



Rapport de stage - 01/07/2022

Le nom du projet : The Green Campus Bot

L'intitulé de la mission	<ul style="list-style-type: none">• Déployer des outils de mesure de la consommation énergétique des serveurs/VM• Proposer une visualisation sous forme de métaphores parlantes des métriques collectées en lieu et place des Watts
l'auteur du rapport	Slim Khiari
le nom de la formation suivie et l'année	IATIC 4 – 2021/2022
le lieu du stage	La côte basque - Anglet
les dates du stage	9 Mai 2022 - 9 Aout 2022 (3 mois)
Encadrant du laboratoire	<ul style="list-style-type: none">• M. Phillipe ROOSE
Encadrant de l'école	<ul style="list-style-type: none">• M. Franck TALBART

Remerciements

Je tiens à remercier tout le personnel de l'IUT de Bayonne et du Pays Basque, et du laboratoire LIUPPA pour le soutien qu'ils ont pu m'apporter ainsi que leurs bonnes humeurs. Je remercie, en particulier, M. Phillipe ROOSE, mon maître de stage professionnel, pour m'avoir permis d'effectuer ce stage dans le laboratoire LIUPPA. Il a été toujours présent pour m'aider à résoudre les différents problèmes rencontrés, en effectuant des mises au point hebdomadaires, tout en me laissant une très grande autonomie de travail, et mon tuteur de stage de l'école ISTEY M. Franck TALBART.

Je remercie aussi, de tout cœur, M. Adel Noureddine, M. Olivier Le Goaer et la Direction du Numérique de l'Université de Pau et des Pays de l'Adour.

Sommaire

Liste des tableaux	4
Liste des figures	4
Le lexique.....	5
Introduction	6
Présentation du laboratoire.....	8
Présentation générale	9
Secteur	9
Organigramme du laboratoire	10
Le service où j'ai effectué le stage	10
Le cadre du stage.....	11
Description de différentes équipes composantes le laboratoire LIUPPA	12
Le projet général et les missions à réaliser	12
Aspects techniques - conception et réalisation	12
Conception	14
Répartition du temps de travail et des tâches	14
Outils utilisés lors de mon stage	16
Réalisation.....	22
Tableau de bord des métriques d'une VM	22
Tableau de bord des métriques de 2 serveurs distants.....	25
Réalisation d'un tableur pour les conversions énergétiques métaphoriquement	28
Conclusion	29
Bibliographie	31
Annexe : Tutoriel pour la visualisation graphique des métriques de la consommation énergétique d'une machine virtuelle avec l'API Scaphandre et Grafana	32
Résumé.....	52
Abstract.....	53

Liste des tableaux

Tableau 1: Conversions énergétiques (les métaphores) à appliquer sur les 2 dashboards créés	28
---------------------------------------------------------------------------------------------------	----

Liste des figures

Figure 1: Organigramme du laboratoire	10
Figure 2: L'écosystème Prometheus - version 1	18
Figure 3: L'écosystème Prometheus - version 2	19
Figure 4: Fichier de configuration pour lancer Prometheus.....	23
Figure 5: Capture d'écran du 1er dashboard réalisé	24
Figure 6: Capture d'écran d'un exemple d'une requête PromQL	24
Figure 7: Schéma explicatif sur l'envoi des données depuis le site de Pau vers le site d'Anglet	25
Figure 8: Schéma explicatif de Telegraf.....	26
Figure 9: Capture d'écran depuis la VM "greenit" montrant l'exploration de la base de données InfluxDB créée	26
Figure 10: La consommation électrique de 2 bare-metals distants (en Wattss) en fonction du temps	27
Figure 11: La consommation du DRAM des 2 bare-metals distants (en Watts) en fonction du temps	27

Le lexique

API	Application Programming Interface
RAPL	Running average power limit
CPU	Central Processing Unit
DSL	Domain-Specific Language
SDLC	Software Development Life Cycle
BBCP	Behavior-Based Consumption Profiles
QoS	Quality of Service
SCP	Systèmes Cyber-Physiques
HTTP	Hypertext Transfer Protocol
SD	Service Directory
TDP	Thermal Design Power
DRAM	Dynamic Random Access Memory
ETL	Extract-transform-load
KVM	Kernel-based Virtual Machine

Introduction

Dans le cadre de ma formation d'ingénieur informatique, intitulée IATIC (Ingénierie des Architectures Technologiques de l'Information et de la Communication) de ISTY (Institut des Sciences et Techniques des Yvelines) de l'UVSQ (Université Saint-Quentin en Yvelines), j'ai été amené à réaliser un stage d'une durée de 3 mois à partir du 9 Mai 2022. Il est effectué au sein de LIUPPA (Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour) sur le site de la Côte Basque.

Le thème général de mon stage est le green IT. Il s'agit d'un ensemble de techniques ayant pour objectif de limiter les conséquences environnementales des technologies de l'information et de la communication. Le green IT est apparu lorsque l'infrastructure informatique est devenue l'une des causes de l'augmentation de l'effet de serre en raison de sa consommation relativement importante de l'énergie électrique.

Mon stage consiste à collecter puis afficher graphiquement les métriques de consommation énergétique, d'un serveur et d'une machine virtuelle en utilisant une API spécifique pour la mesure de la consommation énergétique et un outil de visualisation de données sous forme de graphiques pour la mise en place d'un tableau de bord final résumant, en temps réel, les données collectées pour une machine donnée.

Je commencerai d'abord par présenter le laboratoire et son secteur d'activité. Après la présentation des missions du stage, je détaillerai les travaux effectués. Pour cela, dans un premier temps, j'aborderai la partie de la conception en justifiant le choix des outils utilisés, comme par exemple l'API Scaphandre qui permet d'effectuer les mesures des métriques de consommation énergétique, et en expliquant leurs fonctionnant. Dans un deuxième temps, j'expliquerai la mise en place de l'ensemble des outils utilisés et la présentation des métriques présentes dans les tableaux de bord obtenus.

Présentation du laboratoire

Présentation générale

Le laboratoire LIUPPA fait partie de l'université UPPA. Il a participé à la réalisation de plusieurs projets non seulement européens sur les SCP (Systèmes Cyber-Physiques), le génie logiciel, le développement d'une plateforme de gestion de l'énergie, mais aussi internationaux avec l'Algérie, le Mexique, et l'Espagne.

Ses axes de recherche sont focalisés principalement sur les besoins et les enjeux d'une société numérique dans laquelle les réseaux ont occupé une place non négligeable dans nos activités de tous les jours. Ce phénomène a poussé les systèmes informatiques à devenir de plus en plus complexes à travers non seulement la masse des données de tout types, fortement délocalisées, qui augmente sans arrêt, mais aussi l'évolution des besoins des différents usagers.

Secteur

LIUPPA cherche des solutions sur deux domaines différents avec des préoccupations liées à la sécurité, au traitement de l'image et du signal, à la visualisation, aux systèmes distribués, et à l'interaction et l'adaptation. Le premier domaine est les sciences et les technologies de l'information autour des traitements de l'information, de la connaissance, et du web. Le second principal domaine est le génie logiciel autour de l'ingénierie des modèles, des services et des architectures logicielles.

Le savoir faire du laboratoire se décompose en 3 parties. La première partie concerne les systèmes d'information à travers l'ingénierie des documents électronique, la sémantique des contenus (l'extraction et indexation d'information spatio-temporelle, la recherche par contenus), les interfaces multimodale, les interfaces intelligentes, l'ingénierie collaborative et le E-learning. La deuxième partie concerne le génie logiciel et les systèmes distribués via la modélisation, la vérification, la validation et le codage de systèmes, la conception et le déploiement de solutions à base de composants et d'agents logiciels, les définition et la mise en place de politique de sécurité pour les réseaux et les bases de données, et enfin le Context Aware Middleware. Concernant la troisième partie du savoir faire, il y a les SCP à travers l'analyse des données et apprentissage automatique, la représentation des connaissances, les réseaux et protocoles, la gestion des événements et contrôle des ressources dans les réseaux de capteurs et la protection de la vie privée sur les réseaux sociaux.

Le laboratoire positionne son projet scientifique dans un champ applicatif bien précis qui est la gestion des systèmes d'information et des architectures des SCP. Les SCP sont des systèmes connectés dans lesquels chaque élément est en interaction avec tous les autres éléments. Ainsi, chaque élément contribue à constituer la complexité de cet ensemble d'éléments. Ces systèmes SCP, composés des systèmes logiciels, de capteurs et d'actionneurs, permettent de mettre en relation le monde physique au monde du numérique du traitement de l'information.

Organigramme du laboratoire



Figure 1: Organigramme du laboratoire
(Source : <https://liuppa.univ-pau.fr/fr/organisation/organigramme.html>)

Le service où j'ai effectué le stage

Le stage effectué est réalisé au sein de l'équipe des traitements des informations pour l'adaptation de l'interaction au contexte et à l'utilisateur (T2I). Cette équipe traite plus particulièrement les éléments externes et contextuels d'un SCP.

Le cadre du stage

Description de différentes équipes composantes le laboratoire LIUPPA

Le laboratoire LIUPPA est composé de 3 équipes différentes. La première est l'équipe spécialisée dans l'architecture des SCPs. Elle a pour mission d'élaborer des recherches au tour de la gestion de la sémantique des données à l'intérieur d'SCP, la conception des architectures systèmes, et la maîtrise des échanges entre les équipements. La seconde équipe est l'équipe spécialisée dans les traitements des informations pour l'adaptation de l'interaction au contexte et à l'utilisateur. Elle a pour mission de conception, d'implémentation et de déploiement des applications génériques, interactives et adaptatives, permettant le traitement des différentes données ayant des origines de provenance différentes, qui nécessitent des modèles de représentation et des méthodes d'accès originaux. Le thème général des recherches de cette équipe est la valorisation de l'information et la facilitation des interactions de l'usager. Enfin, la troisième équipe est l'équipe de l'ingénierie dirigée par les modèles. Cette équipe s'intéresse aux langages de spécification et de modélisation semi-formelle pour la conception de logiciels de qualité. Elle mène des recherches sur la conception des logiciels de natures variées.

Le projet général et les missions à réaliser

Le projet sur lequel j'ai eu l'occasion de travailler, ayant pour thème l'informatique responsable, se divise en 3 parties majeures. D'abord, la partie sur le DSL (Domain-Specific Language) réalisé par un étudiant-doctorant, Jorge Andrés LARRACOECHEA. En effet, il travaille principalement sur la création d'un DSL dans le but d'effectuer un profilage du comportement des logiciels à partir des phases initiales du SDLC (Software Development Life Cycle) et d'une méthodologie assistée par un outil personnalisé pour rendre la construction et l'évaluation énergétique des profils plus faciles. Son DSL, nommé BBCEP (Behavior-Based Consumption Profiles), permet aux concepteurs et architectes de logiciels de générer des descriptions du comportement de n'importe quelle unité de logiciel. BBCEP permet également de générer des études diachroniques des logiciels, d'estimer une consommation de ressources locale (profil unique) et globale (collections de profils), et de générer une estimation de la consommation d'énergie et un score énergétique (local ou global). La seconde partie est réalisée aussi par un doctorant, Hernán Humberto Álvarez Valera. Il étudie les méthodes nécessaires pour déployer et redéployer des composants logiciels, c'est-à-dire des microservices, afin d'économiser de l'énergie. Afin d'évaluer ces méthodes, il a développé un simulateur capable d'effectuer une modélisation des différents scénarios distribués. Ces méthodes prennent en compte la relation entre la consommation d'énergie, l'utilisation des composants matériels, et les différentes définitions de QoS (Quality of Service) des applications. Enfin, la troisième partie, constituant donc le sujet de mon stage, est la partie sur l'étude des données de la consommation énergétique réelles. Les missions de ce stage se composent en 2 principales parties. D'abord, la 1ère mission consiste à déployer des outils de mesure et de stockage de la consommation énergétique (l'API Scaphandre, powerAPI, Prometheus et Influxdb), dans le but donc de collecter, en temps réel, des métriques de consommation énergétique et un ensemble d'informations qui y sont liées, provenant de différentes sources (serveurs, machines virtuelles et applications). Par la suite, à l'aide des outils spécifiques de visualisation (Grafana et Chronograf), un langage de requête intégré conçu pour Prometheus et un autre pour InfluxDB, la 2ème mission consiste à visualiser ces données sous forme de graphiques. Ces graphiques seront présentées sous forme d'un tableau de bord efficace en termes de compréhension et d'analyse. Les graphiques obtenues seront facilement compréhensibles, par les futurs usagers du tableau de bord, grâce à des métaphores parlantes de ces consommations permettant de remplacer l'unité de mesure Watt par une autre unité « du monde réel ».

Aspects techniques – conception et réalisation

Conception

Dans cette sous partie, je vais parler essentiellement du planning que j'ai mis en place lors de mes premières semaines du stage, ainsi que des différents outils techniques que j'ai utilisés afin d'atteindre mon objectif fixé au départ qui est l'obtention d'un tableau de bord résumant la consommation énergétique des serveurs et un autre tableau de bord pour une machine virtuelle.

Répartition du temps de travail et des tâches

Afin de bien mener mes tâches, j'ai d'abord réalisé un planning résumant les objectifs à atteindre pour chaque semaine. Le planning, que j'ai fixé, a évolué tout au long du stage, à travers les mises au point hebdomadaires effectuées avec mon tuteur professionnel Phillipe Roose. Pendant ces mises au points, je présente ce que j'ai réalisé dans la dernière semaine passée, d'autres axes de recherche et d'améliorations m'ont été proposés comme la recherche d'autres sources de données de consommation énergétique des serveurs dans le but de les comparer avec les sources de données de consommation énergétique déjà obtenues, et la résolution de quelques problèmes techniques rencontrés lors de mon stage comme le fait de ne pas pouvoir accéder directement aux hyperviseurs de l'IUT pour des raisons de sécurité. Voici donc un tableau résumant, d'une manière hebdomadaire, mon travail effectué au sein du laboratoire pendant les 8 premières semaines de mon stage.

