexte général de la virtualisation et de la softwarization des réseaux et protocoles, actuellement en vogue dans l'Internet de objets. Mon travail a plus spécifiquement porté sur la proposition, le développement et la validation d'une approche de monitoring, basée agents, de fonctions réseau dématérialisées, typiquement des VNFs et de ANFs.

Deux axes ont formé l'épine dorsale de ce travail :

- La conception d'agents adaptés au monitoring des VNFs et ANFs.
- Le déploiement dynamique de ces agents selon le type des conteneurs de virtualisation, par le biais d'un système de gestion basé sur plusieurs composants fonctionnels, dont le principal est le Monitoring Manager (MM).

Ces différents éléments ont été conçus et modélisé (en UML), puis implémentés et testés.

L'architecture REST utilisée par les composants du système de monitoring permet de récupérer les données ou de configurer les agents à l'aide d'un ensemble d'appels prédéfinis et de schémas décrits dans les sections précédentes, peu importe du langage de programmation ou la plateforme.

Le système de monitoring propose trois bus de données séparés pour stocker les données dans une base de données : le premier pour les configurations, les notifications et les dernier pour les données collectées.

L'interface web du Monitoring Manager fournit un moyen efficace pour afficher tous les agents déployés par ce composant et visualiser les données qu'ils collectent. Cependant, cette interface joue aussi le rôle d'un configurateur pour faciliter la gestion et la configuration des agents par un administrateur.

Les perspectives ouvertes par notre travail sont nombreuses :

- La première consiste à améliorer le Monitoring Manager en ajoutant d'autre options pour le déploiement des agents.
- Une deuxième perspective consiste à améliorer l'option d'ajout des agents de monitoring personnalisés.

- Une troisième perspective consiste à rendre la visualisation des données en temps réel.
- Enfin, les tests non fonctionnels de notre système de monitoring sont à effectuer dans le but d'en mesurer l'impact sur les performances du système monitoré.

Bibliographie

[1] *Samir Medjiah and Christophe Chassot, on the enhancement of non-functional requirements for cloud-assisted middleware-based IoT and other applications, Octobre 2017.*

[2] *ETSI - Standards for NFV - Network Functions Virtualisation | NFV Solutions. Disponible sur : https://www.etsi.org/technologies/nfv, consulté le 22-09-2019.*

[3] *Butler, B. (2017, juillet 11). Quelles différences entre SDN et NFV. le monde informatique disponible sur : https://www.lemondeinformatique.fr/actualites/lire-quellesdifferences-entre-sdn-et-nfv-68784.html, consulté le 22-09-2019.*

[4] *Bazargan, Fatma & Yeun, Chan & Zemerly, Jamal. (2011). Understanding the security challenges of virtualized environments. 2011 International Conference for Internet Technology and Secured Transactions, ICITST 2011.*

[5] *Xen project, disponible sur : https://xenproject.org/, consulté le : 22-09-2019.*

[6] *Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action, disponible sur : https://portal.etsi.org/NFV/NFV_White_Paper.pdf, consulté le : 22-09-2019.*

[7] *openNetVM, disponible sur : https://github.com/sdnfv/openNetVM/wiki consulté le : 22-09-2019.*

[8] *Network Functions Virtualisation (NFV);Management and Orchestration disponible sur : https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf, consulté le : 22-09-2019*

[9] *Network Functions Virtualisation (NFV);Architectural Framework disponible sur https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf*

[10] *ETSI Industry Specification Group on Autonomic Network Engineering for the Selfmanaging Future Internet (ETSI ISG AFI) disponible sur : https://www.etsi.org/about/how-we-work/how-we-organize-our-work/industry-specification-groups-isgs, consulté le : 22-09-2019.*

[11] *Network Functions Virtualisation (NFV);Virtual Network Functions Architecture disponible sur https://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV- SWA001v010101p.pdf, consulté le : 22-09-2019.*

[12] *Mickael,Dorigny.it-connect, Monitoring : supervision et métrologie disponible sur :https://www.it-connect.fr/monitoring-supervision-et-metrologie/, consulté le :22-09-2019.*

[13] *Jérémie,POLITO,SUPINFO, Monitoring et supervision, disponible sur : https://www.supinfo.com/articles/single/2789-monitoring-supervision, consulté le : 22-09-2019.*

[14] *monitoring-fr disponible sur : https://www.monitoring-fr.org/, consulté le : 22-09-2019*

