Hes·so VALAIS WALLIS
Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Travail de Bachelor

Informatique de gestion

Un environnement pour l'enseignement de la cybersécurité

Auteur :

Steven Roh

Professeur :

Jean-Luc Beuchat

Déposé à la HES-SO Valais-Wallis à Sierre, le 13 août 2021

Résumé

L'apprentissage de notions en sécurité informatique fait partie intégrante du cursus d'informaticien de gestion et de la formation Certificate of Advanced Studies (CAS) en cybersécurité proposés par la HES-SO Valais-Wallis à Sierre. Pour cela, les professeurs responsables du cours mettent à disposition des étudiants des travaux pratiques en complément des leçons de cours théoriques.

Actuellement, pour les étudiants, l'installation des outils et la mise en place des environnements nécessaires à l'exécution de ces travaux pratiques est complexe et prend beaucoup de temps.

L'objectif de ce travail est de développer des outils d'enseignement pour ces deux formations. Ces derniers doivent permettre un déploiement automatisé des environnements afin que les étudiants ne perdent pas de temps à installer et à configurer divers logiciels.

La première partie de ce travail sera dédiée à l'analyse des sujets traités dans le cours de sécurité ainsi que sur les solutions et outils utilisés dans l'industrie pour ce type de travaux.

Ensuite, après avoir élaboré une solution technique, plusieurs démonstrateurs et exercices seront conçus sur les thèmes abordés en cours afin de tester la création d'environnements.

Mots clés : Conteneurisation, Docker, Virtualisation, Vagrant, cybersécurité, outils pour l'enseignement, déploiement automatisé, DevOps, Packer, Ansible, Terraform

Remerciements

Je remercie tout particulièrement mon professeur Jean-Luc Beuchat pour la transmission avec passion de ses compétences techniques pointues dans le domaine de la sécurité informatique ainsi que de m'avoir suivi, guidé et conseillé durant ce travail.

J'adresse également un grand merci à mon ami Mathias et à ma femme Jessica qui ont eu la gentillesse de relire ce travail afin de réduire le nombre d'erreurs et incohérences.

De plus, je remercie également mes collègues, mes amis et ma famille de m'avoir soutenu durant ma formation.

Table des matières

Table des figures

Liste des tableaux

Introduction

Ce travail de bachelor est réalisé à la HES-SO Valais-Wallis à Sierre pour les étudiants de la formation Bachelor d'informaticien de gestion ainsi que pour ceux de la formation CAS en cybersécurité proposées par la HES-SO Valais-Wallis. Il est proposé et est encadré par le professeur Jean-Luc Beuchat.

L'apprentissage de notions en sécurité informatique fait partie intégrante du cursus d'informaticien de gestion et du CAS en cybersécurité dispensé à la HES-SO Valais-Wallis. Pour cela, les professeurs responsables du cours mettent à disposition des étudiants des travaux pratiques en complément des leçons de cours théoriques.

L'installation des outils et la mise en place d'environnements nécessaires à l'exécution et la reproduction de ces travaux pratiques est complexe et prend du temps. De plus, comme il s'agit d'expérimenter, d'apprendre et de progresser dans un environnement « laboratoire », il est important de ne pas affecter la stabilité de la machine des étudiants.

L'objectif de ce travail est de documenter, tester et mettre en place des environnements complets contenant les logiciels et outils pour le cours de sécurité. Cette solution doit permettre un déploiement totalement automatique d'un environnement créé spécialement pour les étudiants afin qu'ils ne perdent pas de temps à installer et configurer diverses machines ou logiciels.

La première partie de ce travail sera dédiée à l'analyse des approches adoptées dans le cours de sécurité ainsi que des solutions et outils utilisés dans l'industrie.

Par la suite, plusieurs démonstrateurs et exercices seront conçus en accord avec les thèmes abordés durant le cours.

Contexte

Les cours de sécurité des systèmes d'information sont donnés lors du 4e et 5e semestre durant le module Sécurité des Systèmes d'Information (634-2) du bachelor par les professeurs Jean-Luc Beuchat et Xavier Barmaz. Avec le nouveau plan d'étude, ces derniers seront enseignés lors du 2e et 3e semestre dans le cours « 63-22 ». Le CAS en cybersécurité est quant à lui donné sur une période de six mois.

Durant ces cours, sont étudiés de nombreux sujets dont : les annuaires (Active Directory et LDAP), les protocoles d'authentification sécurisés, la gestion de ressources et des utilisateurs/groupes, les outils d'administrations Windows, la protection et sécurisation d'une infrastructure IT, la sécurité locale, la sécurité réseau et Internet, les protocoles, les certificats, l'authentification, le VPN, le pare-feu (firewall), le serveur proxy, les tests d'intrusion réseau, les tests d'intrusion applicatifs, le code sécurisé (secure coding), l'analyse forensique (computer forensics) et la blockchain.

Les étudiants du bachelor disposent de quatre heures de théorie et de démonstrations par semaine ainsi que de deux heures pendant lesquelles ils réalisent les exercices pratiques par eux-mêmes.

Certains documents théoriques sont distribués sur la plateforme Cyberlearn (Moodle). Les étudiants disposent de machines virtuelles existantes distribuées via l'outil HES-SO Tools et sont également amenés à en créer. Certains exercices pratiques sont déjà distribués par le biais d'un dépôt git sur la plateforme gitlab.com [1].

Problématique

Les étudiants sont amenés à installer et configurer de nombreuses machines virtuelles, logiciels et outils afin de réaliser les travaux pratiques.

La reproduction d'environnement similaire à celui du professeur fait perdre un temps considérable aux étudiants. Comme le temps d'apprentissage dédié aux exercices pratique est limité, il est nécessaire de rendre la mise en place plus rapide.

De plus, l'HES-SO Valais-Wallis ayant adopté une politique Bring Your Own Device (BYOD), il est nécessaire que l'exécution fonctionne aussi bien sur les systèmes d'exploitation Windows, MacOS que Linux et que les outils chargés de la création d'environnements soient multi-plateformes.

Finalement, pour les étudiants, le stockage de machines virtuelles et *snapshots* occupe un espace disque important sur les ordinateurs. Ce dernier étant également limité (plus encore sur les disques SSD), il est important d'optimiser le stockage de ces machines et de permettre de libérer l'espace disque qu'elles occupent une fois le travail terminé.

1. https://gitlab.com/jlbhevs/634-2-information-security

Du côté des professeurs, il est très difficile de tester efficacement et de maintenir les exercices et configurations proposés. De nombreux tests manuels sont nécessaires pour s'assurer que chaque exercice soit fonctionnel, et ce à chaque nouvelle édition du cours.

Fonctionnement souhaité

L'objectif de fonctionnement est de créer et de mettre à disposition des étudiants les outils nécessaires à la création d'environnements reproductibles afin d'exécuter leurs travaux pratiques.

Le temps nécessaire à la mise en place et l'espace disque occupé sont des critères déterminants pour le succès de ce travail de bachelor. Comme l'environnement pourra être recréé facilement et rapidement, l'étudiant pourra, une fois le travail pratique terminé, supprimer l'environnement afin de libérer l'espace de stockage utilisé.

L'utilisation de Git fait partie intégrante de la formation. Les étudiants seront confrontés à son utilisation ainsi qu'à la plateforme GitLab durant tous le cours à des fins d'apprentissage.

Les outils à utiliser doivent correspondre à ceux employés en entreprise afin de leur permettre d'en apprendre leur fonctionnement et de pouvoir mettre ces nouvelles compétences à profit dans leur future carrière.

Dans un premier temps, les exercices s'exécuteront sur les machines locales des étudiants car à l'heure actuelle nous ne pouvons pas affirmer que les crédits Cloud offerts (Microsoft Azure) soient suffisants. Cependant, la solution proposée doit déjà tenir compte de la potentielle transition future dans le Cloud afin d'assurer sa compatibilité.

1 État de l'art

1.1 Apprentissage de la sécurité informatique

Le domaine de la sécurité informatique étant en plein développement, les moyens d'apprentissages évoluent également. Depuis quelques années, nous pouvons constater l'apparition de techniques d'apprentissage complémentaires afin d'initier le plus grand nombre de personnes à la sécurité informatique. Ces formations sont axées sur la pratique et permettent de s'exercer sur des exercices pratiques, proposés sous forme de challenge.

1.1.1 Plateformes ou compétitions CTF

Les Capture The Flag (CTF) sont des événements d'une durée limitée, de quelques heures à quelques jours, en ligne ou en présentiel, dans lesquels les participants doivent exploiter des environnements délibérément vulnérables afin de s'y introduire. Une fois la vulnérabilité identifiée et exploitée, un *flag* (drapeau prouvant l'intrusion) permet à l'utilisateur de valider l'épreuve et de gagner des points, le faisant évoluer dans un classement.

Plusieurs sites web proposent ce type de compétitions en ligne en publiant des épreuves sans limite de temps et disponibles continuellement dans des environnements isolés. C'est le cas de Root-Me [1], Hack The Box [2], NewbieContest [3], Hacker101 [4] qui font partie des plateformes les plus populaires.

Root-Me met à disposition publiquement plus de 400 challenges accessibles en tout temps ainsi que plus de 140 machines permettant de démarrer un environnement virtuel sur demande.

1. https://www.root-me.org/
2. https://www.hackthebox.eu/
3. https://www.newbiecontest.org/
4. https://www.hacker101.com/

FIGURE *1.1: Capture d'écran du site Root-Me*
Source: *de l'auteur à partir de root-me.org*

En complément, Root-Me propose une offre professionnelle à destination des entreprises ou écoles souhaitant former son personnel ou ses étudiants sur la plateforme Root-Me Pro [5]. Cette offre permet d'être autonome dans la gestion des utilisateurs et équipes, de donner accès aux épreuves par thème et de disposer de statistiques permettant de suivre la progression pédagogique. Selon le site de Root-Me Pro, les écoles EPITECH, INSA, SUPINFO, HES-SO Fribourg et l'École 42 leur font confiance et utilisent déjà ce service.

5. https://pro.root-me.org/

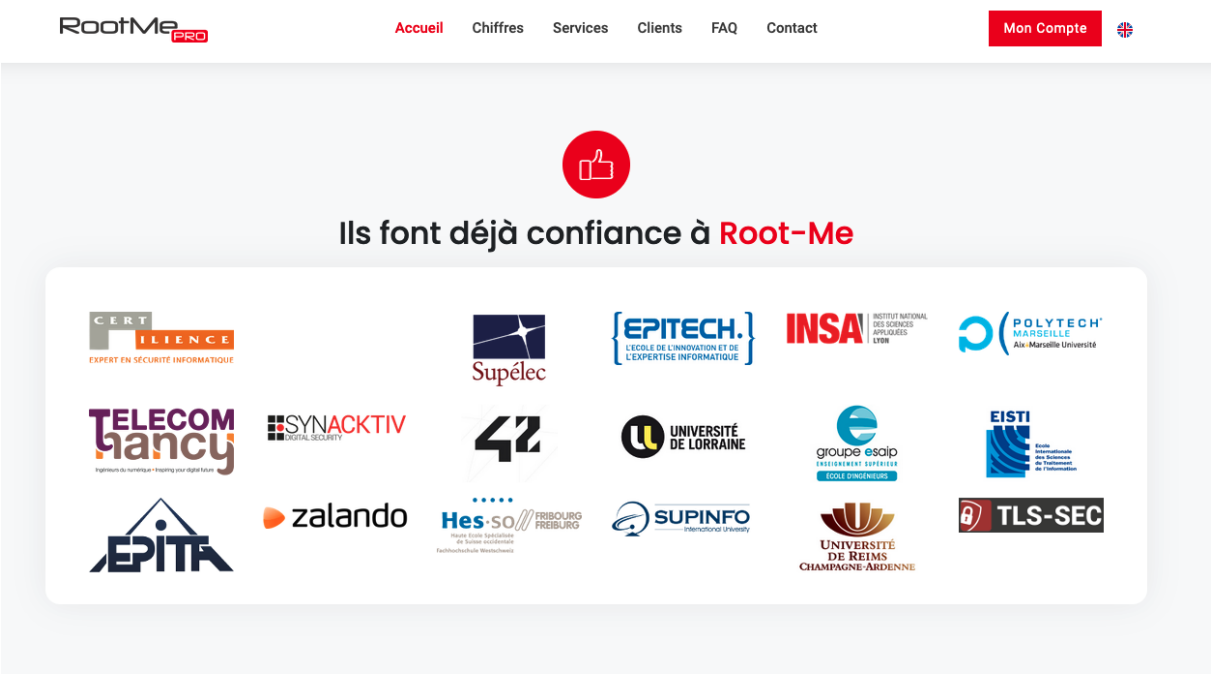


FIGURE *1.2: Capture d'écran du site Root-Me Pro affichant les utilisateurs du service*
***Source:** de l'auteur à partir de pro.root-me.org*

Offensive Security, entreprise proposant des services et formations en sécurité informatique, offre également une plateforme similaire nommée « Proving Grounds Play » [6] qui contient des challenges permettant d'apprendre et de tenter d'exploiter des laboratoires virtuels. Ces derniers s'exécutent dans des machines virtuelles du système d'exploitation « Kali Linux » directement accessibles au travers du navigateur.

Durant ce type de formation axé sur la pratique, l'étudiant évolue seul, exercice après exercice, afin d'explorer un sujet défini.

1.1.2 Cours et certifications en cybersécurité

En plus de ces solutions d'apprentissage autonomes, plusieurs organisations proposent des formations certifiantes. C'est le cas de Offensive Security [7] qui propose des cours sur les tests de pénétration, la sécurité des applications web ainsi que le développement d'*exploits*. Ces formations toujours axées sur la pratique permettent d'apprendre l'état d'esprit du « hacker » et délivrent un certificat (« Infosec Training and Penetration Testing | Offensive Security », s. d.).

L'institut SANS propose également des cours et certifications. Dans ces derniers, l'étudiant est confronté à des exercices dans un environnement virtuel de laboratoire installé en local sur sa machine (« Cybersecurity Courses & Certifications, SANS Institute », s. d.).

6. https://www.offensive-security.com/labs/individual/
7. https://www.offensive-security.com/courses-and-certifications/

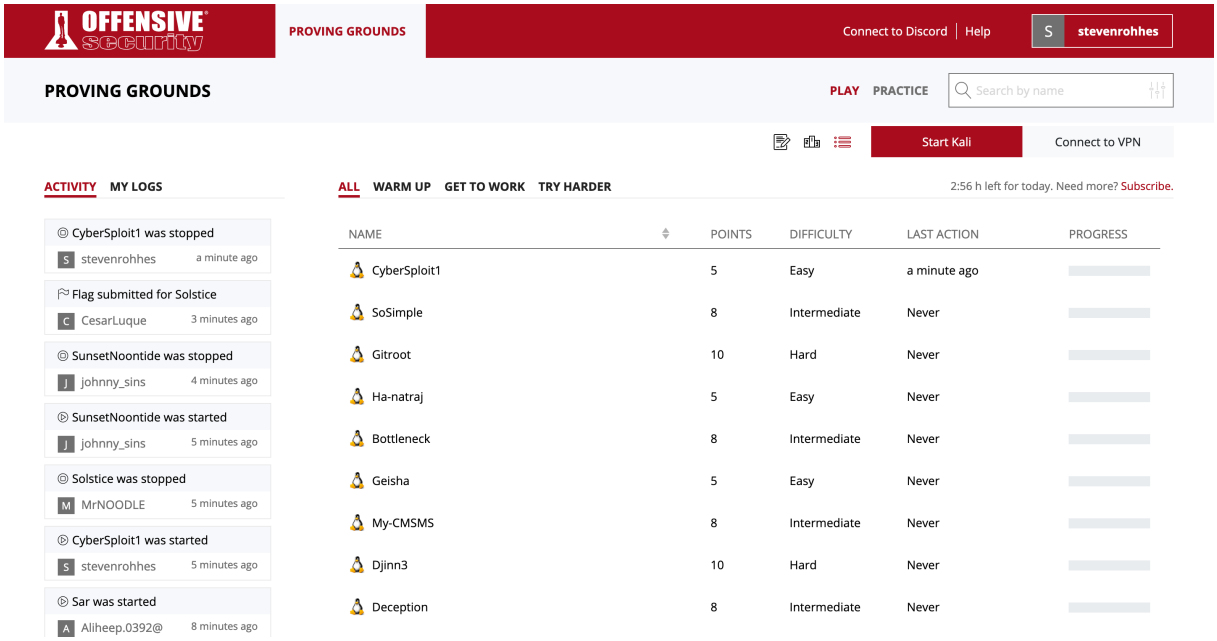


FIGURE *1.3: Liste de challenges « Proving Grounds » sur Offensive Security*
Source: *de l'auteur à partir de portal.offensive-security.com*

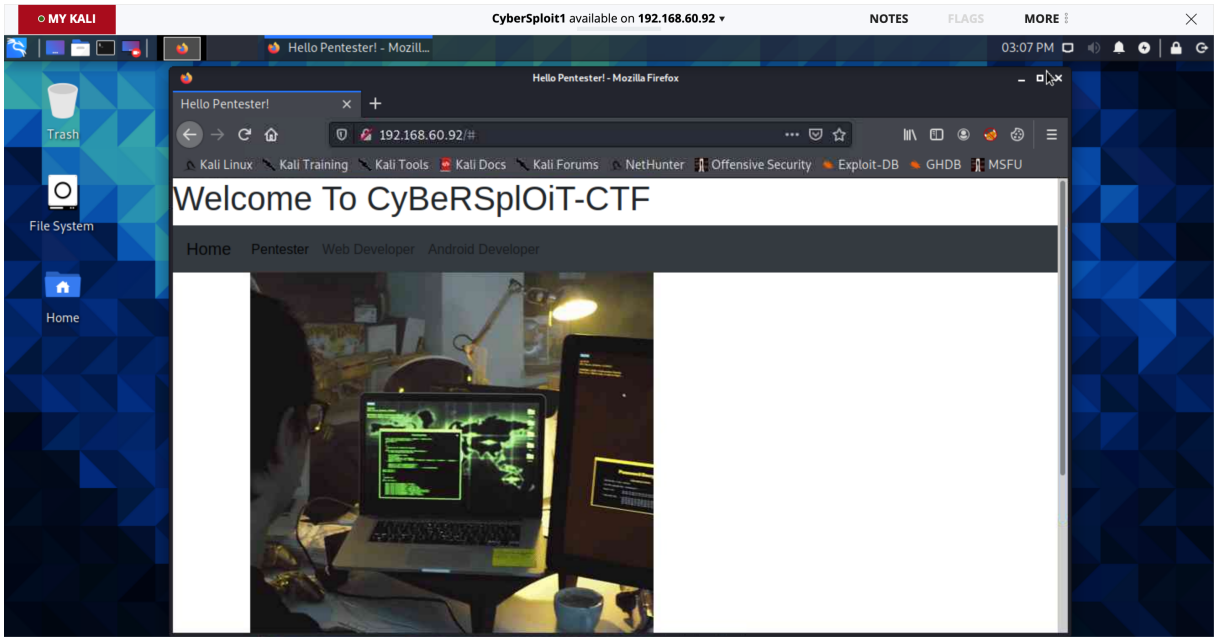


FIGURE *1.4: Challenge « CyberSploit1 » sur Offensive Security*
Source: *de l'auteur à partir de portal.offensive-security.com*

1.2 Solutions pour l'apprentissage de la cybersécurité

Plusieurs entreprises, universités, écoles ou centres de formations ont déjà déployé leur propres infrastructures et outils pour la mise à disposition d'environnements d'apprentissage virtuels similaires dans un but éducatif.

1.2.1 Cyber Sandbox Creator

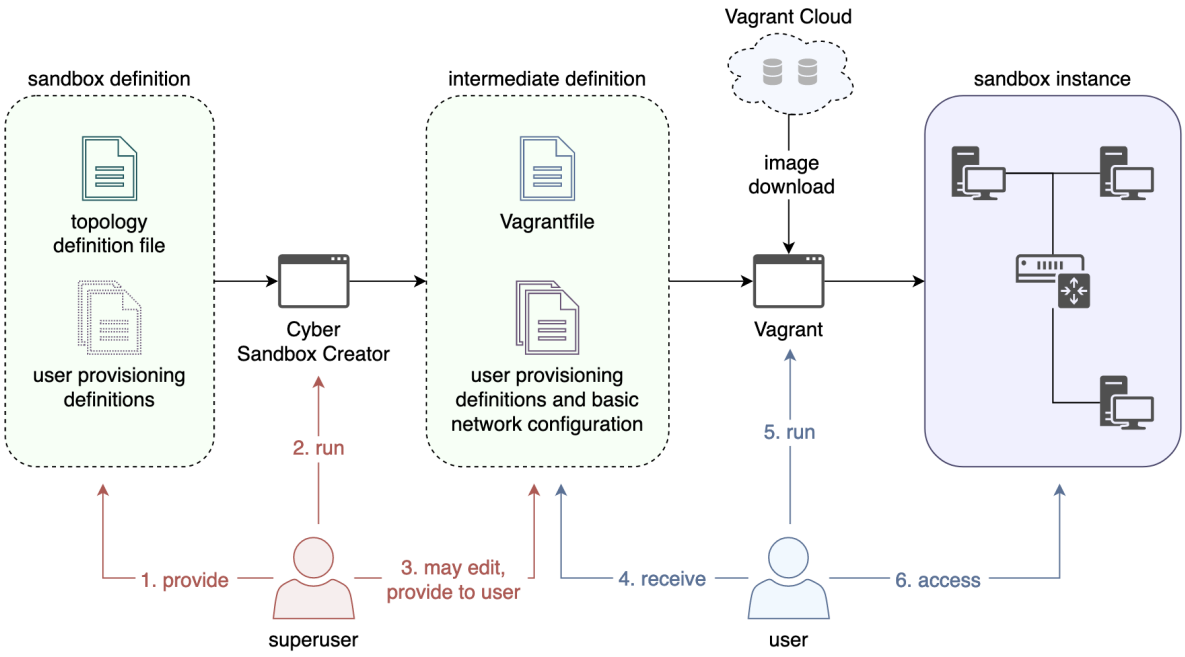


FIGURE *1.5: Démonstration de l'exécution d'un laboratoire dans labtainers*
Source: *https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator*

Dans le cadre du projet de recherche européen « CyberSec4Europe », un outil a été développé sous le nom de « Cyber Sandbox Creator[8] » (« CyberSec4Europe delivers Cyber Sandbox Creator », s. d.) afin de créer des laboratoires virtuels dédiés à la cybersécurité (MATYÁŠ, 2020).

Cet outil permet de créer automatiquement un réseau de machines virtuelles à partir d'un fichier de configuration. Ce dernier crée ensuite les différentes machines virtuelles (VMs) nécessaires à l'aide de l'outil Vagrant et configure les réseaux inter-connectant les VMs.

Cet outil open-source permet de créer un environnement portable et léger pour l'apprentissage, les tests et les certifications dans le domaine de la cybersécurité (« Sandbox Definitions · Wiki · MUNI-KYPO-CSC / cyber-sandbox-creator », s. d.).

8. https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator

1.2.2 CyRIS : Cyber Range Instantiation System

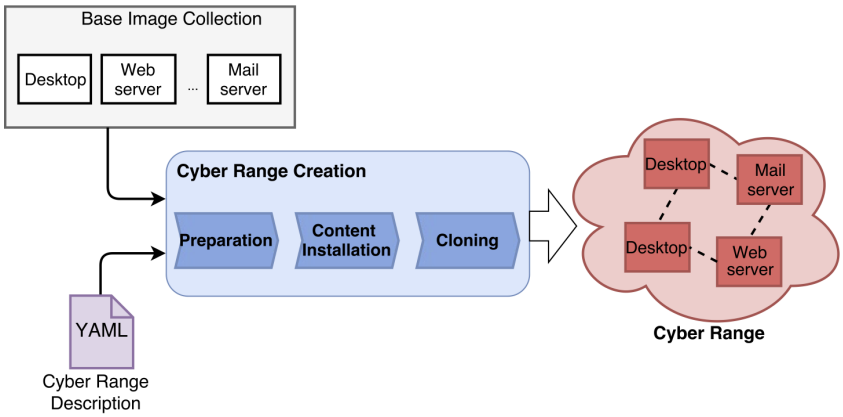


FIGURE *1.6: Schéma de fonctionnement de Cyris*
Source: *https://github.com/crond-jaist/cyris*

CyRIS permet de créer, puis de préparer un environnement virtuel à destination des étudiants. Ce logiciel se base sur un fichier de configuration spécifique décrit avec le langage YAML pour déployer un nombre défini de machines virtuelles sur le même hôte afin de les fournir aux étudiants pour les exercices réalisés simultanément (BEURAN et al., 2018).

Ce projet a également intégré le développement de CyLMS, un ensemble d'outils permettant d'ajouter les contenus de formations au système de gestion de l'apprentissage « Moodle » (« CyLMS: Cybersecurity Training Support for LMS », s. d.).

1.2.3 Lability

Avec l'outil Lability [9] proposé par l'entreprise Virtual Engine Limited, il est possible de procéder au provisionnement d'une machine via Windows Hyper-V. Comme il s'agit de fonctionnalités basées sur PowerShell, il n'est pas compatible avec tous les systèmes d'exploitation (VIRTUALENGINE, 2021).

Cet outil permet la création de réseaux et de machines virtuelles sur Hyper-V puis d'y provisionner majoritairement des systèmes d'exploitation Windows et Windows Serveur mais peut toutefois supporter des machines Linux.

1.2.4 Labtainers

Labtainers est le nom du projet qui regroupe une machine virtuelle ainsi qu'un framework pour la création de nouveaux environnements. Développé par la Naval Postgraduate School à Monterey en Californie, il se présente sous la forme d'une machine virtuelle contenant plus de 50 exercices s'exécutant à l'aide de Docker. L'étudiant peut démarrer un laboratoire à l'aide

9. https://github.com/VirtualEngine/Lability

d'une seule et unique commande qui démarrera un ou plusieurs conteneurs selon l'exercice à réaliser (« Labtainers - Center for Cybersecurity and Cyber Operations - Naval Postgraduate School », s. d.).

Le laboratoire, une fois démarré, fait apparaître plusieurs fenêtres avec interface graphique ou en ligne de commande ainsi que des instructions au format PDF.

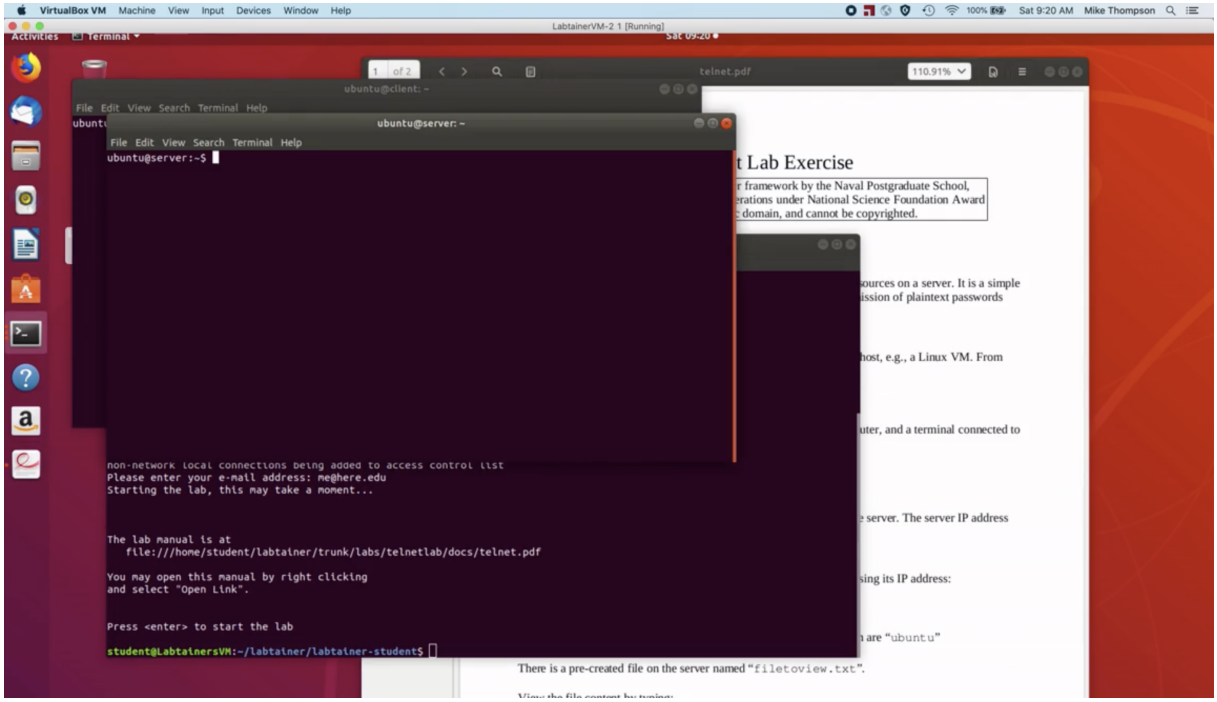


FIGURE *1.7: Démonstration de l'exécution d'un laboratoire dans labtainers*
Source: *https:// nps.edu/ web/ c3o/ labtainers-demo*

1.2.5 Information Systems Education Journal

L'article « Infrastructure Tools for Cybersecurity Exercices » du journal « Information Systems Education Journal » explique que l'outil Vagrant accompagné de VirtualBox sont une combinaison efficace, mature et flexible pour la création d'environnements à la volée (MARQUARDSON, 2018).

1.2.6 Université de Pittsburgh

Dans la présentation « Vagrant and Docker as learning environnement » de l'université de Pittsburgh on mentionne également la portabilité des environnements ainsi que le fonctionnement des outils Vagrant et Docker (TOMER, 2016).

1.2.7 SANS Institute

Dans le document « Using Vagrant to build a manageable and sharable intrustion detection lab » publié par l'institut SANS, il est mentionné que certains environnements tels que celui nommé DVWA (Damn Vulnerable Web Application) est très complexe à mettre en place. Pour remédier à ce problème, Shaun McCullough explique que Vagrant convient parfaitement pour créer, reproduire et déployer un environnement pré-configuré (McCullough, 2016).

1.2.8 Résumé

Nous constatons que toutes ces solutions se basent sur la virtualisation ou la conteneurisation. En effet, l'utilisation de ces technologies amène de nouvelles possibilités pour la création de systèmes de test isolés pour les étudiants. Elles permettent d'obtenir un environnement de laboratoire portable, reproductible afin d'étudier via l'expérimentation sans affecter le système d'exploitation hôte. Cela permet de garder un système principal sain et intact en cas de problème ou d'erreurs de manipulation dans l'environnement invité.