La semaine du	Descriptions
9 Mai 2022	La première semaine du stage est principalement dédiée à la recherche des APIs (Application Programming Interface) de mesure de la consommation énergétique que seront utilisées pendant le stage. Cela a commencé d'abord par la compréhension du problème que pose une VM afin de mesurer sa consommation énergétique (en raison de l'absence du répertoire RAPL (Running average power limit) permettant de fournir les métriques de consommation énergétique), pour enchaîner par la suite par la recherche des outils dédiés à ce type de mesure et enfin le choix entre les 3 outils trouvés Scaphandre, powerAPI et powerJoular. La principale difficulté de cette semaine et celle d'après est de chercher donc la bonne API permettant de collecter les métriques d'une VM parce que le répertoire intel :rapl, fournissant ces métriques que nous cherchons à obtenir, RAPL, n'est pas disponible sous une VM.
16 Mai 2022	Une fois les outils de mesure trouvés, j'ai commencé par regarder le fonctionnement de chaque API trouvée. J'ai commencé donc par powerAPI. Malheureusement, le module de powerAPI permettant de mesurer les métriques de consommation énergétique d'une VM, VirtualWatts, ne fonctionne pas avec une version du kernel récente (5.15.0-33-generic). Concernant la deuxième API trouvée,

	<p>PowerJoular, il ne propose pas la mesure des métriques de la consommation énergétique d'une VM.</p> <p>Après avoir installé un hyperviseur KVM et mettre en place une VM, je me suis donc tourné vers la dernière API trouvée, Scaphandre.</p> <p>Malgré quelques problèmes rencontrés au départ liés au nombre de CPUs virtuels de la VM, cette API a bien fonctionné et donc les différentes métriques proposées par cette API ont bien été récupérées.</p>
23 Mai 2022	<p>Une fois j'ai obtenu les métriques de consommation énergétiques d'une VM, je me suis directement lancé à chercher comment afficher ces métriques, en temps réel, d'une manière graphique. Pour cela, j'ai cherché les outils de visualisation et de stockage de données permettant d'afficher les données d'une source de données en temps réel.</p> <p>Grafana et Prometheus, compatibles parfaitement en termes de fonctionnement par l'API Scaphandre, ont donc été choisis.</p> <p>J'ai donc bien réussi à réaliser, à titre d'exemple, un tableau de bord, composé seulement de 2 graphiques, qui sera amélioré en termes de nombre de métriques et d'organisation, dans les semaines à venir.</p> <p>Afin de concrétiser les étapes choisies, j'ai réalisé un tutoriel détaillé (voir Annexe pour plus de détails), avec quelques remarques importantes liées à la configuration de la VM, permettant de mettre en place un dashboard de visualisation des métriques de consommation énergétique en utilisant l'API Scaphandre.</p>
30 Mai 2022	<p>Après avoir rendu le tutoriel de la semaine précédente, j'ai passé la majorité du temps de cette semaine, à chercher les différentes possibilités d'amélioration du dashboard obtenu en termes de conversions de consommation énergétique en métaphore plus parlantes. Pour cela, j'ai réalisé un tableur afin de les résumer. De plus, j'ai commencé à chercher à comprendre le fonctionnement d>InfluxDB (la base de données de stockage en temps réel qui sera utilisée pour l'envoi de données depuis les serveurs du site de Pau vers la base de données InfluxDB du site d'Anglet). Ensuite, j'ai mis en place une base de données InfluxDB sur la VM du laboratoire (nommée « greenit ») afin de préparer la réception des</p>

	données de la consommation énergétique depuis les deux serveurs distants situés sur le site de Pau ; kvm-0 et kvm-1. J'ai installé aussi Chronograf, l'outil de visualisation des données proposé par InfluxDB, afin de visualiser des données de la base de données InfluxDB créée et donc exécuter les requêtes InfluxQL pour filtrer les métriques reçues.
6 Juin 2022	La récupération des données depuis la bare-metal du site de Pau (sous InfluxDB) et leurs affichages dans le dashboard avec Chronograf ont bien été réalisés pendant cette semaine. J'ai reçu 3 métriques primordiales des 2 serveurs; la consommation du DRAM en Watts, la consommation globale, et le TDP.
13 Juin 2022, 20 Juin 2022, et 27 Juin 2022	<p>Les principales mission de ces 3 semaines sont ; le démarrage de la rédaction du rapport du stage, l'amélioration des dashboards obtenus en termes de choix des métriques, la recherche des outils de monitoring des applications et la préparation des diaporamas afin de présenter, à mon tuteur professionnel, Phillipe Roose, ce que j'ai réalisé pendant ces premières semaines du stage.</p> <p>Pour la semaine du 20 Juin, j'ai participé à l'organisation d'un événement annuel portant sur l'IOT et l'AI (https://ie2022.iutbayonne.univ-pau.fr/).</p> <p>Pendant cette période, j'ai passé du temps aussi à comprendre les nouvelles métriques reçues, des deux serveurs, concernant les métriques pour le CPU et le débit réseau.</p>

Outils utilisés lors de mon stage

Dans cette sous-partie, je vais présenter et expliquer le fonctionnement des différents outils que j'ai utilisés afin d'aboutir à un tableau de bord final permettant d'afficher les métriques de consommation énergétique. Pour cela, je vais d'abord commencer par l'outil permettant uniquement de mesurer ces métriques, par la suite j'aborderai la partie des outils de stockage des données et enfin les outils de visualisation graphique choisis.

L'outil de mesure des métriques de consommation énergétique - Scaphandre

Scaphandre est une API ayant pour objectif de mesurer les métriques de consommation d'énergie des services technologiques. Il permet de fournir l'énergie consommée par un seul processus sur un serveur ou une machine virtuelle.

Afin de comprendre facilement le calcul de la quantité d'énergie consommée par un processus unique exécuté sur une machine (serveur ou machine virtuelle), nous pouvons imaginer un rectangle contenant des traits représentant chacun le temps de calcul allouées à chaque processus. Pour les machines travaillant sur différents processus en même temps (c'est-à-dire travaillant sur un processus pendant un court intervalle de temps, puis un autre

intervalle de temps et ainsi de suite), nous appelons ces intervalles les jiffies. Chaque processus conserve un total cumulé du nombre total de jiffies qui lui sont alloués. Ainsi, pour connaître la quantité de ressources d'une machine utilisées par un processus donné, il suffit de connaître le nombre total de jiffies utilisés. Afin de connaître donc la puissance utilisée par un processus donné, nous comptons d'abord les jiffies utilisées par ce processus lorsqu'il est en cours d'exécution. Ensuite, pour chaque jiffy, nous vérifions la quantité d'énergie consommée à ces moments précis. En regroupant toutes les lectures de puissance pour tous les jiffies sur un intervalle de temps bien précis, nous pouvons arriver à un chiffre utilisable pour la quantité d'énergie utilisée en termes de wattheures.

Sans déployer Scaphandre, ou une API du même type, il est bien évidemment possible de savoir la partie la plus importante des ressources d'une machine en termes d'utilisation par un processus donné. Par contre, si nous voulons trouver la quantité d'énergie utilisée par processus, nous devons obligatoirement savoir quelle quantité d'énergie est utilisée par la machine.

Les informations liées à la consommation énergétique sont extraites à l'aide de la technologie RAPL. Il s'agit d'une technologie intégrée dans les processeurs Intel et les processeurs AMD, ayant l'architecture x86, produits après l'année 2012.

Le module powercap, se trouvant entre Scaphandre et ces données de consommation énergétique, écrit la consommation énergétique dans des fichiers, qui seront, par la suite, lus par Scaphandre. Après avoir été lus, Scaphandre stocke ces données dans des tampons, et permet, par conséquent, un traitement supplémentaire à travers les différents exportateurs.

L'API Scaphandre est composé de deux parties principales ; un capteur et un exportateur.

La 1^{ère} partie, le capteur, est destinée donc à obtenir les métriques de consommation d'énergie de l'hôte, et les mettre à la disposition de l'exportateur. Par exemple, PowercapRAPL obtient et transforme les métriques provenant du noyau powercap Linux, qui sert d'interface pour obtenir les données de la fonction RAPL des processeurs x86. Le capteur PowercapRAPL collecte d'abord les mesures de consommation d'énergie, puis il les convertit en mesures de consommation d'énergie. À chaque fois que l'exportateur, comme l'exportateur prometheus, demande une mesure, par exemple à chaque fois qu'une demande arrive, que nous appelons demande i, PowercapRAPL lit les valeurs des compteurs d'énergie de powercap. Ensuite, ce capteur les stocke et effectue le même fonctionnement pour les statistiques d'utilisation du processeur et pour chaque processus en cours d'exécution sur la machine en temps réel. Maintenant, entre 2 demandes de mesures, demande i et demande i+1, nous avons la possibilité d'obtenir le sous-ensemble de consommation d'énergie lié au PID d'un processus. Donc pour savoir ce qu'un service consomme en réalité, il suffit de joindre la consommation de tous les PID associés. Il est important de noter que cette fonctionnalité n'est pas disponible directement pour les machines virtuelles. Dans ce cas, avec l'exportateur QEMU, il faudra d'abord exécuter Scaphandre sur l'hyperviseur (bare-metal), et ensuite rendre les métriques de la VM disponibles.

Concernant la 2^{ème} partie, l'exportateur est destiné à demander aux capteurs d'obtenir de nouvelles mesures et de les stocker pour une utilisation potentielle ultérieure. L'exportateur permet donc d'exporter les métriques actuelles. Par exemple, l'exportateur prometheus expose les métriques sur un point de terminaison HTTP, pour être extraites par une instance prometheus. Alors que l'exportateur stdout expose simplement les métriques sur la sortie standard. Concernant l'exportateur qemu, il est destiné à collecter des métriques liées à l'exécution de machines virtuelles sur un hyperviseur Qemu/KVM. Ces métriques seront mises à la disposition de chaque machine virtuelle en exécutant le capteur PowercapRAPL avec l'option -vm. L'exportateur Qemu place les métriques de la machine virtuelle dans des fichiers de la même manière que le module de noyau powercap le fait. Il imite ce

comportement afin que le capteur puisse agir de la même manière qu'il le ferait sur une machine non virtuelle.

Pour conclure sur cette partie, afin de mesurer et donc suivre cette quantité d'énergie utilisée par la machine elle-même, nous aurons besoin d'un capteur. Ce qui nous permettra donc de nous fournir un ensemble d'informations portant sur la quantité d'énergie utilisée en Watts. Maintenant, à partir de ces données en wattheures, nous pouvons, dans la partie de la visualisation des données, les convertir en données plus concrètes à l'aide des requêtes promQL ou influxQL, ce qui nous permettra d'avoir une visualisation graphique des données assez efficace et significative.

Les outils de stockage des données

Prometheus

Prometheus est une application logicielle gratuite créée en 2012 par SoudCloud. Depuis 2016, cette application fait partie de la Cloud Native Computing Foundation. Elle sert à surveiller toute série chronologique purement numérique en les enregistrant et traitant. Elle rassemble, organise et stocke des métriques à partir des plates-formes d'infrastructure, des applications et des services, en prenant en compte aussi les métriques des points de terminaison HTTP. Prometheus peut également collecter les métriques de sa propre santé et de les surveiller. Cet écosystème est donc une solution qui englobe, à la fois, la gestion d'un modèle de données multidimensionnel et la collection de métriques évolutive tout en gardant une certaine simplicité opérationnelle.

Elle est développée avec le langage de programmation Go et possède son propre langage de requête, promQL, ce qui facilitera la manipulation et l'analyse des données collectées. Prometheus est un écosystème ayant plusieurs composants fonctionnant ensemble dans le but de générer des rapports sur les performances d'un système. Ainsi, Prometheus collecte et stocke les métriques des applications qui exposent les métriques dans un format de texte brut à travers des points de terminaison HTTP.

Les schémas ci-dessous sont complémentaires. Ils permettent de mieux comprendre l'architecture de l'écosystème prometheus que nous avons utilisé afin de construire notre tableau de bord.

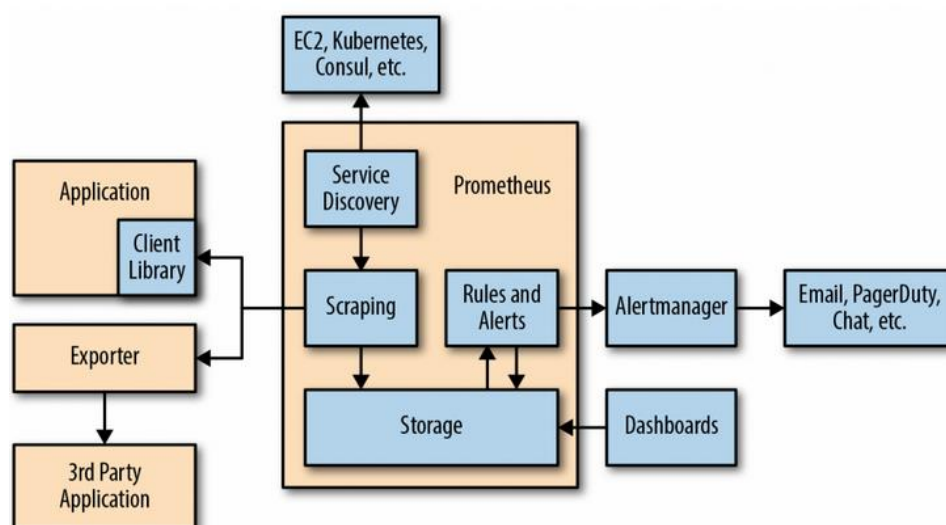


Figure 2: L'écosystème Prometheus - version 1

(Source : <https://www.devopsschool.com/blog/what-is-prometheus-and-how-it-works/>)

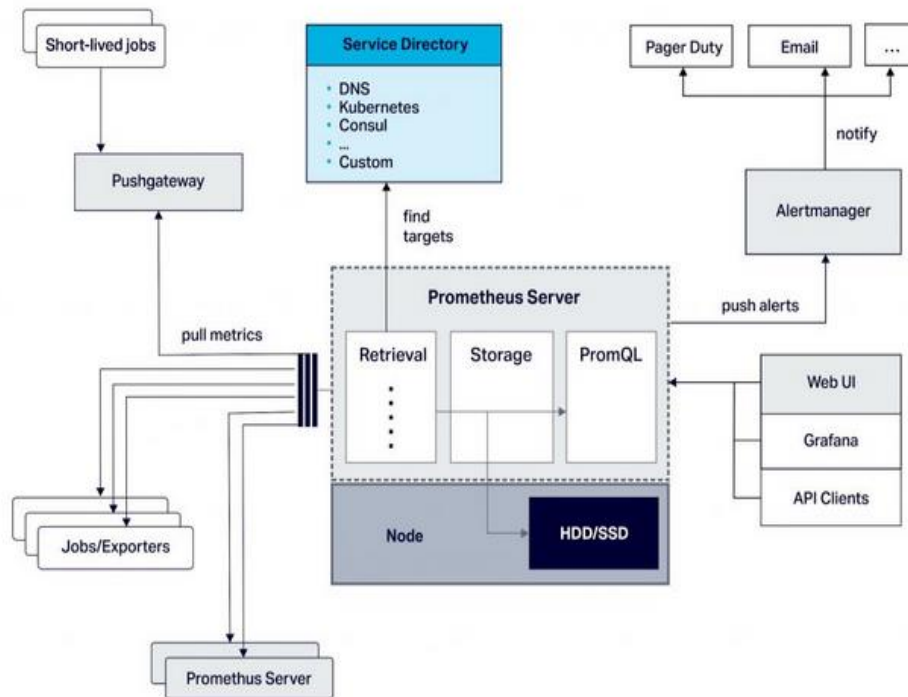


Figure 3: L'écosystème Prometheus - version 2

(Source : <https://www.devopsschool.com/blog/what-is-prometheus-and-how-it-works/>)

Le premier composant de cet écosystème est le serveur Prometheus. Il s'occupe à la fois du stockage des métriques, et de la planification des tâches de surveillance en interrogeant les sources de données à une fréquence d'interrogation prédéfinie. Les tâches de surveillance sont gérées à travers le fichier de configuration YAML, et configurées en utilisant la directive « scrape config ».