[15] *Verma, Dinesh. (2009). Monitoring.Principles of Computer Systems and Network Management. pp111-135*

[16] *Docker Inc., disponible sur :* https://www.docker.com, *consulté le : 22-09-2019*

[17] *http://www-igm.univ-mlv.fr/~dr/XPOSE2014/Docker/fonctionnement.html*

[18] *ALAOUI ECHERIF, A. (2019). Orchestration de fonctions réseaux applicatives Pour la gestion de la QoS dans l'IoT.Mémoire de master inédit,INSA Toulouse.*

[19] *Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.*

[20] *pallets projects, Flask, disponible sur : https://palletsprojects.com/p/flask/, consulté le : 22-09-2019*

[21] *Jean-Michel DOUDOUX, JMX (Java Management Extensions), disponible sur : http://www.jmdoudoux.fr/java/dej/chap-jmx.htm, consulté le : 22-09-2019*

[22] *how to do in java, Jersey (JAX-RS) Tutorials disponible sur : https://howtodoinjava.com/jersey-jax-rs-tutorials/, consulté sur : 22-09-2019*

[23] *Apache Flume, Flume 1.9.0 User Guide disponible sur : https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html, consulté le : 22-09-2019*

[24] *Hadoop, Apache Hadoop 3.2.0, disponible sur https://hadoop.apache.org/docs/r3.2.0/, consulté le : 22-09-2019*

ANNEXES

ANNEXE A : CADRE DE DÉROULEMENT DU STAGE

Présentation de l'organisme d'accueil :

Le stage de fin d'étude a été effectué au sein du Laboratoire d'analyse et d'architecture des systèmes (LAAS-CNRS), plus spécifiquement à l'équipe SARA, dans le cadre de l'une des activités majeures traitent des différents aspects relatifs à l'Internet des objets (IoT).

Le LAAS-CNRS est un laboratoire de recherche scientifique qui axe son activité autour de quatre grandes disciplines : l'informatique, la robotique, l'automatique et les micros et nano systèmes. Le laboratoire comprend 8 départements scientifiques et regroupe 26 équipes de recherche, unités de base de la recherche du laboratoire. Le LAAS travaille sur divers types de systèmes : micro et nano systèmes, systèmes embarqués, systèmes intégrés, systèmes répartis à large échelle, systèmes biologiques, systèmes mobiles, systèmes autonomes et infrastructures critiques, ayant des domaines d'applications tels que : l'aéronautique, l'espace, le transport, l'énergie, les services, la santé, les télécommunications, l'environnement, la production et la défense.