De plus, grâce à de nombreuses automatisations possibles, les étapes chronophages de configuration peuvent être exécutées simplement et rapidement.

1.3 Outils permettant le déploiement d'environnements

1.3.1 Virtualisation

La virtualisation est une technologie très répandue et utilisée dans l'informatique depuis de nombreuses années. Elle consiste à exécuter un système d'exploitation complet dans un environnement isolé de l'hôte sur lequel il fonctionne.

Plusieurs méthodes existent, la première s'exécute via un hyperviseur « bare-metal » dit de « type 1 » car il s'exécute directement sur une plate-forme matérielle (Citrix Xen Server, VMware vSphere, Oracle VM Server, Microsoft Hyper-V Server, KVM).

Dans la virtualisation à l'aide d'un hyperviseur de « type 2 », les machines virtuelles s'exécutent à l'aide d'un logiciel installé sur le système d'exploitation hôte (Parallels Desktop, Parallels Server, Oracle VM VirtualBox, VMware Fusion, VMware Player, VMware Server, VMware Workstation, QEMU).

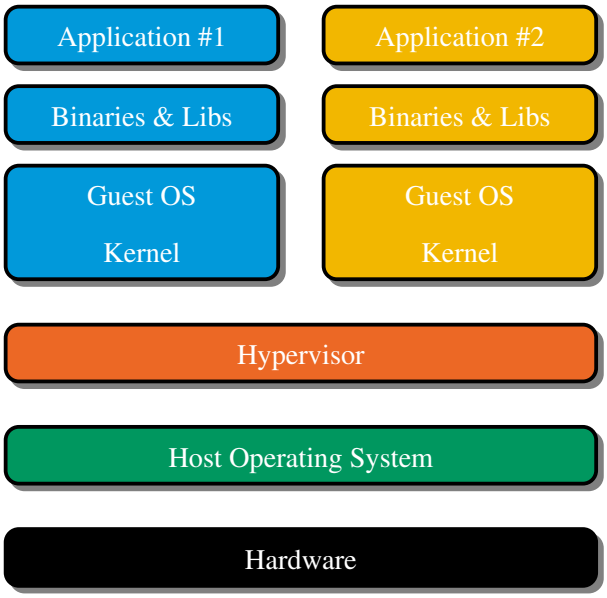


FIGURE *1.8: Schéma de fonctionnement de la virtualisation (type 2).*
Source: *(*BEUCHAT*, 2021)*

1.3.1.1 Oracle VM VirtualBox

Oracle VM VirtualBox est un logiciel libre et gratuit, développé et publié par Oracle, conçu pour la virtualisation professionnelle ou privée. Ses nombreuses fonctionnalités et la possiblité de réaliser des *snapshots* font de lui le logiciel utilisé majoritairement dans le cadre des cours à la HES-SO Valais-Wallis lorsqu'une machine virtuelle est nécessaire.

1.3.1.2 VMware Workstation Pro / Player et VMware Fusion

VMware Workstation Pro, VMWare Workstation Player et VMware Fusion sont trois produits commercialisés par la société VMware. Ils sont propriétaires et sont distribués gratuitement (VMware Workstation Player) ou après l'achat d'une licence d'utilisation (VMware Workstation Pro, VMware Fusion). Cette version gratuite étant limitée en fonctionnalités, elle ne donne pas accès aux *snapshots*.

1.3.1.3 Parallels Desktop

Parallels Desktop est un autre produit similaire commercialisé par Parallels Inc., entreprise spécialisée dans les technologies de virtualisation. Il est disponible exclusivement sur MacOS sous le nom de « Parallels Desktop pour Mac » et ne dispose pas de version gratuite.

1.3.2 Conteneurisation

Alors que la virtualisation nécessite d'exécuter indépendamment et de manière isolée le système d'exploitation invité complet, les conteneurs partagent et utilisent une partie du système d'exploitation hôte (noyau). C'est pour cela qu'une solution basée sur la conteneurisation sera plus légère. Toutefois, il est nécessaire que le noyau de l'hôte offre des mécanismes de conteneurisation.

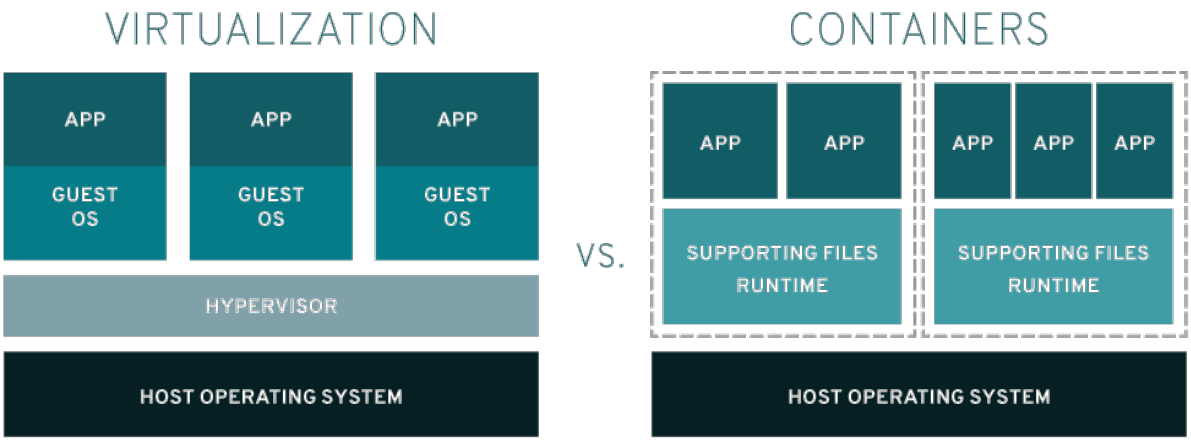


FIGURE *1.9: Virtualisation vs Conteneurisation*
Source: *(*REDHAT*, 2021)*

1.3.2.1 Docker

Docker est une solution de conteneurisation très répandue. Apparue pour la première fois en 2013, son utilisation ne cesse de croître (DATADOG, 2018). Docker permet d'exécuter les conteneurs directement sur Linux et à l'aide d'une machine virtuelle sur Windows et MacOS. Il permet de créer des conteneurs basés sur des images existantes ou de construire des images sur mesure afin d'y exécuter des applications et services.

Le fichier « Dockerfile », contenant les instructions spécifiques pour assembler une image, décrit avec la syntaxe YAML, est utilisé pour définir comment s'exécute le conteneur. En partant d'une image de base existante, des commandes permettent de mettre à disposition des services réseaux en exposant leurs ports respectifs, de définir des variables d'environnement, d'ajouter, de copier des fichiers et de monter des volumes.

En complément, il est possible d'installer l'outil « docker-compose ». À l'aide d'un fichier « docker-compose.yml » utilisant également le langage YAML, ce dernier permet de définir les services devant s'exécuter dans un ou plusieurs conteneurs interconnectés.

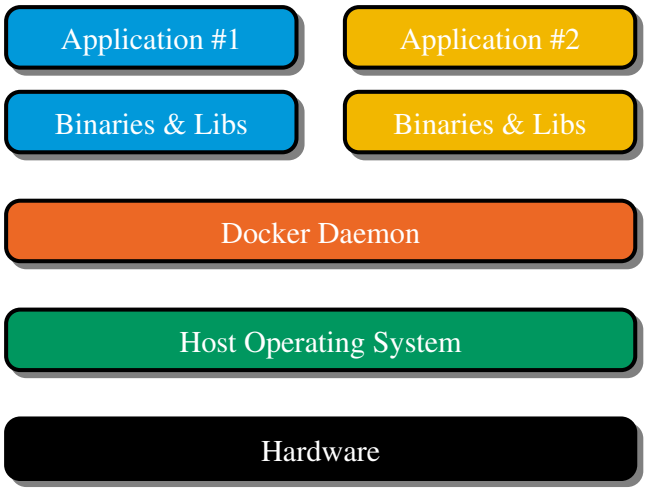


FIGURE *1.10: Schéma de fonctionnement de la conteneurisation.*
Source: *(*BEUCHAT*, 2021)*

1.3.2.2 Podman

Podman est une alternative à Docker proposée par RedHat. Son utilisation étant très similaire à Docker, les commandes sont sensiblement les mêmes. Podman est capable d'exécuter les conteneurs de la même manière que Docker mais il est également capable d'exécuter un nouveau type d'objet : un Pod.

« Un pod est un groupe d'un ou plusieurs conteneurs (comme des conteneurs Docker), ayant du stockage/réseau partagé, et une spécification sur la manière d'exécuter ces conteneurs. Les éléments d'un pod sont toujours co-localisés et co-ordonnancés, et s'exécutent dans un contexte partagé. Un pod modélise un "hôte logique" spécifique à une application - il contient un ou plusieurs conteneurs applicatifs qui sont étroitement liés — dans un monde pré-conteneurs, être exécuté sur la même machine physique ou virtuelle signifierait être exécuté sur le même hôte logique. » (KUBERNETES, 2020).

1.3.2.3 LXC

LXC est un ensemble d'outils open-source pour la conteneurisation. Cette solution permet aux utilisateurs d'un système Linux de créer et de gérer des applications et des systèmes de conteneurs. LXD, développé par Canonical, est une surcouche logicielle de LXC faisant partie du même projet « LinuX Containers ».

1.3.3 Création automatisée de machines virtuelles

En complément de ces solutions de virtualisation ou de conteneurisation, d'autres outils nous permettent d'automatiser la création de ces machines virtuelles ou conteneurs.

1.3.3.1 Vagrant

Vagrant est un outil distribué par HashiCorp destiné à la création et la gestion de machines virtuelles. Cet outil a été développé afin de permettre d'automatiser la mise en place d'environnements virtuels. Il est principalement employé dans un contexte de développement. C'est pour cela qu'il est déconseillé de l'utiliser pour de la production (« Vagrant in production - StackOverflow », 2021).

Les principales fonctionnalités que permet Vagrant sont :

— La création d'une machine à l'aide d'une seule commande (`vagrant up`).

— Le provisioning [10] à l'aide de scripts (Bash, Ansible, Puppet, Chef).

— L'accès SSH simplifié (`vagrant ssh`).

— La synchronisation de dossiers entre l'hôte et le système invité.

— Le partage d'environnements via Internet, permettant d'exposer un service à travers une connexion réseau temporaire (`vagrant share`).

10. Automatisation de tâches variées permettant l'installation et la configuration d'un système.

Il y a plusieurs éléments à connaître pour comprendre le fonctionnement de Vagrant :

1. Le **Provider** est l'environnement dans lequel Vagrant doit créer sa machine virtuelle (hyperviseur). Sont supportés : VirtualBox (gratuit), VMware (payant ; US$ 79) et Parallels (gratuit ; nécessite une licence Parallels Desktop pour Mac en version Pro ou Business) (« Installing Provider - Vagrant Parallels Provider Documentation », 2021).

2. La **Box** est une machine virtuelle prête à l'emploi sous forme d'une image disque compressée. Elle est accompagnée d'un premier fichier de métadonnées au format JavaScript Object Notation (JSON) utilisé lors des échanges avec le catalogue « Vagrant Cloud », ainsi qu'un second fichier facultatif contenant les informations additionnelles telles que l'auteur, son site web ou d'autres valeurs personnalisées (« Box File Format | Vagrant by HashiCorp », 2021).

3. Le fichier **Vagrantfile** contient la configuration basique de l'environnement, la configuration réseau, la configuration SSH, la déclaration des dossiers partagés et les informations et scripts pour le provisionnement. Il peut être rédigé manuellement ou généré à l'aide de la commande d'initialisation `vagrant init`.

Les « Box » Vagrant sont disponibles sur le site officiel HashiCorp Vagrant Cloud [11] ou sur d'autres sites tiers comme vagrantbox.es [12].

Ces dernières sont mises à disposition librement par des personnes ou des entreprises. C'est pour cela que la qualité, la stabilité et la sécurité ne peut être assurée. Tout le monde peut créer une Box Vagrant et la publier. Mise à part l'indication du nombre de téléchargements, le site HashiCorp ne dispose par exemple d'aucun système de notation afin d'évaluer la qualité. Avant d'utiliser l'environnement publié, il faut accorder sa confiance dans l'image fournie. Bien que certains utilisateurs rendent disponibles les sources permettant de créer l'environnement sur un dépôt git (par exemple sur GitHub), ce n'est pas le cas de toutes les images publiées.

Parmi ces « Box », on y retrouve différents systèmes d'exploitation avec des configurations spécifiques à chaque utilisation.

Cependant, les machines officielles basées sur Windows ne sont pas disponibles sur Vagrant Cloud car Microsoft interdit la distribution de son système d'exploiration par les utilisateurs (McCullough, 2016). Seule une Box Windows 10 officielle, équipée du navigateur Edge, est disponible et publiée par Microsoft. Elle n'a cependant pas été mise à jour depuis 2015 [13]. En cas de nécessité d'utilisation d'un système d'exploitation Windows, il est possible de créer sa propre Box à l'aide de l'outil Packer décrit dans la section 1.3.6.1.

11. https://app.vagrantup.com/
12. http://vagrantbox.es
13. https://app.vagrantup.com/Microsoft/boxes/EdgeOnWindows10

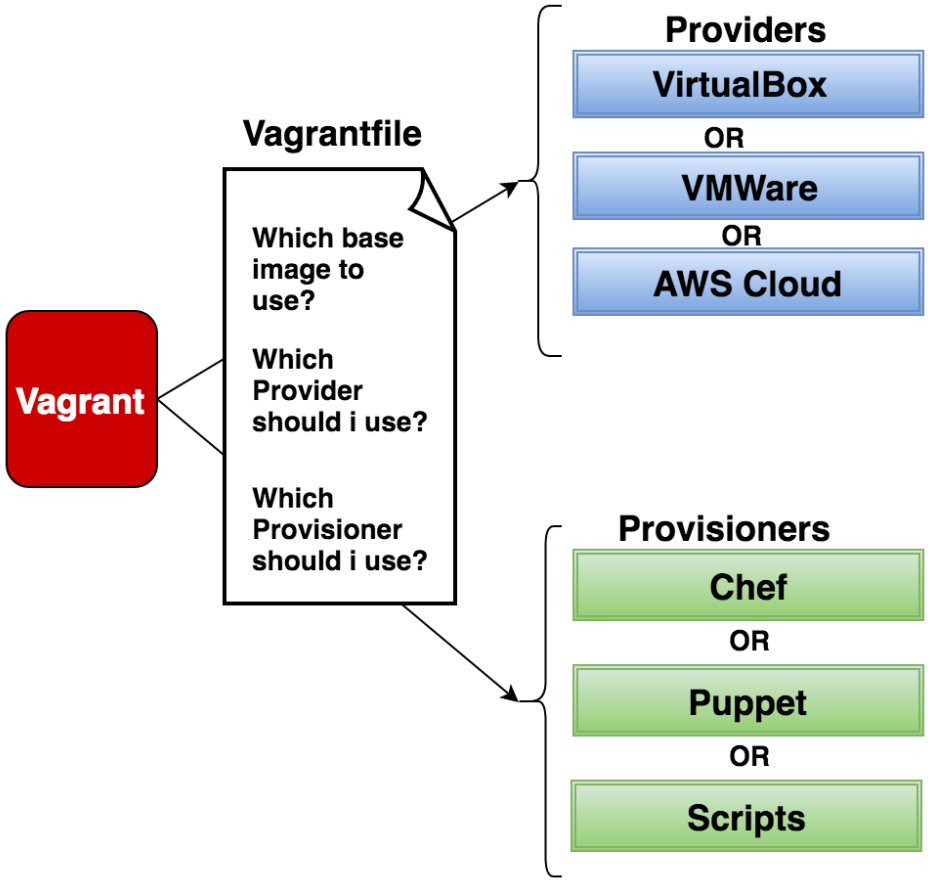


FIGURE *1.11: Schéma de fonctionnement de Vagrant*
Source: *https://www.slashroot.in/what-vagrant-and-how-does-it-work*

Vagrant est disponible en version 2.2.18 en date du 8 août 2021. HashiCorp a annoncé la conception de Vagrant 3.0 et confirme qu'il n'y aura pas d'incompatibilités liées aux fichiers Vagrantfile. Cette nouvelle version sera réécrite en langage Go au lieu de Ruby actuellement (CHRIS ROBERTS, 2021).

1.3.4 Orchestration

1.3.4.1 Kubernetes

Kubernetes est une plate-forme d'orchestration de conteneurs libre et open-source qui automatise l'exploitation de conteneurs. Elle permet de simplifier les processus manuels de publication en exécutant les applications « packagées » dans un format spécifique de conteneurs. Ces derniers sont déployés et exécutés dans un cluster Kubernetes.

Kubernetes fonctionne avec plusieurs plusieurs environnements d'exécution de conteneurs : Docker, containerd, CRI-O et toutes les autres implémentation de Kubernetes Container Runtime Interface (CRI).

Kubernetes se charge du déploiement, de la mise à disposition de l'application ainsi que d'un tableau de contrôle, de la répartition de charge (load balancing), de la surveillance des conteneurs exécutés et du redémarrage de ces derniers en cas de problème.

« Minikube » est un outil permettant d'exécuter un cluster Kubernetes avec un seul noeud [14], en local. Il fonctionne sur Windows, MacOS et Linux à travers plusieurs modes de fonctionnements : à l'aide d'une VM, d'un conteneur, ou en hyperviseur « bare-metal » (THE KUBERNETES AUTHORS, s. d.).

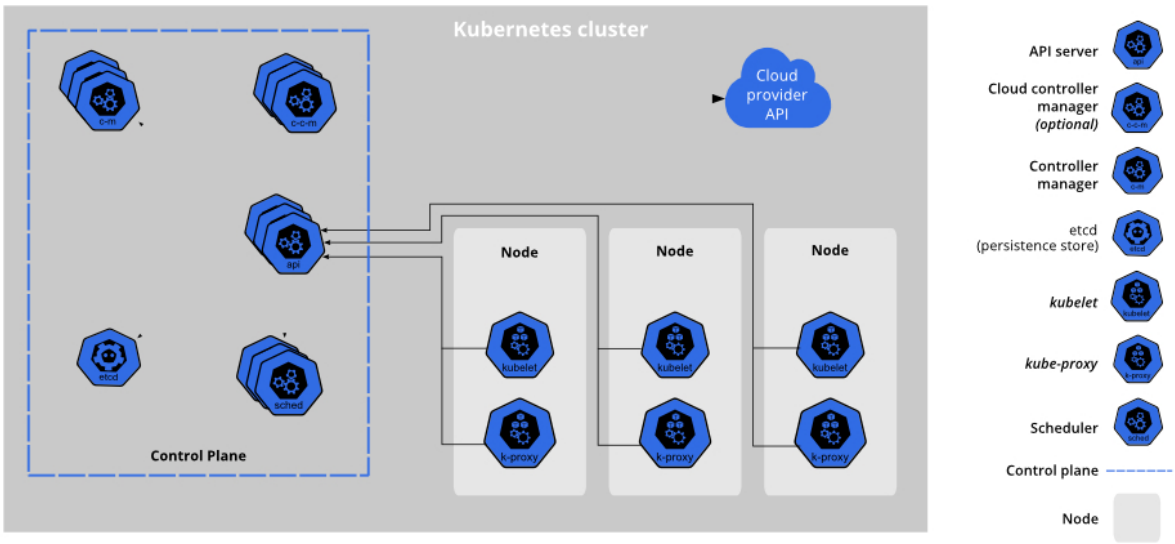


FIGURE *1.12: Schéma de fonctionnement de Kubernetes*
Source: *https://kubernetes.io/docs/concepts/overview/components/*

1.3.5 Gestion de configuration

Une fois la machine virtuelle créée, il est possible de procéder à l'installation et à la configuration de logiciels. Pour cela plusieurs outils permettent d'exécuter automatiquement ces tâches. Afin de choisir la solution adaptée, nous allons comparer les principaux outils utilisés dans l'industrie.

1.3.5.1 Ansible

Ansible est un outil permettant d'appliquer des configurations à partir d'un fichier de configuration YAML nommé « playbook ». Simple d'utilisation et très puissant, il est largement recommandé par les professionnels (RAYOME, ALISON DENISCO, 2019).

```
1 ---
2 - name: Update web servers
3   hosts: webservers
4   remote_user: root
5
```

14. « Un nœud est une machine de travail dans Kubernetes » (KUBERNETES, 2021)

```
  tasks:
  - name: Ensure apache is at the latest version
    ansible.builtin.yum:
      name: httpd
      state: latest
  - name: Write the apache config file
    ansible.builtin.template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest version
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Exemple de recette Ansible

Source : *https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html*

Ansible fonctionne en mode « Push ». C'est-à-dire que les configurations sont « poussées » sur les machines gérées. Ansible ne nécessitant pas d'installation de logiciel client sur les machines cibles, la communication se fait au travers d'une connexion SSH sécurisée. À la différence d'autres solutions, l'état de toutes les machines n'est pas connu par le serveur qui exécute les tâches. Il ne s'agit pas réellement d'un problème car les scripts sont conçus pour réaliser des opérations idempotentes [15]. Il faut cependant veiller à bien rédiger les playbooks afin qu'ils respectent ce principe.

Certaines entreprise utilisent une solution complémentaire nommée « Ansible Tower » (payante) pour centraliser la gestion des machines et améliorer la visibilité de leur état.

1.3.5.2 Chef

Chef est une alternative à Ansible. Conçu initialement par Opscode puis acquis par Progress Software en 2020, il est écrit en Ruby et est conçu pour fonctionner en mode client-serveur (mode « Pull ») mais peut également être utilisé en version *standalone*, sans serveur.

```
node.default['main']['doc_root'] = "/vagrant/web"

execute "apt-get update" do
 command "apt-get update"
end

```

15. « Une opération est idempotente si le résultat de son exécution unique est exactement le même que le résultat de son exécution répétée sans aucune action intermédiaire. » (GOFFINET, 2020)

```
apt_package "apache2" do
 action :install
end

service "apache2" do
 action [ :enable, :start ]
end

directory node['main']['doc_root'] do
 owner 'www-data'
 group 'www-data'
 mode '0644'
 action :create
end

cookbook_file "#{node['main']['doc_root']}/index.html" do
 source 'index.html'
 owner 'www-data'
 group 'www-data'
 action :create
end

template "/etc/apache2/sites-available/000-default.conf" do
 source "vhost.erb"
 variables({ :doc_root => node['main']['doc_root'] })
 action :create
 notifies :restart, resources(:service => "apache2")
end
```

Exemple de recette Chef

Source : *https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-chef-recipes*

1.3.5.3 Puppet

Puppet est également une alternative à Ansible et Chef. À la différence d'Ansible, Puppet fonctionne en mode « Pull ». C'est à dire que la machine cible va elle-même récupérer les configurations à appliquer depuis le serveur primaire.

```
class apache2 {
 if $::osfamily == 'RedHat' {
   $apachename = 'httpd'
 } elseif $::osfamily == 'Debian' {
   $apachename = 'apache2'
 } else {
   print "This is not a supported distro."
 }