Dans le 1er schéma ci-dessus, nous avons un élément qui se trouve dans le serveur Prometheus qui s'appelle SD. En effet, afin de connaître les cibles depuis lesquels la collecte des données se fera, Prometheus s'appuie sur plusieurs mécanismes de SD. Par exemple, nous pouvons avoir une SD basée sur des fichiers que les implémentations personnalisées SD peuvent utiliser. Ceci est donc possible en gérant directement un fichier de type YAML contenant toute une liste de cibles. De plus, Prometheus fournit aussi d'autres implémentations SD comme Kubernetes ou Amazon Elastic Compute Cloud comme le montre les deux schémas ci-dessus.

Le second composant de cet écosystème est l'exportateur Prometheus. Les exportateurs permettent de collecter des métriques à partir d'un système tiers spécifique et les rendre par la suite disponibles afin que les serveurs Prometheus puissent les récupérer. Les bibliothèques clientes, utilisées par des applications, activent un point de terminaison HTTP où les métriques internes sont exposées et collectées ensuite par les serveurs Prometheus.

Le troisième composant est les bibliothèques clients. En effet, les applications ne peuvent fournir des métriques qu'après l'ajout de l'instrumentation à leurs codes en utilisant directement les bibliothèques client Prometheus.

Le quatrième composant est Alertmanager. L'objectif principal de cet élément est de gérer les alertes envoyées par le serveur Prometheus en envoyant des notifications via plusieurs moyens comme le montre les 2 schémas ci-dessus.

Le cinquième composant est le pushgateway. Ce composant permet de récupérer les métriques des services externes à durée courte.

Enfin, nous avons le composant Web UI. Il s'agit d'une application web ayant pour but de visualiser sous forme de graphique ou de tableau, en temps réel, le résultat d'une requête promQL. Afin d'afficher les graphiques des métriques de ma VM, j'ai donc utilisé Grafana.

InfluxDB

InfluxDB est un système de gestion de base de données qui permet d'enregistrer un volume important de séries temporelles. Comme Prometheus, elle est développée en langage de programmation Go. Il est à l'écoute, par défaut, sur le port 8086 et possède son propre langage de requêtes InfluxQL très proche du SQL, et Flux. Les bases de données InfluxDB stockent des mesures. Une base de données InfluxDB est composée d'une colonne représentant le temps en Epoch time, d'une colonne ou plusieurs colonnes représentant les tags key(s), et d'une ou plusieurs field key(s) représentant les field key(s). Les séries représentent l'ensemble des combinaisons, que nous pouvons obtenir, entre la mesure et les tags keys. InfluxDB stocke les données dans des shard groups. Ces groupes stockent les données par shard duration c'est-à-dire par intervalle de temps bien défini. C'est ce que nous appelons la rétention de données.

Les outils de visualisation

Les deux outils de visualisation utilisés sont Grafana et Chronograf. Voici une présentation de ces 2 outils. Nous avons constaté qu'ils sont très similaires en termes de fonctionnalités mais nous allons utiliser ces 2 outils pour des raisons d'organisation.

Grafana

Grafana est une API open-source développée en langage de programmation Go et Node.js. Elle permet d'afficher graphiquement des données chronologiques collectées par d'autres applications ce qui permettra par la suite la facilité d'analyser ces métriques. L'avantage de cet outil est qu'il offre une connexion avec différentes sources de données possibles comme Prometheus, Influx DB, et MySQL. Par exemple, pour visualiser les métriques de la consommation énergétique d'une VM, j'ai utilisé la combinaison Prometheus-Grafana. Grafana constitue donc la partie front-end permettant l'affichage des données et Prometheus la partie back-end permettant de stocker les données de séries chronologiques.

De plus, l'outil Grafana possède une solution d'alerte intégrée. Ceci permet de prévenir l'utilisateur de différents problèmes, qui peuvent arriver, en envoyant des notifications par e-mail par exemple. Il peut utiliser également les règles d'alerte définies par Prometheus, Loki et Alertmanager.

Chronograf

Chronograf est une application web open source. Cette application est développée par InfluxDB. Cet outil donc nous permet de créer des tableaux de bord pré-crés ou personnalisés afin de visualiser graphiquement en temps réel les données collectées dans des bases de données InfluxDB à travers les requêtes InfluxQL ou Flux. Avec l'utilisation de Kapacitor, Chronograf offre la possibilité de création des alertes. Et l'exécution de tâches ETL. De plus, Chronograf nous offre la possibilité d'activation et de désactivation des règles d'alerte, d'affichage des alertes actives, ainsi que l'envoi des alertes à des terminaux comme l'email.

Ces 2 outils se ressemblent beaucoup et sont très riches en termes de fonctionnalités qu'ils proposent. Ainsi, pour travailler avec Prometheus, il est plus judicieux d'utiliser Grafana, et pour travailler avec InfluxDB, il est plus cohérent d'utiliser Chronograf.

Réalisation

Dans cette partie, je vais présenter les missions que j'ai effectuées jusqu'à maintenant. Je vais d'abord commencer par expliquer et décrire la première mission qui est la mise en place d'un tableau de bord pour la visualisation des métriques de consommation énergétique depuis une VM, ensuite je vais enchaîner par l'explication de la mise en place du tableau de bord contenant les métriques de consommation énergétique de 2 serveurs distants.

Tableau de bord des métriques d'une VM

Comme les APIs de mesure de la consommation énergétique existantes actuellement ne permettent pas malheureusement d'accéder directement aux informations de consommation énergétique d'une VM et que je n'ai pas un accès malheureusement aux hyperviseurs du laboratoire, j'ai installé un hyperviseur KVM et configuré une VM parce que l'outil de mesure de la consommation énergétique choisi, l'API Scaphandre, fonctionne seulement avec ce type de système pour la virtualisation de serveur. J'ai commencé par installer tous les packages que j'aurai besoin en exécutant les 2 commandes shell suivantes: **sudo apt -y install qemu-kvm libvirt-daemon bridge-utils virtinst libvirt-daemon-system** pour l'installation du serveur de virtualisation de l'hyperviseur KVM, puis **sudo apt -y install virt-top libguestfs-tools libosinfo-bin qemu-system virt-manager** pour l'installation des outils de gestion de machines virtuelles. Après avoir téléchargé le fichier .iso de Linux Ubuntu, j'ai donc créé une machine virtuelle en utilisant directement le gestionnaire des machines virtuelles que j'ai installé. Une des difficultés rencontrées lors de cette partie est de mon rendre compte du non fonctionnement de l'API Scaphandre avec plus d'un seul processeur virtuel. J'ai donc configuré une machine virtuelle avec un seul vCPU.

Il faut savoir que le problème majeur dans la mesure de la consommation d'énergie est de le faire à l'intérieur d'une VM parce que d'une manière générale la VM n'a pas accès aux mesures de puissance. L'avantage de Scaphandre est qu'il résout cette problématique en permettant une communication entre une instance Scaphandre sur l'hyperviseur et une autre instance s'exécutant sur la VM. Ainsi, l'agent Scaphandre sur l'hyperviseur calculera les métriques pour cette VM et celui sur la VM accédera par la suite à ces métriques.

Après avoir cloné le répertoire « Scaphandre » sur la machine bare-metal et charger le module « intel_rapl_common » permettant de mesurer la consommation énergétique sous Ubuntu, j'ai créé le fichier binaire, qui permet de lancer l'API Scaphandre, avec **cargo build --release**. Ainsi, le fichier binaire se trouve dans le répertoire **scaphandre/target/release/**. Concernant la partie hyperviseur, j'ai enfin lancé Scaphandre avec l'exportateur QEMU en exécutant la commande suivante: **sudo ./scaphandre qemu**. Maintenant je vais pouvoir donner accès à la mv créée à ses métriques mesurées depuis l'hyperviseur. Pour cela, j'ai créé un point de montage tmpfs via l'exécution de la commande shell suivante permettant de monter le répertoire de la VM **/var/lib/libvirt/scaphandre/ubuntu22.04**: **sudo mount -t tmpfs tmpfs_ubuntu22.04 /var/lib/libvirt/scaphandre/ubuntu22.04 -o size=5m** avec comme taille maximale du système de fichiers allouée 5 Mo.

En ce qui concerne la VM, j'ai ajouté un nouveau matériel virtuel en mettant les paramètres suivants. Pour le « Pilote », j'ai sélectionné « virtio-9p ». Pour le chemin de source, j'ai mis « **/var/lib/libvirt/scaphandre/ubuntu22.04** ». Enfin, j'ai mis « scaphandre » pour le chemin cible, et coché la case « Exporter le système de fichiers en lecture seule ».

En ce qui concerne la VM, comme pour l'hyperviseur, j'ai cloné le répertoire « Scaphandre », et j'ai créé le fichier binaire qui permet de lancer l'API Scaphandre. Dans le répertoire « var », j'ai créé un répertoire nommé « scaphandre ». Ensuite, j'ai monté le système de fichiers sur cette VM à l'aide de la commande suivante : **sudo mount -t 9p -o trans=virtio scaphandre /var/scaphandre**. J'ai lancé l'API Scaphandre sur cette VM en exportant les métriques avec l'exportateur Prometheus avec cette commande shell: **sudo ./scaphandre -vm prometheus**.

Les métriques sont ainsi collectables, d'une manière donc non graphique, via le navigateur en tapant directement l'adresse IP de ma VM suivi du numéro de port 8080.

Afin de pouvoir afficher ces métriques de consommation énergétique, j'ai trouvé une solution assez efficace intégrée à l'écosystème Prometheus.

Pour cela, il me faut 3 éléments importants : l'API Scaphandre fonctionnant avec l'exportateur Prometheus, afin de mesurer les métriques de la consommation énergétique en temps réel de la VM, Prometheus afin de les collecter, et Grafana, afin de les visualiser graphiquement en s'appuyant sur une connexion source de données vers le serveur Prometheus.

Pour la visualisation graphique, j'ai commencé donc par installer Prometheus à l'aide de la commande `wget` :

`wget https://github.com/prometheus/prometheus/releases/download/v2.35.0/prometheus-2.35.0.linux-amd64.tar.gz.`

Puis, j'ai créé un fichier de configuration afin de rendre ma VM reconnue par Prometheus, en rajoutant dans « targets » l'adresse IP de la VM, que j'ai créée, suivie du numéro de port 8080 destiné aux métriques récoltées par l'API Scaphandre, 192.168.122.153:8080, ce qui permettra par la suite de stocker les métriques de la consommation énergétique dans le serveur Prometheus. Voici le contenu de ce fichier :



```
slim@slim-Modern-15-A11MU: ~/Documents/prometheus/prometheus-2.35.0.linux-amd64$ cat exporter-config.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: node
    static_configs:
      - targets: ['localhost:9100', '192.168.122.153:8080']
```

Figure 4: Fichier de configuration pour lancer Prometheus

J'ai lancé Prometheus avec l'indicateur «`--config.file`» afin de prendre en compte le fichier créé ci-dessus: **`./prometheus --config.file=exporter-config.yml`**, et j'ai installé Grafana. Pour cela, j'ai d'abord mis à jour les informations du package avec les 3 commandes suivantes : **`sudo apt-get install -y apt-transport-https`**, **`sudo apt-get install -y software-properties-common wget`** et **`wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -`**.

Ensuite, j'ai ajouté un repository de Grafana à l'aide de la commande suivante :

`echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list`. Puis, j'ai installé Grafana avec: **`sudo apt-get install grafana-enterprise`**.

Je suis enfin capable de commencer à construire le tableau de bord avec Grafana et visualiser les graphiques des métriques de consommation énergétique de la VM. J'ai donc lancé Grafana en exécutant les commandes suivantes: **`sudo systemctl daemon-reload`**, afin de recharger la configuration du gestionnaire systemd, et **`sudo systemctl start grafana-server`**.

