



Evolution de la Plateforme de Supervision et Optimisation du Réseau

Mémoire de Stage de Fin d'Etudes

2016-08-18

Auteur:
Thibaut Fabre

Responsables:
M. Arnaud Roudsovsky
M. Frédéric Gracia
M. Romain Ballan



Sommaire

Remerciements	1
Glossaire	2
Introduction	5
1. Contexte Industriel	7
1.1. SQLi Group	7
1.2. ISC France	7
1.3. Pôle CRCI	8
1.4. Infrastructure	10
2. Solution de surveillance et de mesure	12
2.1. Initiation du changement	12
2.2. Remise en cause de la solution actuelle	12
2.3. Environnement de test et mise en production	13
2.4. Apports et limites de cette nouvelle configuration	15
3. Solution de gestion des configurations	17
3.1. Recherche de la solution adaptée	17
3.2. Mise en place	17
3.3. Apports et limites de l'outil	18
Conclusion	19
Annexes	20
1. Virtualisation	20
2. Zabbix	23
2.1. Architecture Zabbix	23
2.2. Communication entre un serveur et un agent	24
2.3. Système d'alertes	26
Bibliographie	27

Remerciements

Je remercie l'entité ISC France de m'avoir offert la possibilité de travailler au sein de leur structure.

Je tiens à remercier également l'équipe du *CRCI* de l'entité pour leur accueil et leur soutien.

M. Arnaud Roudsovsky, responsable du pôle *CRCI*, m'a accordé sa confiance pour cette mission. Il m'a conseillé et m'a permis d'acquérir de l'expérience professionnelle.

Je remercie **M. Frédéric Gracia** et **M. Romain Ballan**, ingénieurs au sein du pôle de m'avoir fait confiance. Leurs expériences et leurs connaissances m'ont fait progresser dans de nombreux domaines. A leurs côtés, j'ai pu découvrir les diverses facettes du métier d'ingénieur réseaux et systèmes.

L'accueil, le soutien et les connaissances du pôle *Expertise* d'ISC France m'ont donné l'opportunité de me familiariser avec la culture *DevOps* et le *continuous delivery*.

La gentillesse et la disponibilité de tous les collaborateurs d'ISC France m'ont permis de me sentir intégré à l'entreprise.

Il me semble nécessaire de remercier **M. Kaninda Musumbu** pour son suivi tout au long de ce stage. Enfin, je remercie **M. Lionel Clément** pour son aide, sa présence et la qualité de ses réponses.

Glossaire

Ansible

logiciel open source permettant la configuration et la gestion à distance des machines.

Cloud Computing

paradigme récent ayant pour but de centraliser et d'utiliser au maximum la puissance de calcul de chaque serveur. Le principe est de limiter les dépenses en matériel et en ressource électrique. De nombreuses entreprises fournissent à leurs clients différents services issus de ce paradigme. Le Cloud Computing se décline en trois catégories de service, *Infrastructure As A Service* (IaaS), *Platforme As A Service* (PaaS) et *Software As a Service* (SaaS).

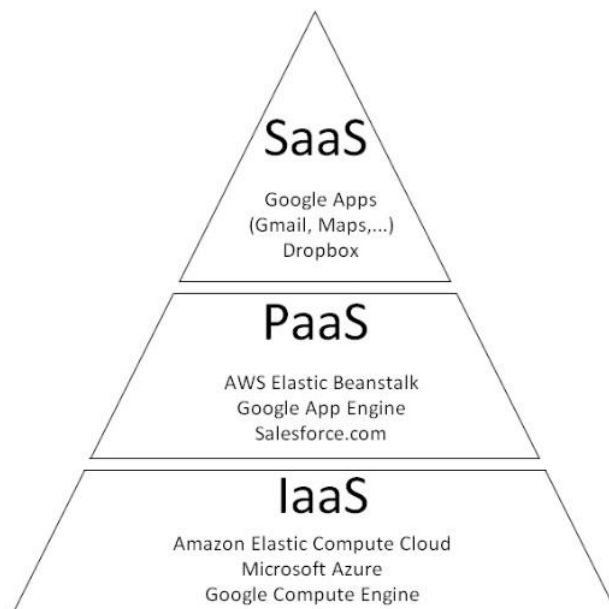


Figure 1: Catégorie des services du Cloud Computing et les solutions connues

Containerisation

sous-catégorie de la virtualisation. Il s'agit d'instancier uniquement le système d'exploitation de chaque environnement virtuel souhaité.

Continuous Delivery

approche de l'ingénierie logiciel visant à assurer le fonctionnement d'une application entre chacune de ses versions. Des cycles courts sont étudiés pour corriger le plus rapidement les éventuels bogues.

DevOps

mouvement informatique visant à donner une cohésion entre les équipes de développeurs et les équipes d'administrateurs au sein d'une même entreprise.

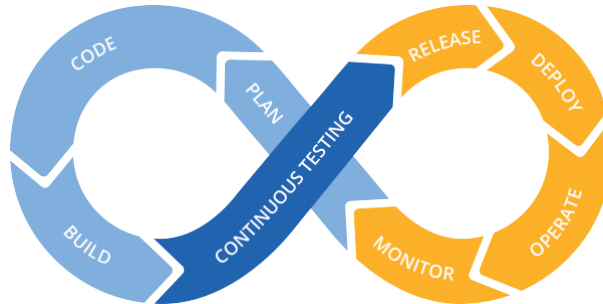


Figure 2: DevOps représenté en cycle

Docker

logiciel open source permettant de créer, gérer et détruire des environnements virtuels de type container.

Entreprise services du numérique (ESN)

société de services proposant des solutions à ces clients pour des missions d'ordre informatique, anciennement nommé société de services en ingénierie informatique (SSI).

InnoDB

moteur de stockage pour *MySQL* et *MariaDB*.

JavaScript Object Notation (JSON)

format de données textuelles issu du langage JavaScript

Paradigme

vision classant des notions informatiques par leur solution.

PowerShell

logiciel *Microsoft* intégrant une interface en ligne de commande, permettant d'interpréter les commandes utilisateurs, et le langage de script orienté objet PowerShell.

Openstack

ensemble de logiciels open source visant à déployer des architectures informatiques à l'aide de la virtualisation.

Redundant Array of Independent Disks (RAID)

ensemble de techniques de stockage offrant la répartition et la sauvegarde des données sur plusieurs disques durs afin d'améliorer les performances, la sécurité et la tolérance aux pannes.

Socket

une interface logicielle qu'utilise un développeur pour exploiter facilement les services d'un protocole réseau.

SNMP (Simple Network Management Protocol)

protocole réseau permettant aux administrateurs de gérer et monitorer tous les types de matériels réseaux.

SSH (Secure Shell)

protocole de communication sécurisée basé sur l'utilisation du couple clef privée/clef publique pour chiffrer et déchiffrer les communications entre deux communicants. L'échange de clefs publiques avant l'interaction est obligatoire.

Supervision

technique industrielle visant à surveiller le bon fonctionnement d'un système et à alerter les éventuelles pannes.

Virtualisation

technique visant à faire fonctionner plusieurs ordinateurs virtuels sur un seul ordinateur physique.

WinRM

protocole de gestion à distance, propriété de *Microsoft*.

