néosoft

SUPINFO International University

DOCUMENT HIGH-LEVEL-DESIGN (HLD)
PRJ3_LYON2

DESCRIPTION

Ce document d'architecture High Level Design (HLD) explicite l'architecture de la solution et donne les détails nécessaires pour servir de base à la bonne compréhension du projet. Il décrit les principaux composants du système et leurs interactions, la politique de sécurité, les bonnes pratiques et conventions utilisées pendant le projet.

GROUPE PRÉSENTANT

RODET Nathan
RIVALLAND Gregory
CHELIAN David
HARMEL Enguerrand

néosoft

SUPINFO International University

Enregistrement des changements

Date d'édition	NOM / Prénom	Version du document	Description
19/09/2023	RODET Nathan	1.1	Reprise du document
30/11/2023	RIVALLAND Gregory	1.2	Correction et mise en page
10/01/2023	CHELIAN David	1.3	Ajout info sur monitoring et log, correction base MySQL DB, ajout d'infos sur la partie Aks Core et Control Panel

néosoft

SUPINFO International University

SOMMAIRE

1. Contexte du projet

a. Rappel du besoin technique et fonctionnel

Le client exprime le besoin d'implémenter une plateforme d'orchestration Kubernetes en s'appuyant sur les services AKS, EKS ou GKE, qui sont les services des principaux fournisseurs de cloud public. Cette plateforme à pour objectif d'héberger un site E-Commerce architecturé sous forme de microservices préalablement développés ainsi que différents services tels qu'un service de cache Redis et une base de données utilisant la technologie SQL.

Il est impératif que la mise en place de cette plateforme respecte les bonnes pratiques recommandées par le fournisseur de service cloud tout en tenant compte des aspects identité, de sécurité réseau et de la bonne gestion des données.

Le client souhaite que notre approche respecte une démarche d'automatisation et de gestion des versions utilisant de l'outillage Devops, et donc l'intégration de pipelines de provisionnement pour son infrastructure via les outils Terraform et Azure Devops. Cette même infrastructure qui devra centraliser la gestion des conteneurs applicatifs et des services pour ensuite les orchestrer à l'aide de la plateforme Kubernetes mise en place.

Il est aussi demandé par le client de fournir plusieurs documents d'architecture et de documentation pour cette solution, à savoir : un document High Level Design (HLD), un Document d'Architecture Technique (DAT) ainsi qu'un Document d'Exploitation (DEX).

Au niveau de notre répartition sur notre temps de travail, nous avons principalement David et Nathan en qualité de développeur sur ce projet, accompagné de Enguerrand et Greg sur la partie documentation.

b. Réponse technique et fonctionnement

Pour répondre à ces besoins, plusieurs solutions technologiques, fonctionnelles et techniques ont été identifiées, notre proposition s'appuiera sur la base du fournisseur de service de cloud public Azure, qui possède le meilleur rapport qualité-prix en plus d'avoir la meilleure interconnexion avec une partie de l'outillage souhaité par le client, à savoir Azure Devops.

Notre solution suivra les recommandations de Microsoft Azure pour les architectures microservices, à laquelle nous ajouterons une couche d'automatisation.

Ainsi, nous proposons l'utilisation d'Azure Devops pour 4 de ses services, à savoir Repos, qui héberge le code source. Pipelines, pour le sous-service éponyme pipelines qui permettra de générer des artefacts à partir du code source et donc d'assurer la partie d'intégration continue de la solution. Le sous-service Release pour réaliser les différentes étapes de déploiement continue à partir des artefacts générés. À

savoir le provisionning de l'infrastructure, la construction et l'envoi des images Docker vers la plateforme de gestion des conteneurs, puis configuration de la plateforme AKS. Et pour finir le sous-service Library, pour la gestion de la configuration, des variables, des secrets et de leurs accès par les autres services.
Pour décrire ce processus plus précisément, nous créerons dans un premier temps un stockage permanent à l'aide d'une étape d'exécution d'Azure Commande Line (AZ-CLI). Ce stockage a pour but d'héberger le fichier ".tfstate" qui gère l'état et le cycle de vie de l'infrastructure provisionnée à l'aide de Terraform. C'est donc une bonne pratique de le rendre partagé, pour éviter toutes pertes de données ou modifications simultanées sur l'infrastructure grâce aux verrouillages opérés par Terraform sur ce fichier pendant qu'il exécute des actions.