Cet outil de visualisation est maintenant accessible via le port 3000. Après avoir défini la source de données Prometheus, et saisi l'adresse IP du serveur Prometheus qui fonctionne sur le port 9090, j'ai bien obtenu les métriques de la consommation énergétique de la VM

fournies par l'API Scaphandre. Grace au langage de requêtes PromQL, j'ai donc créé le tableau de bord suivant :

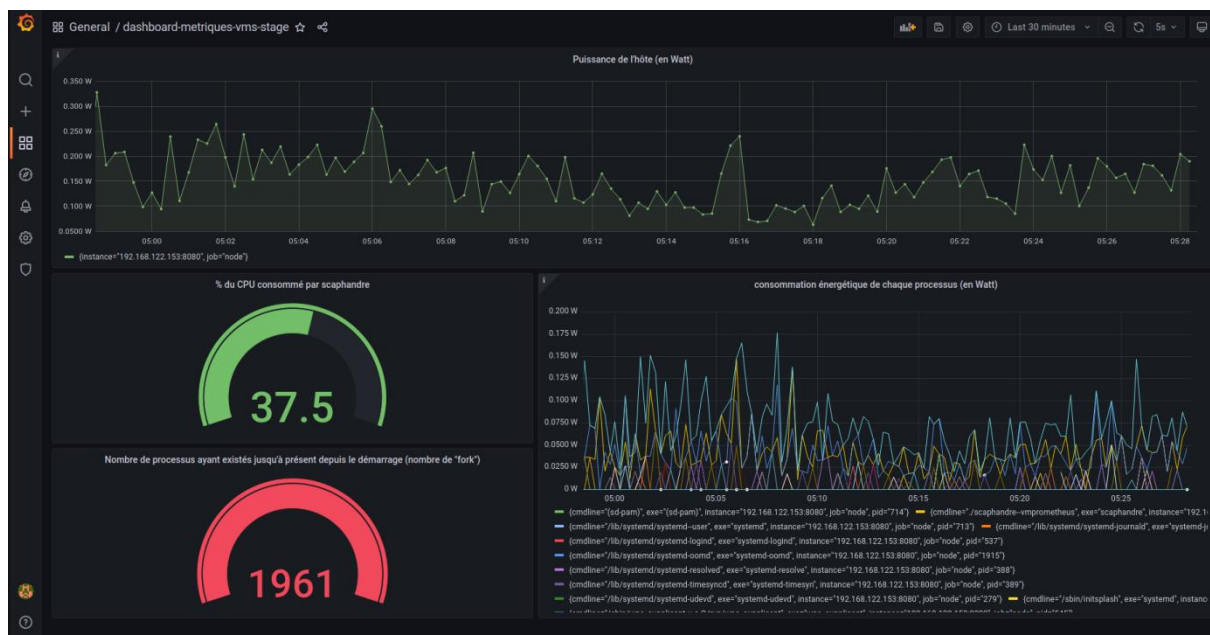


Figure 5: Capture d'écran du 1er dashboard réalisé

Par exemple, pour obtenir le premier graphique de la figure ci-dessous, représentant la mesure de puissance sur l'ensemble de l'hôte Watts « Puissance de l'hôte (en Watt) », j'ai exécuté la requête PromQL suivante en la divisant par 1000000:

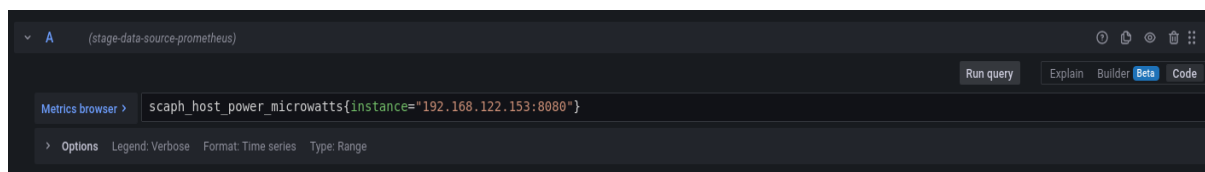


Figure 6: Capture d'écran d'un exemple d'une requête PromQL

Les graphiques à droite de la figure précédente représentent la consommation énergétique en Watts de chaque processus entrain de s'exécuter sur la VM.

Tableau de bord des métriques de 2 serveurs distants

Le but de cette partie est d'extraire et visualiser les métriques de consommation énergétique de 2 hyperviseurs situés à Pau (kvm-0 et kvm-1).

Afin de faciliter la compréhension sur l'envoi des métriques depuis le site, sachant que j'effectue mon stage sur le site d'Anglet, j'ai réalisé un schéma explicatif ci-dessous :

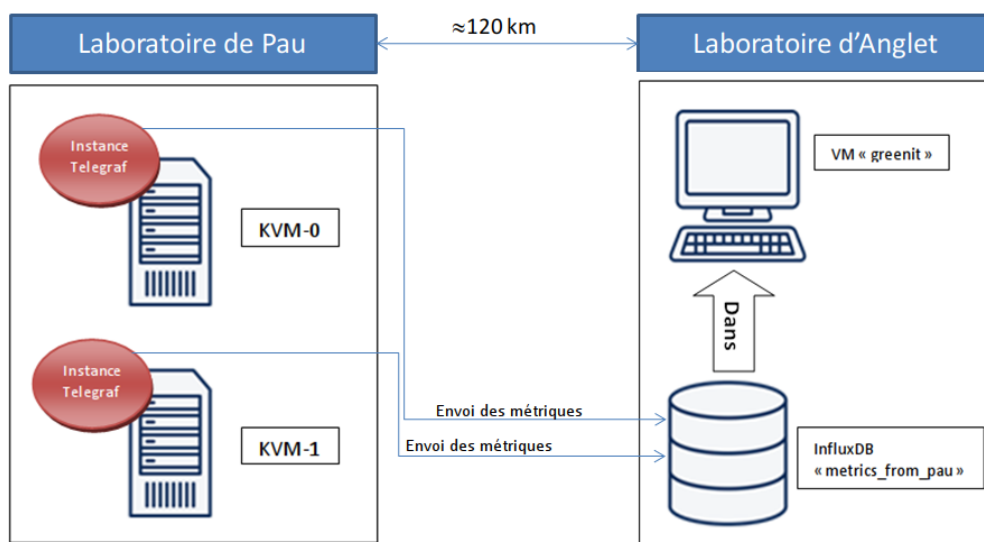


Figure 7: Schéma explicatif sur l'envoi des données depuis le site de Pau vers le site d'Anglet

Pour cela, la DN du laboratoire, située sur le site de Pau, a mis en place, comme le montre le schéma ci-dessus avec les cercles en rouge, des collecteurs, c'est-à-dire des instances de Telegraf, sur les 2 hyperviseurs. Ces instances Telegraf ont pour mission d'envoyer les métriques de consommation énergétique sur 2 bases de données InfluxDB différentes, celle que j'ai créée sur la VM « greenit » sur laquelle j'ai eu pour mission de visualiser les données avec Chronograf, et celle sur une base de données InfluxDB propre à la DN. Sur le schéma ci-dessus, j'ai simplement mis en l'accent sur la partie qui nous intéresse c'est-à-dire l'envoi des données sur la VM du laboratoire d'Anglet.

Comme il n'était pas possible d'avoir accès directement aux hyperviseurs pour déployer l'API Scaphandre (ou une API de même type), la collecte des différentes métriques, concernant la consommation énergétique et le CPU, ont été faites avec ces plugins: <https://github.com/influxdata/telegraf/tree/master/plugins/inputs/cpu> et https://github.com/influxdata/telegraf/tree/master/plugins/inputs/intel_powerstat. De plus, l'avantage d'utiliser des plugins est d'avoir métriques concernant la consommation énergétique.

Voici un schéma résumant le fonctionnement de Telegraf :

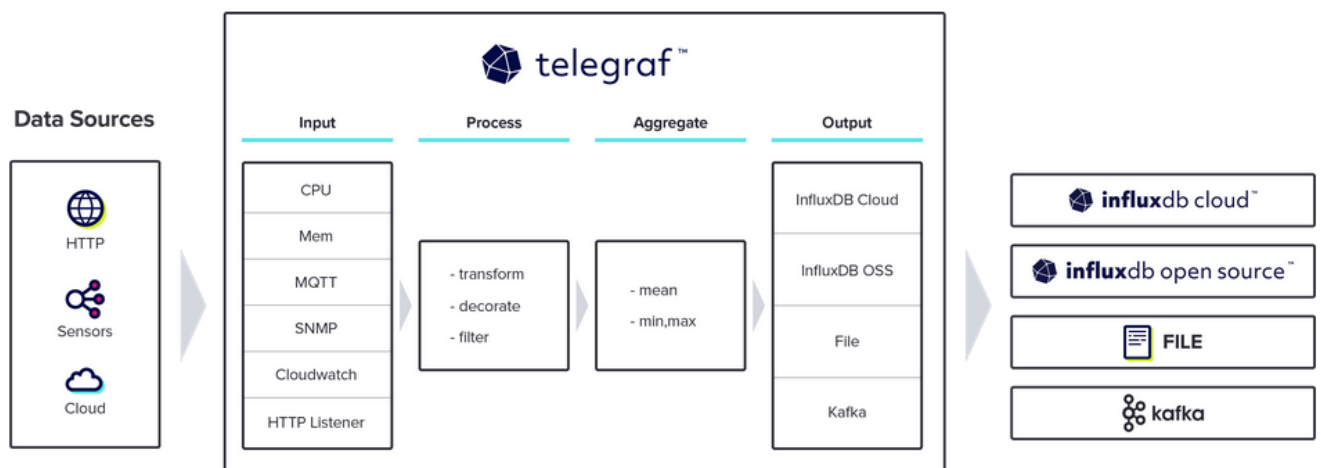


Figure 8: Schéma explicatif de Telegraf

(Source : <https://www.influxdata.com/time-series-platform/telegraf/>)

Telegraf est un agent basé sur un serveur permettant d'effectuer la collecte et l'envoi des métriques à partir de, comme le montre le schéma ci-dessus à gauche, différentes sources de données. D'après le schéma ci-dessus, il existe 4 types de plugins Telegraf (le rectangle au centre) ; les plugins « inputs » qui collectent les métriques des services tierces, les plugins « process » ayant pour but le nettoyage des données avant leurs arrivée, les plugins « aggregate » créant des métriques agrégées, et les plugins « output » qui envoient les données dans différentes sorties possibles comme le montre le schéma ci-dessus à droite.

Afin de mettre en place le tableau de bord avec Chronograf, j'ai créé une base de données InfluxDB afin de pouvoir recevoir le téléversement des données depuis les InfluxDB des serveurs distants. La base de données créée s'appelle « metrics_from_pau ». J'ai créé une politique de rétention de données d'une durée d'une année. Cette rétention s'appelle « forever ». Afin de comprendre l'architecture de cette base de données « metrics_from_pau », j'ai tout d'abord affiché les différentes mesures présentes dans la base de données. Trois packages, contenant les métriques, dont 2 d'entre eux provenant des plugins cités ci-dessus, ont bien été envoyées depuis les serveurs du site de Pau ; « cpu », « net » et « powerstat_package ».

```
> show tag keys from forever.powerstat_package
name: powerstat_package
tagKey
-----
host
package_id
> show field keys from forever.powerstat_package
name: powerstat_package
fieldKey                fieldType
-----
current_dram_power_consumption_watts float
current_power_consumption_watts    float
thermal_design_power_watts         float
```

Figure 9: Capture d'écran depuis la VM "greenit" montrant l'exploration de la base de données InfluxDB créée

Ces métriques, sur la figure ci-dessus, sont collectées par le plugin « Intel PowerStat ». Le plugin PowerStat fonctionne sur Linux. Il a pour but de surveiller les statistiques d'alimentation sur les plates-formes Intel. Les métriques collectées par ce plugin sont collectées à intervalles fixes.

Concernant les tag keys, j'ai eu « package_id ». En effet, les métriques du plugin Intel PowerStat sont collectées par package de processeur. « package_id » est un entier (soit 0 soit 1) qui indique à quel package la métrique se réfère.

Pour les field keys, j'ai reçu 3 métriques, ayant pour unité de mesure les Watts, « current_dram_power_consumption_watts » représentant la consommation électrique actuelle du DRAM du package de processeur, « current_power_consumption_watts » représentant la consommation électrique actuelle du package de processeur et « thermal_design_power_watts » représentant le TDP disponible pour le package de processeur.

Afin de tester le fonctionnement d'InfluxDB avec Chronograf, j'ai seulement effectué la visualisation des 2 premières métriques. En exécutant les requêtes respectivement suivantes pour visualiser la consommation électrique actuelle du package de processeur, la courbe en bleue est pour kvm-0 et la courbe en vert est pour kvm-1:

```
SELECT current_power_consumption_watts FROM  
metrics_from_pau.forever.powerstat_package WHERE "host" = 'kvm-0'
```

et,

```
SELECT current_power_consumption_watts FROM  
metrics_from_pau.forever.powerstat_package WHERE "host" = 'kvm-1',
```

Voici ce que j'ai pu obtenir en Watts :



Figure 10: La consommation électrique de 2 bare-metals distants (en Wattss) en fonction du temps

D'après la figure ci-dessus, la collecte des données dans la base de données InfluxDB a bien débuté le 8 Juin 2022.

De même pour la consommation électrique actuelle du DRAM du package de processeur. Voici les 2 graphiques des 2 hyperviseurs kvm-0 (bleu) et kvm1 (vert) :



Figure 11: La consommation du DRAM des 2 bare-metals distants (en Watts) en fonction du temps

Correspondant à ces 2 requêtes respectivement :

```
SELECT current_dram_power_consumption_watts FROM
metrics_from_pau.forever.powerstat_package WHERE "host" = 'kvm-0'
```

et,

```
SELECT current_dram_power_consumption_watts FROM
metrics_from_pau.forever.powerstat_package WHERE "host" = 'kvm-1'
```

À l'inverse de la première partie avec l'API Scaphandre, la difficulté de cette partie ne réside pas dans l'installation et la configuration des outils utilisés. En effet, la majorité du temps, pour cette partie, est passée sur la compréhension du fonctionnement de la base de données InfluxDB, ainsi que la compréhension des métriques reçues en utilisant les plugins et Telegraf.

Une fois les tableaux de bord réalisés, je cherche maintenant à mieux visualiser les graphiques c'est-à-dire à rendre les graphiques mieux significatifs, ce qui facilitera, pour les futurs usagers ayant n'importe quel type de profil, la compréhension des graphiques en comparant la consommation énergétique à des cas réels de la vie courante.

Réalisation d'un tableur pour les conversions énergétiques métaphoriquement

Pour cela, j'ai réalisé un tableur contenant toutes les conversions énergétiques qui seront appliquées dans les tableaux de bords via les requêtes PromQL et InfluxQL.

Voici le tableur créé :

Tableau 1: Conversions énergétiques (les métaphores) à appliquer sur les 2 dashboards créés

<u>Dispositif</u>	<u>Consommation électrique</u>	<u>Source</u>
Ampoules à LED	9 W	https://www.silamp.fr/consommation-electrique
Batterie de type Galaxy S4 (en la chargeant entièrement)	10 Wh	https://desideespourchangerlemonde.wordpress.com/2015/02/08/combien-ca-consomme-un-smartphone/
Mac Pro (fin 2013) en étant inactif	43 W	https://support.apple.com/fr-fr/HT201796
Console de Sony PS5		https://www.monpetitforfait.com/energie/aides/consommation-console-jeux
en mode veille	1,3 W	
hors jeux + sur le menu principal de la console	47 W	
Vélo équipé d'un kit électrique avec une autonomie de 35 km et une batterie d'une capacité de 0,252 kWh	72 Wh pour 100 km.	https://www.abicyclettepaulette.fr/blogs/blog-velo-electrique/velo-electrique-le-moyen-de-deplacement-le-plus-efficient-en-ville
Téléviseur Samsung 19 pouces	25W	https://www.acheter-tv.fr/conseils-experts/consommation-d-energie

Le tableau ci-dessus pourra évoluer durant la période restante du stage.

Conclusion

En définitive, l'intérêt sur notre recherche repose sur 2 volets importants. La récupération et la visualisation de la consommation énergétique d'une part d'une VM, et d'autre part d'un serveur.