 package { 'apache'
   name => \$apachename,
   ensure => 'present',
 }

 service { 'apache-service':
   name => \$apachename,
   enable => true,
```

```
18      ensure => 'running',
19    }
20  }
```

Exemple de recette Puppet
Source : *https://logz.io/blog/chef-vs-puppet/*

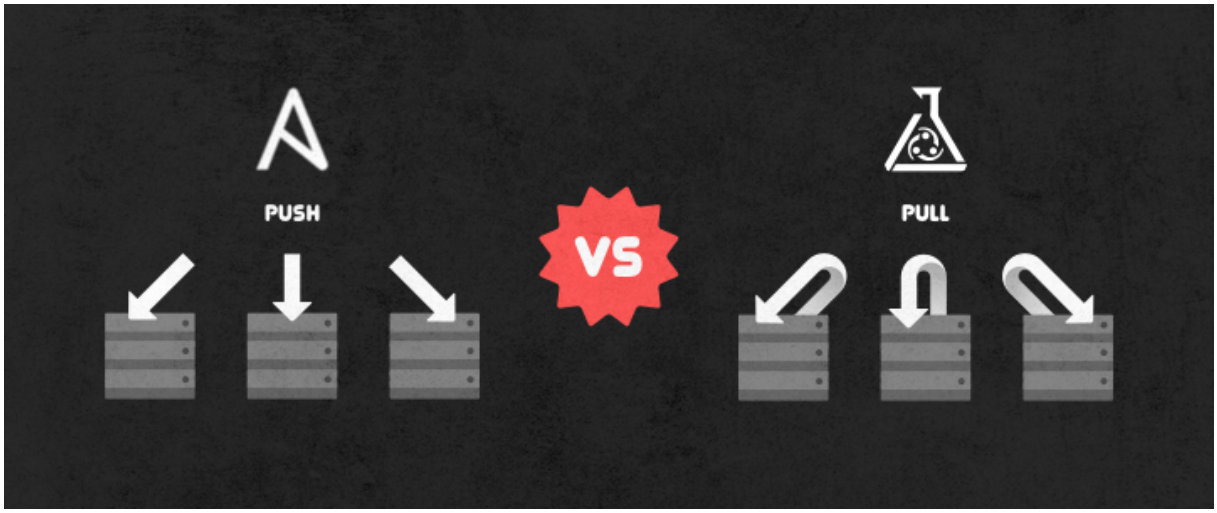


FIGURE *1.13: Fonctionnement Push vs Pull*
Source: *https:// wiredcraft.com/ blog/ getting-started-with-ansible-in-5-minutes/*

1.3.5.4 Comparaison des outils

Outils de gestion de configuration			
Configuration	**Ansible**	**Chef**	**Puppet**
Fichier de configuration, Domain Specific Language (DSL)	DSL basé sur YAML	DSL basé sur Ruby	DSL proche du JSON
Simplicité d'installation	Facile	Modérée	Modérée
Simplicité d'utilisation	Facile	Difficile	Difficile
Popularité (GitHub stars)	49'000	6'600	6'200
Interopérabilité	Haute	Haute	Haute
Client-Serveur	Oui	Oui	Oui
Standalone	Oui	Oui (chef-solo)	Oui (Puppet standalone)
Fonctionnements des mises à jour	Push et Pull (ansible-pull)	Pull	Pull
Language de développement	Python, PowerShell, Ruby	Ruby	Ruby
Développeurs	Red-Hat et Ansible	Opscode	Puppet Labs
Communauté	Très grande	Grande	Grande
Version entreprise disponible	Oui	Oui	Oui
Sortie initiale	2012	2009	2005
Utilisations	Déploiement d'applications, gestion de configuration	CI/CD [16], automatisation d'infrastructure	Déploiement d'applications, déploiement et configuration de serveurs

TABLE *1.1: Comparatif des outils de gestion de configuration*

1.3.6 Création d'images

1.3.6.1 Packer

Packer de HashiCorp est un outil libre et open-source permettant de créer des images de machines pour différentes plateformes à partir d'un fichier de configuration. Ce dernier peut être écrit à l'aide du format JSON mais également à l'aide du format recommandé par HashiCorp, le HashiCorp Configuration Language (HCL).

Le fonctionnement de Packer est semblable à celui de Vagrant. Tout d'abord, les « builders » créent des machines pour une ou plusieurs plateformes spécifiques (Vagrant, VirtualBox, Cloud Azure, Cloud Google, Cloud Amazon, etc..).

Les *provisioners* sont ensuite chargés de configurer la machine après le démarrage. En général, ils sont utilisés pour installer et configurer des logiciels, télécharger les fichiers nécessaires, créer des utilisateurs. Comme avec Vagrant, nous avons accès aux mêmes outils de configuration (Ansible, Chef, Puppet, Shell, etc.).

Après la construction et le provisionnement de l'image, le post-processeur peut être utilisé de manière facultative pour des actions telles que la création d'un *checksum* ou d'un envoi du résultat de l'exécution de Packer vers Vagrant Cloud ou un autre service Cloud.

1.3.7 Publication et mise à disposition d'images

1.3.7.1 Vagrant Cloud

Vagrant Cloud est une plateforme permettant la publication et l'échange de « Box » Vagrant. Il s'agit de l'annuaire principal et officiel sur lequel se trouvent les images qui peuvent être utilisées par n'importe qui. C'est l'endroit où nous pouvons publier nos propres images afin de les rendre publiques.

La publication d'une « Box » publique est gratuite. Cependant, si la « Box » doit être privée, il faut souscrire à un abonnement payant.

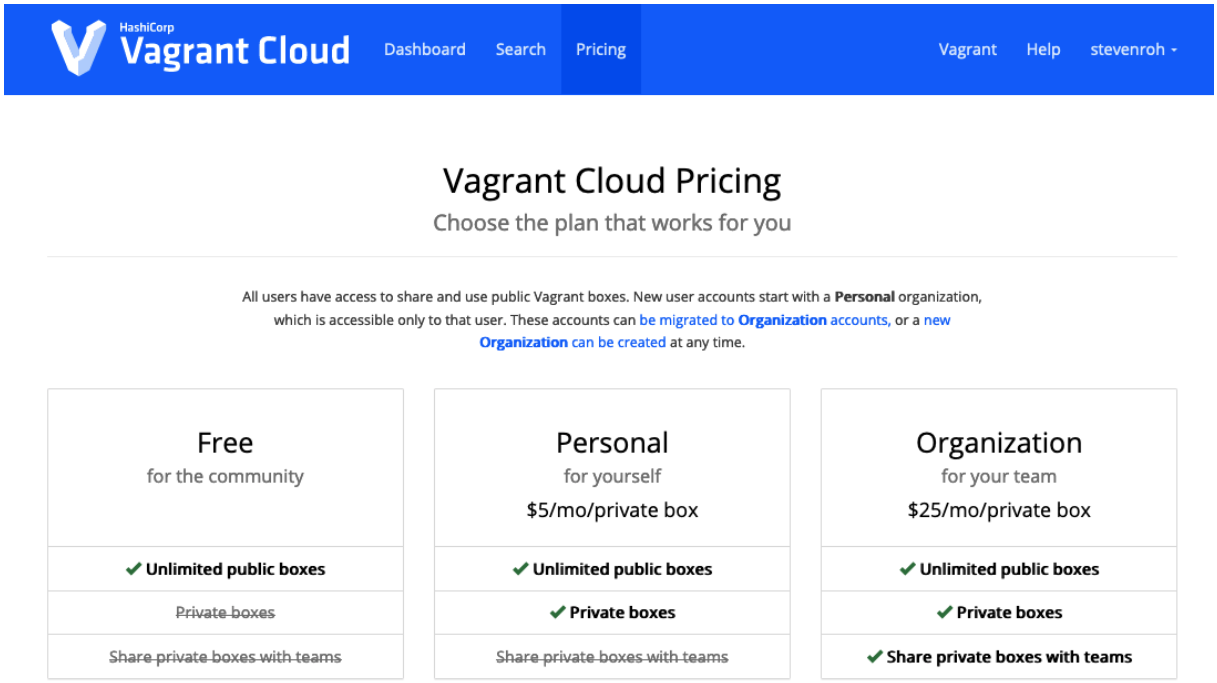


FIGURE *1.14: Capture d'écran du site Vagrant Cloud affichant les tarifs des abonnements mensuels*
Source: *de l'auteur à partir de app.vagrantup.com*

Les filtres permettent de rechercher une « Box » en fonction de l'hyperviseur compatible (provider) ainsi que de trier par date et popularité.

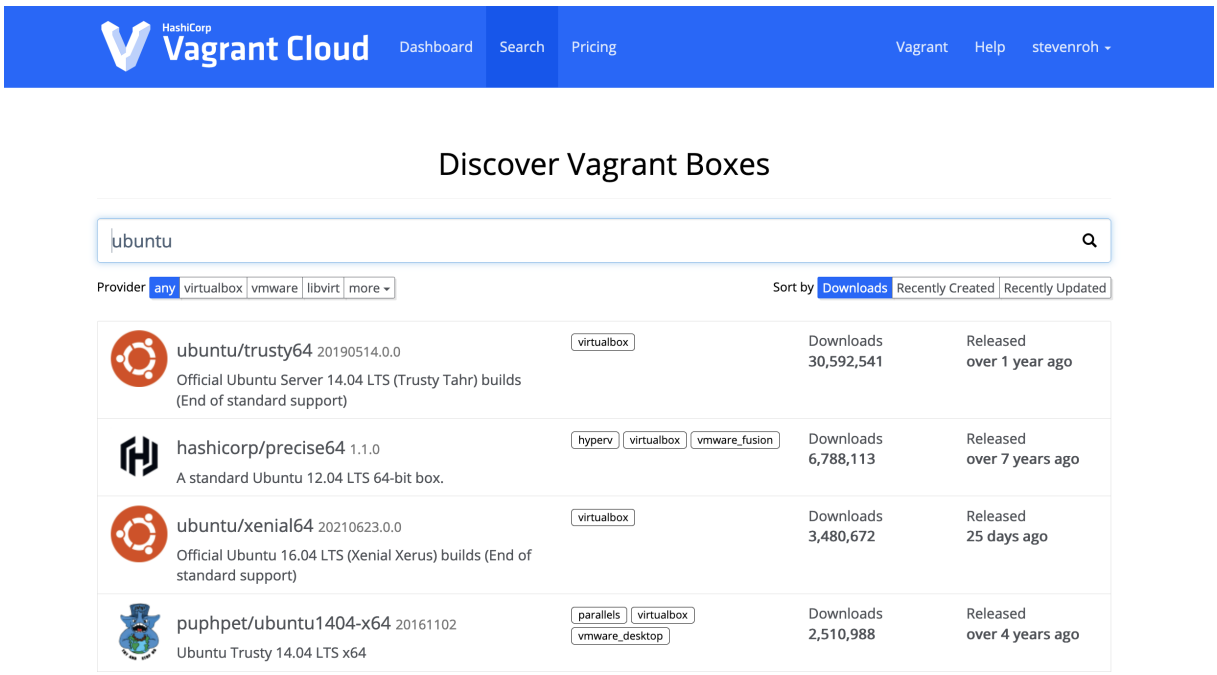


FIGURE *1.15: Capture d'écran du site Vagrant Cloud lors d'une recherche d'une « Box » Ubuntu*
Source: *de l'auteur à partir de app.vagrantup.com*

Une fois inscrit sur la plate-forme nous pouvons publier des images sous notre nom d'utilisateur mais également sous le nom d'une organisation dont nous faisons partie.

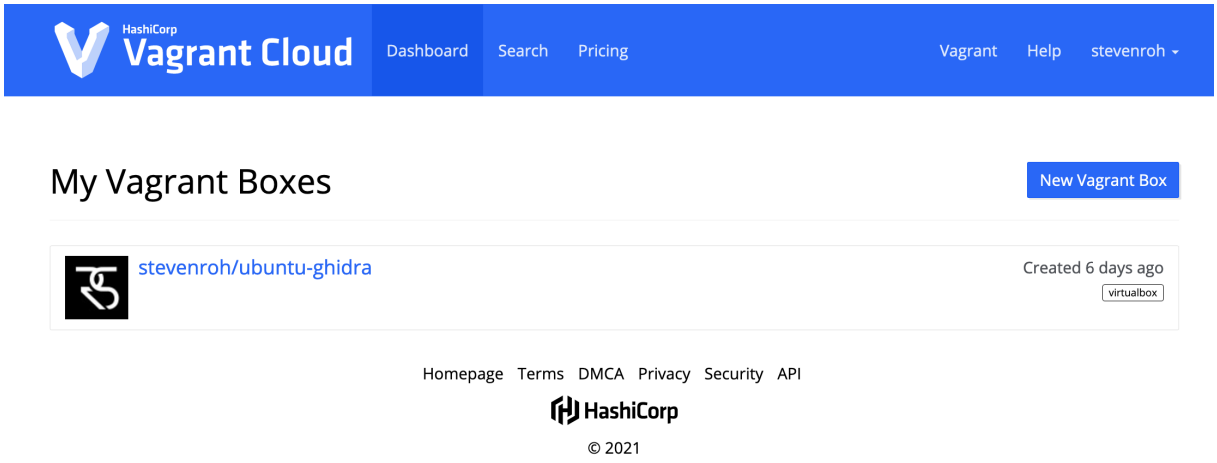


FIGURE *1.16: Capture d'écran du site Vagrant Cloud affichant les « Box » personnelles*
Source: *de l'auteur à partir de app.vagrantup.com*

1.4 Déploiement dans le Cloud

Auprès des entreprises, l'adoption du Cloud ne cesse d'augmenter. Dans le rapport *Flexera 2021 State of the Cloud report* (FLEXERA, 2021), nous pouvons constater que ces dernieres utilisent majoritairement les solutions AWS d'Amazon et Azure de Microsoft comme fournisseur de services Cloud.

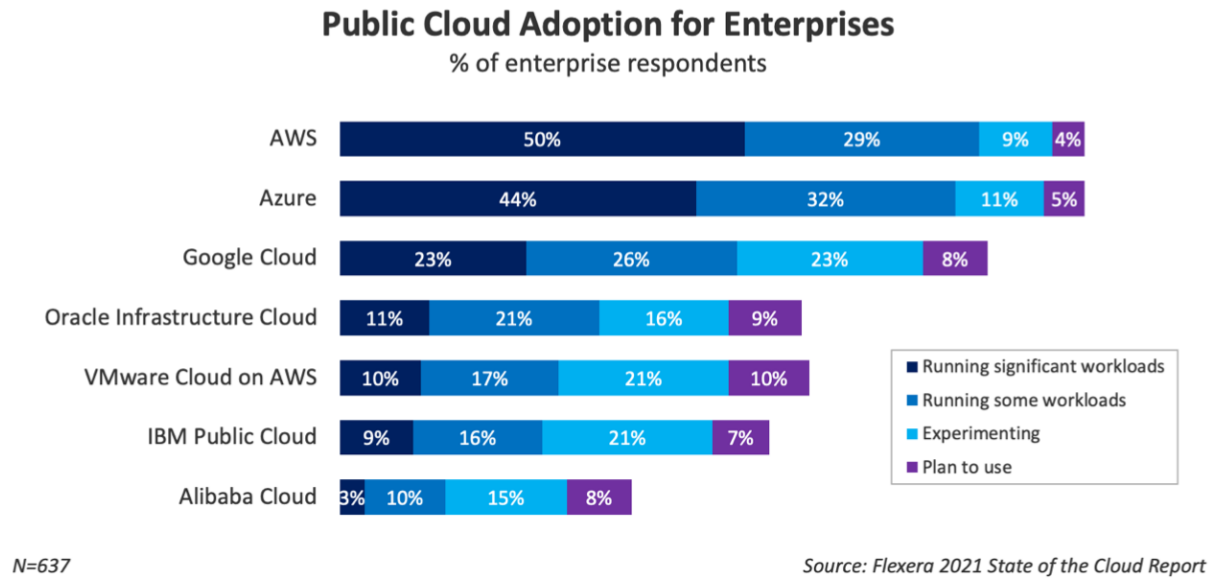


FIGURE *1.17: Adoption du Cloud dans les entreprises*
Source: FLEXERA*, 2021*

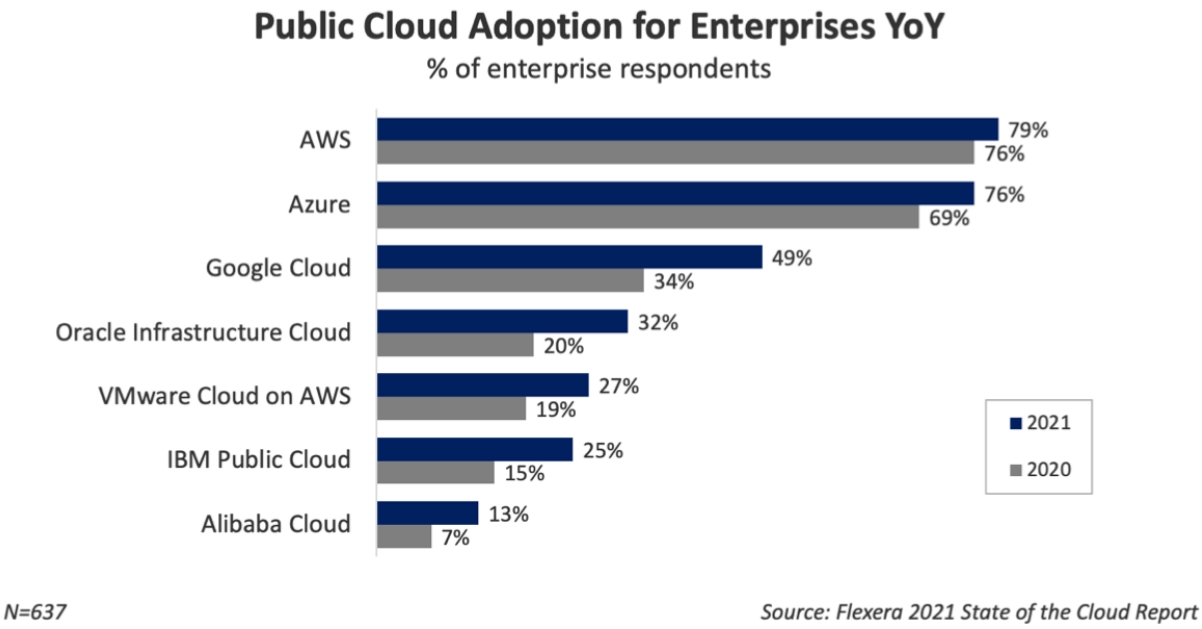


FIGURE *1.18: Adoption du Cloud : comparaison de 2021 par rapport à 2020*
Source: FLEXERA*, 2021*

1.4.1 Fournisseurs de Cloud Public

1.4.1.1 Amazon Web Services

Avec son service Amazon Elastic Compute Cloud (Amazon EC2), Amazon permet d'exécuter des machines virtuelles dans le Cloud. Cela est possible à partir d'une image Amazon Machine Image (AMI) existante ou d'une image sur mesure.

D'autre part, Amazon propose aussi Elastic Kubernetes Service (EKS) pour l'exécution de conteneurs d'applications Kubernetes.

Dans la FAQ d'Amazon Educate, la plateforme d'apprentissage pour les produits Amazon, nous constatons qu'il est possible d'obtenir un crédit de US$ 100 pour les étudiants si on est invité par un professeur : « Students ages 18+ invited by their educator at a member institution to join AWS Educate can enroll with an AWS Educate Starter Account and receive $100 in AWS Promotional Credit and $30 in credit at a non-member institution. » (« Amazon Educate - FAQs », 2020).

1.4.1.2 Microsoft Azure

Microsoft propose également son service Azure Virtual Machine pour déployer des machines virtuelles et Azure Kubernetes Service (AKS) pour l'exécution sur les *clusters* Kubernetes.

Ils proposent gratuitement deux offres pour les étudiants : *Azure for Students Starter pack* pour le développement sur le Cloud, ainsi que *Azure for Students* qui permet aux étudiants de bénéficier d'un crédit de US$ 100 chaque année [17].

La création d'un compte sur cette plateforme se fait à l'aide se son adresse e-mail d'étudiant (HES-SO), sans intervention du professeur et ne nécessite pas de posséder de carte de crédit. L'étudiant étant membre de l'organisation de l'école, il est déjà intégré dans l'annuaire des utilisateurs utilisé sur la plateforme Azure.

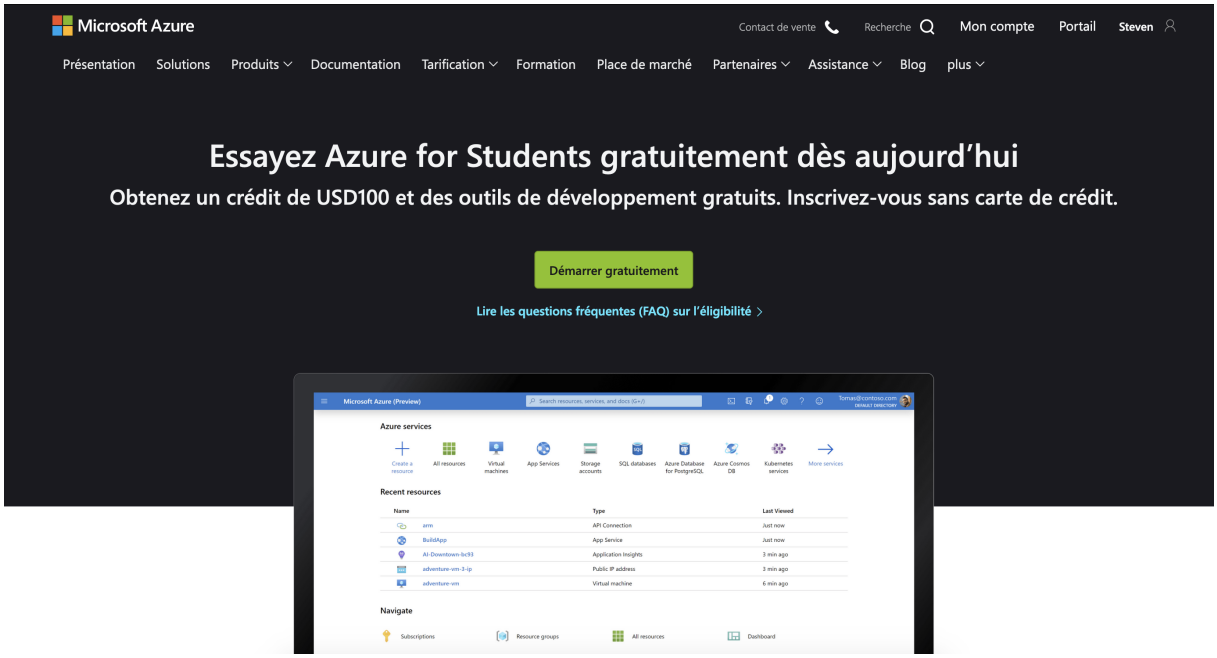


FIGURE *1.19: Crédit Azure offert pour les étudiants*
Source: *de l'auteur à partir de azure.microsoft.com*

1.4.1.3 Autres fournisseurs de services Cloud

Amazon et Microsoft ne sont pas les seuls fournisseurs à proposer des services Public Cloud, nous pouvons également retrouver :

- Alibaba Cloud
- Digital Ocean
- Exoscale
- OVH Public Cloud
- Infomaniak Public Cloud (beta [18])
- Google Cloud Platform

17. https://azure.microsoft.com/en-in/offers/ms-azr-0170p/
18. Annonce de la version beta par Infomaniak : https://twitter.com/infomaniak/status/1414471623718146053

1.4.2 Avantages et inconvénients d'une solution Cloud

Jusqu'à maintenant, les exercices étaient destinés à être exécutés localement. Le déploiement dans le Cloud pourrait apporter certains avantages intéressants.

Le tableau ci-après liste les principaux avantages et inconvénients d'une telle solution.

Utilisation du Cloud	
Avantages	**Inconvénients**
• Les performances de la machine de l'étudiant n'affectent pas la réalisation du travail pratique dans le Cloud. • La machine de l'étudiant n'est pas impactée par une mauvaise manipulation dans l'environnement d'exercice. • L'utilisation d'une infrastructure Cloud donne accès à des machines plus performantes permettant d'exécuter des outils plus gourmands ou à un plus grand nombre de machines simultanément. • Le déploiement dans le Cloud peut permettre la collaboration avec d'autres étudiants sur les mêmes environnements déployés. • L'étudiant peut être familiarisé avec le Cloud et les outils nécessaires au déploiement. Ces compétences techniques lui seront utiles pour sa future carrière.	• Le déploiement dans le Cloud est facturé à l'utilisation et les offres pour étudiants sont limitées. • Si l'étudiant dispose de ressources gratuites limitées, l'oubli de l'arrêt d'un environnement déployé épuisera ses crédits rapidement. • En cas de déploiement dans le Cloud, il faut tester et maintenir la compatibilité des laboratoires avec les services Cloud. • L'utilisation d'une infrastructure dans le Cloud risque de rajouter et de complexifier les étapes de configuration et de l'apprentissage. • Une connexion Internet et une disponibilité du service Cloud sont nécessaires durant tout le travail de l'étudiant. Les exercices ne peuvent être effectués hors-ligne.

TABLE *1.2: Avantages et inconvénients d'une solution Cloud*

1.4.3 Choix du fournisseur de services Cloud pour les laboratoires

Pour le déploiement des exercices pratiques, Microsoft Azure semble être une bonne solution. En effet, le crédit généreux renouvelable chaque année ainsi que l'intégration aux services de l'école sont des arguments positifs en la faveur du service de Microsoft.

1.4.3.1 Stockage et partage d'images

Microsoft Azure propose également un service nommé « galerie d'images partagées » dédié au stockage d'image. Comme son nom l'indique, il permet de partager les images de machines virtuelles envoyées dans la collection avec d'autres utilisateurs de l'annuaire ou même des utilisateurs externes.

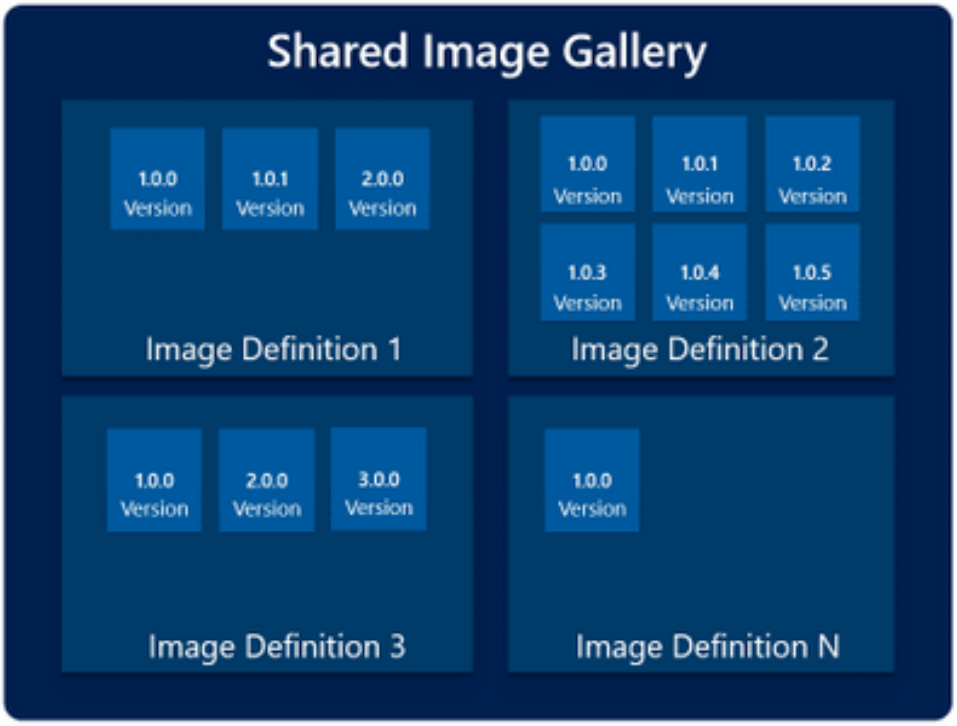


FIGURE *1.20: Schéma d'une galerie d'image partagée, ses définitions et versions*
Source: *https:// techcommunity.microsoft.com/*

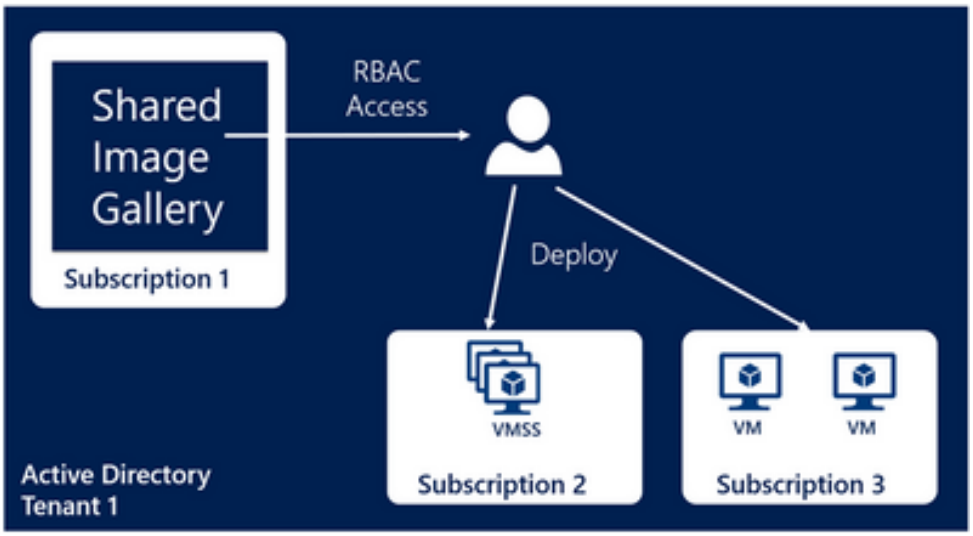


FIGURE *1.21: Utilisation d'une galerie d'image partagée*
Source: *https:// techcommunity.microsoft.com/*

Dans cette galerie, nous pouvons y créer une « définition d'image » qui contiendra toutes les informations sur l'image et ses exigences. Cela inclut le type de système d'exploitation (Linux ou Windows) et les besoins en processeur et mémoire vive.

Chaque définition peut comporter plusieurs versions d'images rendues disponibles dans une région spécifique.

1.4.4 Provisionnement de l'infrastructure

1.4.4.1 Terraform

Terraform est également un outil distribué par HashiCorp. Alors que Vagrant est dédié à la création et gestion d'environnements de développement, Terraform lui se concentre sur la définition d'infrastructures à partir de code (Infrastructure as code) (« Terraform by Hashicorp », 2021).

De ce fait, Terraform n'inclut pas certaines fonctionnalités spécifiques que l'on retrouve dans Vagrant tels que les dossiers partagés, la configuration de réseau automatique, ...

L'objectif primaire de Terraform est la gestion de ressources distantes sur les fournisseurs Cloud tels qu'AWS ou Azure. Il a été conçu pour être capable de gérer de très grandes infrastructures à travers plusieurs fournisseurs de service Cloud (« Vagrant vs. Terraform », 2021).

1.4.5 Distribution continue

La création de l'image à l'aide de Packer puis son téléversement peut prendre du temps en fonction du volume des images ou de la quantité de traitements de préparation de ces dernières. Pour rendre ce processus plus automatisé et surtout l'exécuter en arrière-plan, nous pouvons utiliser les fonctions « CI/CD » de notre dépôt GitLab.

2 | Déploiement en local

2.1 Création d'une machine avec Vagrant

Pour créer une nouvelle machine, nous avons besoin d'une « Box » de base. La liste des « Box » Vagrant disponibles peut être consultée sur HashiCorp Vagrant Cloud [1].

Pour la suite, nous allons nous baser sur la Box Ubuntu 20.04 (LTS) [2], qui est maintenue jusqu'en avril 2025. Le nom de la box est *ubuntu/focal64*.

Pour initialiser cette machine, nous avons les deux options suivantes :

1. Initialiser un nouveau Vagrantfile à partir de la commande `vagrant init` depuis un dossier vide.

```
vagrant init ubuntu/focal64
vagrant up
```

2. Créer et modifier directement un fichier Vagrantfile .

```
Vagrant.configure("2") do |config|
    config.vm.box = "ubuntu/focal64"
end
```

2.2 Configuration

2.2.1 Paramètres généraux

Le fichier Vagrantfile contient également les paramètres pour le Provider qui sera utilisé pour la création de la machine. Par défaut, il s'agit de VirtualBox que nous souhaitons utiliser.

Tous les paramètres de VirtualBox exposés par VBoxManage sont modifiables. Les plus communs disposent de noms de propriétés dédiés (nom de machine, état de l'interface graphique, vCPUs, mémoire vive, type de cartes réseau, ...).

Dans l'exemple suivant, nous créons une nouvelle machine *machine-name*, sans interface graphique, avec 1 CPU virtuel, 1024 Mo de mémoire vive et l'assignons au groupe *VirtualBox-GroupName* afin de l'identifier facilement et de la regrouper visuellement avec d'autres machines dans l'interface graphique de Virtualbox.

1. https://app.vagrantup.com/boxes/search
2. https://app.vagrantup.com/ubuntu/boxes/focal64

```
Vagrant.configure("2") do |config|
    config.vm.box = "ubuntu/focal64"

    config.vm.provider :virtualbox do |v|
        v.name = 'machine-name'
        v.gui = false
        v.memory = 1024
        v.cpus = 1
        v.customize ["modifyvm", :id, "--groups", "/VirtualBoxGroupName"]
      end
end
```

Le nom de la machine peut être spécifié à l'aide de :

```
config.vm.hostname = 'machine-name'
config.vm.define 'machine-name'
```

2.2.2 Dossiers partagés

Par défaut, le dossier du projet (emplacement du Vagrantfile) est automatiquement monté dans la machine invitée à l'emplacement `/vagrant` sur les machines Linux.

Ce comportement peut être désactivé sur demande :

```
config.vm.synced_folder ".", "/vagrant", disabled: true
```

D'autres dossiers peuvent être partagés entre l'hôte et l'invité grâce à l'instruction suivante dans le Vagrantfile :

```
config.vm.synced_folder "www/", "/var/www"
```

D'autres techniques plus spécifiques telles que rsync, SMB et NFS sont disponibles et permettent le montage de dossiers via le réseau.

2.2.3 Redirection de ports

La redirection de ports permet de rendre disponible un port de la machine invitée sur la machine hôte.

Dans l'exemple suivant, le port 80 exposera un éventuel serveur web s'exécutant dans le système d'exploitation invité sur le port 8080 de la machine hôte.

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

Par défaut, il s'agit du protocole TCP mais il est également possible de spécifier le protocole UDP :

```
config.vm.network "forwarded_port", guest: 2003, host: 12003, protocol: "udp"
```

Si les ports spécifiés dans la configuration sont déjà utilisés, Vagrant affichera un avertissement sur une « collision de ports ». Vagrant peut automatiquement choisir un nouveau numéro de port si on active la correction automatique (`auto_correct`). Dans les laboratoires pour étudiants, les ports utilisés doivent être connus et fixes, c'est pourquoi cette option ne sera généralement pas activée.

2.3 Configuration du réseau

2.3.1 Réseau interne

Par défaut, les machines créées sont configurées en « host-only », elles peuvent donc uniquement communiquer avec la machine hôte. Dans certains cas, nous souhaitons créer un réseau privé pour la communication entre les machines virtuelles.

L'instruction suivante permet de spécifier l'utilisation d'un réseau privé, de l'adresse IP à utiliser et d'activer le réseau interne.

```
config.vm.network :private_network, ip: "11.11.11.10", virtualbox__intnet: true
```

2.4 Configuration et installation des outils nécessaires

Une fois la machine configurée et exécutée, nous pouvons procéder à son installation et à sa configuration. Nous disposons de plusieurs possibilités. Les deux premières, permettant d'exécuter des commandes ou des scripts, sont très limitées et ne seront probablement que très peu utilisées pour créer les environnements pour les étudiants. Nous privilégieront des solutions plus robustes citées plus bas.

2.4.1 Provisionning avec shell

Pour exécuter des commandes directement dans le Vagrantfile, nous pouvons le spécifier de plusieurs manières :

1. En englobant le ou les lignes du script avec «`-SHELL` et `SHELL`, où `SHELL` est un mot-clé de votre choix.

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.provision "shell", inline: <<-SHELL
    echo Provisioning...
    date > /etc/vagrant_provisioned_at
  SHELL
end
```

2. En utilisant le même principe mais en améliorant la lisibilité en définissant le script dans une variable.

```
    $script = <<-SHELL
        echo Provisioning...
        date > /etc/vagrant_provisioned_at
    SHELL

    Vagrant.configure("2") do |config|
      config.vm.box = "ubuntu/focal64"
      config.vm.provision "shell", inline: $script
    end
```

3. En spécifiant une ligne de script (`inline`).

```
    Vagrant.configure("2") do |config|
      config.vm.box = "ubuntu/focal64"
      config.vm.provision "shell",
        inline: "echo Hello, World"
    end
```

4. En spécifiant un nom de script local ou sur un serveur web distant, en préfixant avec `http://` ou `https://`.

```
    Vagrant.configure("2") do |config|
        config.vm.provision "shell", path: "script.sh"
    end
```

```
    Vagrant.configure("2") do |config|
      config.vm.provision "shell", path: "https://example.com/provisioner.sh"
    end
```

2.4.2 Ansible

Le provisionner Vagrant Ansible Local permet d'avoir accès à toutes les fonctionnalités Ansible et d'exécuter les playbooks directement sur la machine invitée. Cette méthode ne nécessite pas d'avoir Ansible installé sur le poste de l'étudiant.