Terraform possédant ainsi son environnement et étant prêt à l'emploi, nous pourrons utiliser nos fichiers de configuration préalablement développés pour provisionner les composantes réseaux puis le reste de notre infrastructure.

Une fois l'infrastructure provisionnée, nous passerons aux étapes de construction des images Docker et de leurs envois vers notre plateforme de gestion de conteneurs, à savoir le service Azure Registry Manager déployé par Terraform. Elles seront ensuite disponibles et accessibles par notre plateforme Kubernetes.

Ces étapes réalisées, nous injecterons notre configuration Kubernetes dans notre service Azure Kubernetes Services, qui se chargera de créer les ressources demandées et de réaliser les diverses actions listés dans sa configuration, comme par exemple la création des namespaces ou encore du déploiement des services.

Bien-sûr, cette réponse fonctionnelle n'indique pas tous les éléments pris en compte pour la mise en place de cette solution, elle ne pose qu'une description simpliste ne prenant pas en compte, par exemple, les composantes réseau ou de sécurité.

2. Informations générales

a. Definitions

- **Kubernetes :** Outil open-source d'orchestration de conteneurs et d'automatisation.
- **Docker** : Technologie de conteneurisation permettant entre autres la construction d'images et l'exécution de conteneurs.
- **Conteneur** : Package logiciel constitué de différentes couches : OS, configuration… Utilisé pour isoler le code source d'une application, sa configuration et ses dépendances dans son propre environnement d'exécution.
- **Terraform** : Outil d'automatisation pour la création des ressources d'infrastructures, notamment cloud en gérant leur cycle de vie.

- **AKS, GKE, EKS** : Services Kubernetes fournis par les principaux fournisseurs de cloud public.
- **Azure** : Fournisseur de cloud public appartenant à Microsoft.
- **Azure Devops** : Plateforme contenant les outils pour la gestion et la réalisation d'un projet informatique utilisant les pratiques devops.
- **Devops** : Combinaison de philosophies, pratiques et outillages qui améliorent l'évolutivité, la qualité et la rapidité des produits créés par les entreprises, on confond souvent le Devops avec l'automatisation.
- **Micro-services** : Pattern d'architecture système visant à réduire la taille des composants pour les faire évoluer de manière autonome et les faire communiquer entre eux.
- **Open-source** : Code source ou logiciel accessible publiquement et libre de droit.
- **OWASP** : Organisation à but non lucratif et réseau mondial qui travaille sur la sécurité des logiciels libres, en particulier sur le web.
- **Azure MySQL Database:** Technologie open-source de système de base de données relationnelles SQL.

b. Description générale

Voir partie **1. Contexte du projet**, sous partie **b. Réponse technique et fonctionnement.**

La plateforme Kubernetes d'orchestration de conteneurs pour l'hébergement de l'application micro-service d'e-commerce suivra un modèle multi-couche. La première couche sera applicative et composée des micro-services préalablement développés et possédant leurs propres environnements d'exécution sous forme de conteneur. Sur la seconde couche, nous trouverons les services associées à cette application à savoir la base de données et le service de cache. Ces deux couches seront supportées par une troisième couche, la couche d'infrastructure composée du service Azure Kubernetes Service qui orchestre les conteneurs des micro-services et du service de cache, la base de données étant déployée hors du cluster Kubernetes. Cette même couche portera également le service Azure Container Registry qui héberge les images de conteneur pour la plateforme Kubernetes. Pour finir, une quatrième couche est présente, la couche réseau qui est chargée d'isoler les ressources et la plateforme Kubernetes, mais aussi de réaliser le trafic réseau dans toute la solution.

c. Technologies et outillage utilisés

- **Azure Kubernetes Service (AKS) :** AKS est un cluster Kubernetes managé hébergé dans le cloud Azure. Azure gère le service d'API Kubernetes et seule la gestion des nodes nous incombe.
- **Réseau virtuel (VNET)** : Par défaut, AKS crée un réseau virtuel dans lequel les nodes sont connectées.
- **Ingress** : Un Ingress expose les routes HTTP(S) aux services à l'intérieur du cluster.