Afin de pouvoir élaborer le suivi de la consommation énergétique des serveurs et des VMs, il existe plusieurs outils différents. Dans le cadre de mon stage, pour les 8 premières semaines, j'ai eu l'occasion de découvrir plusieurs APIs permettant de fournir des métriques de la consommation énergétiques des machines. Pour la mesure de la consommation énergétique d'une VM, j'ai utilisé l'API Scaphandre. Ensuite, j'ai utilisé Prometheus pour le stockage des données fournies par l'API Scaphandre et Grafana afin de les afficher graphiquement. Concernant les métriques de la consommation énergétique des serveurs, j'ai utilisé InfluxDB pour le stockage des données pour la réception, en temps réel, sur la vm « greenit », les métriques des serveurs kvm-0 et kvm-1 depuis le site de Pau, et son outil de visualisation Chronograf pour construire le dashboard des graphiques représentant l'évolution de ces métriques en temps réel.

Pour la période restante du stage, je vais d'abord continuer à améliorer les 2 tableaux de bord avec Grafana et Chronograf en rajoutant plus de métriques fournies par l'API Scaphandre et les nouvelles métriques obtenues récemment, dans la base de données « metrics_from_pau », concernant la consommation énergétique du CPU ainsi que les métriques concernant le débit réseau. Par la suite, je vais procéder à l'amélioration des dashboards en appliquant les conversions du tableur sur les requêtes des graphiques des tableaux de bord obtenus avec Grafana et Chronograf.

En ce qui concerne les perspectives, je vais établir la mesure de la consommation énergétique des applications avec l'API powerJoular et l'afficher sous forme de graphiques dans le dashboard construit avec Grafana. Pour le monitoring des applications, je vais utiliser la librairie prometheus-client de Python. Enfin, je vais regarder la gestion des alertes avec Grafana et Chronograf.

Bibliographie

<https://liuppa.univ-pau.fr/fr/index.html>

<https://hubblo-org.github.io/scaphandre-documentation/>

<https://www.devopsschool.com/blog/what-is-prometheus-and-how-it-works/>

https://github.com/influxdata/telegraf/tree/master/plugins/inputs/intel_powerstat

<https://www.influxdata.com/time-series-platform/telegraf/>

<https://jhooq.com/prometheus-grafan-setup/>

Annexe : Tutoriel pour la visualisation graphique des métriques de la consommation énergétique d'une machine virtuelle avec l'API Scaphandre et Grafana

Visualisation graphique des métriques de consommation d'une machine virtuelle avec l'API Scaphandre et Grafana

Ce tutoriel vous expliquera les installations et les configurations à effectuer afin d'obtenir les métriques de consommation énergétique d'une machine virtuelle, en utilisant l'API Scaphandre pour les mesurer, et Grafana pour les visualiser graphiquement.

Il est composé de 4 parties principales :

1. L'installation de KVM
2. La création d'une machine virtuelle
3. La mise en place et lancement de Scaphandre
4. La visualisation graphique des mesures avec Grafana

Toutes les installations et configurations, dans le cadre de tutoriel, se feront sur linux ubuntu 22.04.

Afin de simplifier la compréhension du tutoriel, les commandes à exécuter seront marquées en gras et en noir. Les définitions et les explications seront marquées en italique et en gris, et les avertissements seront en rouge.

Partie 1 : Installation de KVM

Tout d'abord, il faut impérativement installer le serveur de virtualisation de l'hyperviseur KVM (parce que l'outil, que nous allons utiliser pour les mesures des métriques de consommation énergétique, Scaphandre, ne fonctionne que seulement avec ce type de système pour la virtualisation de serveur)

KVM (Kernel-based Virtual Machine) est une solution de virtualisation gratuite et open source pour les systèmes Linux fonctionnant sur du matériel x86. Il est une combinaison de modules de noyau et d'utilitaires nécessaires pour exécuter des machines virtuelles sur un système hôte. Ces modules incluent par exemple l'émulateur QEMU pouvant exécuter, via donc l'hyperviseur KVM, un ou plusieurs systèmes d'exploitation, virt-install, le démon libvirtd, virt-manager et bien d'autres. KVM convertit Linux en un hyperviseur de type 1 (bare metal).

Pour l'installation des packages, nous allons utiliser le gestionnaire de packages apt. Lancez votre terminal et exécutez les commandes, en gras, suivantes.

Etape 1.1

sudo apt update

sudo apt -y install qemu-kvm libvirt-daemon bridge-utils virtinst libvirt-daemon-system

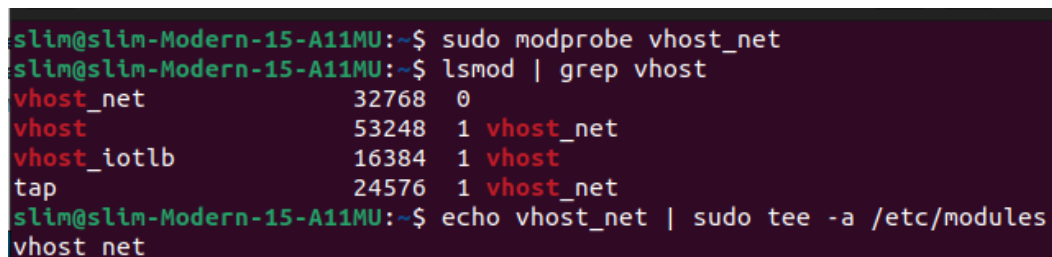
Ensuite, nous allons procéder à l'installation d'autres outils de gestion de machines virtuelles.

Etape 1.2

sudo apt -y install virt-top libguestfs-tools libosinfo-bin qemu-system virt-manager

Assurez-vous que le module vhost_net est chargé et activé. Pour cela, tapez les commandes comme marqué dans la figure ci-dessous.

Etape 1.3



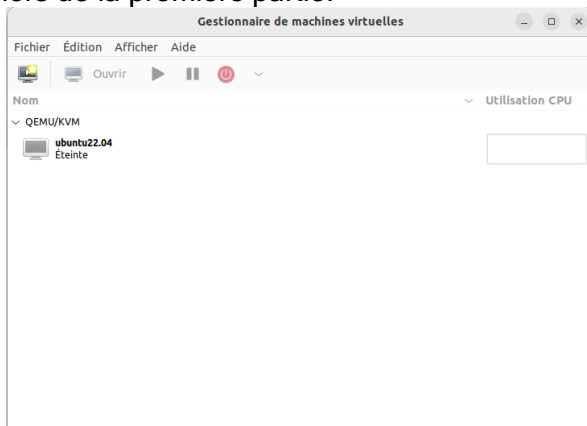
```
slim@slim-Modern-15-A11MU:~$ sudo modprobe vhost_net
slim@slim-Modern-15-A11MU:~$ lsmod | grep vhost
vhost_net          32768  0
vhost              53248  1 vhost_net
vhost_iotlb       16384  1 vhost
tap               24576  1 vhost_net
slim@slim-Modern-15-A11MU:~$ echo vhost_net | sudo tee -a /etc/modules
vhost_net
```

Avant de commencer la seconde étape, veuillez installer un fichier .iso de la machine virtuelle que vous voulez installer. Pour les étapes ci-dessous, j'ai choisi, à titre d'exemple, Linux fedora33.

Partie 2 : Création d'une machine virtuelle en utilisant le gestionnaire des machines virtuelles

Etape 2.1

Dans cette deuxième partie, lancez votre gestionnaire des machines virtuelles, téléchargé lors de la première partie.



Etape 2.2

Comme vous voyez dans l'image ci-dessus, j'ai déjà procédé à une installation d'une machine virtuelle Linux ubuntu 22.04, mais nous allons tout de même procéder à une installation d'une autre machine virtuelle.



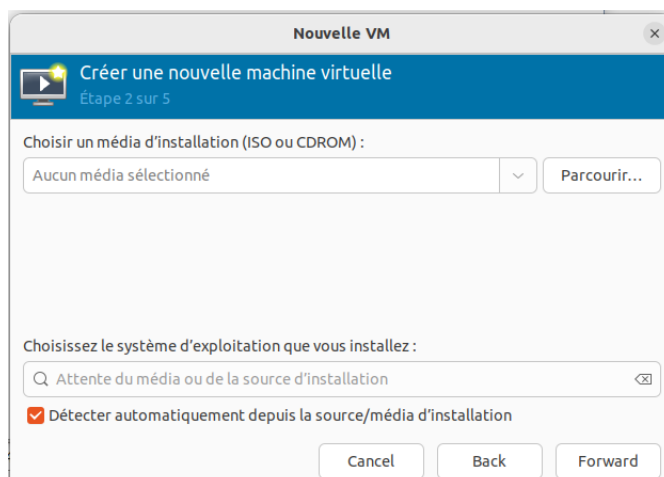
Cliquez sur le bouton, pointé par la flèche, de la figure ci-dessus.

Etape 2.3



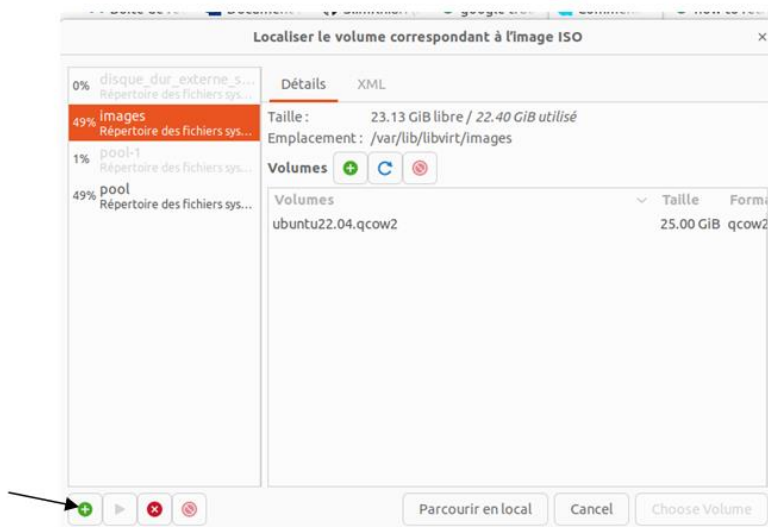
Nous allons donc choisir la première option (parce que nous avons téléchargé un fichier iso), et continuer en cliquant sur le bouton « forward ».

Etape 2.4



Une fois cette la fenêtre ci-dessus obtenu, nous pouvons choisir notre fichier .iso qu'on avait téléchargé avant d'entamer cette deuxième étape. Cliquez sur le bouton « Parcourir... » et vous allez obtenir la fenêtre ci-dessous.

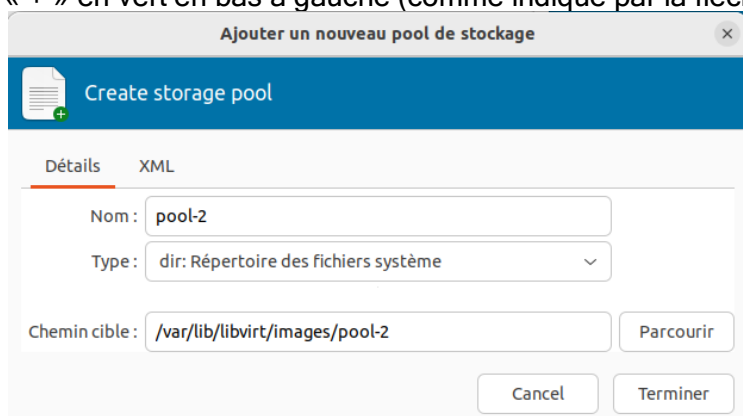
Etape 2.5



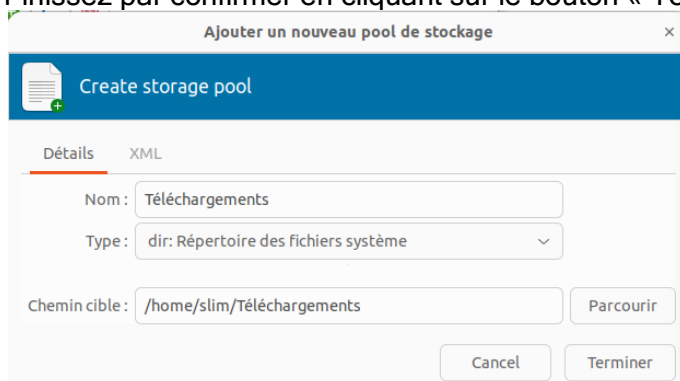
Si vous avez trouvé votre fichier .iso, vous pouvez passer directement à l'étape 2.7. Sinon, regardez l'étape 2.6.

Etape 2.6

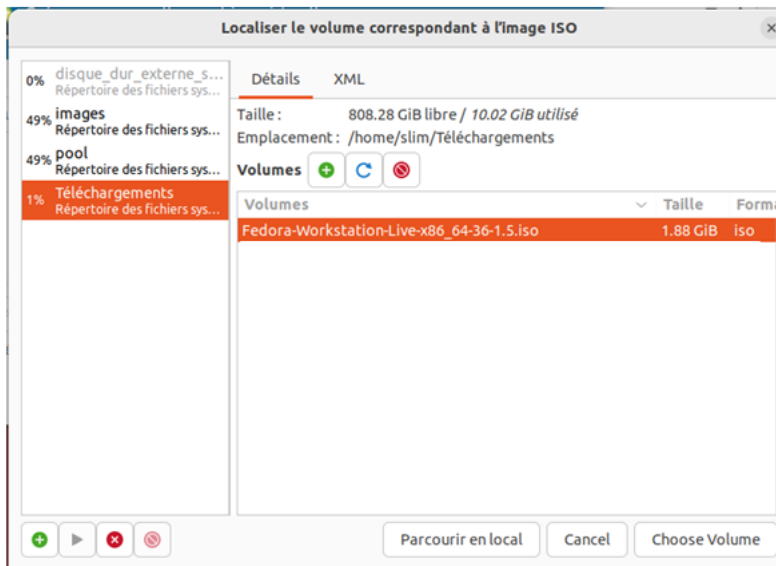
Afin de trouver votre fichier .iso que vous avez téléchargé, vous devez cliquer sur le bouton « + » en vert en bas à gauche (comme indique par la flèche sur la figure ci-dessous).



Une fois la fenêtre ci-dessous obtenue, vous pouvez personnaliser le nom de votre « pool de stockage ». Puisque mon fichier .iso que j'ai téléchargé se trouve dans le répertoire « Téléchargement », je vais changer le nom chemin cible en cliquant sur « Parcourir ». Finissez par confirmer en cliquant sur le bouton « Terminer ».