YAML (YAML Ain't Markup Langage)

format représentant des données par sérialisation, à l'image du format *XML*.

Zabbix

logiciel open source permettant la surveillance et la mesure de l'ensemble des matériels réseaux compatibles d'une infrastructure.

Introduction

Dans une entreprise lambda, posséder des données sur ses finances, sur ses ressources matériels ou encore sur le dévouement de ses collaborateurs dans sa structure peut permettre d'optimiser le rendement et de résoudre certains problèmes. Dans une infrastructure système et réseaux, ces données sont primordiales pour sa stabilité. C'est dans ce but là que la *supervision* a été implémentée. Son principe est de surveiller le bon fonctionnement d'un système et d'alerter dans le cas contraire. L'intérêt de cette technique industrielle est de réagir au plus vite en cas d'erreur sur un environnement et d'utiliser ces données pour optimiser l'infrastructure.

C'est dans ce cadre que l'entreprise *ISC France* a eu le besoin de se développer. Ce centre de service rattaché à *SQLi Group* possède une infrastructure conséquente, grandissante et évolutive. Dans ce contexte d'expansion, les ingénieurs réseaux et systèmes de l'entité ont eu le besoin de faire évoluer leur plateforme de supervision. Le but de cette évolution est d'avoir un outil idéalement configuré pour mesurer le fonctionnement des environnements et pour alerter en cas de défaillance permettant une intervention rapide et précise. De plus avec les mesures rapportées, il est possible d'optimiser les environnements afin de minimiser le coût en ressource de chaque environnement et par conséquent le coût financier de l'infrastructure.

C'est à partir de cette problématique que j'ai été intégré à l'équipe des ingénieurs réseaux et systèmes. Mon objectif principal a été de faire évoluer la plateforme de supervision avec les nouveaux besoins de l'entité pour la rendre plus performante et plus stable à l'avenir. Pour y arriver, j'ai eu accès à tout le parc informatique, tous les documents utiles pour mener à bien mon projet et à la configuration de la plateforme de supervision qui était en fonctionnement. L'équipe m'a fourni un ordinateur avec le système d'exploitation *Ubuntu* en 16.04 LTS.

Depuis quelques mois, la direction de l'entité poussée par quelques collaborateurs passionnés essaie de changer la culture de l'entreprise en adoptant les grands préceptes de la mouvance *DevOps*. Pour comprendre, les nouveaux besoins auxquels l'entreprise se dirige, il m'a fallu comprendre les lignes de ce mouvement. Cette apprentissage était autant intellectuelle que technique. Il a fallu que je me forme sur les outils et les bonnes pratiques que l'entité a décidé d'utiliser et de pourvoir auprès de tous ces collaborateurs. L'apprentissage s'est fait au contact des équipes concernées en premier lieu par le sujet grâce à des discussions et à des exemples techniques.

En plus de l'objectif et de l'apprentissage des nouvelles techniques issues de la culture DevOps, le stage devait me servir d'initiation au métier d'ingénieur réseaux et système. J'ai donc dû aider l'équipe dans leurs tâches quotidiennes comme la gestion du parc informatique et l'assistance des collaborateurs ayant des difficultés de réseaux ou de système.

Ma réflexion s'orientera autour de trois grands axes. Il s'agira dans un premier temps de décrire le contexte industriel global afin d'en comprendre les enjeux actuels et futurs de l'entité ; puis, d'expliquer les raisons de l'évolution de la plateforme de supervision en exposant le panel des différentes méthodes que soulève cette problématique. Il s'agira enfin d'analyser la solution favorable à la résolution du changement de configuration sur un grand nombre d'environnement.

1. Contexte Industriel

1.1. SQLi Group

SQLi Group est une entreprise de services du numérique (ESN) française fondée en 1990. Elle propose à ces clients une transformation digitale allant de la mise à disposition de nouveau moyen de communication jusqu'à la digitalisation complète de services. L'objectif de l'entreprise est de donner une nouvelle vision, combinant collaboration et innovation, à leur client.

SQLi Group articule son objectif autour de trois offres managées par trois entités différentes :

- La "Transformation Digital", conduit par *SQLi Consulting*, offre ses conseils pour faciliter et accélérer la transformation digitale de ses clients.
- La "Performance Business" propose de nouveaux outils pour la communication, le marketing et la vente à ses clients. Cette évolution est guidée par *WAX Interactive*.
- La "Performance Entreprise" pilote ses clients à définir, mettre en oeuvre et piloter leur transition digitale. *SQLi Enterprise* mène cette dernière offre.

SQLi Group est représenté en France (Bordeaux, Lyon, Nantes, Paris, Rouen, Toulouse), en Europe (Luxembourg, Belgique, Pays-Bas, Suisse) et au Maroc au travers de ces différentes agences. De plus, au fil des années, le groupe a racheté certaines entreprises spécialisées permettant d'augmenter ses offres (Alcyonix).

1.2. ISC France

SQLi Group est représenté à Bordeaux par deux entités de SQLi Enterprise, une agence et une sous-entité appelée *ISC* (Innovative Service Centers). Cette dernière est le centre de service de SQLi Group. Il y en a quatre dans la société (deux en France et deux au Maroc). ISC diffère d'une agence dans le fait que les collaborateurs travaillent dans les locaux pour ses clients. Cette dernière donnée oblige ces entités de posséder une infrastructure réseau et système conséquente pour reproduire au maximum les environnements de leur clients.

ISC France est l'entité regroupant les centres de services de Bordeaux et de Nantes. Ils partagent la même direction et travaillent ensemble sur certains projets. De plus, l'infrastructure réseau et système de Bordeaux est le centre d'ISC France ce qui implique qu'un nombre important de serveurs est partagé avec l'entité de Nantes. Ces équipes mènent une veille technologique et culturelle de l'entreprise pouvant faire évoluer leurs méthodes de travail avec de nouvelles bonnes pratiques et l'infrastructure avec une évolution des techniques et des technologies. Sa flexibilité en fait la locomotive de SQLi Group qui se permet de diffuser ces nouvelles avancées à tous les collaborateurs.

1.3. Pôle CRCI

Le Pôle du CRCI -Centre de Ressources et de Compétences Informatiques- est responsable de l'infrastructure système et réseaux, de la gestion du parc et des solutions logicielles et fourni une assistance sur certains points spécifiques. Trois collaborateurs constituent l'équipe du pôle de Bordeaux, **Arnaud Roudsovsky**, responsable du pôle, **Frédéric Gracia** et **Romain Ballan**. Ils interviennent essentiellement sur le site de Bordeaux. Néanmoins, ils coordonnent l'infrastructure de l'entité de Nantes avec l'aide du pôle CRCI distant. Comme expliqué précédemment, ISC France possède une infrastructure conséquente et fondatrice de son travail. Le coeur du business de l'entreprise dépend du bon fonctionnement de l'infrastructure. Le pôle CRCI est un élément primordial de la bonne santé financière de l'entité.