On peut installer automatiquement Ansible dans la machine virtuelle à l'aide de :

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"

  # Execute the ansible playbook using ansible_local
  config.vm.provision "ansible_local" do |ansible|
    ansible.playbook = "ansible/install_script.yml"
```

```
    ansible.install_mode = :default
  end
end
```

2.5 Création d'un environnement multi-machines

Dans certains cas, nous souhaitons créer un environnement contenant plusieurs machines. Pour ce faire, ces dernières peuvent être définies dans le même fichier Vagrantfile.

```
Vagrant.configure("2") do |config|
    config.vm.define "web" do |web| # Première machine
        web.vm.box = "apache"
    end

    config.vm.define "db" do |db| # Deuxième machine
        db.vm.box = "mysql"
    end
end
```

2.6 Création et utilisation de l'environnement de laboratoire

Une fois les scripts de provisonning et le fichier Vagrantfile prêts, nous pouvons exécuter l'environnement à l'aide de la commande `vagrant up`.

S'il s'agit de la première utilisation de l'image spécifiée, Vagrant va tout d'abord télécharger l'image sur la machine. Si l'image est déjà disponible sur le poste, elle sera utilisée et permettra un déploiement plus rapide.

Si les machines virtuelles créées sont stockées dans le dossier par défaut de VirtualBox, les « box » se trouvent quant à elles à l'emplacement :

— `~/.vagrant.d/boxes` pour Linux et MacOS

— `C:/Users/USERNAME/.vagrant.d/boxes` pour Windows

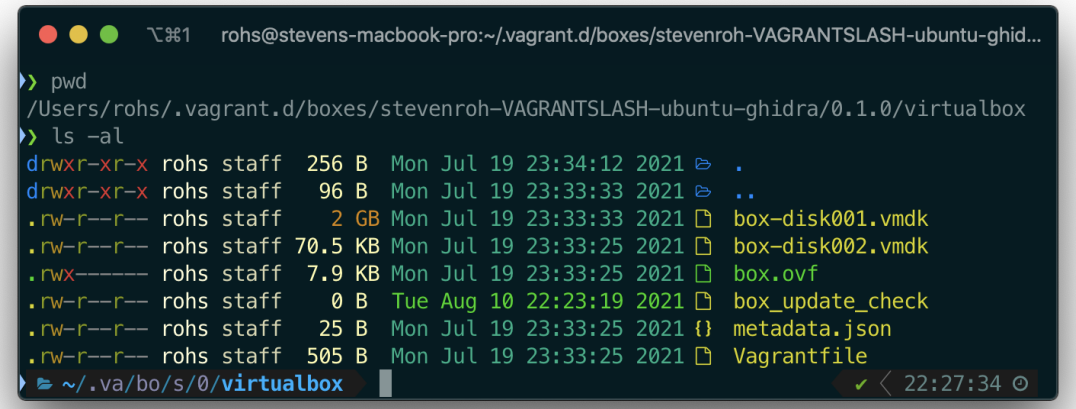


FIGURE *2.1: Affichage d'un dossier contenant tous les fichiers d'une « box »*
Source: *de l'auteur*

Afin de libérer de l'espace disque, l'étudiant pourra exécuter :

```
$ vagrant box prune # Afin de supprimer les anciennes versions de "box"
$ vagrant box remove author/box_name # Afin de supprimer une box spécifique
```

2.7 Partage de l'environnement de laboratoire

Les laboratoires exécutés sont disponibles uniquement en local, sur la machine de l'étudiant. Cependant, l'étudiant peut partager son environnement avec un collègue à l'aide de la commande `vagrant share`. Cette fonctionnalité étant livrée en tant que extension de Vagrant, il sera d'abord nécessaire de l'installer :

```
$ vagrant plugin install vagrant-share
```

A l'aide de la commande `vagrant share`, les services peuvent être exposés sur Internet à l'aide de tunnels « Ngrok » [3].

3. Ngrok est un service qui « expose les serveurs locaux situés derrière des NATs et des pare-feu à l'internet public via des tunnels sécurisés. » (INCONSHREVEABLE, 2021)

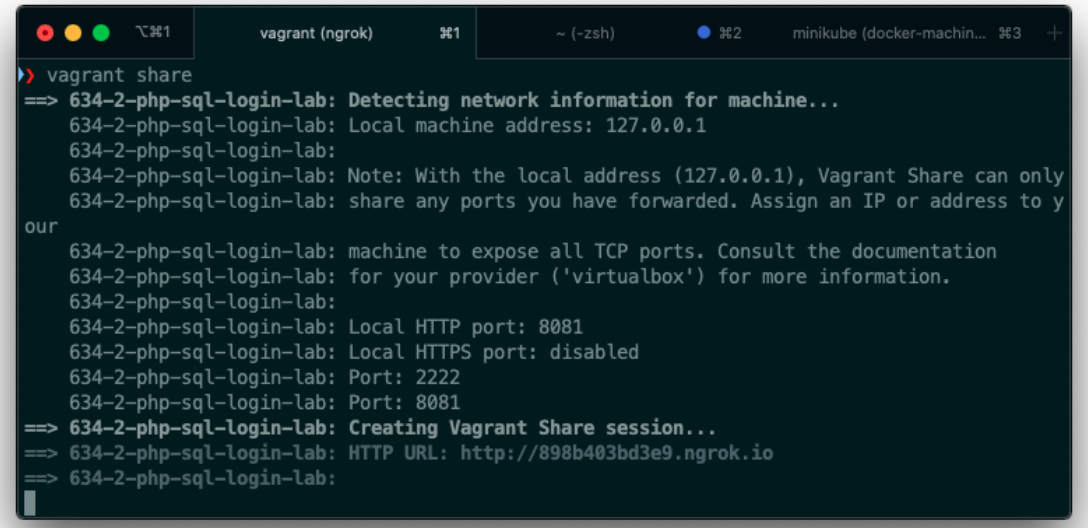


FIGURE *2.2: Partage d'un service exposé avec* `vagrant share`
Source: *de l'auteur*

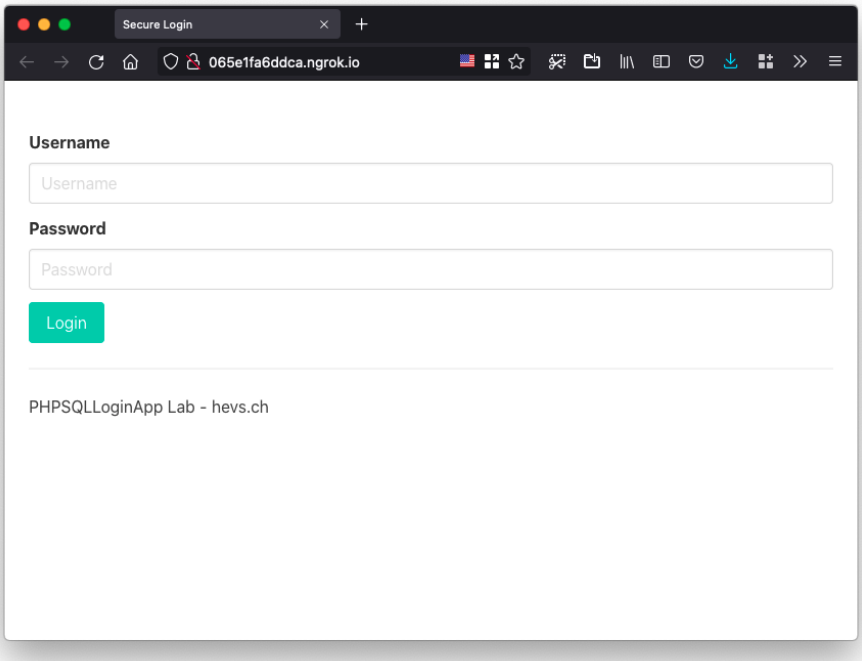


FIGURE *2.3: Accès à un service exposé au travers de Ngrok*
Source: *de l'auteur*

3 | Déploiement dans le Cloud

3.1 Installation de Terraform et de Azure CLI

Avant de pouvoir déployer sur Azure à l'aide de Terraform, il est nécessaire d'installer les outils requis.

Pour cela, nous disposons de plusieurs techniques en fonction du système d'exploitation. Sont supportées Windows, MacOS ou Linux par le biais d'une installation locale ou d'un gestionnaire de paquets (Chocolatey [1] sur Windows, Homebrew [2] sur MacOS ou ceux de la distribution choisie dans le cas de Linux).

Par exemple, pour l'installation sur MacOS :

```
$ brew tap hashicorp/tap
$ brew install hashicorp/tap/terraform

$ brew install azure-cli
```

Ou alors pour Windows :

```
> choco install terraform
> choco install azure-cli
```

Les procédures d'installation, à jour, pour chaque système d'exploitation se trouvent à l'adresse https://learn.hashicorp.com/tutorials/terraform/install-cli pour Terraform et https://docs.microsoft.com/en-us/cli/azure/install-azure-cli pour Azure CLI.

3.2 Connexion avec Azure CLI

Lorsque Terraform CLI est utilisé localement, il est recommandé d'utiliser la connexion via Azure CLI (« Azure Provider: Authenticating via Managed Identity | Guides | hashicorp/azurerm | Terraform Registry », 2021).

Pour réaliser cette connexion, il faut utiliser la commande suivante :

```
$ az login
```

1. Outil libre de gestion de paquets pour Windows https://chocolatey.org/
2. Outil libre de gestion de paquets pour MacOS https://brew.sh/

Ensuite, il sera nécessaire d'ouvrir dans un navigateur web l'URL affichée puis d'y saisir le code fourni. Lorsque la connexion est établie dans le navigateur, Azure CLI affiche les informations du compte utilisateur dans la console.

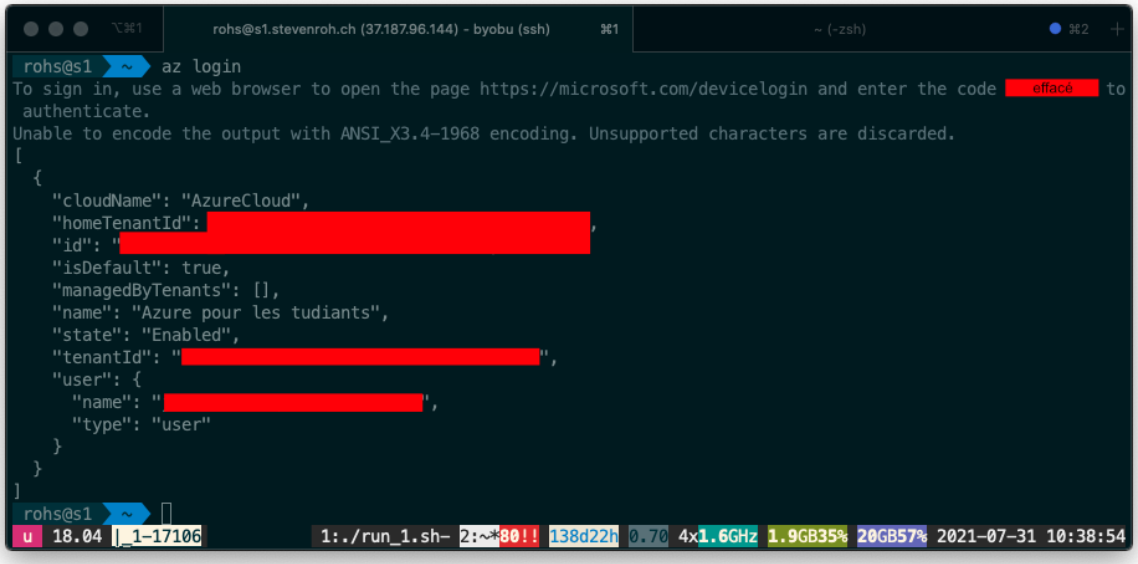


FIGURE *3.1: Connexion de Azure CLI*
Source: *de l'auteur*

Si le compte Azure comporte plusieurs abonnements, la commande suivante permet de les lister. Sans indication contraire, les déploiements se feront dans le compte par défaut, c'est pour cela qu'il est important d'identifier et d'utiliser le bon compte.

```
$ az account list
```

Afin de définir l'abonnement à utiliser par défaut, nous pouvons utiliser la commande suivante en remplaçant « Azure for Students » par le nom (ou l'identifiant) de l'abonnement à privilégier :

```
$ account set --subscription "Azure for Students"
```

3.3 Déploiement d'une machine à partir d'une image existante sur Microsoft Azure

Sur le « Microsoft Azure Marketplace » [3] nous retrouvons plus de 4000 images utilisables.

Pour déployer une image de Ubuntu 18.04-LTS, nous procéderons à la configuration suivante :

```
# Configure the Azure provider
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 2.65"
    }
  }

  required_version = ">= 0.14.9"
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = "STD_rohsworkresourcegroup"
  location = "North Europe"
}

resource "azurerm_public_ip" "ipaddr" {
  name                = "publicip"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  allocation_method   = "Static"
}

resource "azurerm_virtual_network" "virtualnetwork" {
  name                = "virtualnetwork"
  address_space       = ["192.168.2.0/24"]
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_subnet" "subnet" {
  name                 = "internal"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.virtualnetwork.name
  address_prefixes     = ["192.168.2.0/24"]
}

resource "azurerm_network_interface" "public" {
  name                = "nic"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name                          = "remote-access"
```

3. https://azuremarketplace.microsoft.com/

```
    public_ip_address_id          = azurerm_public_ip.ipaddr.id
    private_ip_address_allocation = "Dynamic"
    subnet_id                     = azurerm_subnet.subnet.id
  }
}

resource "azurerm_linux_virtual_machine" "main" {
  name                = var.lab_name
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  size                = "Standard_B1ls"
  admin_username      = "ubuntu"
  admin_password      = "pass123*"

  network_interface_ids = [
    azurerm_network_interface.public.id,
  ]

  admin_ssh_key {
    username   = var.lab_username
    public_key = file("~/.ssh/id_rsa.pub")
  }

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  # Use existing image from Azure
  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }
}

output "public_ip_address" {
  value = azurerm_public_ip.ipaddr.ip_address
  description = "The public IP address of your machine."
}
```

Dans ce fichier, nous pouvons constater qu'une nouvelle infrastructure est créée dans la région `North Europe` dans le groupe de ressources `STD_rohsworkresourcegroup`. L'authentification est réalisée avec l'utilisateur et mot de passe ou à l'aide de la clé `~/.ssh/id_rsa.pub` présente sur le poste de l'étudiant.

Dans le bloc `source_image_reference` de la configuration Terraform nous pouvons constater que la référence de l'image à utiliser est définie.

```
source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
```

```
}
```

Pour terminer, un bloc `output` affichera l'adresse IP de la machine en fin de provisionnement. Cela permet à l'étudiant de s'y connecter ensuite par SSH.

Pour identifier les images disponibles, nous utilisons `az vm image list` :

```
$ az vm image list --output table -all # Récupération de la liste d'images disponibles
$ az vm image list --publisher Canonical --output table -all # Récupération de la liste
↪  d'images disponibles correspondant à l'éditeur Canonical
```

3.4 Création d'images d'environnements pour l'exécution sur Microsoft Azure

3.4.1 Pré-requis

3.4.1.1 Création d'un groupe de ressources

Afin d'accueillir les ressources qui vont suivre, il est nécessaire de disposer d'un groupe de ressources.

FIGURE *3.2: Création d'un groupe de ressources*
Source: *de l'auteur à partir de portal.azure.com*

3.4.1.2 Création d'une galerie d'images partagées

Ensuite, afin de mettre à disposition une image réutilisable sur Microsoft Azure, nous devons procéder à la création d'un lieu de stockage pour cette dernière. Microsoft propose une solution nommée « galerie d'images partagées » (Shared Gallery Image) dans lesquelles nous pouvons déclarer les images, gérer les versions ainsi que définir les règles de partage avec les utilisateurs de l'organisation (Active Directory) ou des personnes externes.

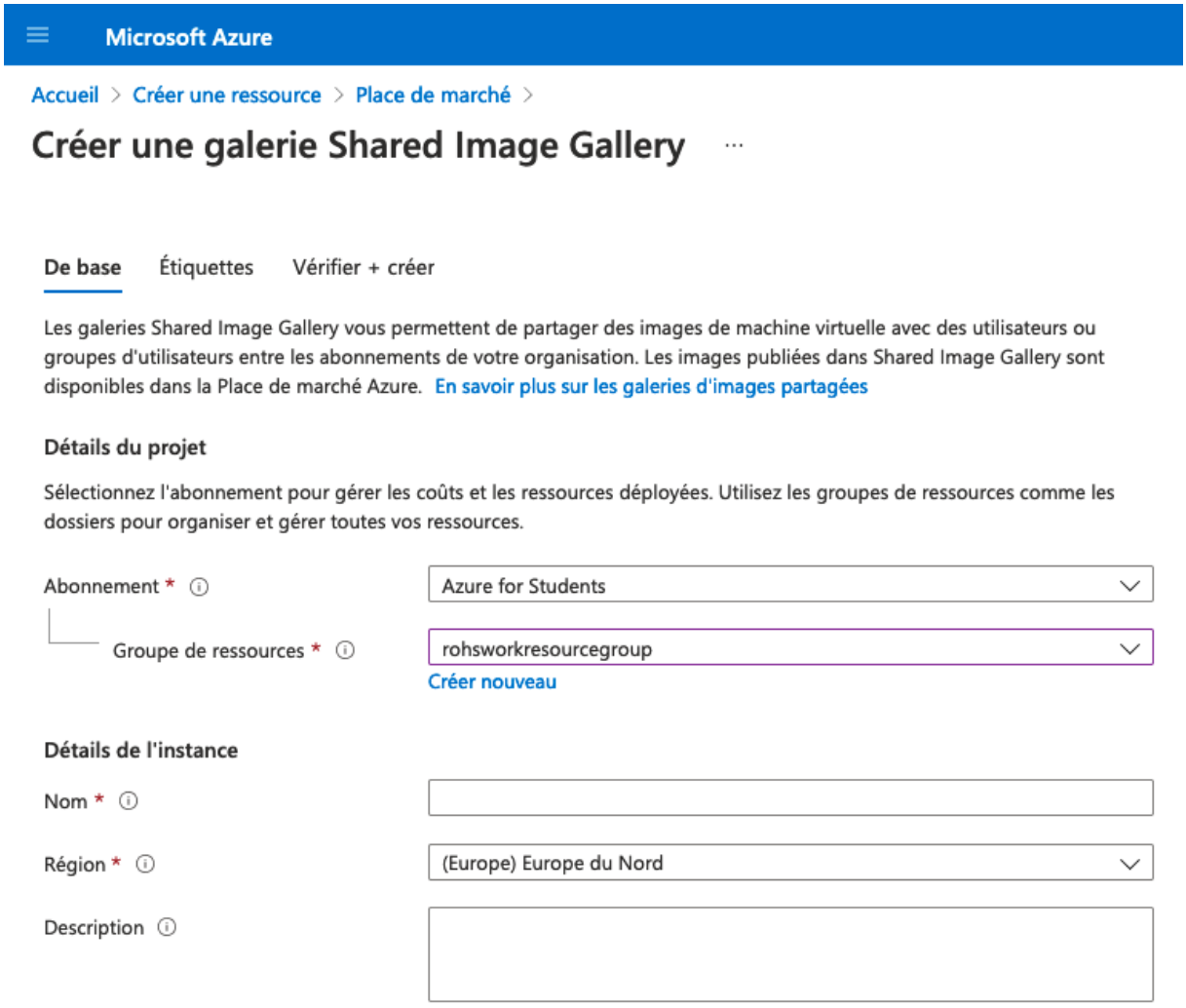


FIGURE *3.3: Création d'une galerie d'images partagées*
Source: *de l'auteur à partir de portal.azure.com*

3.4.1.3 Création d'une définition d'image

Une image est définie dans la galerie d'image à l'aide d'une définition d'image décrite au point 1.4.3.1.

Nous spécifions à cette étape la région dans laquelle sera disponible cette image, le type et l'état du système d'exploitation ainsi que la version de la machine virtuelle (génération compatible).

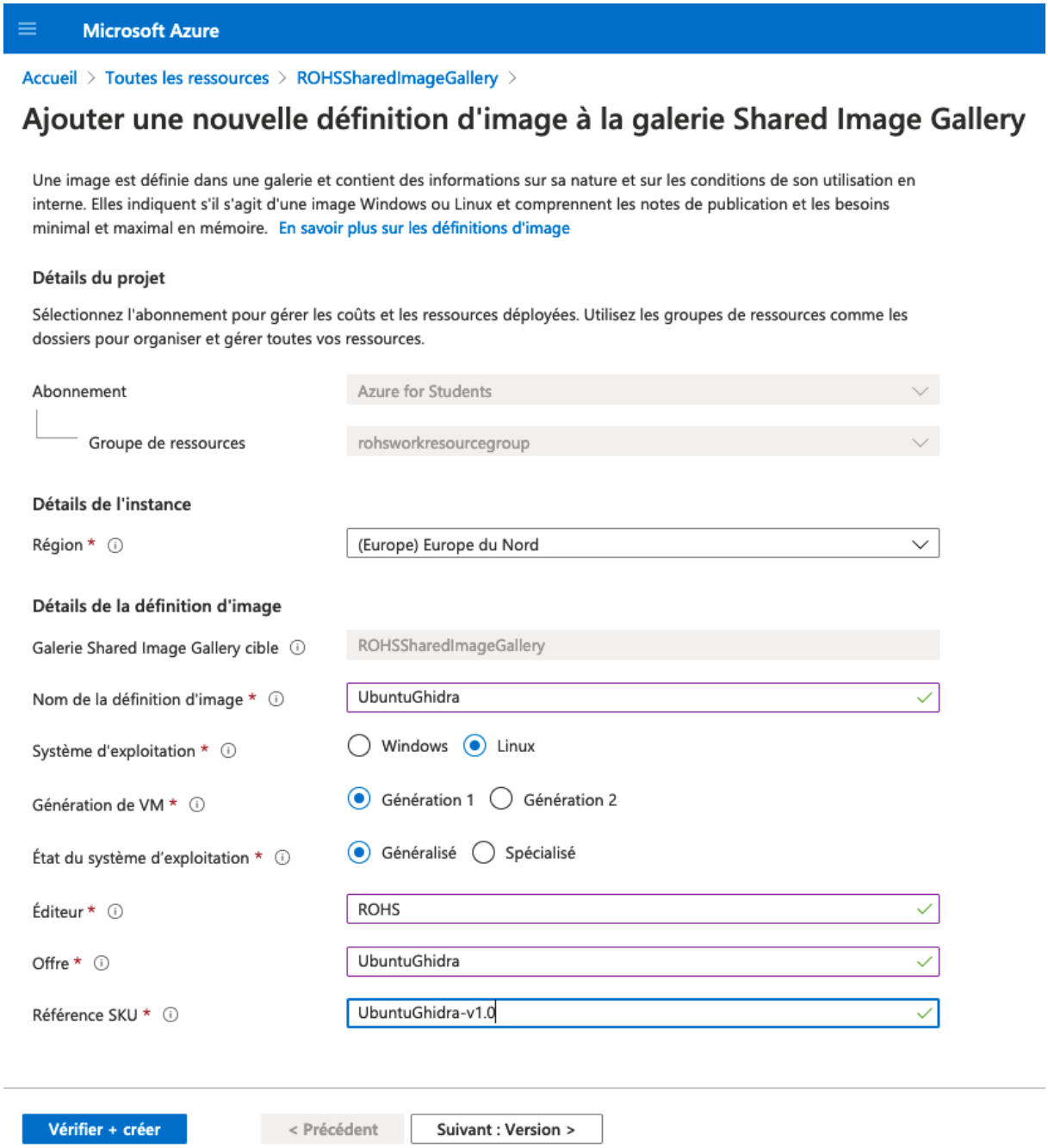


FIGURE *3.4: Création d'une définition d'image dans la galerie d'images partagées*
***Source:** de l'auteur à partir de portal.azure.com*

3.4.2 Définition des règles d'accès

Les règles d'accès définies au niveau de la galerie d'image permettent d'autoriser les utilisateurs à accéder à toutes les images de ladite galerie. Le rôle « Lecteur » est suffisant à l'utilisateur pour afficher la galerie dans son compte Azure et d'utiliser l'image pour ses déploiements.

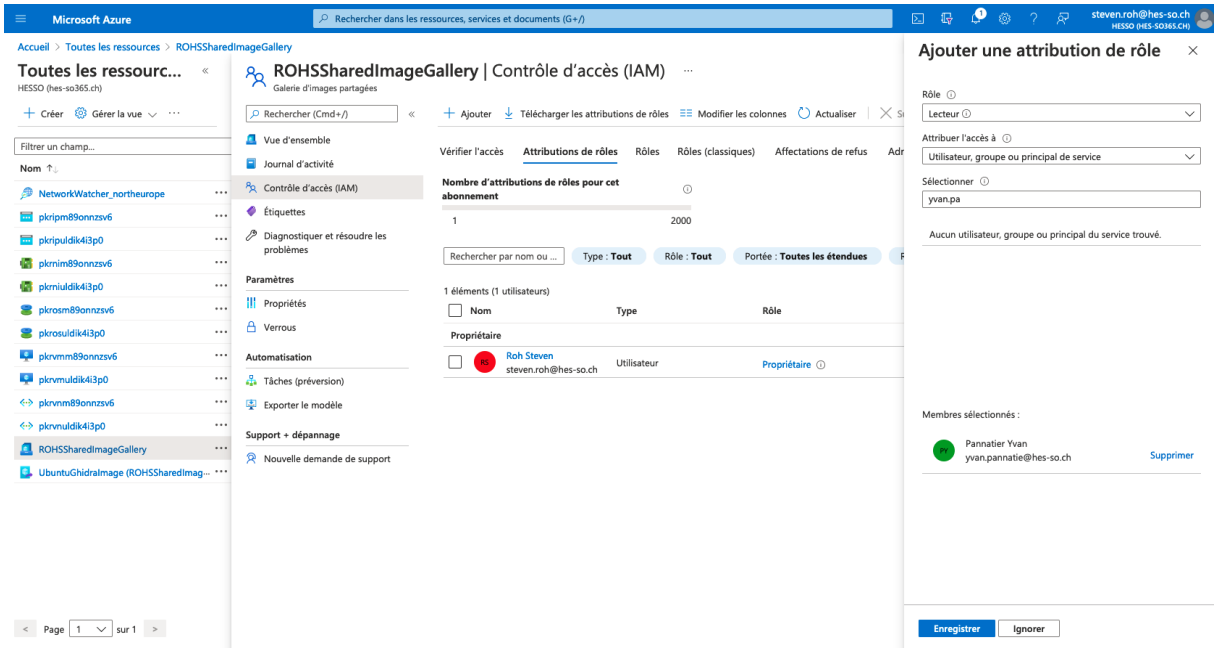


FIGURE *3.5: Attribution de rôles sur la galerie d'images partagées*
Source: *de l'auteur à partir de portal.azure.com*

3.4.3 Création de l'image Azure avec Packer

L'image du laboratoire est construite à l'aide de Packer. Pour cela, le fichier de configuration ci-dessous a été créé.

Dans cette configuration, Packer utilise le « Azure Resource Manager Builder » pour créer et provisionner une machine directement dans Azure et la capture ensuite pour la stocker son image dans la galerie d'images partagées.

Le processus exécuté par Packer correspond à :

1. La récupération de l'image Ubuntu sur Azure.
2. L'installation d'Ansible sur la machine créée.
3. La mise à disposition (copie) des fichiers nécessaires.
4. L'exécution des playbooks Ansible.
5. La sauvegarde de l'image dans la galerie d'images partagées.

```
source "azure-arm" "vmazure" {
    # az account list
    subscription_id = "0fbfd1a9-7aeb-4454-b4b4-d342b9c78ce3"

    managed_image_name = "UbuntuGhidra"
    managed_image_resource_group_name = "rohsworkresourcegroup"

    shared_image_gallery_destination {
        subscription = "0fbfd1a9-7aeb-4454-b4b4-d342b9c78ce3"
        resource_group = "rohsworkresourcegroup"
        gallery_name = "ROHSSharedImageGallery"
        image_name = "UbuntuGhidra"
        image_version = "1.0.0"
        replication_regions = ["North Europe"]
        storage_account_type = "Standard_LRS"
    }

    # from 'az vm image list'
    os_type = "Linux"
    image_publisher = "Canonical"
    image_offer = "UbuntuServer"
    image_sku = "18.04-LTS"

    azure_tags = {
        dept = "seculab"
    }

    location = "North Europe"
    vm_size = "Standard_A1_v2"
}

build {
    sources = ["sources.azure-arm.vmazure"]

    # Install ansible
    provisioner "shell" {
        execute_command = "echo 'vagrant' | sudo -S sh -c '{{ .Vars }} {{ .Path }}'"
        script = "./ansible/ansible.sh"
    }

    # https://github.com/hashicorp/packer/issues/1551#issuecomment-383235951
    provisioner "file" {
        source = "ghidra.desktop"
        destination = "/tmp/ghidra.desktop"
    }

    provisioner "ansible-local" {
        playbook_file = "./ansible/install_ghidra.yml"
    }
}
```

3.5 Création d'un environnement de laboratoire basé sur une image sur-mesure

3.5.1 Définition des variables de l'environnement

Afin d'unifier et de simplifier la création d'environnements, nous pouvons utiliser des variables dans Terraform.

Dans le cas ci-dessous, nous définissons, dans l'ordre :

1. Le nom du laboratoire.
2. L'image (sur-mesure) à utiliser.
3. Le nom de l'utilisateur à créer.
4. Le mot de passe de l'utilisateur précédemment créé.
5. Une option permettant d'utiliser l'authentification par clé ou par mot de passe.

```
variable "lab_name" {
    type        = string
    description = "The practical exercice name"
}

variable "lab_image" {
    type        = string
    description = "The custom image that will be used for the lab"
}

variable "lab_username" {
    type        = string
    description = "Username"
}

variable "lab_password" {
    type        = string
    description = "User password"
}

variable "lab_use_key_auth" {
    type        = bool
    description = "If you want to enable SSH authentication with a key instead of
    ↪  user/password."
    default     = true
}
```

Fichier `variables.tf` contenant les définitions de variables Terraform

Ces variables sont définies dans le fichier de configuration ci-dessous.