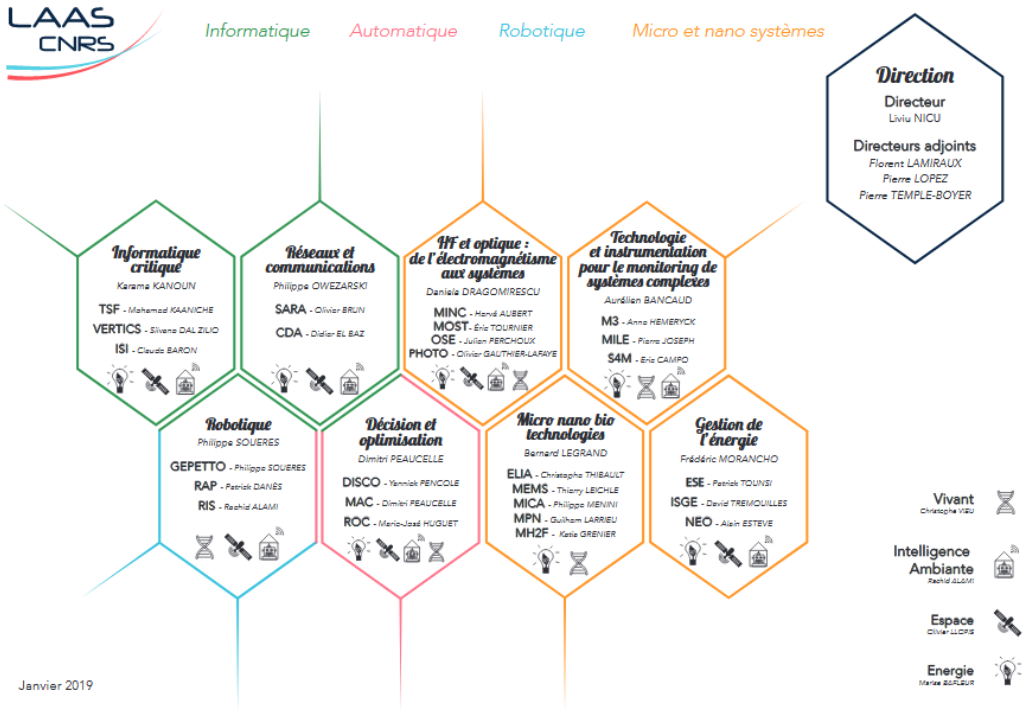


Figure A. 1:organisme de LAAS.

Le CNRS est associé par convention à 6 autres membres fondateurs de la COMUE Université de Toulouse :

- L'Université Toulouse III Paul Sabatier (UPS)
- L'Institut National des Sciences Appliquées de Toulouse (INSA)
- L'Institut National Polytechnique de Toulouse (INPT)
- L'Université Toulouse - Jean Jaurès (UT2J)
- L'Université Toulouse I Capitole (UT1)
- L'Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO)

1.2. Équipes de Recherche :

Le LAAS-CNRS compte 21 équipes (Figure) de recherche qui sont :

- Microsystèmes d'Analyse MICA,
- Matériaux et Procédés pour la Nanoélectronique MPN,
- Nano Ingénierie et Intégration des Systèmes N2IS,
- Intégration de Systèmes de Gestion de l'Énergie ISGE,
- Photonique PHOTO,
- Microondes et Opto-microondes pour Systèmes de Télécommunications MOST,
- Micro et Nano systèmes pour les Communications sans fils MINC,
- Nano bio Systèmes NBS,
- Méthodes et Algorithmes en Commande MAC,
- Diagnostic, Supervision et Conduite DISCO,
- Modélisation, Optimisation et Gestion Intégrée de Systèmes d'Activités MOGISA,
- Tolérance aux fautes et Sûreté de Fonctionnement Informatique TSF,
- Vérification des Systèmes Temporisés Critiques VERTICS,
- Services et Architectures pour Réseaux Avancés SARA,
- Ingénierie Système et Intégration ISI,
- Calcul Distribué et Asynchronisme CDA,
- Robotique, Action et Perception RAP,

- Robotique et Interactions RIS,
- Mouvement des Systèmes Anthropomorphes GEPETTO,
- Micro et nano-systèmes Hyperfréquences Fluidiques MH2F,
- Optoélectronique pour les Systèmes Embarqués OSE.

Départements de recherche :

Les recherches menées au LAAS-CNRS visent une compréhension fondamentale des systèmes complexes tout en considérant l'usage qui peut en découler. A l'inverse, de nombreuses problématiques sociétales ou industrielles, par exemple dans les domaines de l'aéronautique, de l'espace, de la santé, de l'énergie ou des réseaux de communication soulèvent des questions fondamentales qui nourrissent à leur tour l'inspiration des chercheurs.

Ainsi, à la fois défricheur de problématiques émergentes et promoteur de solutions intégrées, le laboratoire a identifié trois axes stratégiques fondés sur les quatre champs disciplinaires qui constituent la marque de fabrique du laboratoire depuis sa création (informatique, robotique, automatique et micro et nano systèmes) :

- Adream : Architectures dynamiques reconfigurables pour systèmes embarqués autonomes mobiles,
- Alive : Analyses des interactions avec le vivant et l'environnement,
- Synergy : Systèmes pour une gestion intelligente de l'énergie.

Au sein de ces disciplines, 8 départements scientifiques définissent les orientations des prochaines années :

- Informatique critique,
- Réseaux et communication,
- Robotique,
- Décision et optimisation,
- Gestion de l'énergie,
- Micro nano biotechnologies,
- Micro Nano systèmes RF et optiques,

- Nano Ingénierie et intégration.

1.4. Département Réseaux et Communication :

Le département Réseaux et Communication est composé de deux équipes de recherche : l'équipe SARA (Services et Architectures pour les Réseaux Avancés) et l'équipe CDA (Calcul Distribué et Asynchronisme).