Depuis 2015, l'entreprise a décidé de faire évoluer son infrastructure et ses méthodes de travail en s'appuyant sur le mouvement *DevOps*. Comme le présente le livre de Len Bass, Ingo Weber et Liming Zhu, DevOps est un ensemble de pratiques cherchant à réduire le temps entre la demande et la concrétisation d'une modification d'un système en production tout en assurant son bon fonctionnement [1]. Ce mouvement repose sur cinq piliers essentiels comme l'énonce Joonas Hamunen dans son document [2]:

- *Culture* : changer l'organisation en silos par une organisation plate et collaborative.
- *Automation* -automatisation- : rendre autonome le déploiement des environnements de travail et de production.
- *Lean* : fluidifier et optimiser l'organisation et les modes d'interactions entre les différents acteurs.

- *Measurement* -mesure- : mettre en place des indicateurs de performance adéquats et partagés par tous les métiers.
- *Sharing* -solidarité- : accroître la communication et la collaboration entre les métiers.

Ce nom est la contraction de *Development* (référence au métier de développeur) et *Operation* (référence aux différentes discipline du métier d'infogérant). Il symbolise le thème émergent de cette culture, à savoir l'étroite collaboration de ces métiers dans la réalisation des projets.

Au sein d'ISC France, cette culture est poussée par le pôle CRCI et le pôle Expertise. Ce dernier est le garant de la qualité technique des projets de part un support technique, la dispense de formations et la mise en place de bonne pratique à exécuter par les développeurs. Dans la culture DevOps, les experts sont, notamment, responsables de la mise en place et de la maintenance des outils de production tel que les plateformes d'intégration continue. Le *Continuous Delivery* est une approche de développement en lien avec la mouvance DevOps. Il a pour but de développer, de tester et de sortir des versions de façon rapide et répéter jusqu'à avoir un logiciel de haute qualité [3]. En plus d'insuffler aux développeurs la culture DevOps, le pôle expertise est le lien entre les équipes de production et le pôle CRCI dans le calcul quantitatif des ressources utiles au bon déroulement des projets et la qualité des environnements de développement fournis à chaque développeur.

Les pôles CRCI et expertise travaillent ensemble pour donner à l'entité les moyens d'avoir une infrastructure de type *cloud computing*. C'est un nouveau *paradigme* avec pour objectif de centraliser et d'utiliser au maximum la puissance de calcul de chaque serveur afin de limiter les dépenses en matériel et en ressource électrique. Dans cette collaboration, l'expertise possède à sa charge l'administration du gestionnaire de version privée à l'entité, *GitLab*, et des plateformes d'intégration continue. Quand aux pôle CRCI, son rôle est d'administrer, de rendre disponible et de faire évoluer l'infrastructure dans son ensemble.

1.4. Infrastructure

Historiquement, SQLi Group travaille avec des environnements virtualisés. Son objectif consiste à faire fonctionner plusieurs systèmes d'exploitation sur un ou plusieurs serveurs [4]. De nombreuses entreprises de tout secteur adoptent ce système car il permet une réduction des coûts et rend la gestion plus aisée. L'infrastructure d'ISC France est en évolution depuis l'appropriation de la mouvance DevOps. Des investissements infrastructureux de la part de la direction ont été ordonnés. Ils ont provoqué une infrastructure hétérogène et plus difficile à administrer.

VMWare ESXi est la solution de virtualisation la plus ancienne de l'entreprise. Cette plateforme propriétaire est un hyperviseur de type I signifiant qu'il est installé en lieu et place d'un système d'exploitation. Il permet une gestion plus efficace des environnements virtualisés tel que la migration dynamique des machines virtuels [5]. *OpenStack* est la seconde plateforme de virtualisation utilisée par ISC France. Malgré sa jeunesse, cette solution est stable. Elle est déjà utilisée par des grandes entreprises aux architectures massives et complexes [6]. De plus, c'est un logiciel *open source* avec une communauté de plus en plus importante.

En plus du système de virtualisation dit classique, l'entreprise utilise deux solutions basées sur l'isolation des processus dans le noyau *Linux*. Le système d'exploitation invité, appelé *container*, est contenu dans un processus isolé des autres à l'aide de deux fonctionnalités du noyau *Linux* [7]. Le premier est *OpenVZ*, un noyau modifié *Linux*[8]. Ce logiciel a été l'un des premiers à utiliser la technologie de *containerisation*. Il apparait un peu en retrait de nos jours. Le second est *Docker* [9], devenu en quelques années un des outils de référence dans le mouvement DevOps. Il a l'avantage de s'installer sur tous les noyaux *Linux* comme un logiciel standard et va bientôt être supporté nativement dans les environnements *Windows* et *MAC OS*.

A court terme, l'entreprise a pour but de migrer les machines virtuelles des *VMWare ESXi* vers *OpenStack* et les environnements virtuels d'*OpenVZ* vers *Docker*.

Au sein du pôle CRCI, de nouvelles pratiques ont émergé en lien avec la culture DevOps.

L'automatisation du déploiement des environnements de production est la première. Le but est de pouvoir détruire une machine virtuelle avec la possibilité de la redéployer à l'identique en un minimum de temps. OpenStack est connu pour permettre nativement cette automatisation. A la création de l'instance virtuelle, un script (Bash, Python ou Perl), passé en paramètre, s'exécute et rend la machine utilisable dès son premier lancement.

La seconde pratique est la configuration adéquate d'un logiciel de mesure permettant de récupérer les ressources utilisées par une machine virtuelle. D'un point de vue du CRCI, les mesures à calculer sont au niveau physique de la machine virtuelle (la quantité de mémoires vives (RAM) ou le stockage minimum nécessaire pour une application d'un projet). Ces relevés peuvent être remontés aux équipes de développement pour optimiser l'application. De plus, ce type de logiciel est utile pour le pôle car il permet d'effectuer une surveillance de l'infrastructure dans son ensemble et d'envoyer une notification aux collaborateurs du pôle en cas de problème.

2. Solution de surveillance et de mesure

2.1. Initiation du changement

Depuis le second trimestre 2014, le pôle CRCI s'est doté d'un serveur *Zabbix*, un logiciel open source mesurant et surveillant l'infrastructure. Cependant, l'expansion de l'activité, l'évolution de l'infrastructure et la mise en place de bonne pratique en lien avec la culture DevOps ont réduit le temps des ingénieurs du pôle pour s'occuper du logiciel. Il était devenu une tare de part ses alertes nombreuses, fréquentes et non importantes.

A partir de ces reproches, nous avons décidé de repartir du commencement et un processus de mise en production s'est déroulé en deux étapes. Pour commencer, nous avons remis en cause le logiciel par la recherche de nouvelles solutions. Ensuite, un environnement de test a été créé avant de déployer l'outil et la configuration adéquat vers l'infrastructure. Enfin un bilan sera tiré de cette nouvelle configuration pour le pôle CRCI.

2.2. Remise en cause de la solution actuelle

Pour débiter, notre première action a été de lister les objectifs actuels et futurs auxquels l'outil devra répondre. L'infrastructure d'ISC France héberge des applications web internes. Certaines sont primordiales pour la gestion quotidienne du travail des collaborateurs, d'autres facilitent la vie de l'entreprise. Le pôle CRCI doit pouvoir recevoir une alerte rapide lorsque l'application n'est pas accessible. La solution à envisager doit pouvoir simuler un scénario web comme un simple accès sur une page web ou une tentative de connexion.