- **Load Balancer** : Dès qu'un cluster AKS est créé, il est prêt à utiliser le load balancer. Une fois le service NGINX déployé, le loadbalancer est configuré avec une nouvelle adresse IP publique pour l'Ingress Controller. Le load balancer achemine ainsi le trafic Internet vers l'Ingress.
- **External data store** : Les microservices sont généralement sans état et écrivent l'état dans des magasins de données externes, tels qu'Azure SQL Database ou Azure Cosmos DB.
- **Azure Active Directory**. AKS utilise une identité Azure Active Directory (Azure AD) pour créer et gérer d'autres ressources Azure telles que les Azure LoadBalancer. Azure AD est également recommandé pour l'authentification de l'utilisateur dans les applications clientes.
- **Azure Container Registry**. Utilisez Container Registry pour stocker les images Docker privées, qui sont déployées sur le cluster. AKS peut s'authentifier auprès de Container Registry à l'aide de son identité Azure AD. AKS ne nécessite pas Azure Container Registry. Vous pouvez utiliser d'autres registres de conteneurs, tels que Docker Hub. Assurez-vous simplement que votre registre de conteneurs correspond ou dépasse le contrat de niveau de service (SLA) pour votre charge de travail.
- **Azure Pipelines**. Azure Pipelines fait partie des services Azure DevOps et permet d'exécuter des builds, des tests et des déploiements automatisés. Vous pouvez également utiliser des solutions CI/CD tierces telles que Jenkins.
- **Helm**. Helm est un gestionnaire de package pour Kubernetes qui permet de regrouper et de généraliser les objets Kubernetes en une seule unité qui peut être publiée, déployée et mise à jour, et dont la version peut être contrôlée.
- **Azure Monitor**. Azure Monitor collecte et stocke les métriques et les journaux, les données de télémétrie des applications et les métriques de plateforme pour les services Azure. Utilisez ces données pour superviser l'application, définir des alertes, configurer des tableaux de bord et effectuer une analyse de la cause racine des échecs. Azure Monitor s'intègre à AKS pour collecter des métriques à partir des contrôleurs, nœuds et conteneurs.

d. Recommandations de sécurité

Pour répondre aux exigences du client, nous recommandons une approche DevSecOps. Cette méthode intègre les préoccupations de sécurité et les bonnes pratiques telles que les recommandations de l'OWASP dès les premières étapes des développements et tout au long du cycle de vie des produits.

Elle intègre aussi différents besoins, à savoir l'identification des différentes données sensibles pour répondre aux réglementations, dans notre cas : les données des utilisateurs et bancaires sont concernées. Et être capable de repérer les menaces et failles potentielles récurrentes sur les applications, la gestion des dépendances et les processus de déploiement pour être capable d'y remédier.

Dans ce contexte, l'ajout de solutions d'analyse de DAST (Dynamic Application Security Testing), telle que OWASP ZAP, de IAST (Interactive Application Security Testing), telle que Checkmarx, et de SAST (Static Application Security Testing), telle que Trivy, sont des outils, en plus de s'intégrer directement

dans les pipelines, permettent de détecter et de remonter rapidement les alertes de sécurité qui pourraient apparaître sur l'application.

Nous pouvons aussi prendre en considération la mise en place d'un Firewall d'Application Web (WAF) grâce au service Azure Web Application Firewall. Ce Firewall protège actuellement contre la majorité des vulnérabilités de l'OWASP top 10, voici un exemple tiré de la documentation :

- SQL injection protection.
- Cross-site scripting protection.
- Protection against common web attacks such as command injection, HTTP request smuggling, HTTP response splitting, and remote file inclusion attack.
- Protection against HTTP protocol violations.
- Protection against HTTP protocol anomalies such as missing host user-agent and accept headers.
- Prevention against bots, crawlers, and scanners.
- Detection of common application misconfigurations (that is, Apache, IIS, and so on).

Le WAF peut aussi protéger des attaques DDOS en appliquant cette protection sur la Gateway du Virtual Network où il est déployé.

Cette démarche permet de garantir la sécurité du code et des processus sans affecter l'intégrité ou le fonctionnement des applications elles-même. Cependant, dans le cas spécifique du projet, cette prestation ne sera réalisée que sous forme de bonus si le temps alloué au projet le permet avec l'aide de Trivy qui est une solution open-source.

3. Architecture globale

a. Convention et bonne pratiques

Securité

Dans le cadre de notre projet, nous adopterons l'approche "Security by Design" en conformité avec la philosophie DevSecOps. Cela signifie que nous intégrerons la sécurité dès la conception du projet, en vue de prévenir toute compromission et création de failles potentielles dès les phases initiales. Cette démarche sera mise en œuvre tout en respectant les principes du RGPD et du triptyque CIA (Confidentialité, Intégrité, Accessibilité) pour garantir le respect des données personnelles et la sécurité de l'information.