Le périmètre scientifique de ces deux équipes est le suivant :

- Objets des recherches :
  - Réseaux, systèmes de communication de nouvelle génération, leurs applications.
- Objectifs des études :
  - Maîtrise de la conception, planification, gestion du déploiement, supervision.
- Contributions :
  - Élaboration de méthodes, de modèles et d'outils, proposition d'architectures, de protocoles et de services.
  - Analyse, évaluation des performances, contrôle et prototypage des logiciels et des plates-formes de communication.
- Défis :
    - Modélisation et contrôle des systèmes dynamiques à grande échelle.
    - Conception de systèmes à fortes exigences (fonctionnelles, non-fonctionnelle) et à fortes contraintes (énergie, ressources)
    - Continuité de service et de sa qualité dans les réseaux de systèmes mobiles.
    - Architectures logicielles dynamiques et autonomiques orientées services, composants et événements.

Ces deux équipes travaillent également sur les Plateformes expérimentales suivantes :

- Architectures protocolaires et mécanismes
- Qualité de service, Sécurité et Supervision
- Réseaux de capteurs, Internet des objets, M2M
- Réseaux de grande taille, réseaux dynamiques

1.5. Équipe SARA :

Le stage s'est déroulé au sein de l'équipe SARA (Services et Architectures pour Réseaux Avancés). Il a été conjointement encadré par M. Christophe CHASSOT et M. Samir MEDJIAH, tous deux enseignants-chercheurs au sein de l'équipe SARA.
L'équipe SARA est sous la responsabilité de M. Khalil DRIRA. Cette équipe est l'une des équipes du thème Réseaux et Communication.

Les travaux de l'équipe SARA concernent les réseaux, les systèmes de communication de nouvelle génération et leurs applications. Les études visent la maîtrise de leur conception, planification, gestion du déploiement, supervision.

Les contributions de l'équipe portent notamment sur l'élaboration de méthodes, de modèles et d'outils ainsi que sur la proposition d'architectures, de protocoles et de services. En particulier, les membres de cette équipe s'intéressent à l'analyse, à l'évaluation des performances, au contrôle et au prototypage des logiciels et des plates-formes de communication.

Les défis comportent notamment la maîtrise :

- de la modélisation et du contrôle des systèmes dynamiques à grande échelle ;
- de la conception de systèmes à fortes exigences (Qualité de Service, sécurité) et à fortes contraintes (énergie, ressources) ;
- de la continuité de service et de sa qualité dans les réseaux de systèmes mobiles
- des architectures logicielles dynamiques et autonomiques orientées services et composants.

Les thématiques de recherches explorées sont :

- la supervision, la modélisation et l'analyse du trafic : l'objectif est la conception et l'analyse de modèles mathématiques permettant de prédire, d'optimiser et de contrôler les performances des réseaux de nouvelle génération. Les contributions sont à la fois de nature théorique, avec la conception de nouvelles approches de modélisation stochastique ou de contrôle distribué, et appliquée, avec le développement d'outils de métrologie et de logiciels pour l'étude des réseaux ;
- les architectures de communication autonomiques : les membres de cette équipe considèrent à la fois les contextes fortement contraints et les contextes hétérogènes, dynamiques et à grande échelle. Il s'agit de traiter de façon cohérente à la fois les exigences des applications que les réseaux supportent et les contraintes des environnements dans lesquels ils doivent opérer ;

- les mécanismes des protocoles et des services adaptatifs et garantis : l'objectif est de concevoir, pour un niveau de communication donné, des protocoles et des mécanismes élémentaires de nature à fournir de manière adaptative ou garantie un service de communication à un niveau de communication supérieur, au niveau applicatif ou au niveau de l'utilisateur final ;

- la planification et optimisation des réseaux : Les études portent d'une part sur l'optimisation du routage en vue de garantir une qualité prédéfinie, sur le dimensionnement à moindre coût des équipements, ainsi que sur la recherche de topologies optimales. D'autre part, ils travaillent sur l'optimisation du déploiement des nouveaux réseaux, qu'ils soient sans infrastructure ou même spontanés sans organisation ni configuration préalable ;

- le développement des plates-formes expérimentales : L'objectif est de fournir le support technique pour expérimenter et valider les architectures protocolaires et mécanismes réseaux issus de la recherche, principalement sur des problématiques de qualité de service, de sécurité et de supervision dans les réseaux de capteurs, l'Internet des objets, les réseaux de grande taille et les réseaux dynamiques.