Le pôle CRCI rend un rapport chaque semaine concernant la disponibilité de l'infrastructure. Comme l'évoque le papier scientifique de Pankesh Patel, Ajith H. Ranabahu et Amit P. Sheth [10], dans une infrastructure dite cloud computing, il est important de signer un accord de disponibilité de la plateforme entre les consommateurs et un prestataire. Dans le cas d'ISC France, les consommateurs sont le directeur, les managers et les directeurs de projet et les prestataires sont le pôle CRCI. Un *SLA* -Service Level Agreement- doit être négocié entre les deux parties pour définir précisément un accord sur les mesures à prendre en compte.

La mise en place de la culture DevOps a emmené l'utilisation, au sein de l'infrastructure, de l'outil Docker combiné au système d'exploitation *CoreOS*, système d'exploitation léger et adapté pour l'utilisation de cet outil comme l'indique Mathijs Jeroen Scheepers [11]. La solution à choisir doit pouvoir surveiller ce genre d'environnement virtuel sans devoir installer de composant sur le système d'exploitation. Cette mouvance qui est en train de s'installer dans l'entreprise doit pousser les équipes de développement à quantifier les ressources matérielles utilisées par leurs programmes. L'outil doit donc pouvoir fournir des graphes et des valeurs lisibles et compréhensibles à des non-initiés à l'infogérance.

A partir de cette liste, nous avons pu nous mettre à la recherche d'un logiciel remplissant ces conditions. Il existe de nombreuses offres sur le marché. Néanmoins, très peu d'entre elles répondent entièrement à nos attentes. Notre choix s'est donc porté sur la solution que nous utilisons, Zabbix, pour plusieurs raisons. Ce logiciel détient les éléments de base de nos exigences, à savoir l'envoi d'alerte en cas d'indisponibilité sur une application web que nous hébergeons et le calcul de SLAs calculant la disponibilité de l'infrastructure.

De plus, l'offre est open source et possède une large communauté active offrant des fonctionnalités supplémentaires utiles à nos exigences, comme par exemple la surveillance et la mesure des ressources utilisées par les environnements virtuels sous Docker. Ensuite, la nouvelle version de ce logiciel offre une interface plus ergonomique. Cette nouveauté rendra l'utilisation des données et des graphes plus lisibles aux équipes de développement. Enfin, les membres du pôle CRCI sont déjà formés sur cet outil. Cet avantage permet de gagner un temps considérable sur son approche basique. Cette conjoncture nous sera utile pour configurer plus précisément la solution et mettre en place les fonctionnalités souhaitées.

2.3. Environnement de test et mise en production

Après le choix du logiciel, nous avons dû le déployer. Notre but a été de configurer la future solution d'une manière optimale tout en gardant en fonctionnement la solution présente sur l'infrastructure. Pour cela, on a recréé un environnement de tests comportant le minimum d'environnement nécessaire à une configuration adéquate.

Zabbix est une offre de surveillance et de mesure de l'activité infrastructurelle d'une entreprise. Il est basé sur le système de client-serveur signifiant que l'agent, dans le cas de cette solution, envoie au serveur les données que celui-ci souhaite recevoir. La solution possède trois logiciels.

- *Serveur* qui est le centre du service. Il est composé de deux logiciels, *server* et *frontend*. Ces outils doivent être combinés à une base de données servant à stocker les données récupérées et à un serveur web qui permet de déployer une interface web permettant de configurer les mesures que l'on souhaite récupérer et aidant la lecture de celle-ci. Le serveur récupère les données envoyées par les protocoles *HTTP* (en utilisant le format *JSON*) et *SNMP*.
- *Agents* qui sont les noeuds du système. Ils sont installés sur chaque environnement à surveiller. Chaque agent récupère et envoie les mesures que lui demande le serveur. Dans une infrastructure, un agent doit obligatoirement avoir un serveur à qui il peut communiquer.
- *Proxys* qui font office d'intermédiaire entre un agent et un serveur. Son but est de récolter les informations d'un groupe d'agents et de les envoyer au serveur principal. Ce logiciel peut ne pas être utilisé.

A partir de notre plateforme de tests, on a pu configurer notre serveur et nos agents à notre convenance. Dans un premier temps, nous avons décidé de nous passer de la fonctionnalité proxy de la solution. Il y a eu deux phases dans la configuration. La première concerne les fichiers de configuration des agents et du serveur. Il faut que l'agent connaisse le nom de la machine sur lequel il est installé et l'adresse IP du serveur. Pour ce dernier, nous avons dû le configurer pour qu'il puisse se connecter à la base de données et que l'interface web soit accessible. De plus, nous avons dû le configurer pour qu'il démarre assez de processus système afin de traiter toutes les données qui lui sont destinées.

La seconde étape de configuration a été de définir ce que l'on souhaitait mesurer sur chaque environnement. Cette phase se fait par l'intermédiaire de l'interface web du serveur Zabbix. On choisit des *items* qui représentent une donnée à récupérer sur les agents, comme par exemple l'utilisation de la RAM. Ensuite, ces items sont utilisés pour déclencher ou non des *triggers*. Ce sont un ensemble de conditions qui provoquent une *alerte*, courriel par exemple, si elles sont vérifiées. Par exemple, si l'utilisation de la RAM dépasse les 90% du total alors une alerte est déclenchée. Enfin c'est avec ces données que les SLAs demandés sont calculés.

Dans la configuration de la solution qui était en place sur l'infrastructure, les environnements à monitorer ainsi que leurs mesures étaient enregistrés par l'équipe du pôle CRCI grâce à l'application web du serveur. Zabbix permet de rendre ces actions dynamiques. Il suffit de renseigner dans le fichier de configuration de chaque agent une liste de *métadatas*. Lors du lancement du service sur l'agent, un échange de requêtes a lieu avec le serveur. Celui-ci enregistre la machine dans la base de données avec son nom et son adresse IP. Puis il analyse les métadatas et affecte à l'agent les items souhaités. Ce gain de temps entre dans l'esprit DevOps.

Comme expliqué, l'entité utilise de plus en plus le logiciel Docker. Nous possédons plusieurs environnements "Dockerisés" qui n'étaient pas surveillés par l'ancien serveur Zabbix. Nous souhaitons un système qui mesure l'activité de chaque environnement tout en évitant d'installer un agent dans le système d'exploitation CoreOS. Grâce à la communauté active, nous avons trouvé une fonctionnalité qui a pu contenter nos attentes. Grâce à la large communauté de Zabbix, une solution a été développée. Néanmoins, celle-ci ne nous satisfaisait pas entièrement, nous avons donc décidé d'implémenter notre propre outil. Il se traduit par un container Docker dans lequel un agent et le composant ont été compilés. Cet agent récupère les données de tous les containers s'exécutant sur la machine. Cette action est effectuée par l'intermédiaire du *socket* du logiciel Docker. Ensuite, il envoie les mesures au serveur.

A la fin de la phase de test, nous avons décidé de remplacer le serveur existant par le nouveau. La migration s'est faite en deux parties. Pour commencer, nous avons installé les logiciels serveurs sur le nouvel environnement. Puis, il a fallu migrer le fichier de configuration du serveur test. Enfin, une importation de la saisie des items et des triggers du serveur de test a dû être effectué. Dans un second temps, il a fallu modifier la configuration de chaque agent pour qu'ils aient leurs nouvelles informations. Pour cette phase, un outil d'automatisation a été utilisé.