Gouvernance des données

Comme tout projet d'hébergement d'un site d'E-Commerce, il y a des responsabilités qui accompagnent le processus d'hébergement, comme le respect de la réglementation générale sur la protection des données personnelles, le respect de la vie privée, le respect de l'intégrité des données, la qualité des données.

Nous allons donc gérer l'hébergement d'un site sur un environnement cloud clusterisé sous Azure et en Europe.

Azure est responsable de la sécurité de l'infrastructure sous-jacente, telle que les centres de données, les réseaux et les serveurs physiques. Ils s'engagent à fournir une infrastructure sécurisée et à appliquer des mesures de sécurité pour protéger les données. Cependant, nous sommes toujours responsable de la configuration et de la gestion appropriées de notre environnement sur Azure. Cela comprend la configuration des paramètres de sécurité, la gestion des accès et des autorisations, la protection des données hébergées et la conformité aux réglementations applicables (RGPD). En cas de non conformité, c'est notre entreprise qui est responsable, notamment sur l'origine des données, sur la nature des données.

Nous partons du principe que nous gérons et ne stockons pas d'autres données que celles d'un site internet. Pas de données bancaires par exemple. Par contre, nous allons potentiellement héberger des données utilisateurs sur PostgreSQL, tels que des informations utilisateurs dites "personnelles", des informations de télémétries, et plus largement d'autres données comme des données de facture.

Convention de nommage

Nous mettrons en œuvre les bonnes pratiques dans divers aspects du projet. Pour l'infrastructure Azure, nous adopterons des conventions de nommage claires et cohérentes. Pour justifier les coûts et optimiser les ressources, nous fournirons une justification détaillée des spécifications techniques des ressources déployées.

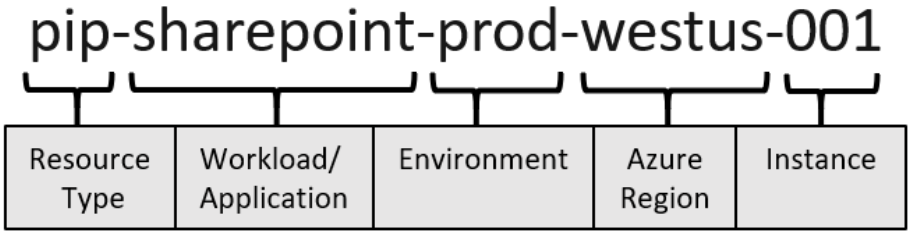

Figure 1 - Exemple de convention de nommage

Terraform

De plus, le fichier tfstate de Terraform sera partagé pour éviter toutes modifications simultanées qui pourraient perturber l'état de l'infrastructure. Cette précaution est essentielle pour maintenir la stabilité et la cohérence de celle-ci.

Monitoring

Nous installerons un système de monitoring performant pour surveiller l'infrastructure en temps réel. Cela nous permettra d'être réactifs en cas de problème, d'identifier rapidement les éventuelles anomalies et de prendre des mesures correctives dans les meilleurs délais.

b. Présentation des services

Azure Devops

Azure DevOps offre des fonctionnalités de contrôle de version, de build et d'intégration automatisé, de test et de gestion des versions… Il couvre l'ensemble du cycle de vie des applications et permet de mettre en œuvre les capacités DevOps.

On utilisera diverses fonctionnalités d'Azure DevOps dans notre projet, le repository héberge le code de l'application ainsi que le code source de notre infrastructure. Nous utiliserons aussi la fonctionnalité Pipelines et son service éponyme pipelines pour créer des artefactes à partir de notre code source. Aussi, nous analyserons notre code source avec Trivy, une solution SAST open-source d'analyse de code source qui pourra détecter les secrets, les principales failles de sécurité et problèmes dans les dépendances.

Nous utiliserons aussi le service Release et Library, qui permettront de créer des déploiements personnalisés pour l'infrastructure et l'application, sous forme de différents stages qui comprennent différentes tâches (Jobs). Le service Library permettra de sécuriser les secrets et d'y accéder depuis nos tâches (Jobs) les agents des runners.

Azure Container Registry

La gestion des images dans un container fait partie des bonnes pratiques, car elle permet de créer, stocker, sécuriser, répliquer et gérer des images et artefacts de conteneur avec une instance complètement managée de distribution OCI (Open Container Initiative), et cela s'implémente bien avec AKS.