ANNEXE B : LE PROJET SDCI

Au cours des années précédentes, un nouveau concept, l'internet des objets (IoT), a été largement traité dans les domaines de recherche, l'équipe SARA vise à maîtriser la conception, la planification, et la gestion du déploiement des systèmes de communication de nouvelle génération (IoT) et leurs applications.

Ainsi, une architecture Machine-to-Machine (M2M) [2] a été proposée pour ce système, cette architecture se base sur 4 niveaux (Figure 2) :

- Niveau d'application où les applications IoT métiers situés, offrant des solutions finales.
- Niveau du middleware (MW) qui vise à cacher les détails de divers réseaux sous-jacents et technologies pour faciliter l'interopérabilité.
- Niveau de réseau comprenant différents types de réseaux à interconnecter les équipements.
- Niveau d'équipement incluant les dispositifs capteurs / actionneurs IoT.

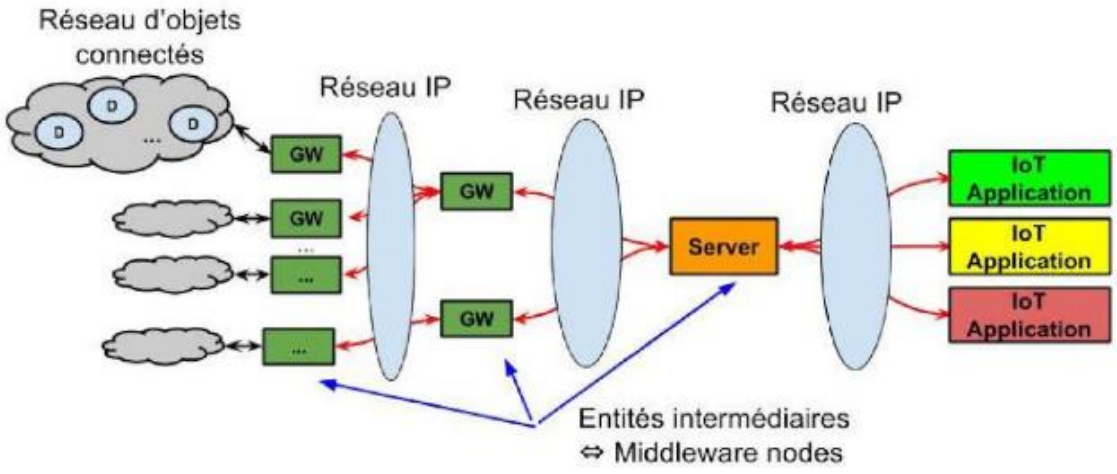


Figure B. 1:Le middleware dans l'IoT

Pour permettre à l'Internet des objets et à son architecture d'atteindre leur plein potentiel et concrétiser les scénarios qui en découlent, plusieurs aspects doivent être étudiés. En effet, les

concepts prometteurs imaginés par la communauté scientifique nécessitent de résoudre un certain nombre de problématiques : gestion de la qualité de service (QoS), hétérogénéité, impact du monde physique (grands aux données continues, variabilité de l'environnement). Ainsi, plusieurs défis déjà abordés dans l'Internet "classique" doivent être pris en compte, en particulier la performance de l'ensemble du système de communication (y compris les couches réseau et la couche Middleware) en réponse aux exigences de QoS au niveau de l'application, par exemple, en termes de haute disponibilité du service ou de temps de réponse limité.

Dans ce contexte, l'avènement des technologies dites de « Virtualisation » initialement liées au Cloud permet désormais d'envisager le déploiement des traitements et mécanismes non seulement sur des équipements dédiés (typiquement un Middleware dans le monde de l'IoT), mais en privé ou en des Datacenters publics avec des hyperviseurs offrant les capacités fonctionnelles requises. Pour cela, l'approche proposée avec le projet Software Defined Communication Infrastructure (SDCI) [1] consiste à concevoir, développer et expérimente des modèles d'architectures logicielles génériques pour une gestion auto adaptative de la QoS aux différents niveaux du système de communication, en :

- tirant partie des opportunités technologiques liées au déploiement dynamique de fonctions de réseau, des réseaux programmables et de l'Autonomic Computing ;
- tenant compte de l'hétérogénéité des solutions en cours de déploiement ;
- assurant la cohérence des choix de configuration effectués aux différents niveaux par le biais d'outils théoriques adéquats.