2.4. Apports et limites de cette nouvelle configuration

Après la mise en production du nouveau serveur de surveillance et le changement de configuration de tous les agents vers ce dernier. Les mesures récupérées ont été instantanées. A partir de ce moment, on a pu ajuster certaines mesures et alertes jugées sans intérêts ou trop sensibles et en ajouter des nouvelles.

Les apports de cette nouvelle configuration sont importants et primordiaux pour le pôle CRCl. Pour commencer, les alertes reçues en cas de problème sur les environnements surveillés sont précises ce qui nous permet d'intervenir rapidement et efficacement. Ensuite, les mesures effectuées sur l'infrastructure nous donnent la capacité de suivre la consommation de chaque machine efficacement. Cette avantage nous offre la possibilité de l'optimiser.

Enfin, pouvoir monitorer les environnements virtuels sous Docker est un plus pour l'évolution et la popularité de ce système dans notre infrastructure. Malgré tout, le logiciel Zabbix n'est pas un outil parfait. Sa principale limite est le temps à consacrer dans la configuration des mesures, des alertes et des hôtes. Effectivement, une machine qui a été supprimée sur l'infrastructure ne l'est pas sur le serveur. Les limites de Zabbix ainsi que l'évolution de notre infrastructure nous obligeront à garder du temps pour effectuer une veille technologique autour de cet outil et de ses composants annexes. Garder notre serveur de supervision en adéquation avec notre système est essentiel pour la bonne santé de celui-ci.

Actuellement nous utilisons une base de donnée *MySQL InnoDB* sans réplication. Pour notre infrastructure mesurant plus de deux cent hôtes, cette configuration est suffisante. Cependant, elle peut s'avérer être une limite en cas d'accroissement massif de l'infrastructure à superviser. En effet, selon les documents officiels de Zabbix, notre configuration sera optimale jusqu'à cinq cent hôtes. Notre marge de manoeuvre est encore importante. Néanmoins, la mise en place de sauvegarde de la base de donnée serait une bonne pratique à mettre en place. L'utilisation d'un standard *RAID* pourrait rendre cette pratique automatique.

3. Solution de gestion des configurations

3.1. Recherche de la solution adaptée

La problématique concernant le changement de configuration des agents Zabbix en place sur le réseau a été l'élément déclencheur de cette recherche. Il y avait plus de deux cent configurations à modifier. Un nombre important d'action commune devaient être faite sur chaque environnement. Nous avons souhaité automatiser et rationaliser, notions importantes de la mouvance DevOps. Les solutions de gestion de configuration sont des outils puissants puisqu'ils offrent la possibilité de copier des scripts et de les exécuter en parallèle sur un nombre illimité d'ordinateurs. Pour nous, l'outil doit posséder une installation légère, être simple dans sa configuration de requêtes et pouvoir gérer les environnements Linux et Windows.

Les recherches nous ont amenés à choisir *Ansible*, solution open source implémentée en langage Python. Nous avons été motivé par plusieurs points. Premièrement, c'est un outil dit "agentless", ça signifie que l'outil n'a besoin que d'un serveur pour effectuer la gestion. Les solutions réputées telles que *Chef* et *Puppet* utilisent un serveur et des agents sur chaque noeud à gérer, comme le logiciel Zabbix. Ensuite, Ansible utilise *YAML*, un format de représentation des données que nous pratiquons régulièrement, et peut gérer plusieurs environnements. Enfin, cette solution est gratuite pour l'utilisation que nous souhaitons en faire.

3.2. Mise en place

Avant de mettre en place le serveur sur l'infrastructure, nous avons effectué divers tests dans un environnement dédié. Son installation est très aisée puisqu'il s'agit d'un logiciel qui se trouve dans les répertoires des environnements Linux. Ansible utilise le protocole *SSH* -Secure Shell- pour transmettre ces requêtes vers les environnements Linux. Ce protocole réseau est un protocole de communications sécurisées basé sur l'utilisation du couple clef privée et clef publique et l'échange des clefs publiques entre communicants. Un couple de clefs a été mis en place pour l'utilisation de ce service sur notre infrastructure. La clef publique a été transmise à chaque hôte Linux de notre architecture réseau. Pour les environnements Windows, ce protocole de sécurisation n'existe pas. Il faut donc utiliser une fonctionnalité native appelé *WinRM* -Windows Remote Manager. C'est un protocole de gestion à distance qui permet la communication entre deux matériels réseaux. Ansible embarque diverses modules comme le ping, fonction qui retourne si la

connexion entre le serveur et le noeud est fonctionnelle.

Dans notre cas, notre but était d'automatiser la mise à jour ou l'installation des agents sur les environnements à monitorer et les configurer. Nous avons créé deux scripts, un pour les environnements Linux et l'autre pour les environnement Windows. Le premier devait installer la plus haute version possible sur chaque distribution Linux (*Ubuntu*, *Debian* et *Redhat/CentOS*). Quand au second, il devait détecter si l'environnement Windows installé était en 32 bits ou en 64 bits. Des scripts robustes et stables devaient être implémentés afin d'éviter les pertes de temps inutiles dans le débogage de l'exécution. Grâce à Ansible, nous avons pu copier nos scripts sur chaque environnement et les exécuter.

3.3. Apports et limites de l'outil

Ce logiciel a été d'une grande aide dans la migration définitive de notre nouveau serveur Zabbix. Il nous a permis de déployer les nouvelles configurations sur un large nombre d'hôtes. Le gain de temps a été positif pour le pôle. D'autres utilisations en ont découlé comme effectuer des mises à jour ou des montées de version sur un logiciel sur tous les environnements utilisant cette version. Cependant, cet outil possède des failles. Elles concernent les destinataires étant sous l'environnement Windows. Pour rendre leur logiciel fonctionnel sur cet environnement, les concepteurs ont dû utiliser une solution de secours au protocole SSH, à savoir WinRM. Malheureusement, cette fonctionnalité n'est pas native et dépend d'une version du logiciel *PowerShell*. C'est un logiciel qui permet l'interprétation de commandes inscrites par un utilisateur. Certaines versions de ce système d'exploitation ne permettent pas la mise à jour de ce logiciel et donc l'utilisation d'Ansible. Néanmoins, notre infrastructure possède un nombre limité d'hôtes possédant ce problème.

Conclusion

L'objectif principal de ce stage a été de faire évoluer la plateforme de supervision existante vers une configuration plus adéquate aux nouveaux besoins du pôle et de l'entité. Selon mon analyse, cette tâche a été menée à bien. J'ai pu comprendre ce que l'équipe attendait de mon travail, analyser l'outil sélectionné, optimiser la configuration de celui-ci et trouver des solutions pour permettre de répondre aux nouvelles solutions de l'entreprise. Grâce aux notifications en cas de panne, nous avons déjà pu intervenir rapidement pour les résoudre et comprendre leur déclenchement.

L'objectif de s'intégrer dans la culture *DevOps* prônée par les équipes du pôle CRCI et des experts a été accompli. Grâce aux notions et technique apprises grâce aux équipe et à mes lectures, j'ai pu proposer et mettre en place des solutions qui seront utiles à l'avenir.