```
# Lab definition, do not edit
lab_name = "ubuntu-ghidra"
```

```
lab_image =
↪  "/subscriptions/0fbfd1a9-7aeb-4454-b4b4-d342b9c78ce3/resourceGroups/rohsworkresourcegro
↪  up/providers/Microsoft.Compute/galleries/ROHSSharedImageGallery/images/UbuntuGhidra"

lab_username = "ghidra"
lab_password = "pass123*"
lab_use_key_auth = true
```

Fichier `terraform.tf` contenant les variables du laboratoire Terraform

Le chemin de l'image (variable `lab_image`) provient du portail Azure dans la vue JSON de l'image. Après plusieurs tentatives, il s'agit de la seule technique fonctionnelle permettant de créer une machine basée sur l'image d'une galerie d'images partagées d'un tiers. Malgré plusieurs recherches, une meilleure solution n'a pas été trouvée [4].

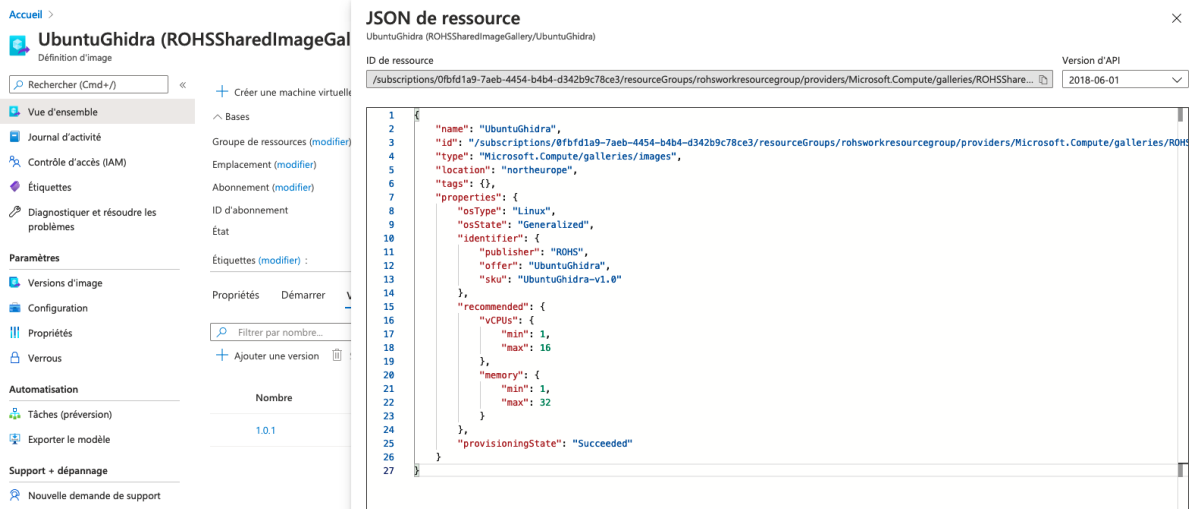


FIGURE *3.6: Récupération du chemin de l'image à partir d'Azure*
Source: *de l'auteur à partir de portal.azure.com*

3.5.2 Exécution

Une fois toute la configuration du provisionnement Terraform spécifiée, le laboratoire peut être inspecté avec la commande :

```
$ terraform plan
```

À la suite de l'exécution de cette commande, sont affichés tous les modifications prévues par l'application des changements dans l'infrastructure.

L'étudiant peut déployer tout l'environnement dans le Cloud à l'aide d'une commande :

```
$ terraform apply
```

4. Problème similaire sur GitHub : https://github.com/terraform-providers/terraform-provider-azurerm/issues/4378

L'adresse IP publique de la machine sera ensuite affichée à la dernière ligne suite à l'exécution de la commande ci-dessus. C'est cette dernière que l'étudiant doit utiliser pour se connecter à son laboratoire.

```
Outputs:
public_ip_address = "137.116.249.138"
```

4 Démonstrateurs

4.1 Laboratoire Keycloak

4.1.1 Objectif

Ce laboratoire déployable en local, dédié à l'outil Identity and Access Management (IAM) Keycloak, permet à l'étudiant de comprendre le fonctionnement de la configuration de cette solution, d'OpenID et d'une connexion SSO.

Deux applications différentes seront configurées afin de permettre l'authentification (une application Python/Django et une application web en JavaScript)

4.1.2 Schéma de fonctionnement du laboratoire

Cet environnement contient plusieurs machines (multi-machines Vagrantfile). La machine Django et la machine Keycloak sont connectées par le biais d'un réseau privé interne.

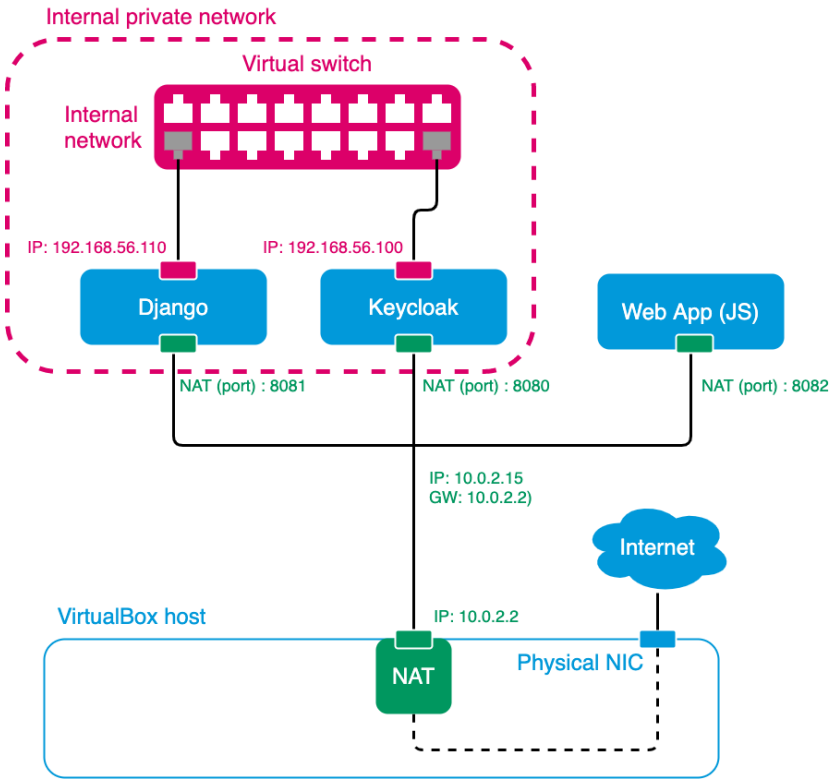


FIGURE *4.1: Schéma de fonctionnement du laboratoire Keycloak*
Source: *de l'auteur*

4.1.3 Configuration de Keycloak

Grâce à cet environnement, l'étudiant dispose de la solution Keycloak installée et prête à l'emploi.

Après avoir configuré un nouveau royaume (Realm), il pourra configurer chacune des deux applications, créer des utilisateurs ou configurer un « authentification provider » compatible (SAML, OpenID, Google, Facebook, LinkedIn, GitHub, GitLab, ...) afin de pouvoir authentifier ses utilisateurs.

L'application doit préalablement avoir été déclarée dans Keycloak. De plus, la propriété *Access Type* doit être modifiée à *Confidential* pour permettre l'authentification de l'application avec Django à l'aide d'identifiants.

Afin d'utiliser les mêmes URLs dans toute la configuration et pour éviter les problèmes liés à l'utilisation d'une adresse IP locale mentionnés au point 4.5.3, le nom de domaine local (`seculab.local`) a été configuré en créant une entrée dans le fichier « hosts » sur la machine hôte (`/etc/hosts` ou `C:\Windows\System32\drivers\etc\hosts`) et dans la machine Django (`/etc/hosts`).

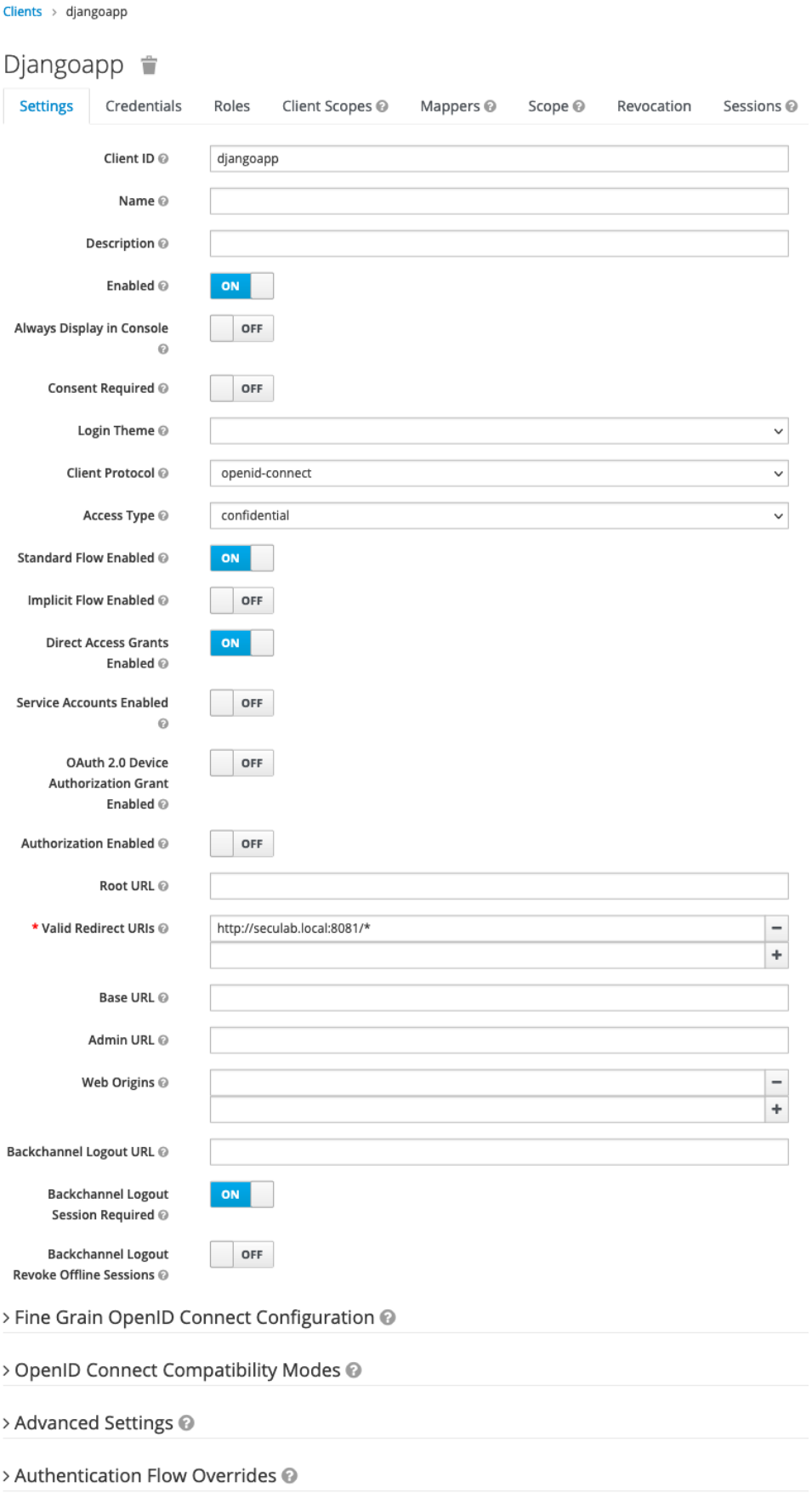


FIGURE *4.2: Déclaration de l'application Django dans Keycloak*

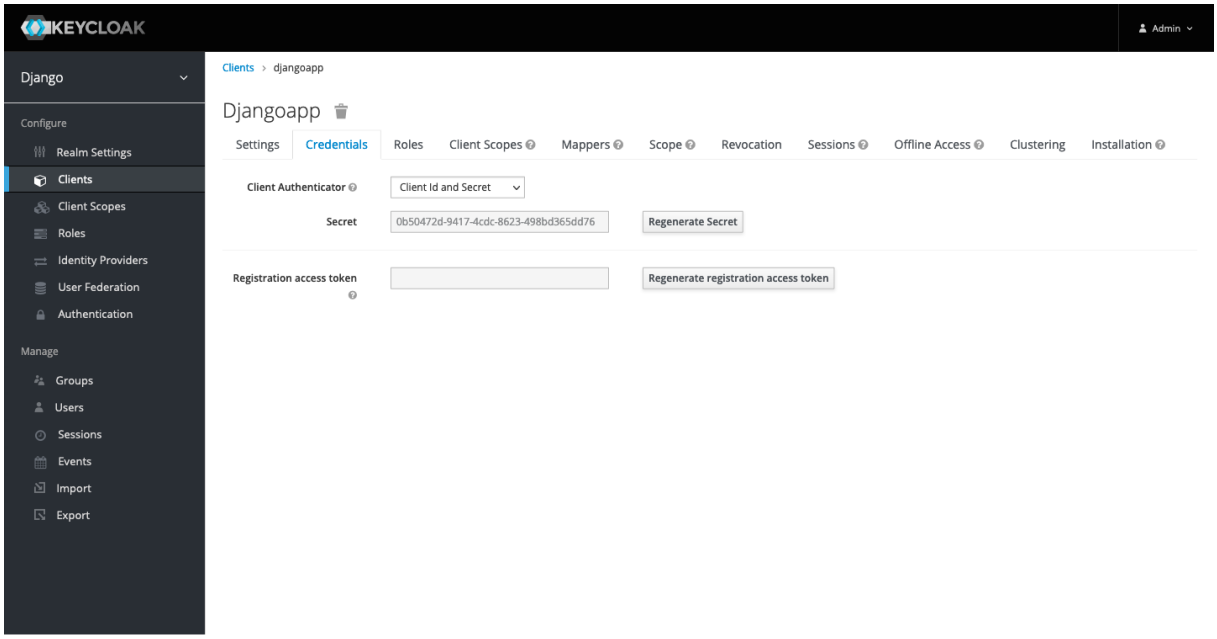


FIGURE *4.3: Récupération des identifiants pour la configuration de l'application Django*
Source: *de l'auteur*

4.1.4 Application Django

Une application Django est fournie dans une seconde machine exécutant Python dans sa version 3 sur Ubuntu. L'application dispose de la bibliothèque `mozilla-django-oidc` pré-installée et pré-configurée. Après quelques étapes d'ajustements de configurations de cette dernière, l'étudiant pourra tester la connexion à Django à travers Keycloak.

En consultant les documentations respectives de Django et de `mozilla-django-oidc`[1], nous pouvons constater que la bibliothèque OIDC s'intègre à Django via un « authentification provider ». Il est possible de l'étendre afin de créer une classe d'authentification sur-mesure, en fonction de ses besoins spécifiques (contrôle des autorisations, attributions de champs spécifiques au profil utilisateur, ...).

Pour la démonstration, le fichier `custom_auth.py` a été créé dans le projet `djangoLoginApp` dans l'application `products` à l'emplacement `djangoLoginApp/products/custom_auth.py`. Ce code étant vraiment simplifié, redondant et non-optimisé, il n'est pas destiné à de la production.

```
1  from mozilla_django_oidc.auth import OIDCAuthenticationBackend
2
3  ADMIN_ROLE = 'superadm'
4
5  class CustomOIDCAB(OIDCAuthenticationBackend):
6      def create_user(self, claims):
7          user = super(CustomOIDCAB, self).create_user(claims)
8
```

1. https://mozilla-django-oidc.readthedocs.io/en/stable/installation.html#quick-start

```
        print('create_user')

        user.first_name = claims.get('given_name', '')
        user.last_name = claims.get('family_name', '')

        user.is_staff = False
        user.is_superuser = False

        roles = claims.get('roles', '')

        if ADMIN_ROLE in roles:
            user.is_staff = True
            user.is_superuser = True

        user.save()
        return user

    def update_user(self, user, claims):
        print('update_user')

        user.first_name = claims.get('given_name', '')
        user.last_name = claims.get('family_name', '')

        user.is_staff = False
        user.is_superuser = False

        roles = claims.get('roles', '')
        if ADMIN_ROLE in roles:
            user.is_staff = True
            user.is_superuser = True

        user.save()
        return user
```

4.1.4.1 Intégration de GitHub en tant qu'Identity Provider

Keycloak se connecte avec d'autres « Identity Provider » en jouant le rôle de « broker » [2]. Les protocoles compatibles sont les suivants :

— SAML v2.0

— OpenID Connect v1.0

— OAuth v2.0

De plus, Keycloak fourni un support clé en main pour certains réseaux sociaux populaires : Google, Facebook, Twitter, Github, GitLab, LinkedIn, Microsoft, StackOverflow, ... (« Server Administration Guide », 2021).

Dans l'exemple suivant, nous pouvons tester l'authentification avec GitHub.

2. Composant permettant de fournir un pont entre de multiples système de gestion d'identité

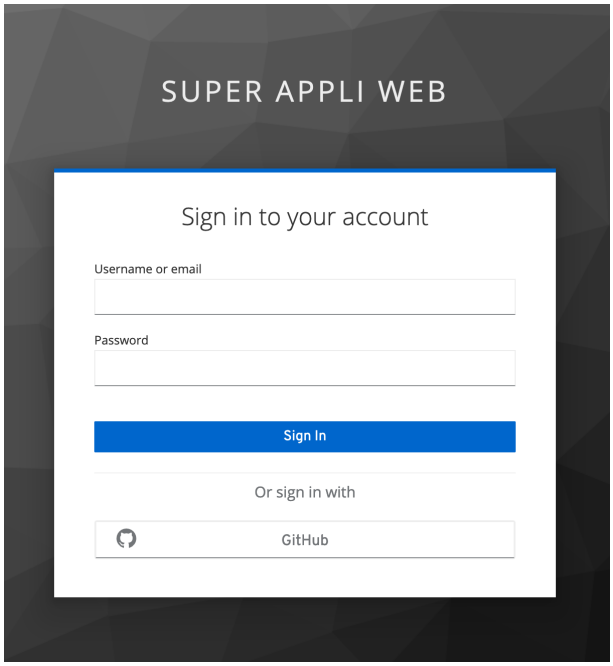


FIGURE *4.4: Formulaire de connexion utilisateur proposant la connexion avec GitHub*
Source: *de l'auteur*

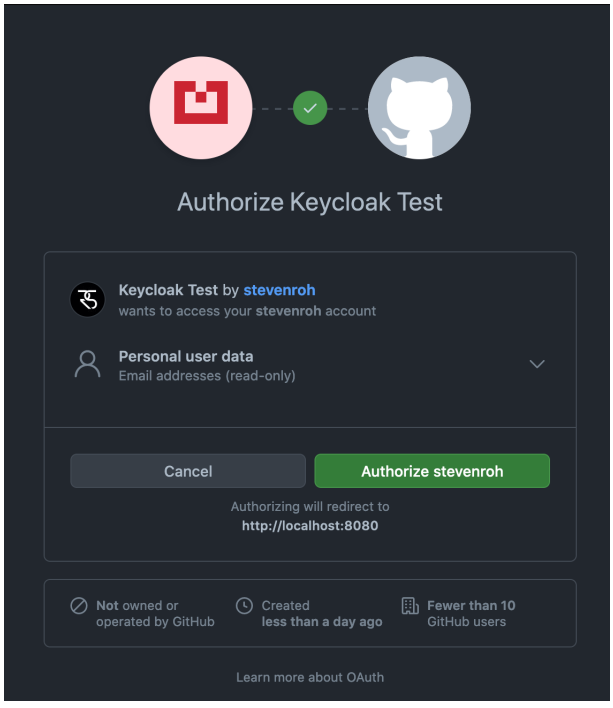


FIGURE *4.5: Demande d'autorisation sur GitHub*
Source: *de l'auteur*

Pour chaque utilisateur connecté avec un profil social, nous retrouvons l'utilisateur correspondant dans la liste des utilisateurs Keycloak.

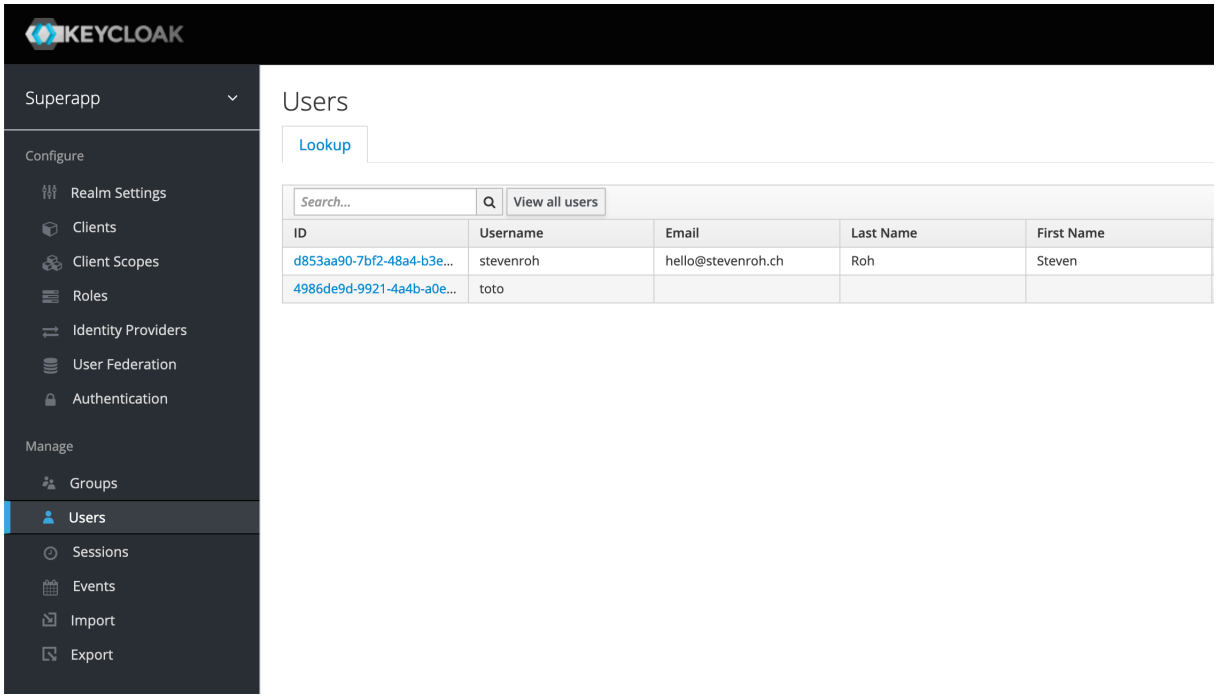


FIGURE *4.6: Keycloak affichant l'utilisateur provenant de GitHub*
Source: *de l'auteur*

4.1.5 Application web JavaScript

Une application web minimale écrite en JavaScript permet également d'expérimenter l'intégration Keycloak et l'authentification côté client.

```
<html>
<head>
    <script src="http://localhost:8080/auth/js/keycloak.js"></script>
    <script>
        var keycloak = new Keycloak();

        keycloak.init({
            onLoad: 'login-required',
        }).then(async function(authenticated) {
            console.log(authenticated ? 'authenticated' : 'not authenticated');

            var info = await keycloak.loadUserInfo();
            console.log(info);
            document.getElementById('info').innerHTML = info.name != undefined ? info.name :
            ↪  info.preferred_username;
            document.getElementById('token').innerHTML = 'Token : ' + keycloak.token;
            document.getElementById('actions').style.display = 'block';
        }).catch(function() {
            alert('failed to initialize');
        });

        var logout = function() {
            console.log('Logout');
            keycloak.logout();
        }
```

```
    </script>
</head>
<body>
    <h1>App content</h1>

    <div id="info"></div>
    <div id="token"></div>

    <div id="actions" style="display:none">
        <button onClick="logout()">Logout</button>
    </div>
</body>
</html>
```

Le client JavaScript peut être chargé directement depuis le serveur keycloak `http://seculab.local:8080/auth/js/keycloak.js`. Le fichier de configuration JSON spécifique à la bibliothèque JavaScript doit se trouver quand à lui à la racine de l'application au même niveau que le fichier `index.html`.