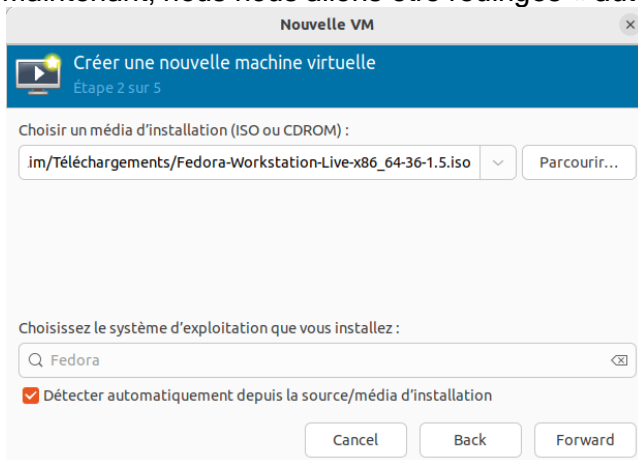
Maintenant, nous allons dans notre « pool de stockage » qu'on avait appelé « Téléchargements » et nous allons choisir simplement notre fichier .iso de notre machine virtuelle comme indiqué dans la figure ci-dessous.



Etape 2.7

Confirmez en cliquant sur « Choose Volume ».

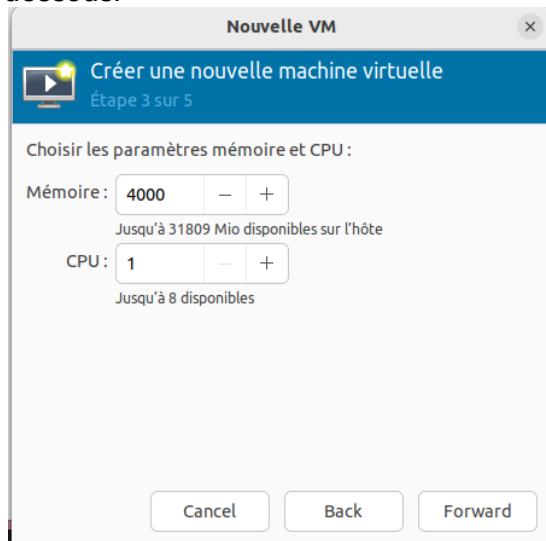
Maintenant, nous nous allons être redirigés « automatiquement » à cette fenêtre.



Continuez en cliquant sur le bouton « Forward ».

Etape 2.8

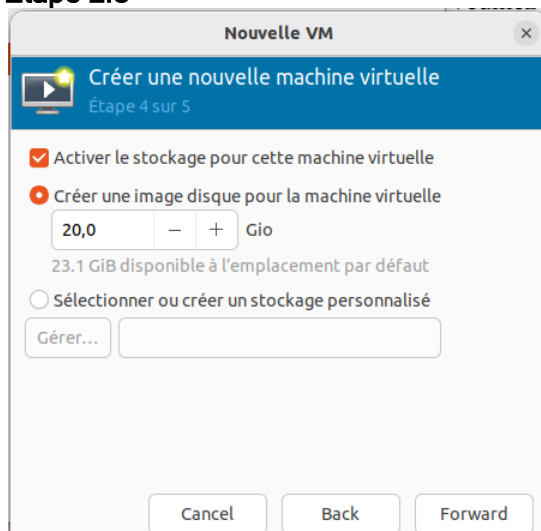
Veillez à ne pas cliquer sur le bouton « Forward » directement sans lire le texte en rouge ci-dessous.



Afin de pouvoir attribuer une mémoire de plus attribuer de 2 Go à votre machine virtuelle que vous êtes entrain de créer, vous aurez besoin d'un kernel Linux de 64 bits.

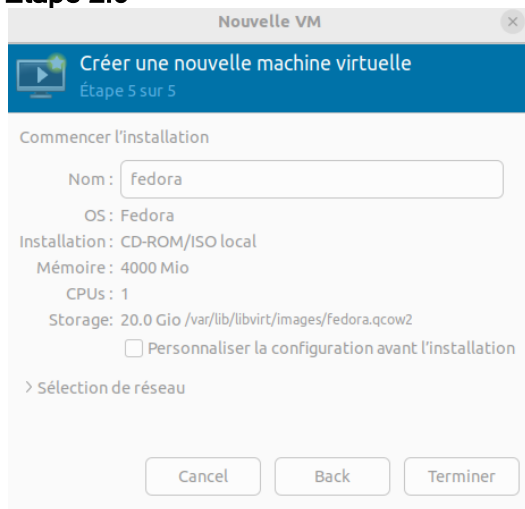
Attention: L'API de mesure des métriques de consommation énergétique, Scaphandre, que nous allons utiliser, ne fonctionne pas, pour le moment, avec des machines virtuelles avec plus d'un seul CPU virtuel! Cela pourra être possible dans les prochaines mis à jour de Scaphandre. Donc, en ce qui concerne le nombre de vCPUs, ou de CPU virtuels, nous allons choisir qu'un seul, comme le montre la figure ci-dessus.

Etape 2.8



Si vous le souhaitez, vous pouvez garder les options choisies par défaut. Ensuite, cliquez sur « Forward », vous allez obtenir la fenêtre ci-dessous.

Etape 2.9



KVM vous offre, par défaut, un réseau ponté de type NAT, cela veut dire que votre machine virtuelle aura accès au réseau via le système d'exploitation hôte.

Si vous souhaitez qu'un logiciel serveur, exécutée sur votre machine virtuelle, soit accessible à partir d'autres appareils sur le réseau, vous devrez modifier les paramètres réseau en cliquant sur « Sélection de réseau ». Mais, cela ne sera pas traité dans notre tutoriel.

Finissez par cliquer sur le bouton «Terminer ». Maintenant vous pouvez accéder à votre machine virtuelle et accéder à son installation en suivant simplement les instructions indiquées par la VM¹. Veuillez d'être patient, l'installation peut durer quelques minutes.

Dans la suite du tutoriel, je vais vous montrer les détails des installations et des configurations à effectuer, et les commandes à exécuter sur Linux ubuntu 22.04. La logique des étapes sera donc la même pour les autres distributions de Linux mais avec des commandes à exécuter différentes.

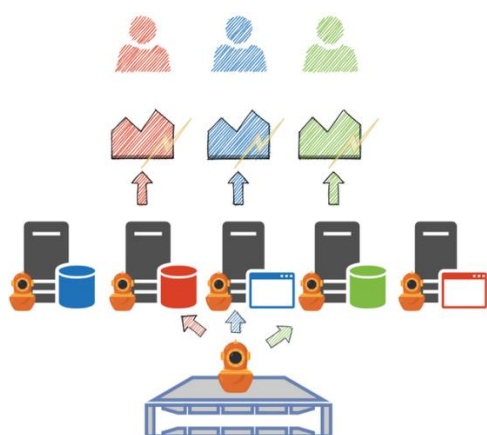
¹ Virtual machine

Partie 3 : Obtention des métriques sur la vm avec l'API Scaphandre

Une fois l'hyperviseur est installé, et la VM configurée comme indiqué dans la partie 2, nous allons maintenant voir comment installer et lancer l'outil de mesure de métriques de consommation énergétique Scaphandre sur la VM depuis l'hyperviseur. Comme indiqué plus haut, la mesure de la puissance avec Scaphandre ne fonctionne que sur les hyperviseurs Qemu/KVM pour la version de Scaphandre actuelle.

Il faut savoir que le problème majeur dans la mesure de la consommation d'énergie est de le faire à l'intérieur d'une VM parce que d'une manière générale la VM n'a pas accès aux mesures de puissance. L'avantage de Scaphandre est qu'il résout cette problématique en permettant une communication entre une instance Scaphandre sur l'hyperviseur et une autre instance s'exécutant sur la VM. Ainsi, l'agent Scaphandre sur l'hyperviseur calculera les métriques pour cette VM et celui sur la VM accédera par la suite à ces métriques.

Vous trouverez le schéma explicatif ci-dessous afin de mieux comprendre ce fonctionnement.



Cette 3^{ème} partie se compose de 2 sous-parties, étape 3.1 concerne la configuration à effectuer sur l'hyperviseur, et étape 3.2 à effectuer sur la VM.

Etape 3.1 : Configurations de la machine hôte

Etape 3.1.1

Nous allons commencer par vérifier la présence du logiciel de gestion de versions centralisé git, en vérifiant sa version, sur la machine hôte en tapant la commande ci-dessous.

```
slim@slim-Modern-15-A11MU:~$ git --version
git version 2.34.1
```

Si vous n'obtenez pas la même sortie que sur la figure ci-dessus (c'est-à-dire une sortie du type «git version xx.xx.xx »), installez git en exécutant simplement les commandes suivantes: **sudo apt update** ensuite **sudo apt-get install git-all**

Etape 3.1.2

De même, vérifiez la présence du gestionnaire de paquets de Rust. Pour cela, exécutez :

```
slim@slim-Modern-15-A11MU:~$ cargo --version
cargo 1.57.0
```

Comme dans 3.1.1, si la version de cargo n'apparaît pas, alors exécutez ces commandes afin de l'installer : **sudo apt update** ensuite **sudo apt install cargo**

Faites aussi la même chose pour le compilateur de Rust.

```
slim@slim-Modern-15-A11MU:~$ rustc --version
rustc 1.58.1
```

Etape 3.1.3

Une fois git, cargo, et rustc sont installés, clonez le repository « scaphandre » en exécutant cette commande : **git clone** <https://github.com/hubblo-org/scaphandre.git>

Etape 3.1.4

Entrez dans le répertoire « scaphandre », créé à l'étape 3.1.3, en exécutant : **cd scaphandre**. Avant de passer à l'étape 3.1.5, veuillez installer « **pkg-config** » et « **libssl-dev** », si cela n'a pas été déjà fait en exécutant les commandes suivantes:

```
sudo apt install pkg-config  
sudo apt install libssl-dev
```

Etape 3.1.5

Par la suite, exécutez : **cargo build --release**

Ainsi, le fichier binaire « scaphandre », que nous allons l'exécuter pour le côté hyperviseur, se trouve dans le répertoire **scaphandre/target/release/**

Etape 3.1.6

Selon la version de votre kernel, vous devrez peut-être modifier le module **intel_rapl** ou **intel_rapl_common** avant de lancer scaphandre, ainsi vous devez exécuter la commande suivante si votre kernel est > 5: **modprobe intel_rapl_common**

Sinon, exécutez cette commande: **modprobe intel_rapl**

Etape 3.1.7

En étant dans la répertoire « scaphandre » de l'étape 3.1.4, accédez maintenant au répertoire **target/release/**, et exécutez la commande suivante afin d'exécuter Scaphandre avec l'exportateur qemu: **sudo ./scaphandre qemu**

Etape 3.1.8

Maintenant, ouvrez un nouveau terminal. Pour chaque machine virtuelle à laquelle vous souhaitez donner accès à ses métriques, créez un point de montage tmpfs en exécutant la commande suivante :

```
sudo mount -t tmpfs tmpfs_DOMAIN_NAME /var/lib/libvirt/scaphandre/DOMAIN_NAME -o size=5m
```

 avec DOMAIN_NAME le nom de domaine libvirt de la machine virtuelle.

Par exemple, puisque ma VM est ubuntu22.04, la commande que j'ai donc exécutée est :

```
slim@slim-Modern-15-A11MU:~$ sudo mount -t tmpfs tmpfs_ubuntu22.04 /var/lib/libvirt/scaphandre/ubuntu22.04 -o size=5m
```

Passons à la configuration au niveau de la VM.

Dans l'étape ci-dessous 3.2, je vais, personnellement, utiliser ma machine virtuelle ubuntu 22.04, mais si vous souhaitez utiliser une machine virtuelle de votre choix, cela est bien évidemment possible (suivez tout de même les étapes de la partie 2).

Etape 3.2 : Configurations de la machine virtuelle

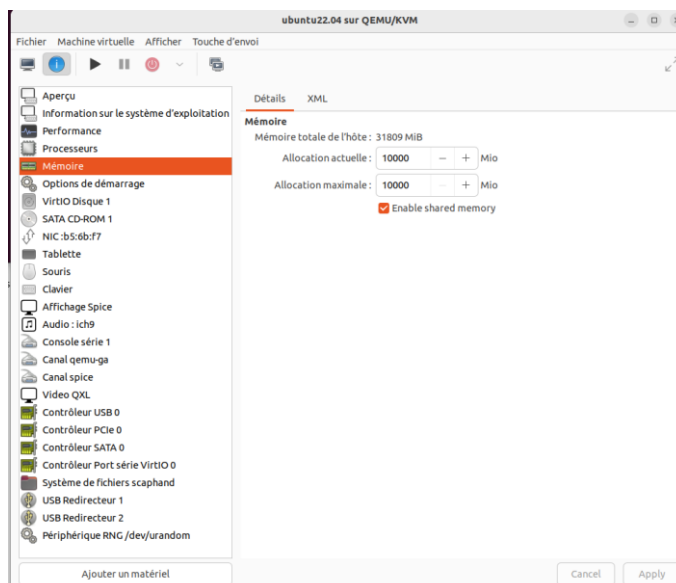
Etape 3.2.1

Lancez votre gestionnaire de machines virtuelles, puis sélectionnez votre VM et appuyez sur le bouton «Ouvrir» comme l'indique la flèche ci-dessous.



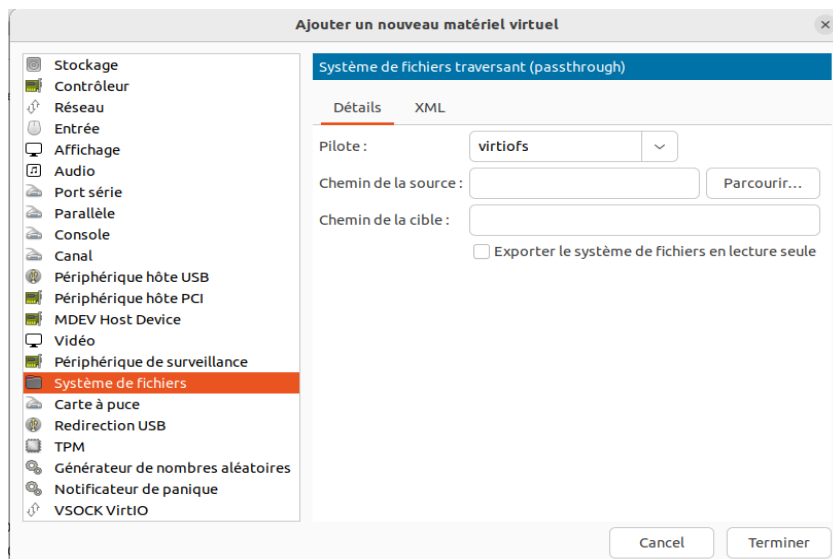
Etape 3.2.2

Une fois une nouvelle fenêtre affichée, allez dans « Affichez les détails du matériel virtuel » en cliquant sur le bouton « i » en bleu. D'abord, dans la partie « Mémoire », vérifiez si la case « Enable shared memory » est bien sélectionnée, si ce n'est pas le cas, veuillez la sélectionner.