Le travail réalisé durant ce stage n'aidera certainement pas l'équipe à optimiser les environnements ou l'infrastructure. Elle nous permet de réagir rapidement lors des défaillances en ciblant la cause grâce aux données collectées par l'outil. J'ai seulement réussi à réduire le temps entre le démarrage du problème et la résolution de celui-ci. De plus, l'outil de gestion des configurations nous permettra de faciliter les évolutions massives des environnements tout en réduisant les ressources temporelles de l'équipe.

Malheureusement, maintenir la plateforme de supervision est un travail de tous les jours. Si nous souhaitons qu'elle reste un atout pour notre service, il faut que nous gardions du temps pour continuer son évolution et son optimisation puisque la configuration actuelle sera potentiellement obsolète d'ici quelques mois.

Annexes

1. Virtualisation

La virtualisation est une technique permettant de faire fonctionner plusieurs ordinateurs virtuels à l'aide d'un seul ordinateur ou serveur physique. Cette technologie a l'avantage de limiter l'achat de matériels physiques et de gérer efficacement et plus simplement les environnements virtuels qui sont et seront créés. Il existe plusieurs types de virtualisation.

La plus répandue est la virtualisation complète. Elle utilise un hyperviseur servant de gestionnaire pour les machines virtuelles instanciées et d'isolateur entre elles et le système d'exploitation de la machine physique. Il est possible que certains hyperviseurs comme VMWare ESXi soient installés à la place d'un système d'exploitation, c'est la virtualisation de type I. Le principe de cette technologie est de créer un environnement virtuel en simulant un ordinateur complet. L'hyperviseur alloue les ressources que l'on souhaite attribué au nouvel environnement en fonction des ressources physiques disponibles. Le lien entre les ressources physique et les machines virtuelles est fait grâce à l'hyperviseur.

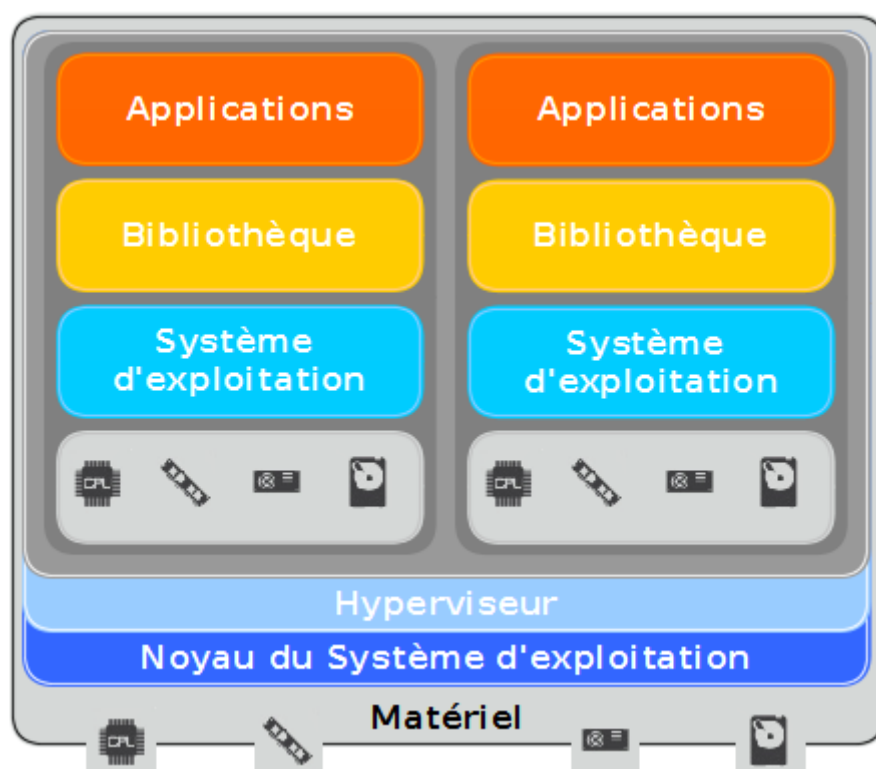


Figure 3: Représentation de la virtualisation de type I

L'avantage de cette technologie est qu'il est possible de créer un environnement virtuel Windows sur un serveur Linux. Cependant, sa principale limitation est son coût en matière de ressources. Par exemple, un utilisateur alloue à un environnement virtuel deux gigaoctets de mémoire. Cette quantité est déduite de la quantité totale disponible matériellement et ce, même si l'environnement n'en utilise que la moitié.

La nouvelle technologie émergente est la *containerisation*. Elle est poussée par la culture DevOps et de nombreux logiciels tels que *LXC* et surtout *Docker*. Au lieu de simuler un ordinateur complet comme la précédente technique évoquée, le logiciel instancie un environnement en l'isolant avec son propre espace mémoire. Les ressources systèmes et matériel sont partagées entre le kernel de la machine hôte et les "containers". Le principe de cette technologie est de pouvoir créer et détruire rapidement et simplement une application avec toute sa configuration.

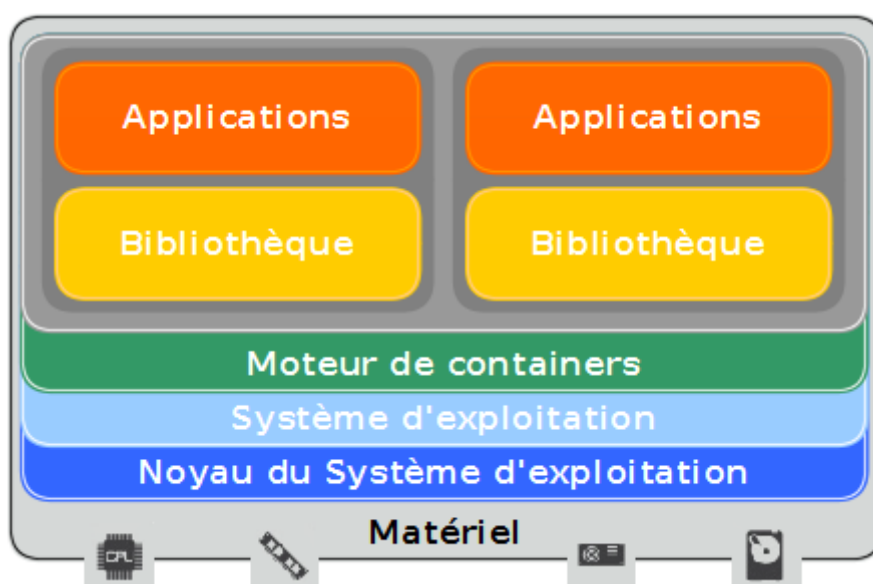


Figure 4: Représentation de la containerisation

L'avantage est de pouvoir avoir des applications que l'on peut migrer d'une machine à une autre et redéployer rapidement et simplement en cas de panne sur le container. Néanmoins, vu que la technologie utilise le partage des appels système entre l'hôte et les machines virtuelles, il n'est pas possible d'instancier un environnement Windows sur un système d'exploitation Linux. De plus, cette technologie n'est native uniquement dans un environnement Linux. Certains logiciels tels que *Docker* la développe sous les environnements Windows et MacOS en ajoutant une couche d'abstraction entre le système d'exploitation et les containers.