```
{
  "realm": "Django",
  "auth-server-url": "http://seculab.local:8080/auth/",
  "ssl-required": "external",
  "resource": "webappclient",
  "public-client": true,
  "confidential-port": 0
}
```

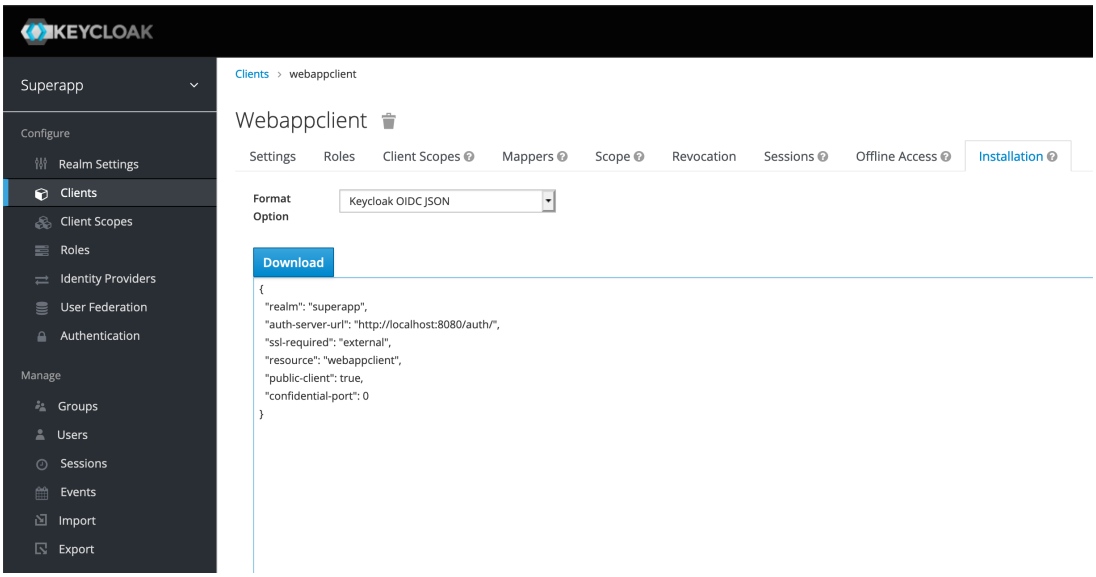


FIGURE *4.7: Fichier de configuration pour l'application web récupéré sur Keycloak*
Source: *de l'auteur*

Pour cette démonstration, le JSON Web Token (JWT) est affiché sur la page une fois connecté. Ce jeton peut ensuite être inspecté par les étudiants avec l'outil JWT.io [3].

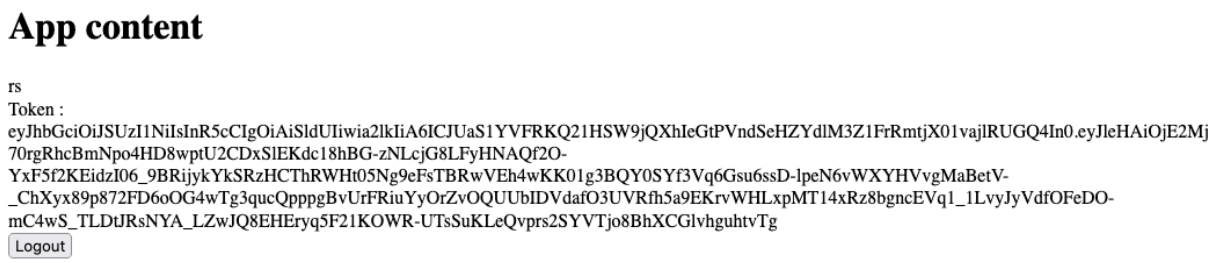


FIGURE *4.8: Application web minimale permettant de tester l'authentification*
Source: *de l'auteur*

3. https://jwt.io/

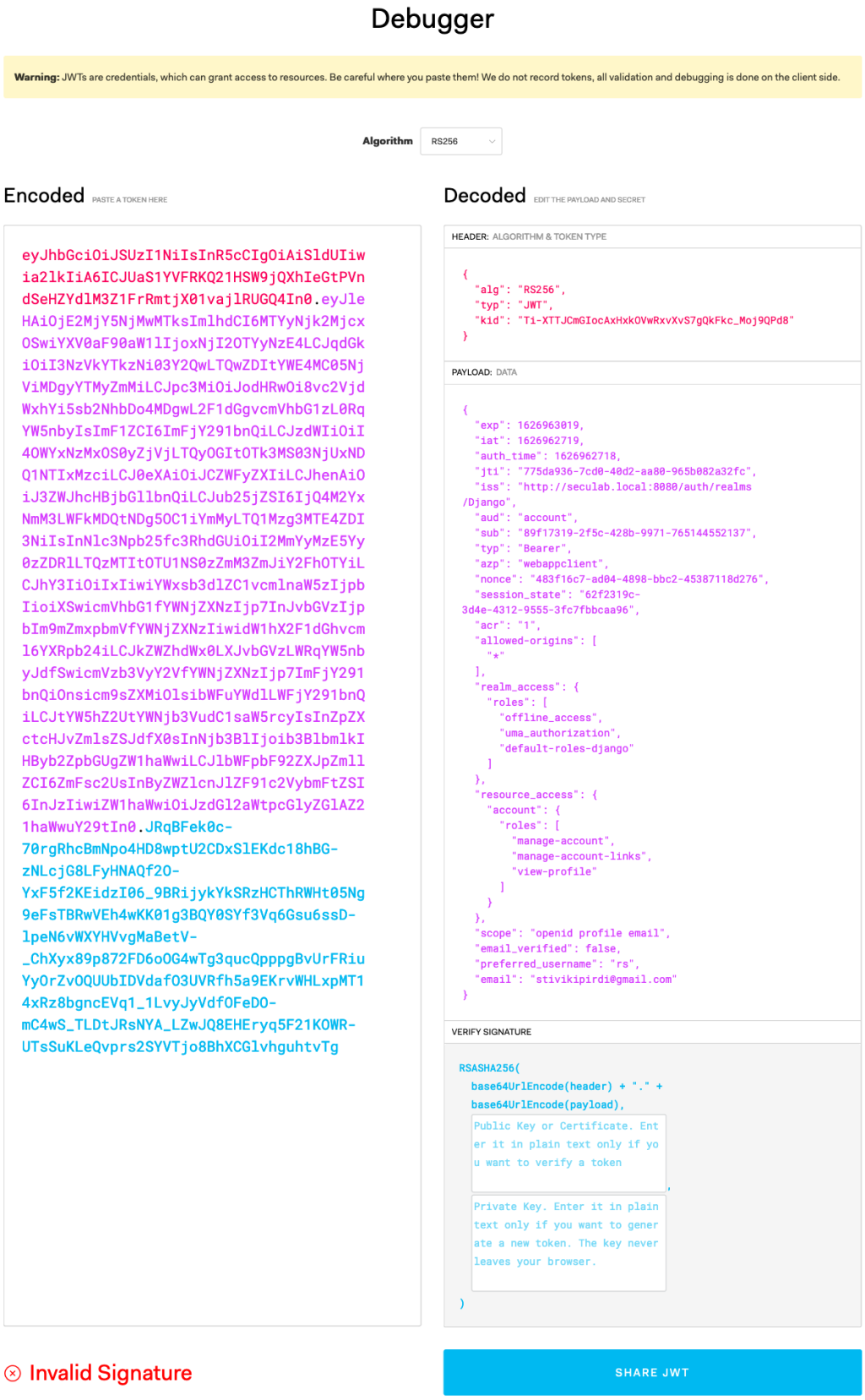


FIGURE *4.9: Capture d'écran du site JWT.io permettant de visualiser un token JWT*
***Source:** de l'auteur à partir de jwt.io*

La bibliothèque JavaScript étant instanciée, il est possible de tester directement à l'aide de la console et du langage JavaScript, l'obtention de données du profil connecté avec la méthode `loadUserInfo`.

FIGURE *4.10: Console du navigateur web affichant les informations utilisateur depuis client JavaScript*
Source: *de l'auteur*

4.1.6 « Mapper »

Dans certains cas, nous souhaitons obtenir des informations sur le profil utilisateur afin de les utiliser dans notre application. La fonctionnalité « Mapper » permet de rendre disponible des attributs personnalisés aux applications via le jeton JWT ou dans le profil utilisateur.

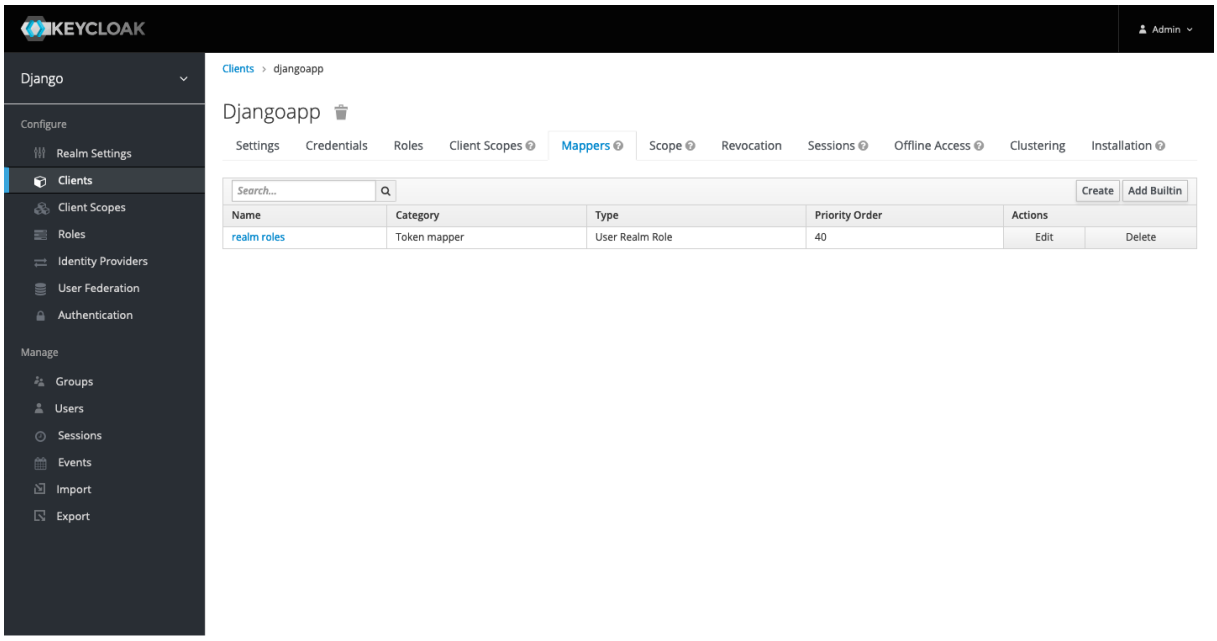


FIGURE *4.11: Liste des « Mappers » dans Keycloak*
Source: *de l'auteur*

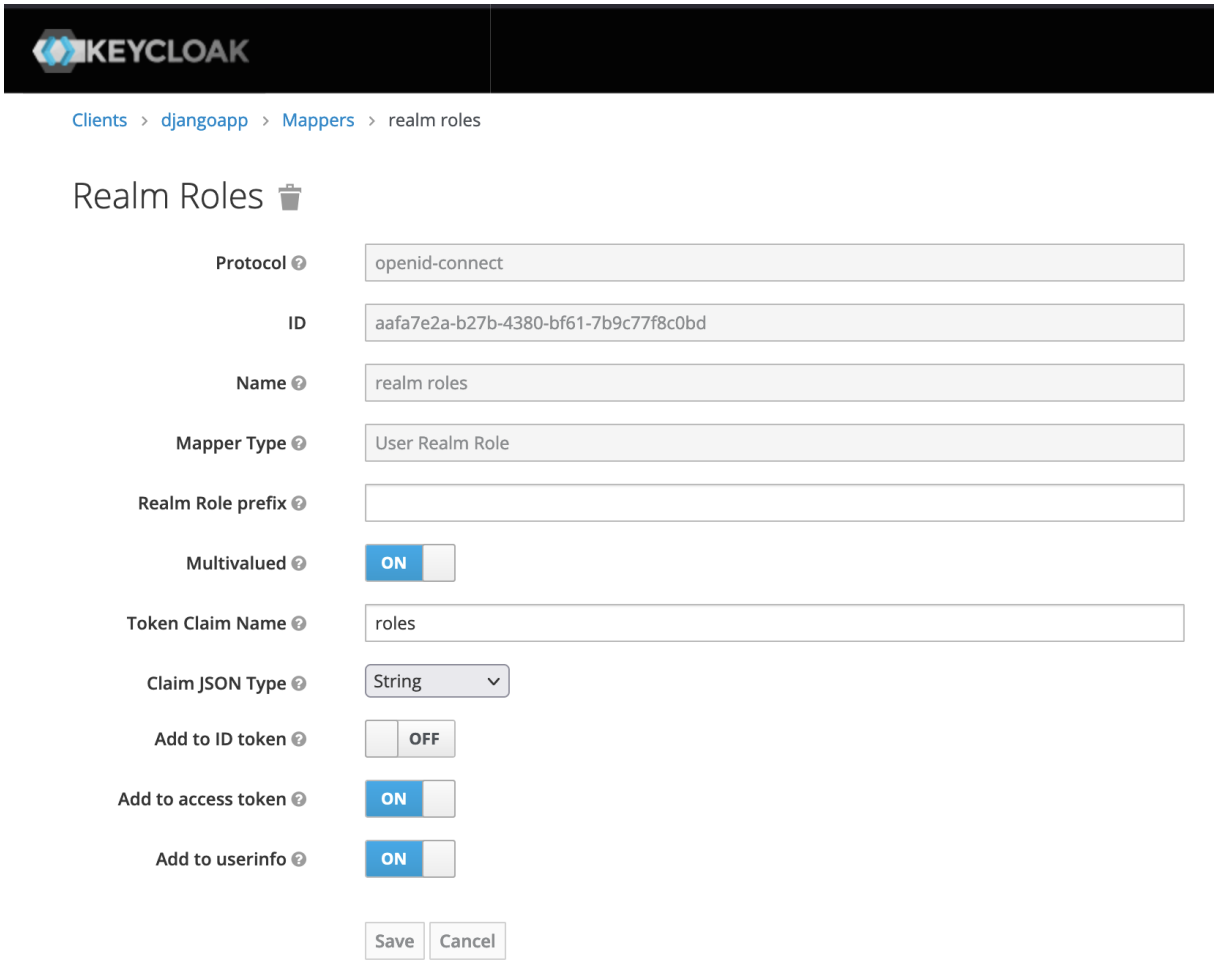


FIGURE *4.12: Configuration d'un « Mapper » permetant de récupérer le rôle utilisateur*
***Source:** de l'auteur*

4.2 Laboratoire Cuckoo

4.2.1 Objectif

« Cuckoo Sandbox » est un outil d'analyse de logiciels malveillants (malware) exécutant les fichiers de manière isolée dans une sandbox. Le logiciel est libre et le code source est disponible sur GitHub [4].

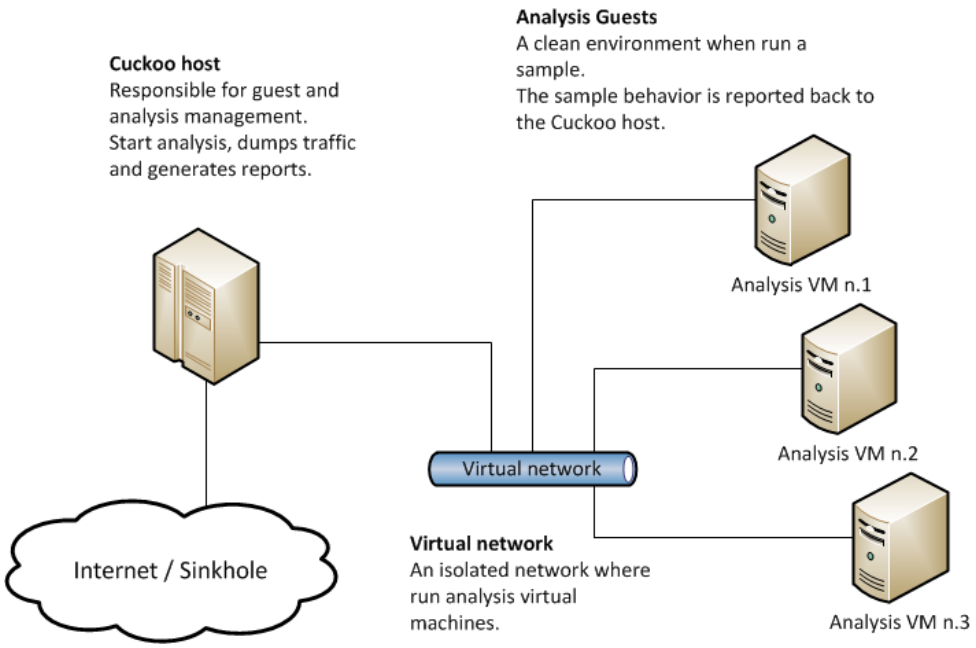


FIGURE *4.13: Schéma de fonctionnement de l'environnement Cuckoo*
Source:
https:// medium.com/ @warunikaamali/ cuckoo-sandbox-installation-guide-d7a09bd4ee1f

Cuckoo était déjà proposé par le professeur Jean-Luc Beuchat dans les laboratoires de détection d'intrusions. La dernière version du projet n'a plus été mis à jour depuis 2018 et est écrit en Python version 2, obsolète depuis le 1er janvier 2020. Finalement, lors de ce travail, le projet Cuckoo a été archivé sur GitHub.

4. https://github.com/cuckoosandbox/cuckoo

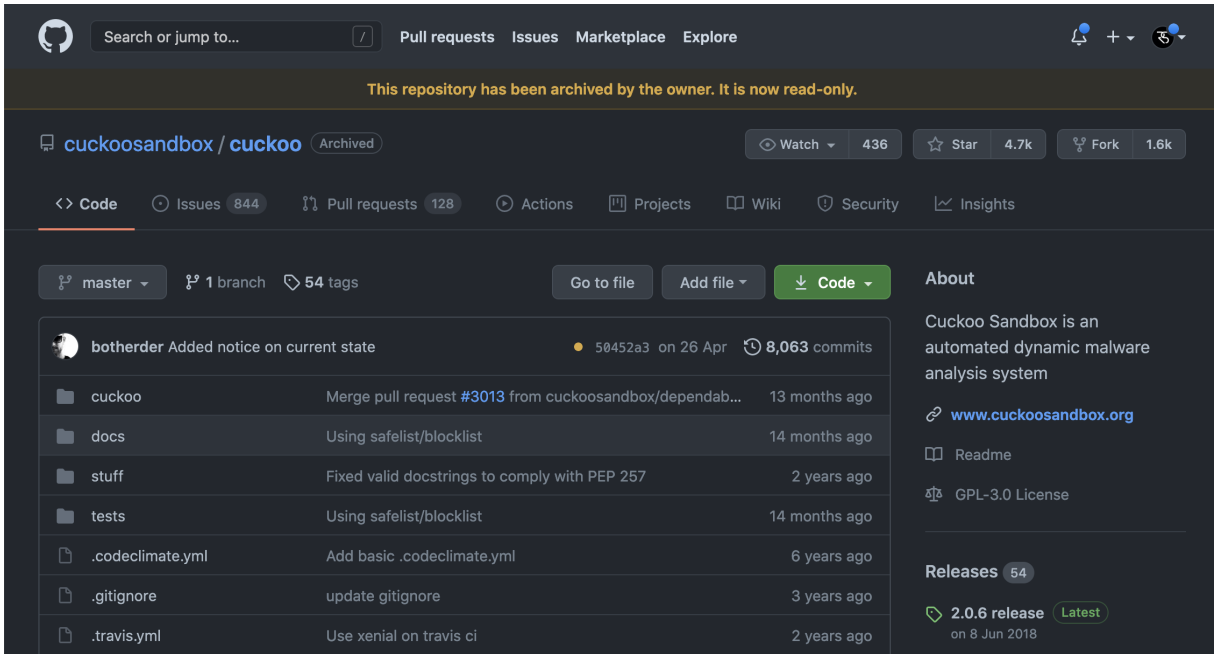


FIGURE *4.14: Dépôt GitHub du projet Cuckoo Sandbox archivé*
Source: *de l'auteur à partir de github.com*

Plusieurs indications mentionnent une future version 3 de Cuckoo [5]. Elle sera rendue disponible après une réécriture complète en Python 3 [6].

En attendant la future mise à jour, d'autres solutions peuvent être proposées aux étudiants :

— CAPEv2 [7]

— Any.run [8]

— Hybrid Analysis [9]

— Joe Sandbox [10]

Sur le blog de Zeltser, nous pouvons également consulter une liste d'alternatives à l'adresse https://zeltser.com/automated-malware-analysis/ (« Free Automated Malware Analysis Sandboxes and Services », 2021).

5. « Cuckoo 3 is still in development. This will part of its namespace. »https://pypi.org/project/Cuckoo3/
6. https://hatching.io/cuckoo/
7. https://github.com/kevoreilly/CAPEv2/
8. https://any.run/
9. https://www.hybrid-analysis.com/
10. https://www.joesecurity.org/joe-sandbox-cloud

4.3 Laboratoire SQL Injection

4.3.1 Objectif

L'objectif de ce laboratoire est de sensibiliser les étudiants à la problématique de l'injection SQL. L'environnement du laboratoire contient une application web écrite en PHP avec une base de donnée MariaDB. Un formulaire de connexion, vulnérable aux injections SQL, permet de s'authentifier.

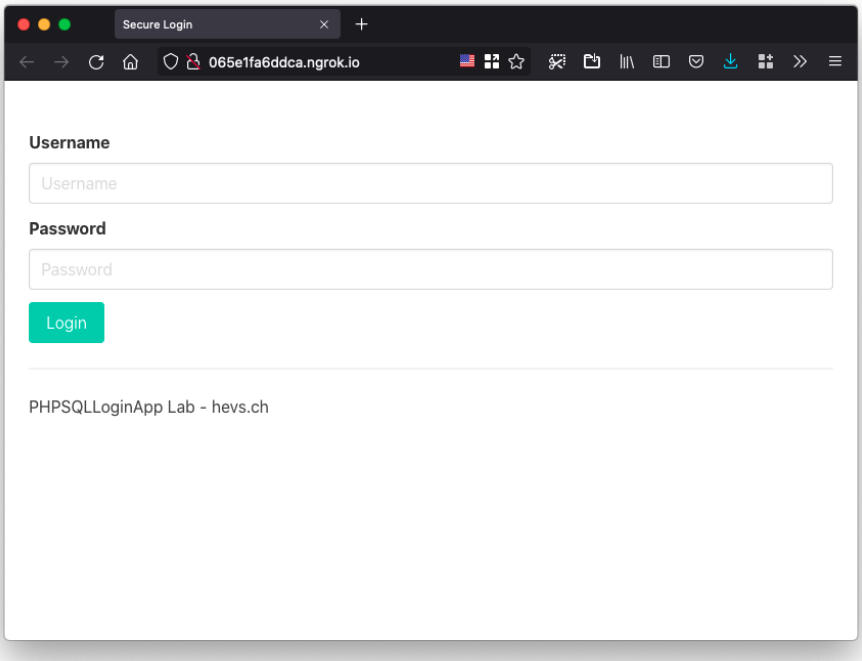


FIGURE *4.15: Formulaire de connexion vulnérable aux injections SQL*
Source: *de l'auteur*

Le code conçu pour cette application comporte une vulnérabilité à la ligne 34 :

```
1  <?php
2
3  $db = mysqli_connect('127.0.0.1', 'lab', 'lab', 'secure_login', '3306');
4
5  if (mysqli_connect_errno()) {
6      echo "Error establishing database connection";
7      echo mysqli_connect_error();
8      exit();
9  }
10 ?>
11
12 <!DOCTYPE html>
13 <html lang="en">
14         <head>
15                 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
16                  <title>Secure Login</title>
17          <link rel="stylesheet"
            ↪  href="https://cdn.jsdelivr.net/npm/bulma@0.9.2/css/bulma.min.css">
18          <meta name="viewport" content="width=device-width, initial-scale=1">
19      </head>
20          <body>
21
22          <section class="section">
23              <div class="container">
24
25                  <?php
26                      if(isset($_POST['login'])) {
27                          $username = $_POST['username'];
28                          $username = stripslashes( $username );
29
30                          $pass = $_POST['password'];
31                          $pass = stripslashes($pass);
32                          $pass = md5($pass);
33
34                          $query  = "SELECT * FROM `users` WHERE username='$username' AND
                            ↪  password='$pass';";
35                          // echo $query;
36
37                          $result = mysqli_query($db, $query) or die(mysqli_error($db));
38
39                          if( $result && mysqli_num_rows( $result ) == 1 ) {
40                              // Login Successful...
41                              echo "Congrats, you have logged in ! <br/>";
42                              echo "Welcome to the admin area ! <br/>";
43
44                              echo '<iframe src="https://giphy.com/embed/g9582DNuQppxC"
                                ↪  width="480" height="270" frameBorder="0"
                                ↪  class="giphy-embed" allowFullScreen></iframe>';
45
46                              die();
47                          } else {
48                          ...
```

Après avoir étudié le fonctionnement d'une faille d'injection SQL, l'étudiant pourra tenter de se connecter à l'application en tant qu'administrateur, sans avoir connaissance du mot de passe. Il pourra par exemple utiliser :

```
1  admin' #
2  admin' or 1=1 --␣
```

Il pourrait également corriger cette faille en modifiant le code source dans le dossier `www/` monté sur sa machine hôte grâce à Vagrant.

FIGURE *4.16: Exploitation de l'injection SQL*
Source: *de l'auteur*

4.4 Laboratoire Crowdsec

4.4.1 Objectif

L'objectif de ce laboratoire est de configurer le logiciel CrowdSec afin de détecter et de bloquer les tentatives d'intrusions sur une machine dédiée. L'étudiant doit configurer et vérifier l'acquisition et le « parsing » des journaux de connexion (logs). Il doit également installer les scénarios permettant de détecter des attaques sur le serveur web. Finalement, la configuration du « bouncer » permettra de bloquer l'adresse IP après la détection de l'attaque.

4.4.2 Fonctionnement

CrowdSec est un logiciel libre de type Endpoint Detection and Response (EDR) visant à identifier et partager les adresses IP malveillantes attaquant un système.

CrowdSec est inspiré du très connu Fail2Ban [11]. Ce nouvel outil est conçu pour protéger les serveurs ainsi que leurs services exposés sur Internet. Selon le site Developpez.com, l'outil de prévention des intrusions est présenté comme une version modernisée et collaborative de Fail2Ban (BRUNO, 2021).

Une fois installé CrowdSec observe les logs en temps réel et de manière rétroactive. Les différents *parsers* permettent d'extraire et de détecter les différentes informations essentielles dans les journaux. La plateforme peut enrichir automatiquement les données avec des informations complémentaires telles que le code de pays, la latitude, la longitude et les informations sur l'adresse IP via une fonctionnalité GeoIP [12].

Comme il s'agit d'une solution communautaire, les adresses IP malveillantes sont transmises et récupérées en communiquant avec le serveur central nommé CAPI.

11. Fail2ban est un outil de prévention d'intrusions écrit en Python
12. Outil et technique permettant de déduire approximativement la position géographique d'un appareil à partir de son adresse IP

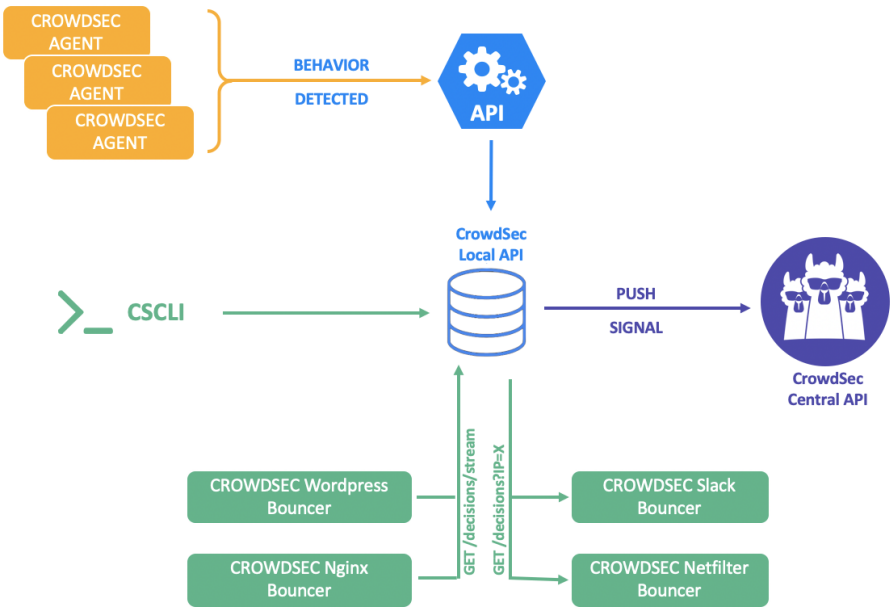


FIGURE *4.17: Schéma de fonctionnement de CrowdSec et des APIs*
Source: *https:// doc.crowdsec.net/ Crowdsec/ v1/*

4.4.3 Composants

4.4.3.1 LAPI

« LAPI » est l'API Crowdsec locale. Elle permet aux machines configurées avec l'agent `crowdsec-agent` d'ajouter des alertes et décisions dans la base de données. Les machines s'authentifient avec une combinaison login-password et peuvent lire et créer des décisions.

Elle permet également au bouncer d'avoir accès aux alertes et décisions afin de bloquer le trafic en fonction du scénario. Les bouncers sont authentifiés à l'aide d'une clé d'API en lecture seule.

4.4.3.2 CAPI

L'API Central de Crowdsec (CAPI) est disponible à l'adresse https://api.crowdsec.net. C'est cette dernière qui permet le partage collaboratif d'adresses malveillantes reportées par toutes les installations de Crowdsec.

Le report d'informations envoyées à la Central API contient uniquement les méta-données de l'attaque tels que :

— L'adresse IP de l'attaquant.

— Le nom du scénario.

— La date de début et de fin de l'attaque.

4.4.3.3 Outil cscli

La commande `cscli` permet d'interagir avec le service CrowdSec et installer des parsers, des collections et des scénarios.

- `cscli collections [action]`
  Permet d'installer, de lister, de mettre à jour et d'inspecter une collection.
- `cscli parsers [action]`
  Permet d'installer de lister, de mettre à jour et d'inspecter les parsers.
- `cscli scenarios [action]`
  Permet d'installer, de lister, de mettre à jour et d'inspecter les scénarios.
- `cscli dasboard [action]`
  Permet de configurer, de démarrer ou d'arrêter le tableau de bord.
- `cscli decisions [action]`
  Permet d'afficher, d'ajouter ou de supprimer les décisions.
- `cscli metrics [action]`
  Permet d'afficher les statistiques de crowdsec.

4.4.3.4 Tableau de bord Metabase

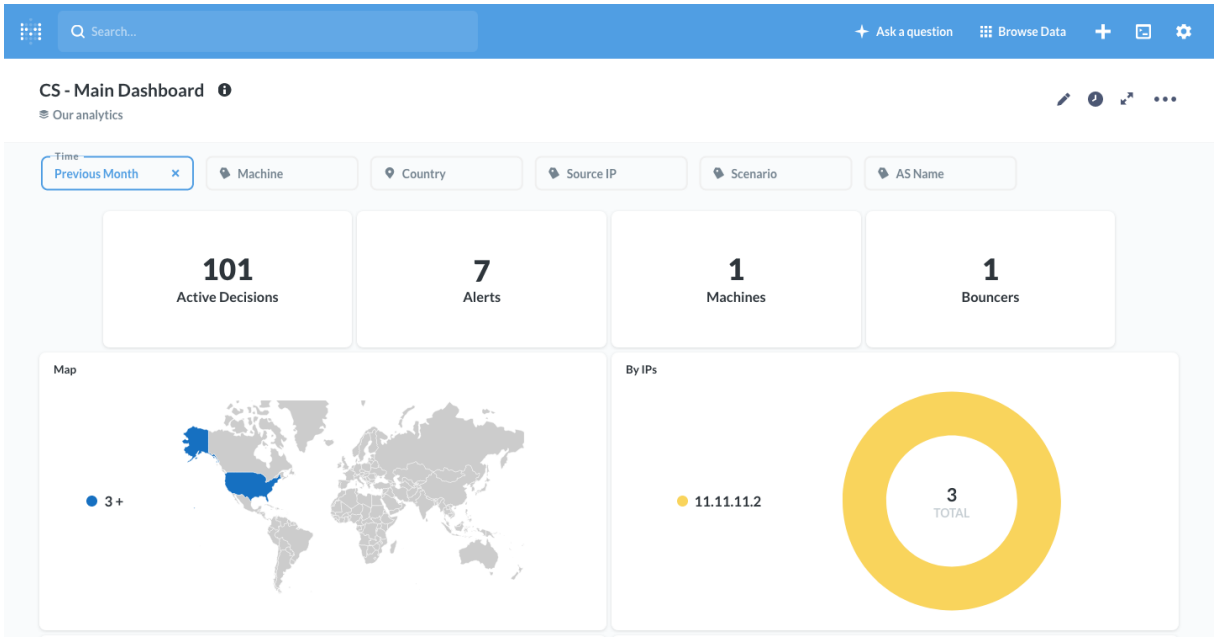


FIGURE *4.18: Capture d'écran du tableau de bord Metabase*
***Source:** de l'auteur*

Crowdsec inclut un tableau de bord basé sur l'outil Metabase exécuté à l'aide d'un container Docker.

FIGURE *4.19: Capture d'écran du tableau de bord Grafana*
Source: *https:// doc.crowdsec.net/ Crowdsec/ v0/ observability/ prometheus/*

Géré à l'aide de la commande `cscli`, il peut être configuré avec `cscli dashbord setup` et démarré avec `cscli dashbord start`.

Metabase permet d'afficher les informations sur les attaques reçues/bloquées et les machines gérées à l'aide de graphiques simples. Metabase étant un logiciel libre de « Business Intelligence », les fonctionnalités sont limitées.

4.4.3.5 Tableau de bord Grafana

Grafana, l'outil libre de visualisation de données, permet également d'afficher les données d'un serveur Prometheus [13] dans des tableaux de bord personnalisables. Cette configuration permet une meilleure visualisation et une meilleure flexibilité que Metabase. Grafana n'est pas intégré dans ce laboratoire mais pourrait être installé par l'étudiant dans le cadre d'un travail pratique.

Le fichier Vagrantfile permet de créer l'environnement complet permettant à l'étudiant d'étudier Crowdsec. Il est composé de deux machines (Attacker, Target). Lorsqu'il est déployé sur le Cloud, seule la machine cible est nécessaire. Les attaques peuvent être réalisées avec le poste local.

13. Logiciel libre de surveillance de systèmes informatiques.

FIGURE *4.20: Capture d'écran du tableau de bord Grafana*
Source: *https:// doc.crowdsec.net/ Crowdsec/ v0/ observability/ prometheus/*

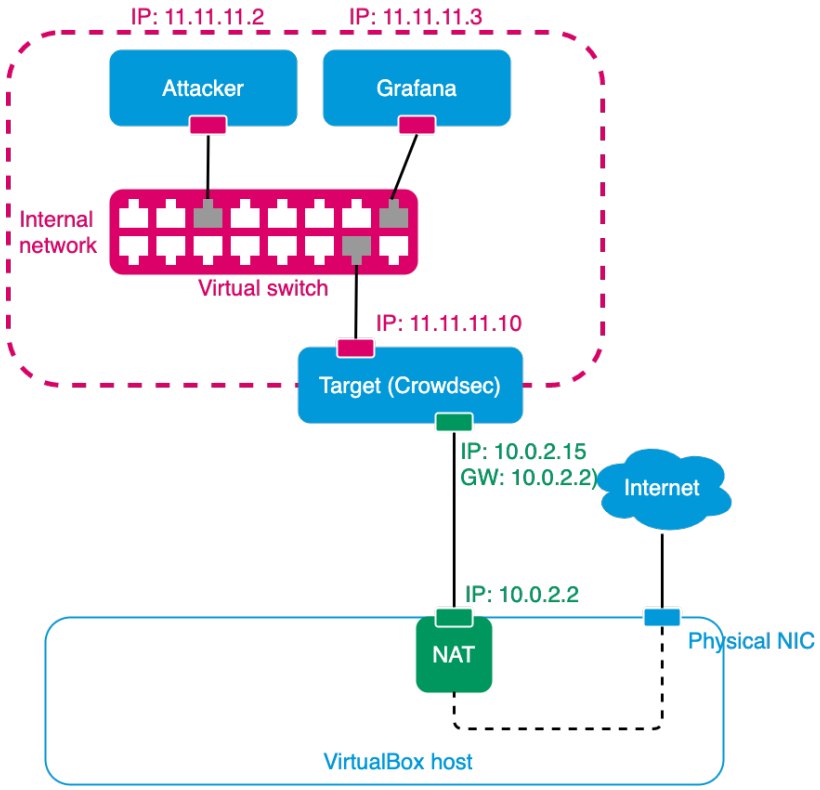


FIGURE *4.21: Schéma de fonctionnement du laboratoire CrowdSec*
Source: *de l'auteur*

La machine attaquante de ce laboratoire (`attacker`) contient un script écrit en Python 3 permettant d'effectuer une attaque par force brute sur le formulaire de connexion utilisateur. Cette dernière comprend également les outils `wapiti` [14] et `nikto` [15]. L'utilisation de ces derniers donne la possibilité aux étudiants de réaliser un audit de sécurité d'une application et du serveur web. Les vulnérabilités découvertes (Injections SQL, Injections de commandes, Cross Site Scripting (XSS), Xml eXternal Entity (XXE), ...) seront listées dans un rapport.

4.4.4 Scénarios

Afin de protéger les attaques du serveur web (HTTP), nous pouvons installer une collection de scénarios nommée `base-http-scenarios` permettant de détecter :

— Les robots agressifs (aggressive crawl).
— Les attaques par sondage (scanning/probing).
— Les « User-Agent » malicieux.
— Les attaques par traversée de répertoire (path traversal).
— Les tentatives d'accès à des données sensibles.
— Les tentatives d'injections SQL.
— Les attaques par force brute des authentifications basiques HTTP.

L'installation de cette collection se réalise à l'aide de la commande :

```
1 $ cscli collections install crowdsecurity/base-http-scenarios
```

Le scénario qui nous intéresse particulièrement est `http-generic-bf` car il permettra de bloquer l'attaque par force brute sur la connexion. Ce dernier est inclus dans la collection proposée ci-dessus.

4.4.5 Parsers

Pour pouvoir lire et comprendre les journaux du serveur web Apache ou nginx, nous utilisons `http-logs`.

Ce dernier est également intégré dans la collection ci-dessus.

La prochaine étape consiste à configurer le fichier de configuration qui définit les journaux à acquérir.