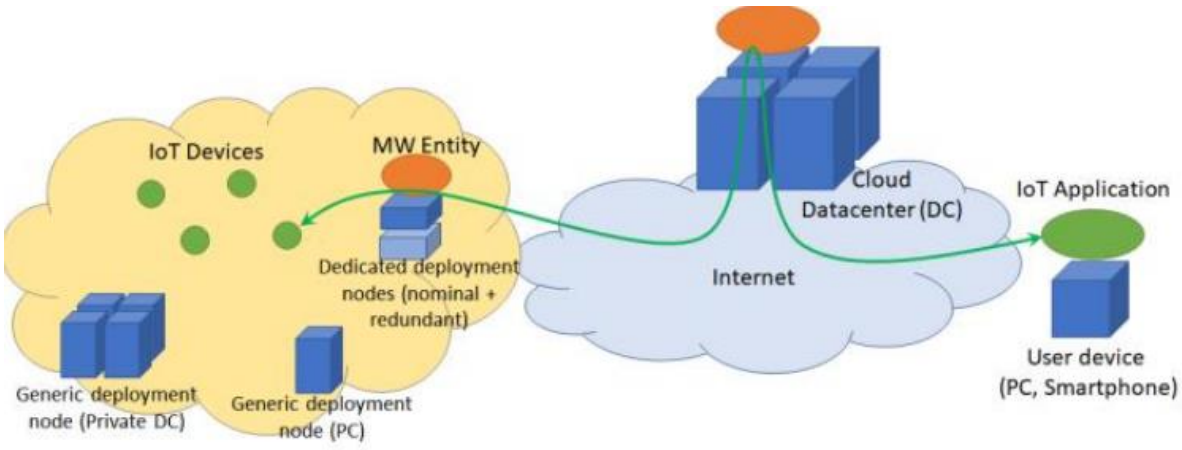


Figure B. 2:Infrastructure de déploiement typique des solutions IoT basées sur le middleware assisté par Cloud.

De plus, le déploiement basé sur le cloud peut également permettre de répondre à des besoins fonctionnels, par exemple, par le déploiement dynamique de fonctions de traitement de données / paquets implémentées dans des logiciels précédemment intégrés dans des conteneurs de virtualisation (des machines virtuelles ou des conteneurs système : LXC, Docker, etc.) (Figure B.3). Par exemple, une application peut avoir besoin que son format de données soit converti d'un format texte (par exemple JSON) en un format binaire (par exemple BSON) avant d'être échangé par le MW afin d'optimiser le transfert de données et par la destination finale. Un autre exemple est le déploiement de la fonction d'agrégation de messages entre l'application et le MW afin d'adapter les communications MW aux réseaux sous-jacents (par exemple, les réseaux satellites avec de grandes trames de données).