Ces technologies sont de plus en plus utilisées dans les infrastructures des entreprises, soit à travers la souscription d'une offre de *cloud computing* ou par le déploiement d'une architecture interne à l'aide de solution tels que *OpenStack* ou *VMWare*.

2. Zabbix

Zabbix est un ensemble de logiciels permettant la surveillance de l'infrastructure à travers la récupération de mesures de ressources sur chaque environnement souhaité. Il existe trois logiciels pour le fonctionnement de cette solution, serveur, agent et proxy.

2.1. Architecture Zabbix

Une architecture Zabbix peut être constituée d'au minimum, un serveur auquel on peut greffer deux autres outils, des agents et des proxys. Le serveur est le maître de l'architecture et les proxys (si utilisés) sont les maîtres d'un certain nombre d'agents. En utilisant tous les outils, il est donc possible d'avoir une architecture distribuée. La transmission des données se fait au moyen de protocoles de communication réseaux.

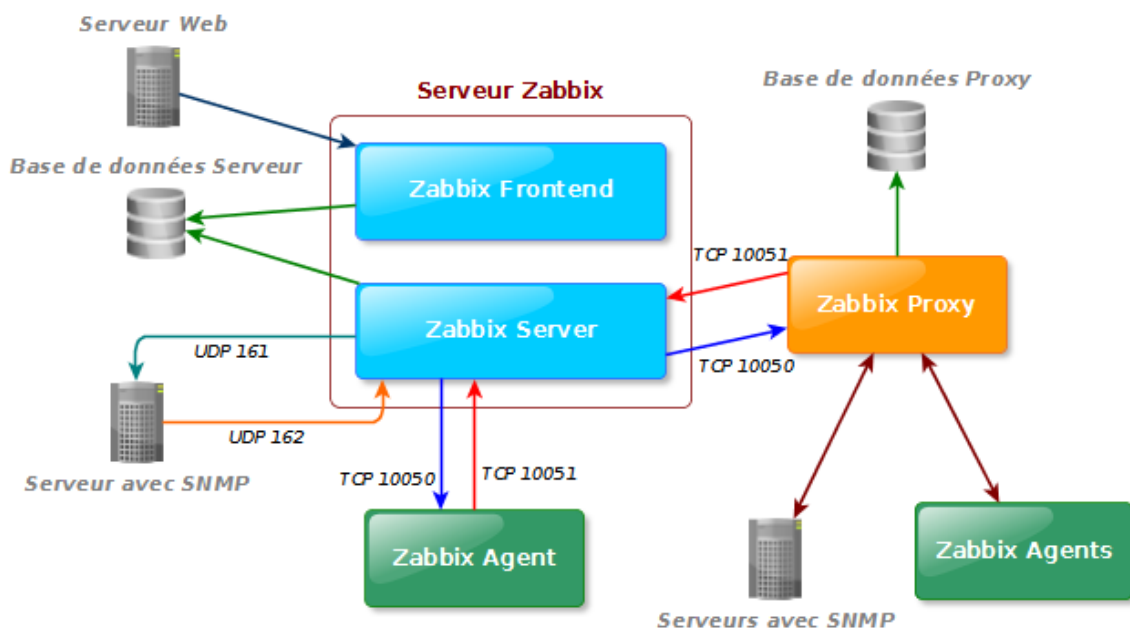


Figure 5: Architecture complète d'une supervision par les outils Zabbix

Les outils *server* et *frontend* sont le noeud central d'une architecture Zabbix. Ils doivent être couplés à une base de données (*MySQL*, *Oracle*, *PostgreSQL* ou *SQLite*) pour le stockage des données et à un serveur web. Le serveur Zabbix est le centre de l'architecture Zabbix. Son objectif est de récupérer et d'analyser les données, de calculer les déclencheurs d'alertes et d'envoyer des notifications (courriel, SMS,...) aux utilisateurs en cas de problèmes détectés. Le serveur Zabbix a la capacité d'utiliser le protocole *SNMP* afin de récupérer les données souhaitées par un utilisateur.

L'outil *proxy* sert d'intermédiaire entre le serveur et les agents. Il est installable dans un environnement Linux. Son objectif est de limiter la charge en processus système et en ressource du serveur. Il doit être couplé à une base de données pour fonctionner. Il stocke les mesures des différents hôtes dont il est maître et transfère tout le contenu de la base au serveur.

Le logiciel *agent* de Zabbix est un outil installable sur les environnements Linux, MacOS ou Windows. Il a pour but de récupérer au travers du système d'exploitation les mesures souhaitées et de les transmettre à son noeud supérieur.

Dans le cas de notre infrastructure, nous avons décidé de ne pas mettre en place le mode distribué de l'outil. Seul des agents et un serveur sont utilisés. Cette décision n'est pas définitive mais à l'heure actuelle, son utilisation n'est pas nécessaire puisqu'elle complexifierait notre architecture qui l'est déjà.

2.2. Communication entre un serveur et un agent

Dans Zabbix, la communication entre un noeud et son maître se fait par le protocole *HTTP* et les données transmises sont au format *JSON*. Il existe deux modes d'échange entre les noeuds.

Le premier est le "check" dit passif. Le zabbix serveur envoie une requête de demande de données et l'agent lui répond en transmettant les mesures souhaitées.

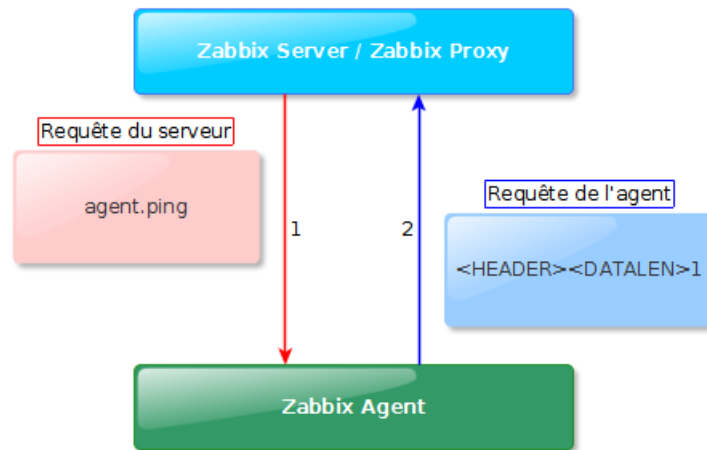


Figure 6: Exemple de check passif

Le second est le "check" actif. Dans ce cas, c'est l'agent qui amorce les échanges. La communication est décomposée en deux étapes distinctes. Pour commencer, l'agent initialise la connexion avec son maître en lui demandant la liste de toutes les données qu'il souhaite récupérer. Une fois fait, l'agent démarre sa collecte de données à intervalle fixe pour chacune des mesures. Une fois que la liste est complétée, il transmet au serveur les résultats.