```
1 filename: var/log/apache2/*.log
2 labels:
```

14. https://wapiti.sourceforge.io/
15. https://github.com/sullo/nikto

```
3     type: apache2
4   ---
5   filename: var/log/nginx/*.log
6   labels:
7     type: nginx
8   ---
```

Fichier `/etc/crowdsec/acquis.yaml`

4.4.6 Bouncer

Pour que la machine attaquante soit bloquée, nous devons installer un « bouncer ». Un bouncer est un logiciel indépendant chargé d'agir suite à une décision de Crowdsec.

Ces différents logiciels interagissent avec la LAPI de Crowdsec, authentifiés à l'aide d'une clé d'API.

Il existe des « bouncers » pour les firewalls (iptables, nftables, ipset et pf), pour les applications et frameworks express.js et WordPress et pour les serveurs web Caddy et nginx. Nous pouvons également en créer de nouveaux sur-mesure.

Pour la mise en place de ce laboratoire, le « bouncer » pour *firewall* a été installé. Il permettra de rejeter les paquets réseaux des machines bloquées avec « iptables ».

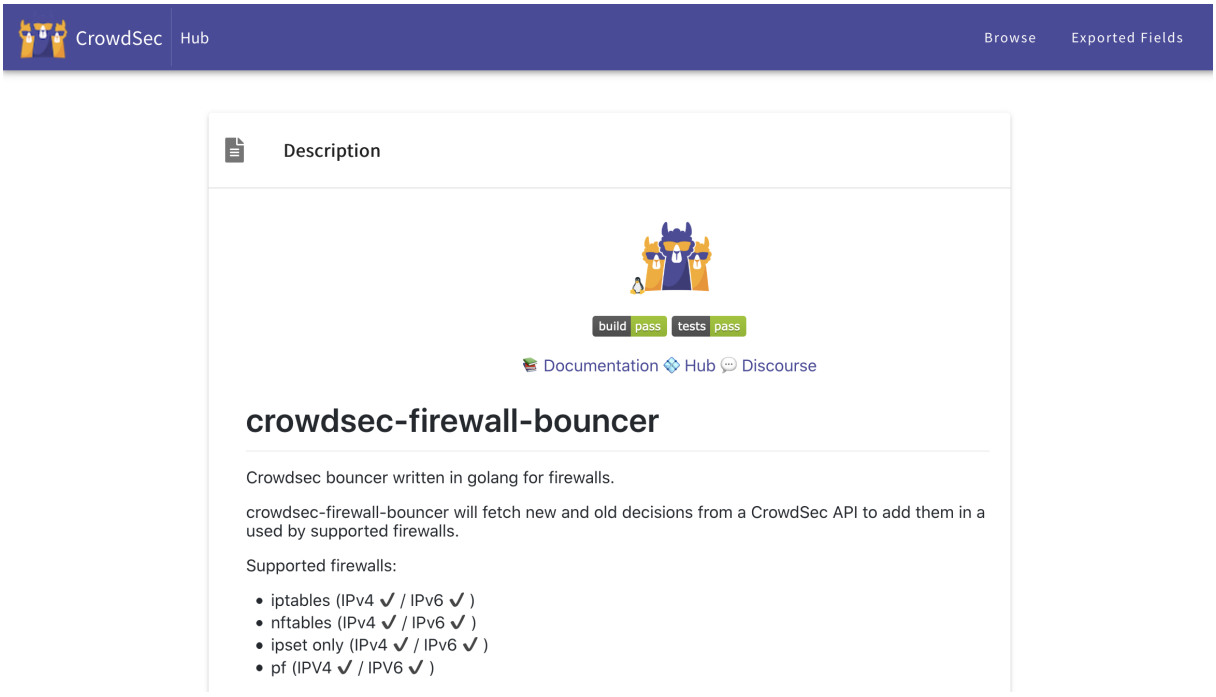


FIGURE *4.22: Bouncer Firewall sur le Crowdsec Hub*
Source: *de l'auteur à partir de hub.crowdsec.net*

4.4.7 Tests d'intrusions

Dès que l'installation et la configuration est terminée, l'étudiant peut lancer le script python `http_bruteforce_login_attack.py` dans la machine de l'attaquant. Si la configuration est correcte, l'attaque devrait être bloquée peu après son exécution.

4.5 Déploiement vers Vagrant Cloud

4.5.1 Objectif

L'objectif de ce démonstrateur est de créer une box prête à l'emploi et de la mettre à disposition sur le Vagrant Cloud. Cette machine, basée sur le système d'exploitation Ubuntu 20.04 LTS, est équipée de Ghidra et de ses dépendances.

Ghidra étant une application avec interface graphique, l'environnement de bureau XFCE a été installé.

4.5.2 Création et publication

La construction de l'image se fait avec Packer et du fichier de configuration ci-dessous. Ce fichier réalise dans l'ordre les étapes suivantes :

1. Récupération de l'image Vagrant Ubuntu.
2. Installation d'Ansible sur la machine invitée/distante.
3. Mise à disposition (copie) des fichiers nécessaires.
4. Exécution des playbooks Ansible.
5. Envoi du fichier généré sur Vagrant Cloud.

Le fichier de configuration générant cette « Box » sur Vagrant Cloud est le suivant :

```
locals {
    box = "stevenroh/ubuntu-ghidra"
    version = "0.1.0"
}

variable "cloud_token" {}

source "vagrant" "box" {
    communicator = "ssh"
    source_path = "ubuntu/bionic64"
    provider = "virtualbox"
    add_force = true
}

build {
    name = "ubuntu-ghidra"
    sources = [
```

```
        "source.vagrant.box"
    ]

    provisioner "shell" {
        execute_command = "echo 'vagrant' | sudo -S sh -c '{{ .Vars }} {{ .Path }}'"
        script = "./ansible/ansible.sh"
    }

    # https://github.com/hashicorp/packer/issues/1551#issuecomment-383235951
    provisioner "file" {
        source = "ghidra.desktop"
        destination = "/tmp/ghidra.desktop"
    }

    provisioner "ansible-local" {
        playbook_file = "./ansible/install_ghidra.yml"
    }

    post-processor "vagrant-cloud" {
        access_token = "\${var.cloud_token}"
        box_tag = "\${local.box}"
        version = "\${local.version}"
    }
}
```

Fichier ubuntu-ghidra-vagrantcloud.pkr.hcl

4.5.3 Utilisation

Peu après l'envoi de l'image sur Vagrant Cloud, elle sera disponible publiquement. Nous pouvons ensuite l'utiliser dans nos fichiers de configuration Vagrant.