Etape 3.2.3

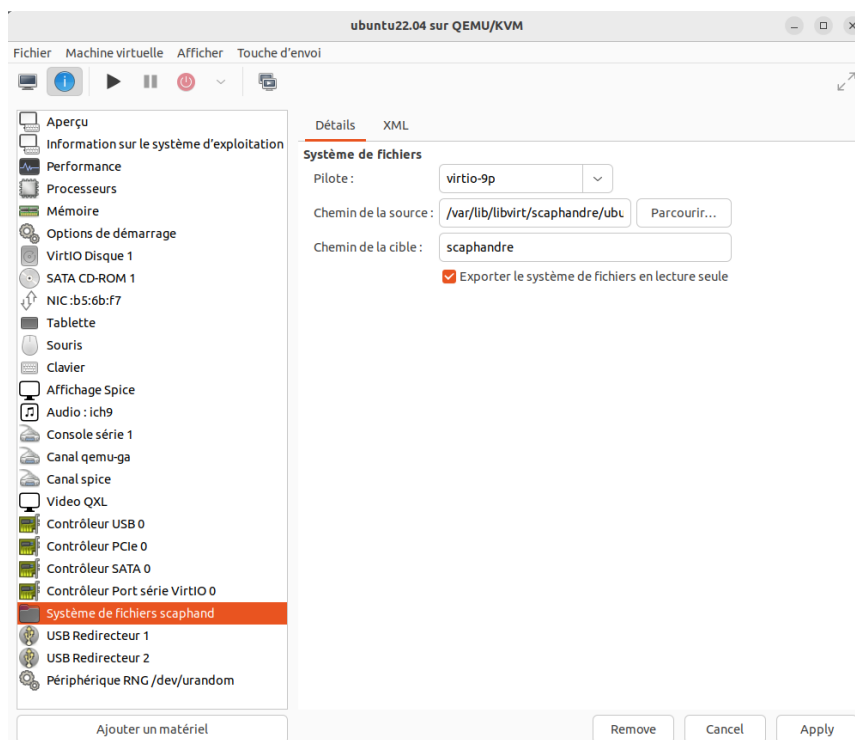
Faites maintenant un clic droit sur « USB Redirecteur 1 », et choisissez « Ajouter du matériel ». Une fenêtre, comme le montre la figure ci-dessous, devrait s'afficher.



Etape 3.2.4

Dans cette étape, nous allons remplir les différents champs dans « Détails ». En ce qui concerne le « Pilote », choisissez « virtio-9p ». Pour le chemin de source, tapez « /var/lib/libvirt/scaphandre/DOMAIN_NAME » en remplaçant bien évidemment DOMAIN_NAME par le nom que vous avez saisi à l'étape 3.1.8 (pour mon cas, le chemin de source est /var/lib/libvirt/scaphandre/ubuntu22.04). Enfin, mettez « scaphandre » pour le chemin cible, et cochez la case « Exporter le système de fichiers en lecture seule ».

Vous trouverez, dans la figure ci-dessous, le résumé du remplissage des champs pour mon cas.



Enfin, validez en cliquant sur « Apply ».

Etape 3.2.5

Une fois toutes les précédentes étapes effectuées, lancez maintenant votre VM (ou la redémarrer si elle est déjà en fonctionnement).

Etape 3.2.6

Refaites exactement les étapes suivantes 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, et 3.1.6.

Avant d'entamer l'étape 3.2.7, veuillez créer un répertoire, en étant dans le répertoire « var », en le nommant « scaphandre » : **sudo mkdir scaphandre**

Etape 3.2.7

Vous pouvez maintenant monter le système de fichiers sur votre VM avec cette commande : **sudo mount -t 9p -o trans=virtio scaphandre /var/scaphandre**

Etape 3.2.8

Enfin, nous pouvons lancer Scaphandre sur notre VM en exportant les métriques avec l'exportateur que nous avons choisi « prometheus ».

Exécutez donc simplement la commande suivante : **sudo scaphandre --vm prometheus**

Etape 3.2.9

Vous pouvez, à ce stade, collecter les métriques de consommation énergétique spécifiques à votre machine virtuelle en tapant cette adresse dans votre navigateur <http://adressesIPvm:8080/metrics> en changeant adressesIPvm par votre adresse ip de votre VM.

Attention : Après avoir rafraîchi la page <http://adressesIPvm:8080/metrics>, si vous avez redémarré votre machine (hyperviseur) et que vous avez effectué les mêmes étapes ci-dessus, et que vous avez obtenu une valeur nulle, par exemple, pour la métrique suivante scaph_process_power_consumption_microwatts, alors veuillez recommencer les étapes 3.2.1, 3.2.2, 3.2.3, 3.2.4 et 3.2.5.

Maintenant, nous cherchons à trouver une manière efficace afin de visualiser ces métriques sous forme de graphiques. Pour cela, nous allons utiliser les outils suivants :

- « Scaphandre » entrain de fonctionner avec prometheus exporter pour mesurer les métriques de la consommation énergétique en temps réel de la VM (ce qui a été déjà fait avec la partie 3),
- Prometheus afin de collecter ces métriques, et
- Grafana pour les visualiser graphiquement. Il est important de noter ici que cet outil ne stocke pas de données mais s'appuie plutôt sur la connexion « Datasource » vers le serveur Prometheus.

Partie 4 : Visualisation graphique des métriques sur la VM avec l'API de mesures des métriques Scaphandre

***Remarque :** Avant d'entamer les étapes à suivre ci-dessous pour cette 4^{ème} partie, il faut savoir qu'il y aura 2 terminaux fonctionnant en même temps au total concernant la partie serveur : un terminal pour lancer Scaphandre, un pour lancer Prometheus. Pour la partie VM, il n'y aura que Scaphandre à lancer.*

Etape 4.1 : Installations et configurations au niveau du serveur

D'abord, soyez sûr que vous êtes bien sur le bare-metal ensuite créez un répertoire qui correspondra à Prometheus.

```
slim@slim-Modern-15-A11MU:~/Documents$ ls
nodeexporter prometheus scaphandre
```

Etape 4.1.1

Après avoir accédé au répertoire « prometheus » que vous avez créé, commencez par installer Prometheus. Pour cela, allez sur ce site <https://prometheus.io/download/>, et choisissez le bon lien correspondant à votre système d'exploitation.

prometheus

The Prometheus monitoring system and time series database. [prometheus/prometheus](https://prometheus.io/)

2.35.0 / 2022-04-21 Release notes				
File name	OS	Arch	Size	SHA256 Checksum
prometheus-2.35.0.darwin-amd64.tar.gz	darwin	amd64	77.01 MiB	a4f4d0ea38addc179bd37a0282738a8b03a6e58ae3706f09be3178e883c44abc
prometheus-2.35.0.linux-amd64.tar.gz	linux	amd64	76.89 MiB	e4546960688d1c85530ec3a93e109d15b540f3251e1f4736d0d9735e1e857faf
prometheus-2.35.0.windows-amd64.zip	windows	amd64	78.41 MiB	782323b31ef8f99159ec7a9042b3d553c4e49c39af8e3afe3f85e6711dad0018

Pour mon cas par exemple, je vais donc copier le second lien.

Ensuite lancez un nouveau terminal, et exécutez la commande suivante (wget + le lien que vous avez choisi) :

```
wget https://github.com/prometheus/prometheus/releases/download/v2.35.0/prometheus-2.35.0.linux-amd64.tar.gz
```

Etape 4.1.2

Une fois Prometheus est installé, extrayez le fichier binaire de téléchargement en exécutant cette commande : **tar xvfz prometheus-2.35.0.linux-amd64.tar.gz** (remplacez bien évidemment prometheus-2.35.0.linux-amd64.tar.gz par votre version que vous venez de télécharger).

Avant de configurer directement Prometheus afin qu'il puisse récupérer les données depuis la VM, nous allons rajouter une étape qui sert à vérifier que Prometheus fonctionne correctement (étape 4.1.3).

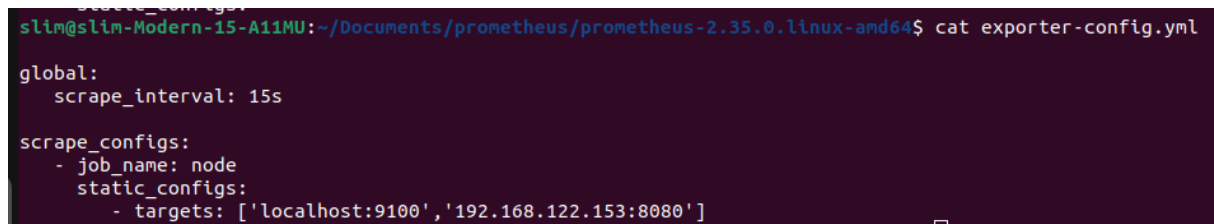
Etape 4.1.3

Accédez au répertoire que vous venez d'obtenir et lancez Prometheus avec cette commande : **./prometheus**

Vous pouvez vérifier le bon fonctionnement de Prometheus en accédant à son interface graphique en visitant <http://localhost:9090/graph>

Etape 4.1.4

Arrêter Prometheus maintenant. Vous devez créer un fichier de configuration `.yaml`, (que j'ai personnellement appelé « exporter-config »). L'adresse ip de ma VM est 192.168.122.153. Donc dans « targets », nous allons rajouter cette adresse ip suivi du numéro de ports des métriques issues de l'API Scaphandre qui est 8080. Vous trouverez dans la figure ci-dessous le contenu à mettre dans votre fichier de configuration en mettant bien évidemment l'adresse ip de votre machine virtuelle.



```
slim@slim-Modern-15-A11MU: ~/Documents/prometheus/prometheus-2.35.0.linux-amd64$ cat exporter-config.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: node
    static_configs:
      - targets: ['localhost:9100', '192.168.122.153:8080']
```

Après ces changements, vous relancez Prometheus mais cette fois-ci avec l'indicateur `--config.file : ./prometheus --config.file=exporter-config.yml`

Etape 4.2 : Installation de Grafana

Etape 4.2.1

Dans cette étape, nous allons mettre à jour les informations du package. Pour cela, exécutez les commandes ci-dessous :

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

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

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

Etape 4.2.2

Ensuite, Ajouter un repository de Grafana à l'aide de la commande suivante :

```
echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

Etape 4.2.3

Enfin, mettez le repository et installer Grafana :

```
sudo apt-get update
```

```
sudo apt-get install grafana-enterprise
```

Etape 4.3 : Lancement de Grafana et visualisation des métriques

Etape 4.3.1

Afin de lancer Grafana, il suffit d'exécuter les commandes ci-dessous :

```
sudo systemctl daemon-reload
```

```
sudo systemctl start grafana-server
```

Pour vérifier son état actuel (c'est-à-dire activé ou non), lancez cette commande :