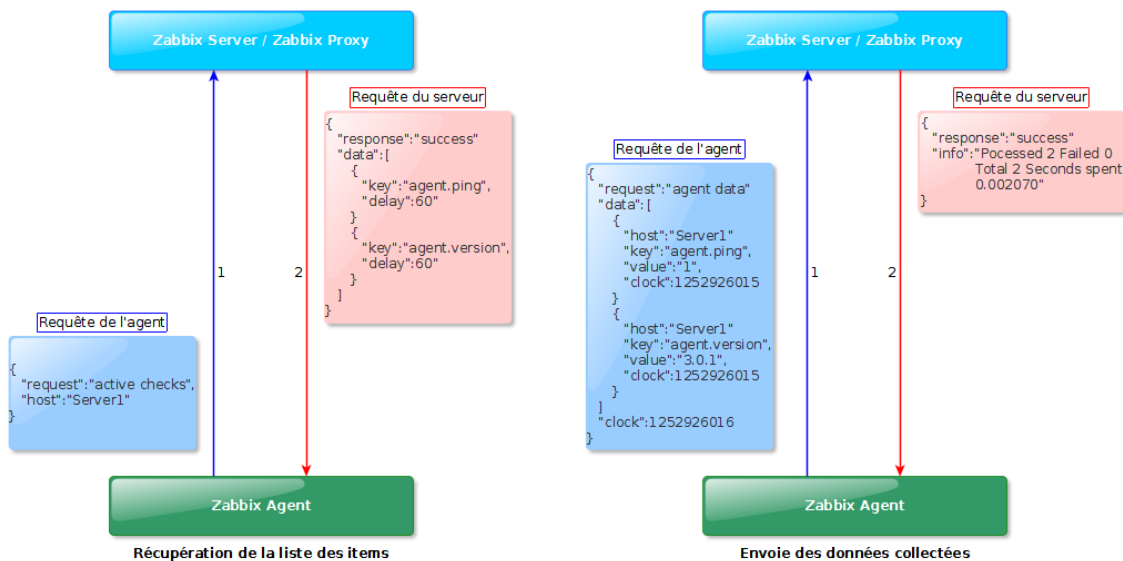


Figure 7: Exemple de check actif

L'intérêt de ce dernier check est de limiter la bande passante et les processus système des noeuds maîtres. En effet, dans le premier check, le serveur envoie une requête pour chaque item. A titre d'exemple, dans notre configuration, le serveur devrait gérer plus de deux cents échanges par seconde. Si tel était le cas, notre serveur aurait une charge importante entraînant de possibles problèmes.

2.3. Système d'alertes

Zabbix propose de générer une alerte lorsque certaines conditions sont remplies. Le principe de la surveillance est de notifier les administrateurs en cas de problème sur l'infrastructure. Ces alertes utilisent trois types de fonctionnalité zabbix. Les *items* sont des éléments qui définissent les mesures à effectuer. Ensuite, les triggers génèrent un type événement dépendant d'une valeur collectée à partir d'un item. Enfin, les actions sont l'envoi d'une alerte à des utilisateurs selon le type d'évènement généré par un trigger.

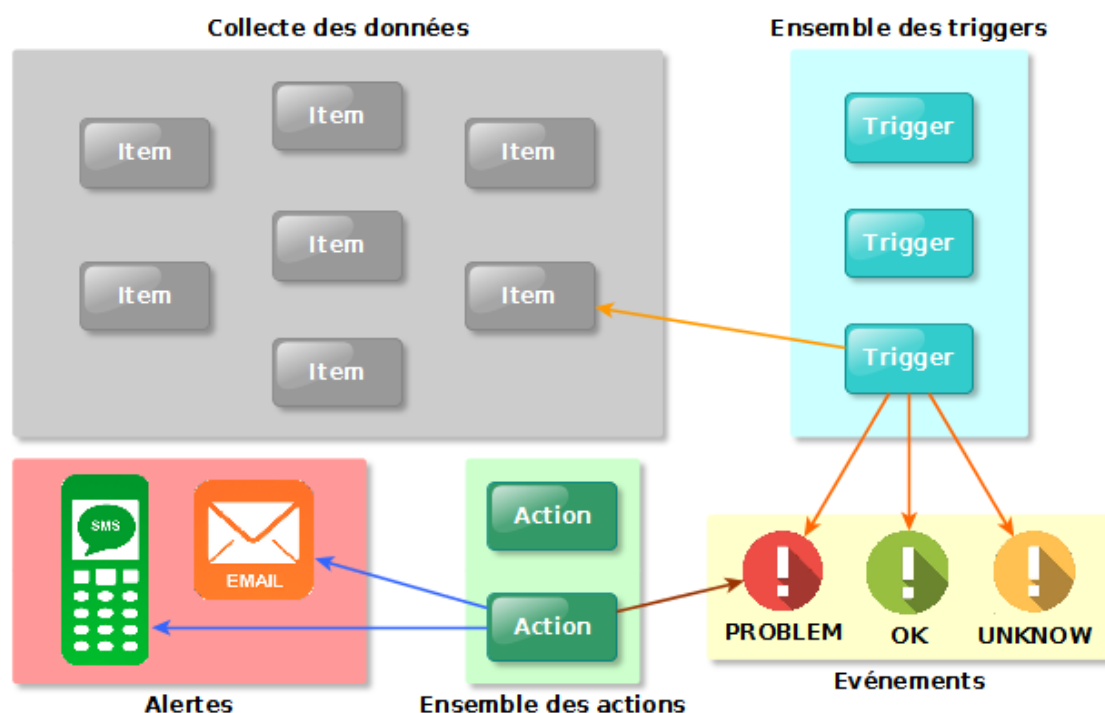


Figure 8: Processus de génération d'une alerte

Les évènements dits "problème" possèdent une certaine sévérité, définie par l'utilisateur. Il existe six degrés de gravité dans Zabbix. Dans le cas de ISC France, nous avons fait le choix d'effectuer une action uniquement si la sévérité est élevée. Nous avons été motivés dans ce choix par le nombre élevé de courriels reçu sans ce filtre.

Bibliographie

- [1] L. Bass, I. Weber, L. Zhu, *DevOps: A Software Architect's Perspective*, Addison-Wesley Professional, 2015.
- [2] J. Hamunen, *Challenges in Adopting a Devops Approach to Software Development and Operations*, Thèse de master, Université de Aalto, 69p, 2016.
- [3] J. Humble, D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, Addison-Wesley Professional, 2010.
- [4] T. Anderson, L. Peterson, S. Shenker, J. Turner, *Overcoming the Internet Impasse Through Virtualization*, Computer, 38:34-41, 2005.
- [5] S. Sridharan, *A Performance Comparison of Hypervisors for Cloud Computing*, Thèse de master, Université de North Florida, 127p, 2012.
- [6] O. Sefraoui, M. Aissaoui, M. Eleuldj, *OpenStack: Toward an Open-Source Solution for Cloud Computing*, International Journal of Computer Applications, 55:38-42, 2012.
- [7] A. Sampathkumar, *Virtualizing Intelligent River : A comparative study of alternative virtualization technologies*, Thèse de master, Université de Clemson, 85p, 2013.
- [8] K. Kolyshkin, *Virtualization in Linux*, White paper OpenVZ, 3:39p, 2006.
- [9] D. Merkel, *Docker: Lightweight Linux Containers for Consistent Development and Deployment*, site de Linux Journal [En Ligne], Adresse URL: <http://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment>
- [10] P. Patel, A. H. Ranabahu, A. P. Sheth, *Service Level Agreement in Cloud Computing*, 2009.
- [11] M. J. Scheepers, *Virtualization and Containerization of Application Infrastructure: A Comparison*, 2014.