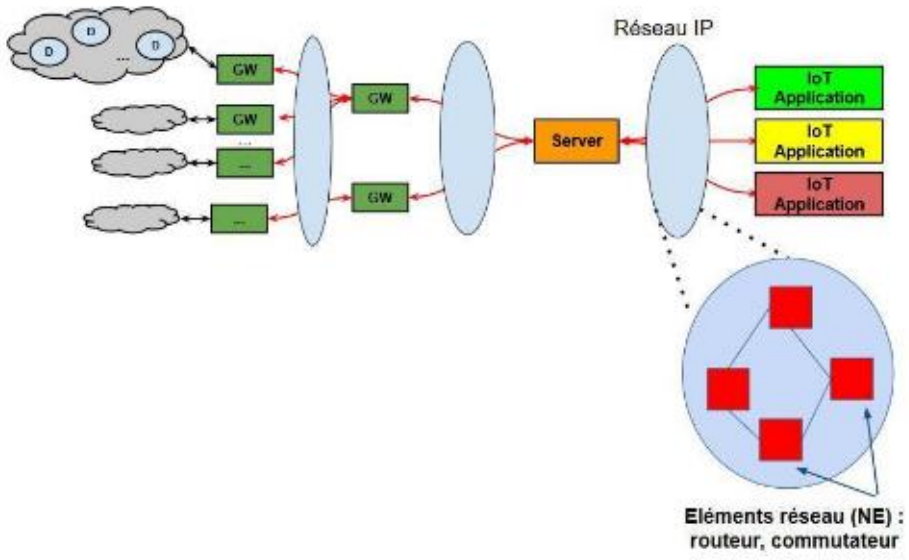


Figure B. 3:Architecture M2M en vision sur la couche réseau

ANNEXE C : DIAGRAMME DE SÉQUENCE

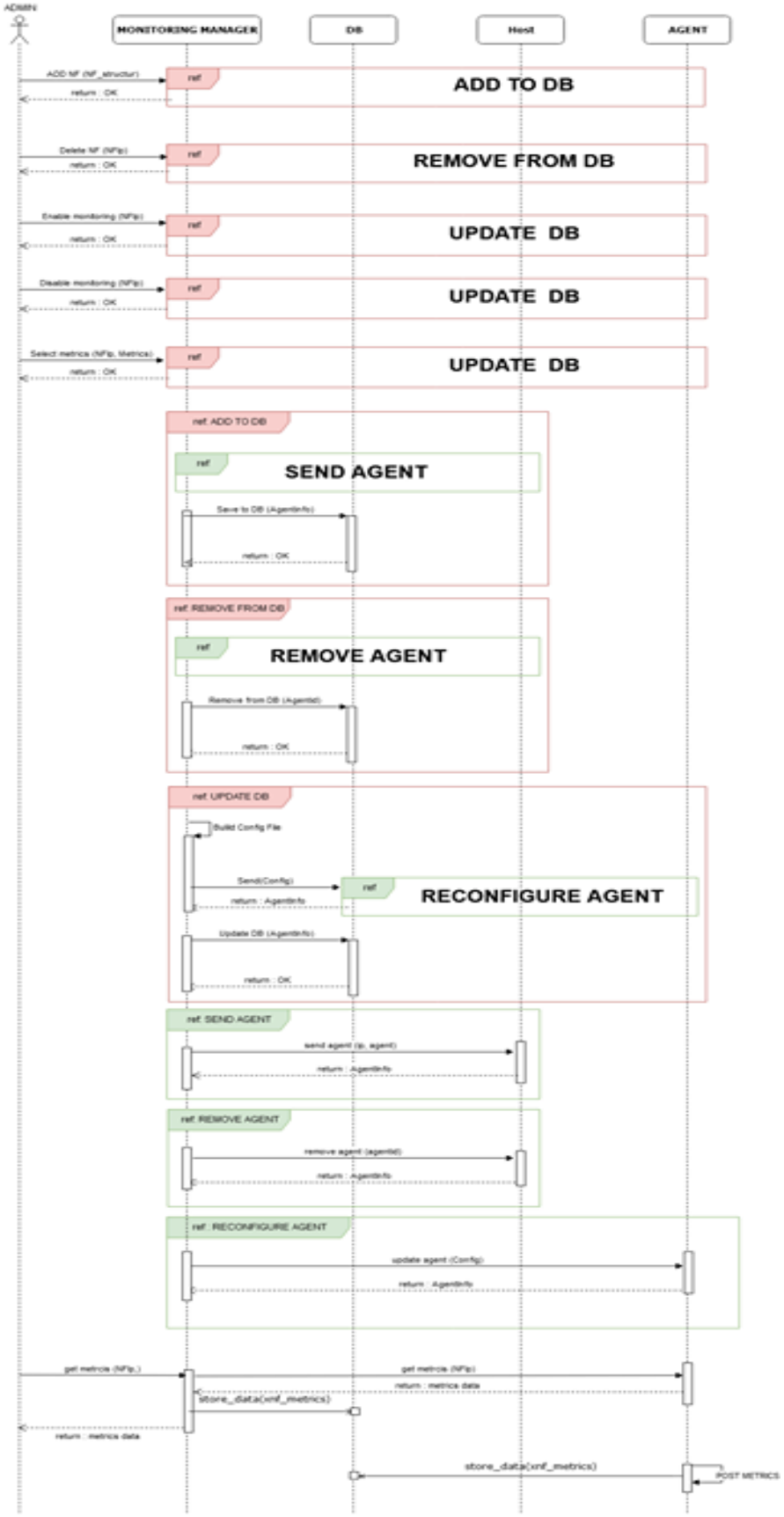


Figure C. 1:DIAGRAMME DE SÉQUENCE

ANNEXE D : Terminologie

Agent de monitoring : un agent de Monitoring est une application résidant dans un périphérique et chargée de collecter un ensemble d'informations sur des métriques prédéfinies

Applicative Network Functions (ANF) : est une fonction réseau déployable sous forme de logiciel sur des plateformes compatibles à une structure modulaire installée sur un matériel standard

Bundle OSGi : est un fichier d'archive Java contenant du code Java, des ressources et un manifeste décrivant le bundle et ses dépendances.