```
$ vagrant init stevenroh/ubuntu-ghidra
```

```
Vagrant.configure("2") do |config|
    config.vm.box = "stevenroh/ubuntu-ghidra"
    config.vm.box_version = "0.1.0"
end
```

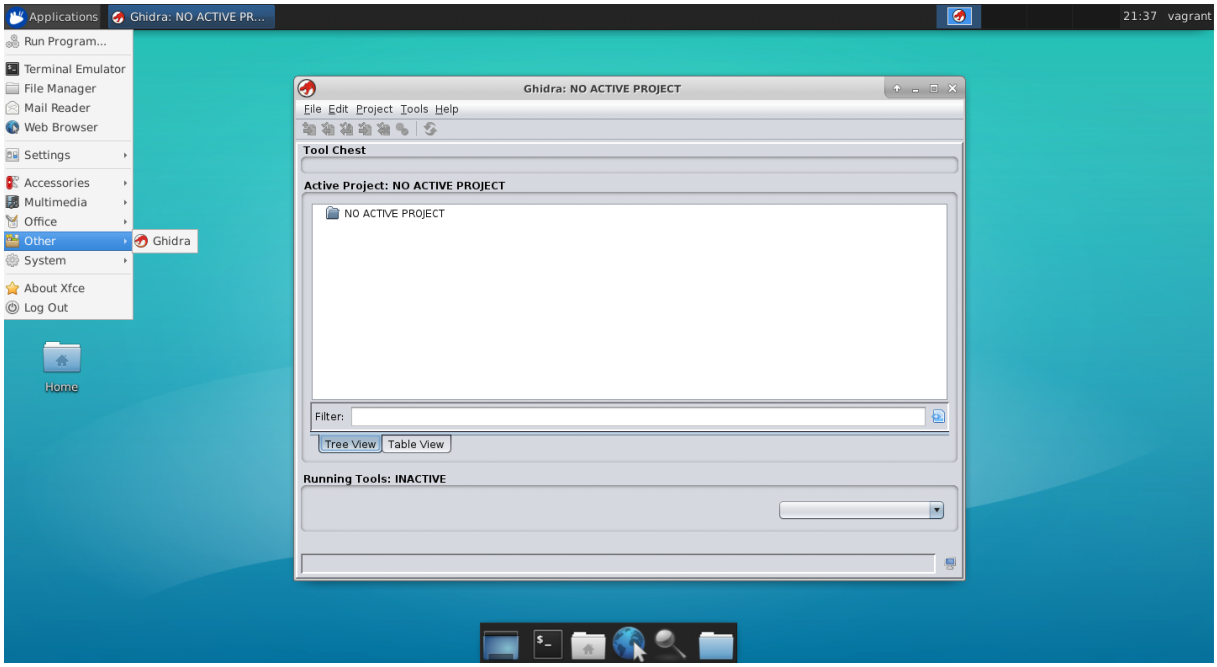


FIGURE *4.23: Ghidra executé dans une machine virtuelle Ubuntu*
Source: *de l'auteur*

Conclusion

Grâce à Vagrant, Ansible, Packer et Vagrant Cloud, le déploiement d'environnements peut se faire en local, sur les postes des étudiants. Terraform fait partie des solutions compatibles et utilisables pour déployer ces mêmes environnements dans le Cloud.

Améliorations possibles

Les solutions proposées permettent facilement aux étudiants de créer les environnements respectifs. Une amélioration possible serait l'ajout d'un outil de suivi de la progression de l'étudiant par le professeur. Grâce à celui-ci, le professeur pourrait vérifier la progression de l'apprentissage de l'étudiant voire de lui apporter du soutien en cas de blocage.

Actuellement, les environnements et les données des exercices pratiques sont disponibles sur GitLab. Une amélioration pourrait être la mise à disposition d'une plateforme web à la manière d'une plateforme CTF sur lequel l'étudiant trouverait les données des exercices et ressources utiles ainsi que des questions pour guider son apprentissage.

L'exécution des laboratoire fonctionne à l'aide de machines virtuelles. Il est possible d'améliorer ce procédé en utilisant des conteneurs. Cela permettrait non seulement une utilisation plus faible des ressources du système mais également de créer une procédure configuration unifiée grâce à l'utilisation de conteneurs.

Le déploiement de conteneurs sur Kubernetes à l'aide de Minikube sur les machines en local ou sur le Cloud Azure Container Service (AKS) permettrait de déployer les mêmes environnements à l'aide des mêmes fichiers de configuration.

Difficultés rencontrées

Intégration de mozilla-oidc-django

Lors de l'installation et la configuration de `mozilla-oidc-django` au point 4.1.4, nous pouvons constater que les réglages définissent les URLs avec lesquelles Django se connecte à Keycloak. Ces mêmes URLs sont aussi utilisées lors des redirections du coté du client visitant l'application. Cependant, la connexion de Django à Keycloak se fait via un réseau privé et des adresses IP internes ce qui n'est pas le cas lorsque le navigateur web se connecte à l'application Django et à Keycloak.

Afin de simplifier ce processus, un domaine local (`seculab.local`) a été mis en place. Sur la machine de l'étudiant, ce domaine pointera vers sa propre machine (`127.0.0.1`) grâce à l'ajout d'une entrée dans le fichier `hosts` de sa machine. Sur la VM Django, il pointera sur l'adresse IP privée de la machine (`192.168.56.100`), également à l'aide d'une entrée dans le fichiers `hosts`.

Crowdsec ne bloque pas l'attaque exécutée

Dans la section 4.4, l'exécution d'attaques par force brute n'était pas bloquée. Il s'agit effectivement du comportement correct lorsqu'un *bouncer* n'est pas installé. Une fois mis en place et configuré pour communiquer avec Crowdsec, le « bouncer » est chargé de bloquer la connexion à la machine depuis l'adresse IP de l'attaquant lors de la détection par un scénario. Les « bouncers » utilisables sont disponibles sur le Crowdsec Hub, à l'adresse https://hub.crowdsec.net/browse/#bouncers.

Échec de la construction de l'image pour Azure avec packer

Lors des premiers tests de déploiement, Packer retournait une erreur indiquant que la taille de la ressource (machine virtuelle) n'était pas disponible dans cette région. Cependant, selon la documentation Azure, cette taille de machine est bien disponible dans cette région [16].

Après vérification, il se trouve qu'avec l'abonnement destiné aux étudiants (Azure for Students), il existe des restrictions sur les types de machines utilisables avec le crédit offert.

16. https://azure.microsoft.com/fr-fr/global-infrastructure/services/

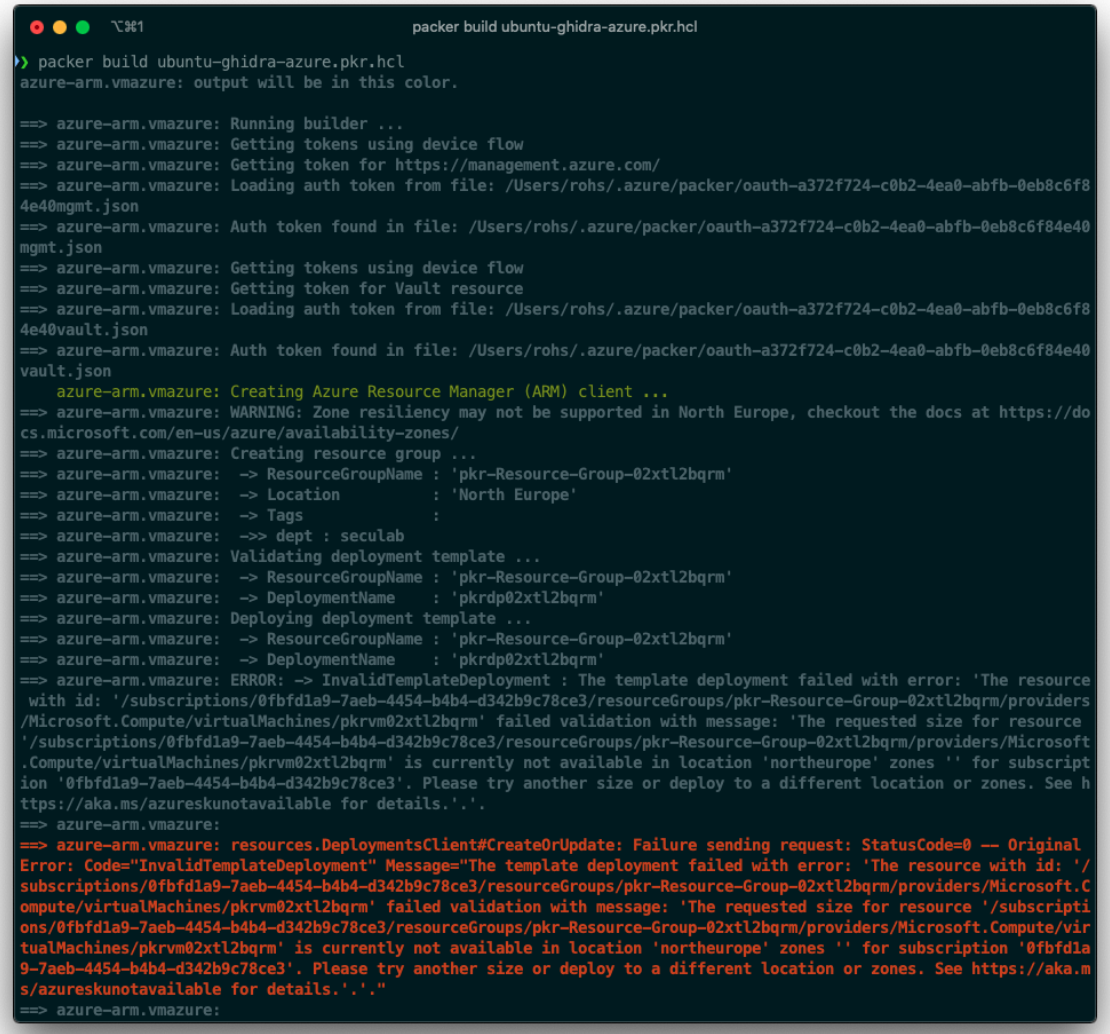


FIGURE *4.24: Erreur lors de l'exécution de la commande* `packer build`
Source: *de l'auteur*

La commande `az vm -list-skus` a permi d'identifier la disponibilité en fonction de la région et du compte étudiant.

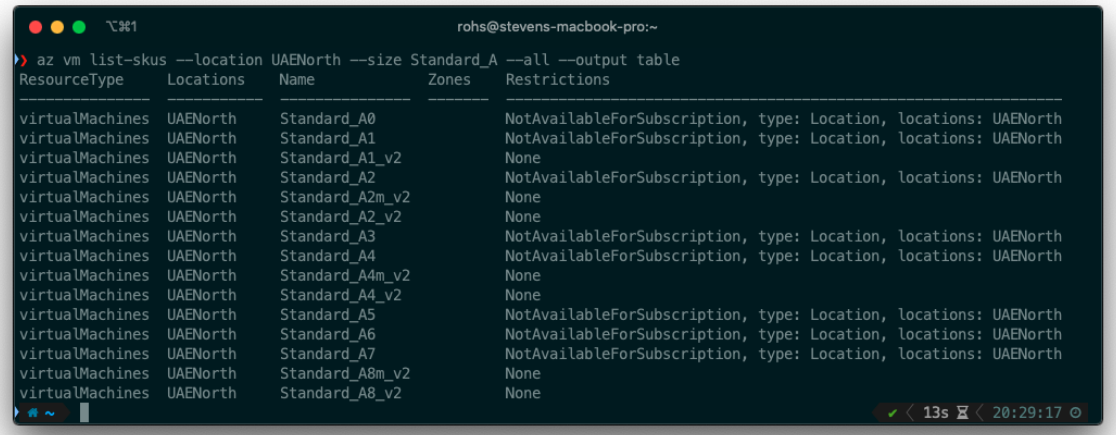


FIGURE *4.25: Liste de type de machines disponibles par région avec le compte étudiant*
Source: *de l'auteur*

I | Vagrant - Commandes utiles

`vagrant init [name [url]]`
Initialise un dossier d'environnement Vagrant en créant un Vagrantfile s'il n'existe pas déjà.

`vagrant up`
Démarre de l'environnement actuel (dossier courant).

`vagrant halt [-f] [-force]`
Éteint des machines de l'environnement.

`vagrant reload`
Redémarre l'environnement. Équivalent d'un halt suivi d'un up.

`vagrant ssh [machine]`
Permet la connexion à l'environnement [machine] à travers SSH.

`vagrant status`
Affiche l'état de l'environnement courant.

`vagrant destroy`
Détruit l'environnement courant.

`vagrant box list`
Affiche la liste des « Box » disponibles localement.

`vagrant box add`
Ajoute une nouvelle « Box » à la liste des « Box » disponibles localement.

`vagrant box remove`
Supprime une « Box » à la liste des « Box » disponibles localement.

II | Terraform - Commandes utiles

- `terraform validate`
  Vérifie la configuration.
- `terraform plan`
  Affiche les changements requis par la configuration.
- `terraform apply`
  Applique la création ou la mise à jour de l'infrastructure.
- `terraform destroy`
  Détruit l'infrastructure précédemment crée.

III | Packer - Commandes utiles

- `packer init`
  Récupère et installe les extensions définies dans le bloc de configuration `required_plugins`.
- `packer inspect [template]`
  Affiche le détail d'un modèle Packer.
- `packer validate [template]]`
  Vérifie le modèle Packer.
- `packer build [template]`
  Construit l'image à partir d'un modèle Packer.
- `packer hcl2_upgrade [template]`
  Convertit un fichier modèle du format JSON vers le format HCL (version 2).

IV | Création d'un compte Azure pour étudiant

Ce guide s'adresse aux étudiants et permet de créer un compte *Azure for Students*. Avec celui-ci, un crédit de US$ 100 est octroyé chaque année à condition de fréquenter une université reconnue comme la HES-SO Valais-Wallis.

IV.1 Mise en route

Le compte Azure pour étudiant peut être créé à l'adresse https://azure.microsoft.com/fr-fr/free/students/ en cliquant sur le bouton *Démarrer gratuitement*.

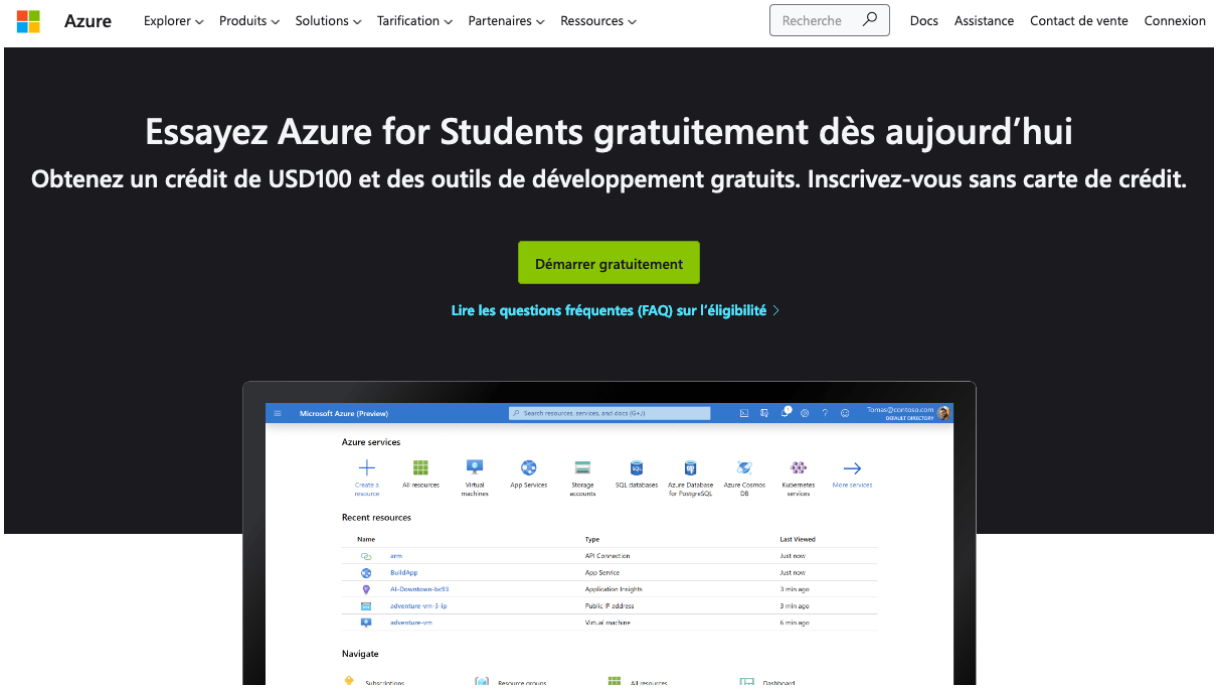


FIGURE *IV.1: Page d'accueil Azure pour Étudiants*
Source: *de l'auteur à partir de azure.microsoft.com*

Afin de bénéficier de l'offre, il est nécessaire d'utiliser l'addresse e-mail `@hes-so.ch`.

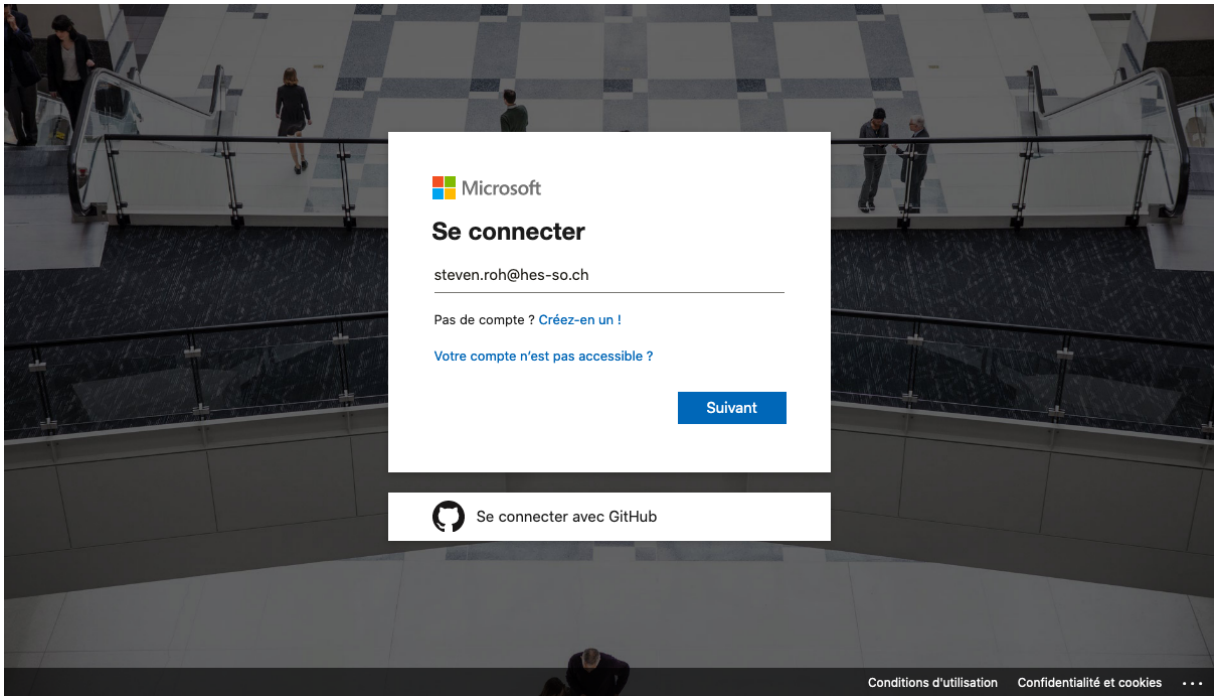


FIGURE *IV.2: Écran de connexion où il faut se connecter avec ses identifiants scolaires*
***Source:** de l'auteur à partir de azure.microsoft.com*

À cette étape, le profil utilisateur doit être rempli avec les informations personnelles.

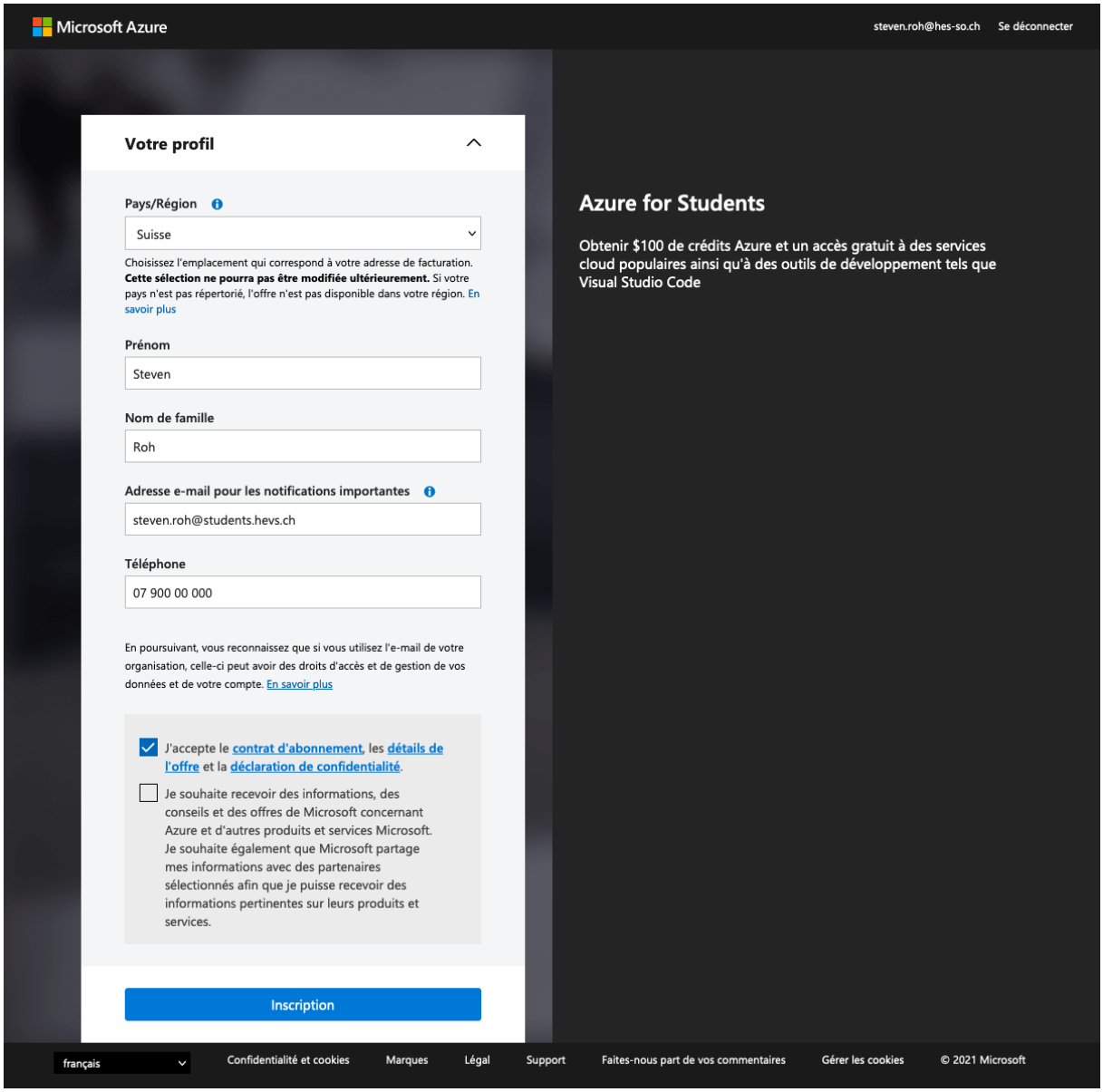


FIGURE *IV.3: Formulaire de profil utilisateur*
Source: *de l'auteur à partir de azure.microsoft.com*

IV.2 Vérification de l'abonnement

Une fois terminé, l'abonnement peut être vérifié en se dirigeant sur la page d'accueil puis dans *Abonnements*

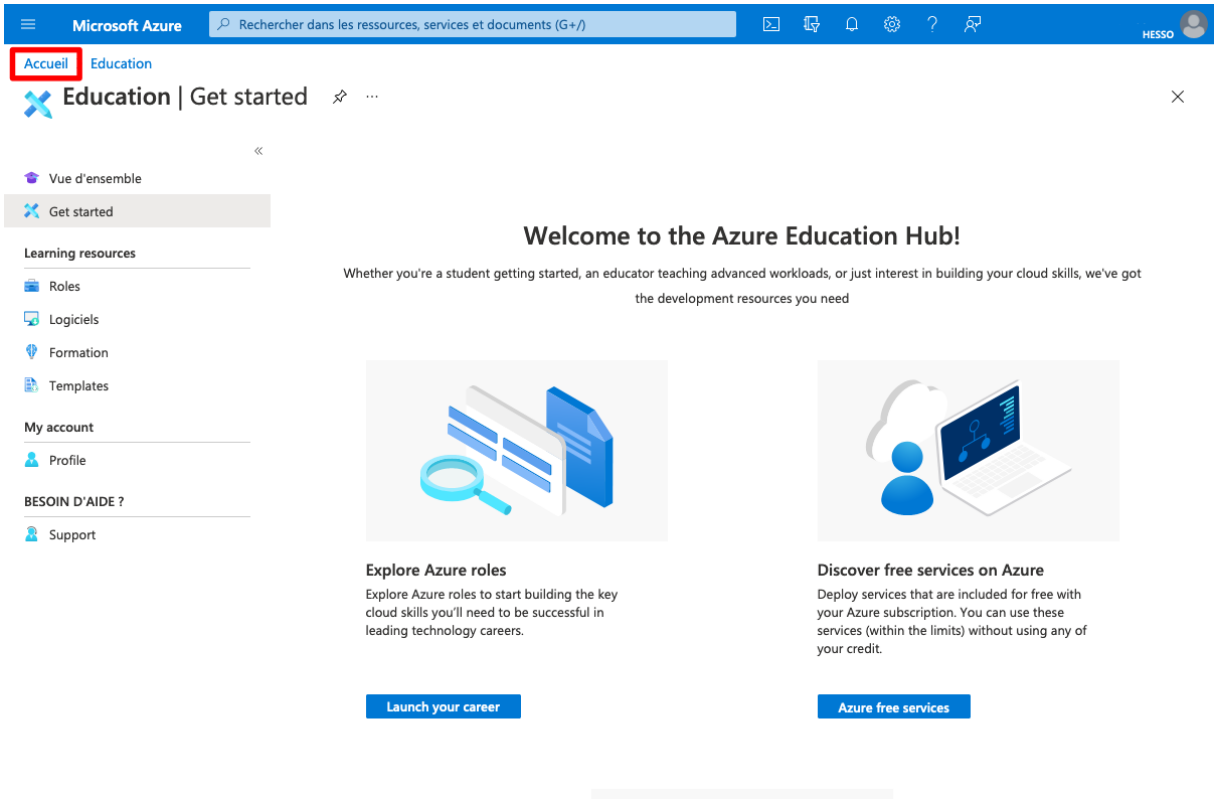


FIGURE *IV.4: Page de bienvenue et de confirmation au programme éducation*
Source: *de l'auteur à partir de azure.microsoft.com*

Dans la page *Abonnements* doit figurer le nouvel abonnement *Azure pour les étudiants*.

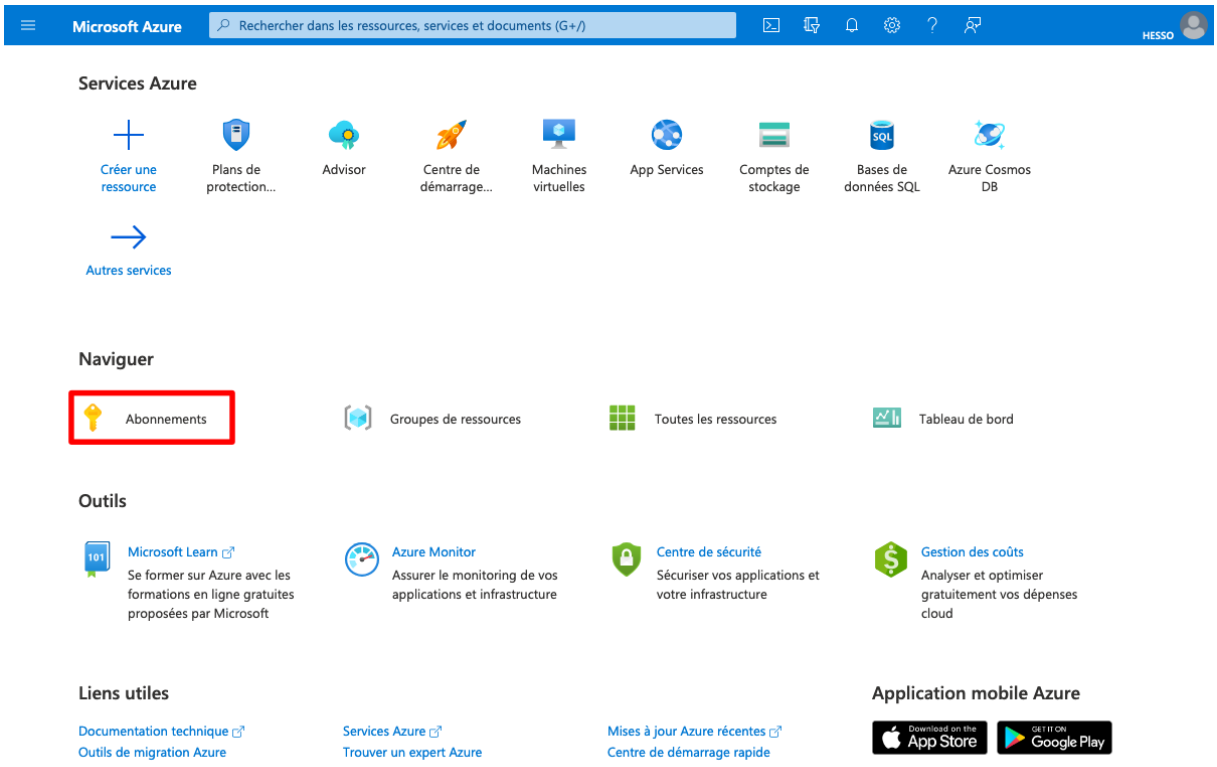


FIGURE *IV.5: Page d'accueil Azure avec la liste des services à disposition*
***Source:** de l'auteur à partir de portal.microsoft.com*

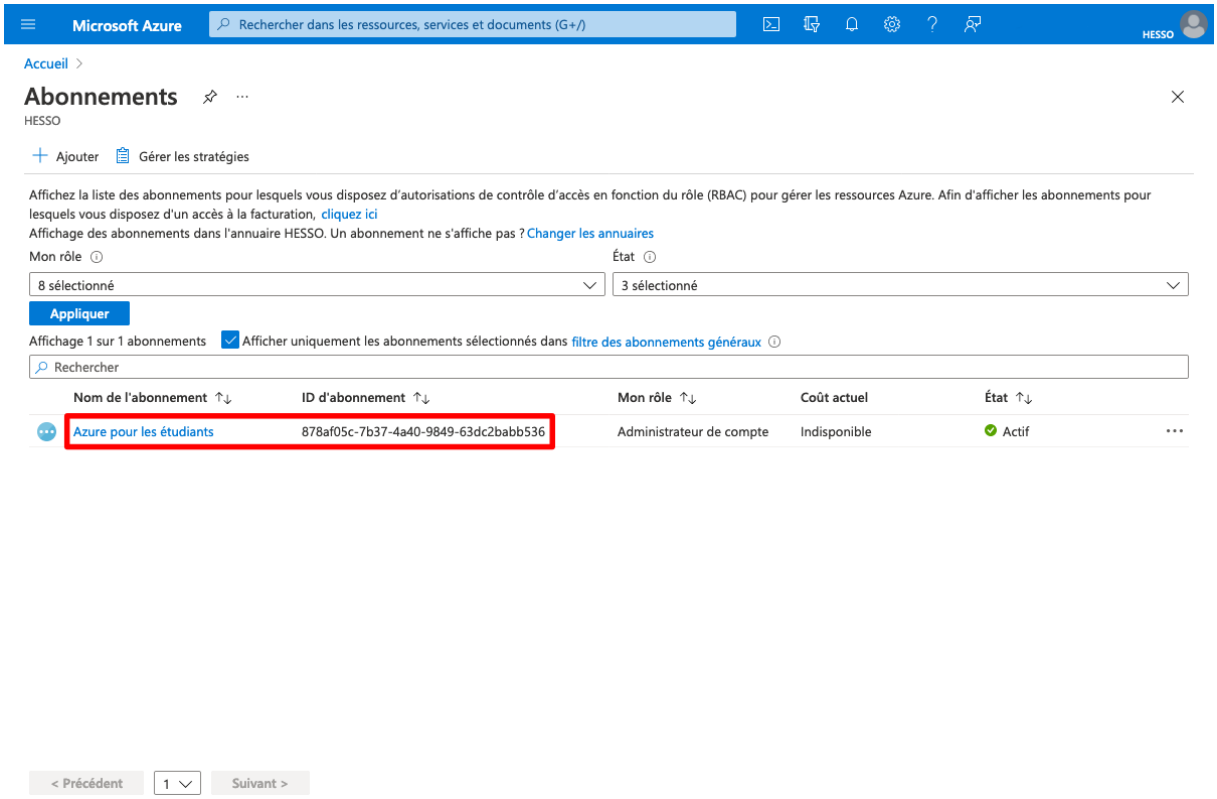


FIGURE *IV.6: Liste des abonnements de l'utilisateur*
Source: *de l'auteur à partir de portal.microsoft.com*

IV.3 Vérification du crédit

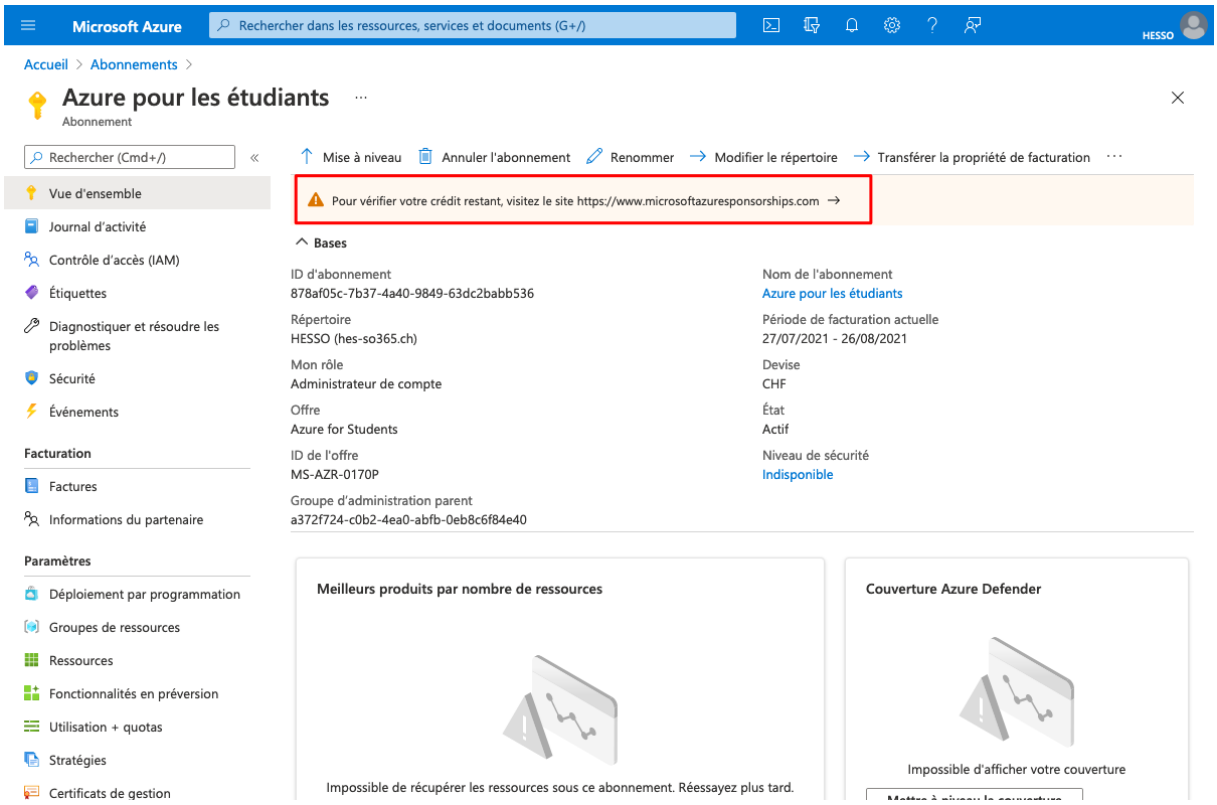


FIGURE *IV.7: Détail de l'abonnement avec lien pour la consultation du solde*
Source: *de l'auteur à partir de portal.microsoft.com*

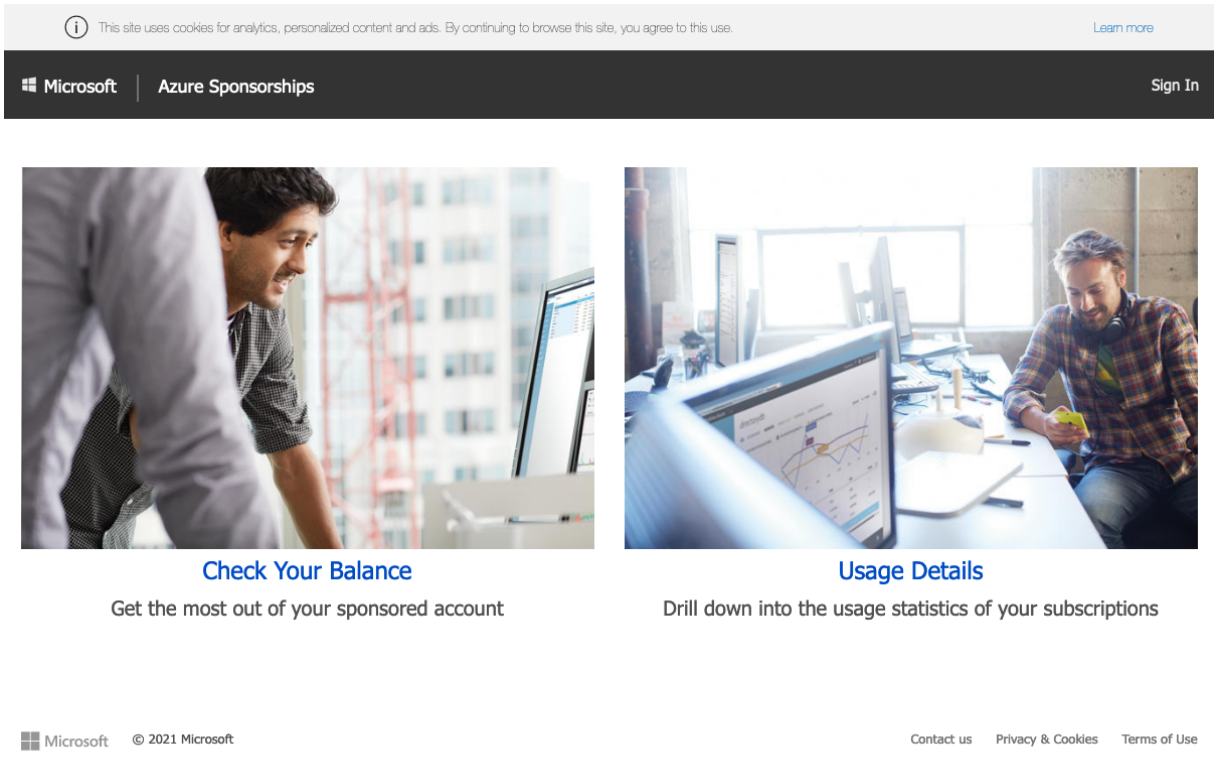


FIGURE *IV.8: Page dédiée à la consultation du solde disponible et du détail d'utilisation*
***Source:** de l'auteur à partir de microsoftazuresponsorships.com*

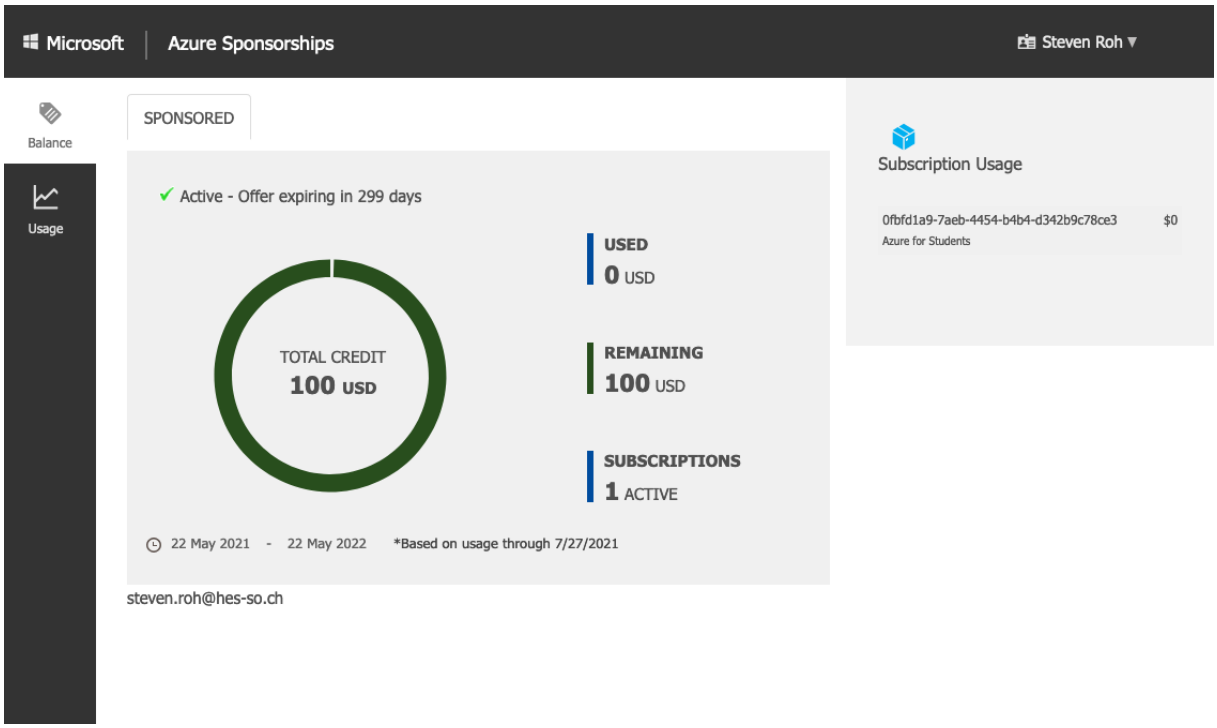
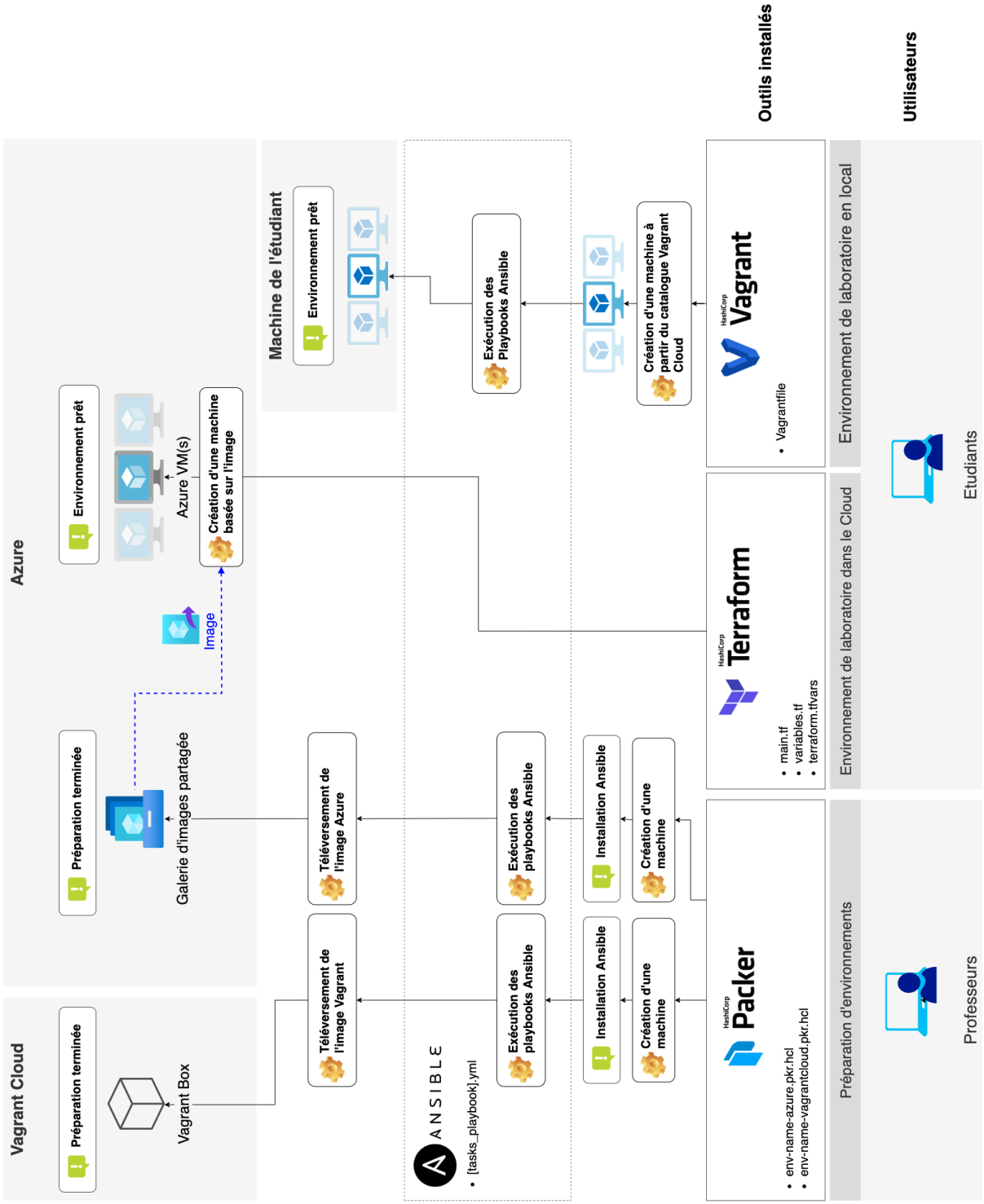


FIGURE *IV.9: Page affichant le solde disponible*
***Source:** de l'auteur à partir de microsoftazuresponsorships.com*

V | Schéma de fonctionnement



Références

Amazon Educate - FAQs. (2020). Récupérée le 19 juillet 2021, à partir de https://www.awseducate.com/student/s/faqs

Azure Provider: Authenticating via Managed Identity | Guides | hashicorp/azurerm | Terraform Registry. (2021). Récupérée le 31 juillet 2021, à partir de https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/managed_service_identity

BEUCHAT, J.-L. (2021). 634-2 - Théorie et exercices pratiques.

BEURAN, R., PHAM, C., TANG, D., CHINEN, K.-i., TAN, Y. & SHINODA, Y. (2018). Cybersecurity Education and Training Support System: CyRIS. *IEICE Transactions on Information and Systems*, *E101.D*, 740-749. doi :10.1587/transinf.2017EDP7207

Box File Format | Vagrant by HashiCorp. (2021). Récupérée le 19 juillet 2021, à partir de https://www.vagrantup.com/docs/boxes/format

BRUNO. (2021). Crowdsec, un outil de prévention d'intrusions, conçu pour protéger les serveurs, les services et les conteneurs. Récupérée le 10 juillet 2021, à partir de https://securite.developpez.com/actu/313207/Crowdsec-un-outil-de-prevention-d-intrusions-concu-pour-proteger-les-serveurs-les-services-et-les-conteneurs-presente-comme-une-version-modernisee-et-collaborative-de-Fail2Ban/

CHRIS ROBERTS, S. C. (2021). Toward Vagrant 3.0. Récupérée le 13 juin 2021, à partir de https://www.hashicorp.com/blog/toward-vagrant-3-0

CyberSec4Europe delivers Cyber Sandbox Creator. (s. d.). Récupérée le 2 mars 2021, à partir de https://digital-strategy.ec.europa.eu/en/news/cybersec4europe-delivers-cyber-sandbox-creator

Cybersecurity Courses & Certifications, SANS Institute. (s. d.). Récupérée le 16 juillet 2021, à partir de https://www.sans.org/cyber-security-courses/

CyLMS: Cybersecurity Training Support for LMS. (s. d.). Récupérée le 5 avril 2021, à partir de https://github.com/crond-jaist/cylms

DATADOG. (2018). 8 Surprising Facts About Real Docker Adoption | Datadog.

FLEXERA. (2021). Flexera 2021 State of the Cloud report: Europe Spotlight.

Free Automated Malware Analysis Sandboxes and Services. (2021). Récupérée le 19 juillet 2021, à partir de https://zeltser.com/automated-malware-analysis

GOFFINET, F. (2020). Glossaire Ansible. *Linux Administration*. Récupérée le 10 août 2021, à partir de https://linux.goffinet.org/ansible/glossaire-ansible

INCONSHREVEABLE. (2021). ngrok - secure introspectable tunnels to localhost. Récupérée le 6 août 2021, à partir de https://ngrok.com/product

Infosec Training and Penetration Testing | Offensive Security. (s. d.). Récupérée le 15 juillet 2021, à partir de https://www.offensive-security.com/

Installing Provider - Vagrant Parallels Provider Documentation. (2021). Récupérée le 10 août 2021, à partir de https://parallels.github.io/vagrant-parallels/docs/installation

KUBERNETES. (2020). Pods. Récupérée le 8 août 2021, à partir de https://kubernetes.io/fr/docs/concepts/workloads/pods/pod

KUBERNETES. (2021). Noeuds. Récupérée le 8 août 2021, à partir de https://kubernetes.io/fr/docs/concepts/architecture/nodes

Labtainers - Center for Cybersecurity and Cyber Operations - Naval Postgraduate School. (s. d.). Récupérée le 16 juillet 2021, à partir de https://nps.edu/web/c3o/labtainers

MARQUARDSON, J. (2018). Infrastructure Tools for Efficient Cybersecurity Exercises.

MATYÁŠ, V. (2020). Virtual lab for open-source tools education and research.

MCCULLOUGH, S. (2016). Using Vagrant to Build a Manageable and Sharable Intrusion Detection Lab.

RAYOME, ALISON DENISCO. (2019). Ansible overtakes Chef and Puppet as the top cloud configuration management tool. *TechRepublic*. Récupérée le 8 août 2021, à partir de https://www.techrepublic.com/article/ansible-overtakes-chef-and-puppet-as-the-top-cloud-configuration-management-tool

REDHAT. (2021). Conteneurs et machines virtuelles.

Sandbox Definitions · Wiki · MUNI-KYPO-CSC / cyber-sandbox-creator. (s. d.). Récupérée le 5 avril 2021, à partir de https://gitlab.ics.muni.cz/muni-kypo-csc/cyber-sandbox-creator/-/wikis/Sandbox-Definitions

Server Administration Guide. (2021). Récupérée le 22 juillet 2021, à partir de https://www.keycloak.org/docs/latest/server_admin/#_identity_broker

Terraform by Hashicorp. (2021).

THE KUBERNETES AUTHORS. (s. d.). Drivers | minikube.

TOMER, C. (2016). *Vagrant and Docker as Learning Environments*. Récupérée le 10 mars 2021, à partir de http://d-scholarship.pitt.edu/32407/

Vagrant in production - StackOverflow. (2021).

Vagrant vs. Terraform. (2021). Récupérée le 13 juillet 2021, à partir de https://www.vagrantup.com/intro/vs/terraform

VIRTUALENGINE. (2021). Lability. Récupérée le 20 juillet 2021, à partir de https://github.com/VirtualEngine/Lability

Glossaire

AKS Azure Kubernetes Service. 26

AMI Amazon Machine Image. 26

Bash Bourne-Again shell. 15

BYOD Bring Your Own Device. 2

CAS Certificate of Advanced Studies. ii, 1

CRI Container Runtime Interface. 17

CTF Capture The Flag. 4, 77

DSL Domain Specific Language. 22

EDR Endpoint Detection and Response. 67

EKS Elastic Kubernetes Service. 26

HCL HashiCorp Configuration Language. 23, 83

IAM Identity and Access Management. 51

JSON JavaScript Object Notation. 16, 22, 23, 83

JWT JSON Web Token. 59, 61

SSO Single Sign-On est un procédé par lequel un utilisateur peut se connecter à plusieurs applications en ne procédant qu'une seule fois à son authentification. 51

XSS Cross Site Scripting. 72

XXE Xml eXternal Entity. 72

YAML Yet Another Markup Language est un language qui permet de représenter des informations complexes d'une manière simple et lisible. 9, 13, 22

Informations sur ce travail

Informations de contact

Auteur : Steven Roh

HES-SO Valais-Wallis

E-mail : *steven.roh@students.hevs.ch*

Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

Lieu, date : _____

Signature : _____