```
sudo systemctl status grafana-server
```

À partir de l'étape suivante, nous présentons seulement les étapes nécessaires afin d'obtenir un graphique des métriques fournies par l'API Scaphandre. Pour plus d'informations sur Grafana, veuillez vous référer à la documentation (<https://grafana.com/docs/grafana/latest/>).

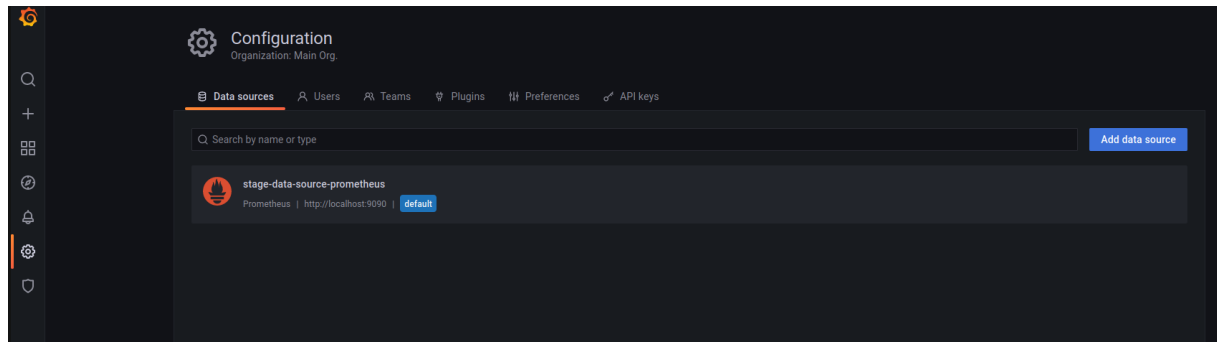
Etape 4.3.2

Pour visualiser les graphiques des métriques mesurées, accédez au tableau de bord Grafana en allant sur <http://localhost:3000/login>

Le nom d'utilisateur de connexion par défaut est admin et le mot de passe par défaut est admin. Veuillez ensuite modifier le mot de passe administrateur par défaut après la connexion.

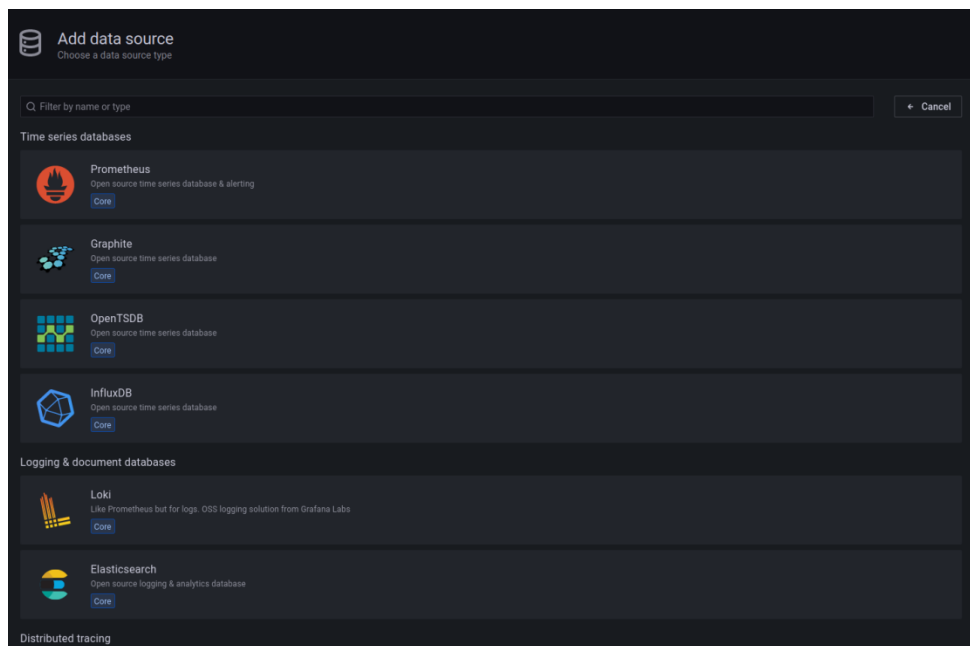
Etape 4.3.3

La première chose à faire est de définir une source de données en cliquant sur le symbole « engrenage » de la barre menu et choisir « Data sources ».



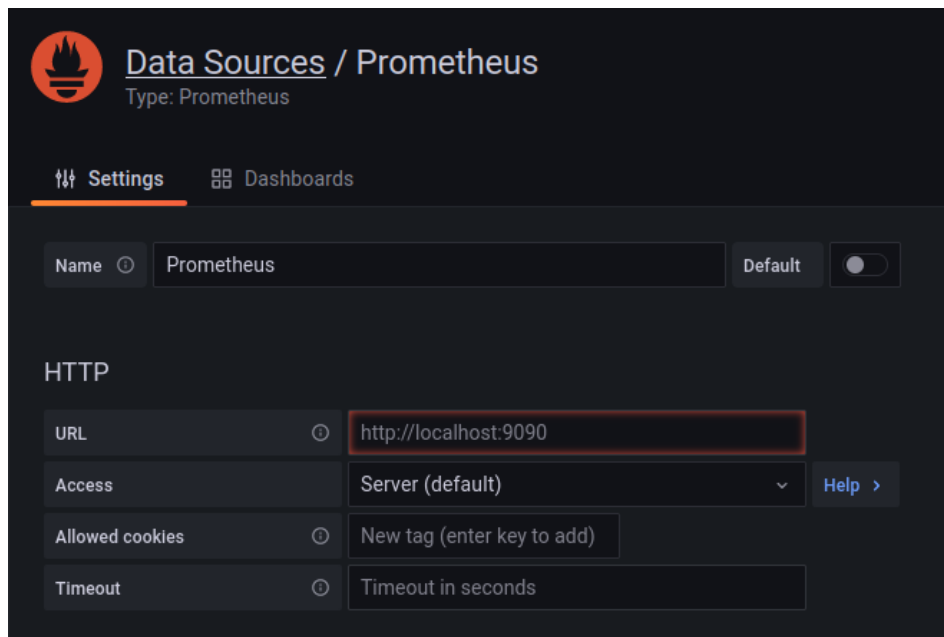
Pour mon cas par exemple, j'ai déjà défini une source de données qui s'appelle « stage-data-sources-prometheus ». Nous allons tout de même rajouter une nouvelle source de données.

Donc pour rajouter une nouvelle source de données, cliquez directement sur le bouton bleu « Add data source », ce qui permettra d'obtenir un choix de base de données temps séries.



Comme dans notre cas, il s'agit du prometheus, nous allons alors cliquer sur le premier choix, « Prometheus ».

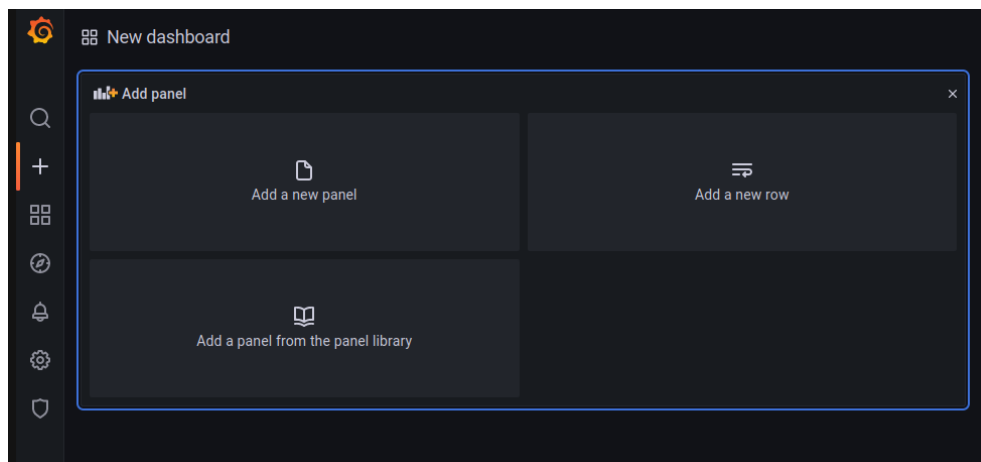
Nous allons nous concentrer sur seulement 2 choses principales ; le nom de la source de données et l'URL à saisir.



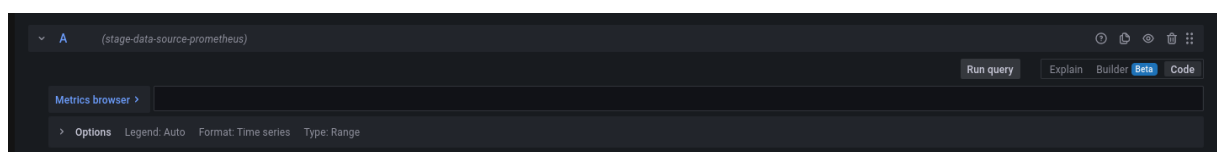
Pour l'URL, entrez le nom d'hôte ou l'adresse IP du serveur prometheus. Pour mon cas, je vais saisir plutôt l'adresse IP du serveur prometheus. Enfin, avant de descendre en bas de page, et cliquer sur « Save & Test », ayez sûr d'avoir lancé le serveur Prometheus comme indiqué à l'étape 4.1.4.

Etape 4.3.4

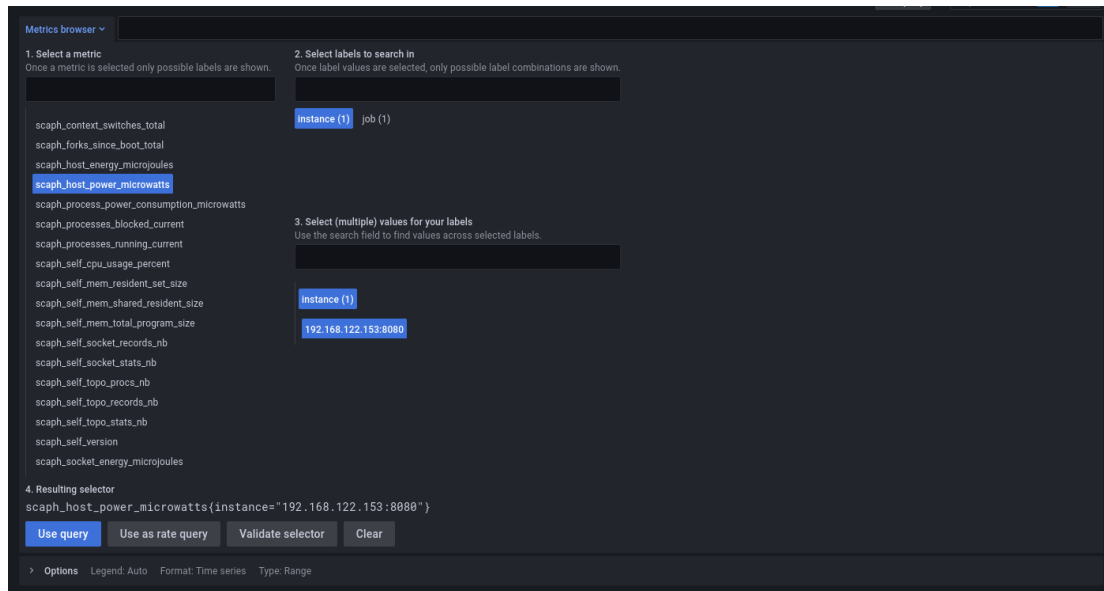
Une fois la source des données est définie, vous pouvez maintenant créer votre dashboard en choisissant les données que vous voulez mettre en évidence. Pour cela, il suffit d'aller dans le symbole « plus » de la barre menu puis en cliquant sur « Add a new panel » comme le montre la figure ci-dessous.



Maintenant, vous pouvez cliquer sur « Metrics browser » afin de choisir la métrique que vous voulez afficher dans votre dashboard.



Afin de choisir une métrique parmi les métriques proposées par l'API que nous avons utilisé, Scaphandre, veuillez sélectionner la bonne instance c'est-à-dire une instance de type ADRESSE_IP_VM :8080. Concernant la métrique, comme vous voyez dans la figure ci-dessous, j'ai sélectionné par exemple « scaph_host_power_microwatts » qui correspond à la consommation énergétique de l'hôte en Watt.

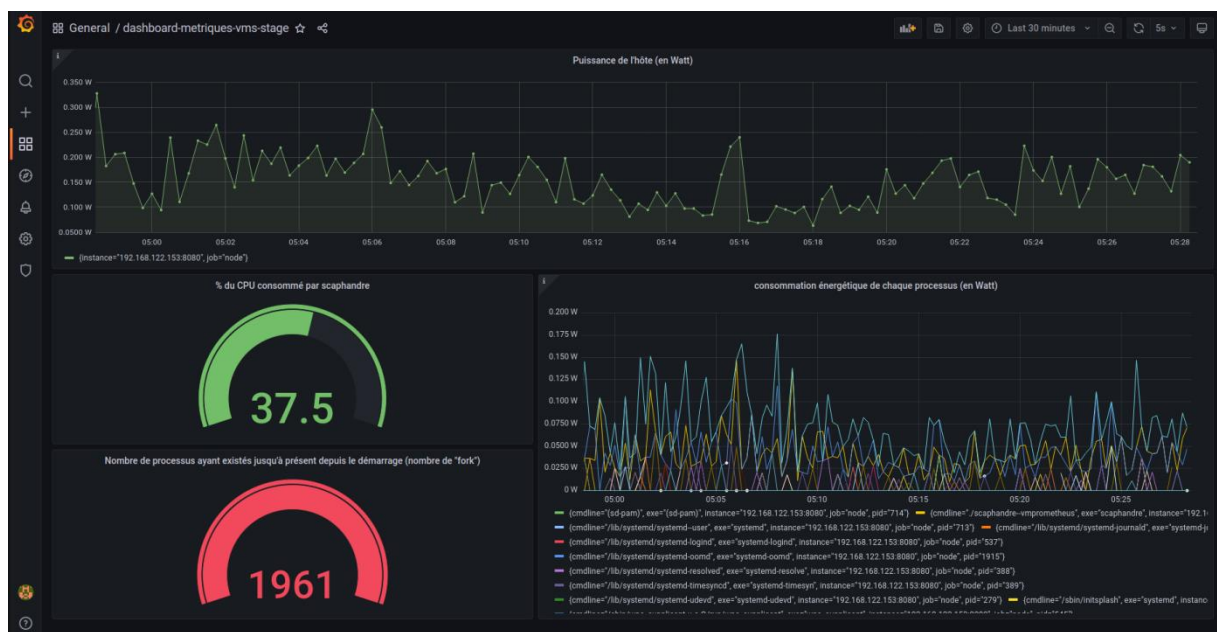


Confirmez en cliquant simplement sur « Use query ».



Vous obtenez ainsi une requête PromQL, un langage de requête intégré conçu pour Prometheus. Enfin pour visualiser le graphique correspondant à la métrique choisie, cliquez sur « Run query ».

Voici, à titre d'exemple, mon dashboard que j'ai obtenu :



Résumé

Ce rapport est réalisé dans le cadre d'un stage de 3 mois au sein du laboratoire LIUPPA du 9 Mai au 9 Aout 2022.

Sur le plan technique, le thème général du stage est le greenit. Le but est de mettre en place des tableaux de bord permettant de visualiser la consommation énergétique des serveurs et machine virtuelle en temps réel. Le stage se décompose principalement en 2 parties. La première consiste à une récupération des métriques de la consommation énergétique d'une machine virtuelle en utilisant l'API Scaphandre pour mesurer uniquement, ensuite en mettant en place Prometheus et Grafana afin d'afficher sous forme de graphiques les métriques collectées. La seconde partie est la récupération des métriques des serveurs (bare-metal) en utilisant la technologie InfluxDB. Dans ce 2^{ème} cas, les métriques sont collectées à l'aide des plugins et Telegraf, et la visualisation est faite avec l'outil de visualisation d'InfluxDB, Chronograf.

Sur le plan personnel, ce stage est une très bonne expérience. En effet, cette expérience m'a non seulement donnée une initiation sur le domaine de la recherche et appris beaucoup d'outils technologiques pour élaborer une étude dans le domaine de la greenit, mais aussi elle m'a permis de travailler le coté relationnel à travers notamment l'événement international qui a eu lieu durant toute la semaine du 20 Juin 2022.

Concernant les améliorations à faire, il en existe beaucoup de pistes. Pour le moment d'abord, je propose d'établir des tests de mesures de la consommation énergétique des applications avec l'API powerJoular (ou PowerAPI), et de proposer par la suite une visualisation des métriques collectées dans le dashboard construit avec Grafana. Pour le monitoring des applications, je vais utiliser la librairie prometheus-client de Python. Enfin, je vais regarder la gestion des alertes avec Grafana et Chronograf.

Abstract

This report is produced as part of a 3-month internship in the LIUPPA laboratory from May 9 to August 9, 2022.

On a technical level, the general theme of the course is greenit. The goal is to set up dashboards to visualize the energy consumption of servers and virtual machines in real time. The internship is mainly divided into 2 parts. The first consists of retrieving the energy consumption metrics of a virtual machine using the Scaphandre API to measure only, then setting up Prometheus and Grafana in order to display the metrics used in graphs. The second part is the recovery of server metrics (bare-metal) using InfluxDB technology. In this 2nd case, the metrics are imposed using the plugins and Telegraf, and the visualization is done with the InfluxDB visualization tool, Chronograf.

On a personal level, this internship is a very good experience. Indeed, this experience not only gave me an introduction to the field of research and learned many technological tools to develop a study in the field of greenit, but also it allowed me to work on the relational side of notably through the international event that took place throughout the week of June 20, 2022.

Regarding the improvements to be made, there are many tracks. For the moment first, I propose to establish tests for measuring the energy consumption of applications with the powerJoular API (or PowerAPI), and then to propose a visualization of the metrics provided in the integrated dashboard with grafana. For application monitoring, I will use Python's prometheus-client library. Finally, I will look at alert management with Grafana and Chronograf.