Cloud computing : est un modèle qui permet un accès omniprésent, pratique et à la demande à un réseau partagé et à un ensemble de ressources informatiques configurables (comme par exemple : des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être provisionnées et libérées avec un minimum d'administration

Conteneur : est un environnement virtuel résultant d'une isolation d'un ensemble des processus et des ressources (CPU, mémoire, réseau, etc.) afin d'exécuter séparément une tâche bien spécifique.

Conteneur de Virtualisation : Un Conteneur de Virtualisation est défini par le groupe ETSI comme étant une partition d'un nœud de calcul qui fournit un environnement de calcul virtualisé isolé.

Docker : est un logiciel open source qui permet d'automatiser le déploiement de conteneurs sur les OS Linux, Windows et Mac en accédant aux fonctionnalités des noyaux des systèmes d'exploitation. Docker est constitué de 2 principaux projets : Docker Engine (création/exécution des conteneurs Docker) et Docker Compose (définition d'applications à plusieurs conteneurs). Les containers Docker suivent le standard d'OCI (Open Container Initiative) de la fondation Linux.

Edge computing : est une architecture informatique répartie dans laquelle les données client sont traitées à la périphérie du réseau, aussi près que possible de la source d'origine

Firmware : est un programme intégré dans un matériel tel qu'un clavier, un disque dur, un BIOS ou une carte vidéo. Il est conçu pour donner des instructions permanentes pour

communiquer avec d'autres périphériques dans un système et exécuter des fonctions comme des tâches d'entrée / sortie de base.

Fog computing : étend le cloud pour se rapprocher de ce qui produit et agit sur les données IoT. Ces dispositifs, appelés nœuds de Fog, peuvent être déployés n'importe où avec une connexion réseau : en usine, au sommet d'un poteau électrique, le long d'une voie ferrée, dans un véhicule ou sur une plate-forme pétrolière. Tout périphérique doté d'une connectivité informatique, de stockage et réseau peut être un nœud de Fog.

Framework OSGI (Open Service Gateway Initiative) : est un framework Java permettant de développer et de déployer des programmes logiciels modulaires (bundle) et des bibliothèques.

Monitoring : est un processus consistant à superviser un système et d'obtenir les informations d'une manière permanentes des différents éléments de ce système et les consolider pour les analyser [verma2009.pdf] et les tracer.

Monitoring Manager : est le composant central du monitoring qui est responsable à créer des agents de monitoring à partir d'un descripteur (section 3.3) et les configurer. Ce composant gère aussi le déploiement et la modification de la configuration des agents de monitoring.

Métrique : est un attribut mesuré sur un système qui permet de mieux le décrire, et de mieux le caractériser.

Métrique De Performance : un ensemble de paramètres pour évaluer un système.

Orchestrateur ANF : est conçue pour gérer et orchestrer des services réseau basés sur des composants logiciels déployés dans une infrastructure type multisite.

Orchestrateur VNF : bloc fonctionnel qui gère le cycle de vie du service de réseau (NS) et coordonne la gestion du cycle de vie du NS, du cycle de vie de VNF (pris en charge par le VNFM) et des ressources NFVI (pris en charge par le VIM) afin de garantir une allocation optimisée des ressources et la connectivité nécessaires

SDN : La définition académique, qui a depuis largement évolué, consistait à voir le SDN comme une architecture qui découpait les fonctions de contrôle et de transfert des données du

réseau (data plane) afin d'avoir une infrastructure physique complètement exempte de tout service réseau.

Virtualisation: est une technologie qui permet de créer des services informatiques utiles à l'aide de ressources qui sont généralement liées au matériel. Elle permet d'exploiter toute la capacité d'une machine physique en la répartissant entre de nombreux utilisateurs ou environnements différents grâce à une couche d'abstraction qu'on l'appelle moniteur de machine virtuelle (VMM) ou hyperviseur.

Virtual Network Functions (VNF) : correspond aux fonctions réseaux indépendamment de leur langage de programmation et leur package.

VM : est un environnement de calcul virtualisé, qui simule un ordinateur/serveur physique avec tous ses composants (processeur, mémoire / stockage, interfaces / ports).