On va pouvoir build et push les images depuis notre pipeline Azure, et pull les images nécessaires aux services depuis notre cluster AKS.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) simplifie le déploiement d'un cluster Kubernetes géré dans Azure en déchargeant la surcharge opérationnelle sur Azure. En tant que service Kubernetes hébergé, Azure prend en charge les tâches critiques, comme la surveillance de l'état de santé et la maintenance. Lorsque vous créez un cluster AKS, un plan de contrôle est automatiquement créé et configuré. Ce plan de contrôle est fourni gratuitement en tant que ressource Azure gérée et abstraite de l'utilisateur. Nous ne payerons et ne gérerons que les nœuds attachés au cluster AKS.

Avec AKS, notre application e-commerce profite de la mise à l'échelle automatique, nous pouvons adapter les ressources en fonction de la demande, optimisant ainsi les coûts d'exploitation. Les pics de trafic lors de périodes d'affluence seront alors gérés efficacement garantissant la disponibilité.
Notre Cluster AKS sera déployé à partir des Pipelines d'Azure Devops et d'un Job Terraform avec une configuration initiale. Les accès au cluster se feront à l'aide des credentials Azure.

Storage Account

Un compte de stockage Azure contient tous les objets de données Azure Storage : blobs, fichiers, files d'attente et tables. Le compte de stockage fournit un espace de noms unique pour vos données Azure Storage, accessible depuis n'importe où dans le monde via HTTP ou HTTPS. Les données de votre compte de stockage sont durables et hautement disponibles, sécurisées et massivement évolutives.

Dans notre cas, le Storage Account sera utilisé pour partager le fichier "State" de Terraform, répondant ainsi aux problématiques de modifications simultanées de l'infrastructure et à la bonne pratique Terraform évoquée dans le rapport.

Pour son implémentation, le storage account sera créé à partir d'un Job utilisant la CLI Azure depuis nos Pipelines dans Azure Devops. Ainsi, il sera créé si non existant avant l'exécution de Terraform.

AutoScaling AKS, Nginx Ingress Controller (load balancing et Gateway)

Pour la création d'un environnement clusterisé performant et ajusté AKS, nous allons implémenter après le déploiement, un component d'autoscalling. En effet, le Cluster AutoScaller s'implémente dans un cluster déployé et va gérer la puissance en fonction de l'utilisation. L'autoscaler de cluster surveille les pods qui ne peuvent pas être planifiés sur les nœuds en raison de contraintes de ressources. Le cluster augmente ensuite automatiquement le nombre de nœuds en cas de besoin.

Avec le loadbalancer, nous allons pouvoir fournir des connexions sortantes aux nœuds de cluster en traduisant l'adresse IP privée en une adresse IP publique faisant partie de son pool sortant, et fournir un

accès aux applications via les services Kubernetes, ce qui vous permet de faire évoluer facilement nos applications et de créer des services hautement disponibles.

L'Ingress Controller et le load balancer, ici nginx est utilisé pour répartir la charge des requêtes vers les différentes instances en respectant des règles définies dans notre configuration Kubernetes, tandis que le Gateway se charge de l'accès en HTTPS entre nos services et nos applications .

Le load balancer sera implémenté et configuré après le déploiement du cluster AKS par Terraform depuis nos Pipelines avec une tâche dédiée dans un Job.

Azure Monitor, Application Insights, Log Analytics Workspace et Alerts & Actions

Azure Monitor est une solution de surveillance pour la collecte, l'analyse des télémétries à partir de vos environnements cloud et locaux. Nous allons utiliser Azure Monitor pour surveiller la disponibilité et optimiser les performances de nos conteneurs d'applications et services.

Application Insights est un service d'Azure Monitor et fournit des fonctionnalités de surveillance des performances des applications (APM). Les outils APM sont utiles pour surveiller les applications depuis le développement, aux tests et jusqu'à la production.

Log Analytics est un outil du portail Azure permettant de modifier et d'exécuter des requêtes de journal à partir des données collectées par les journaux Azure Monitor et d'analyser leurs résultats de manière interactive. Nous pouvons utiliser des requêtes Log Analytics pour récupérer des enregistrements qui correspondent à des critères particuliers, identifier des tendances, analyser des modèles et fournir diverses informations sur nos données.

Nous allons mettre en place des alertes et des actions qui nous aideront à surveiller, maintenir et optimiser notre infrastructure basée sur Kubernetes dans Azure, en garantissant une disponibilité élevée, des performances optimales et une gestion efficace des ressources.

Une fois que les alertes sont atteintes, nous recevrons un email pour nous prévenir de l'alerte et par la suite une action pourra être mise en place automatiquement.

c. Diagramme de l'architecture

Architecture Global

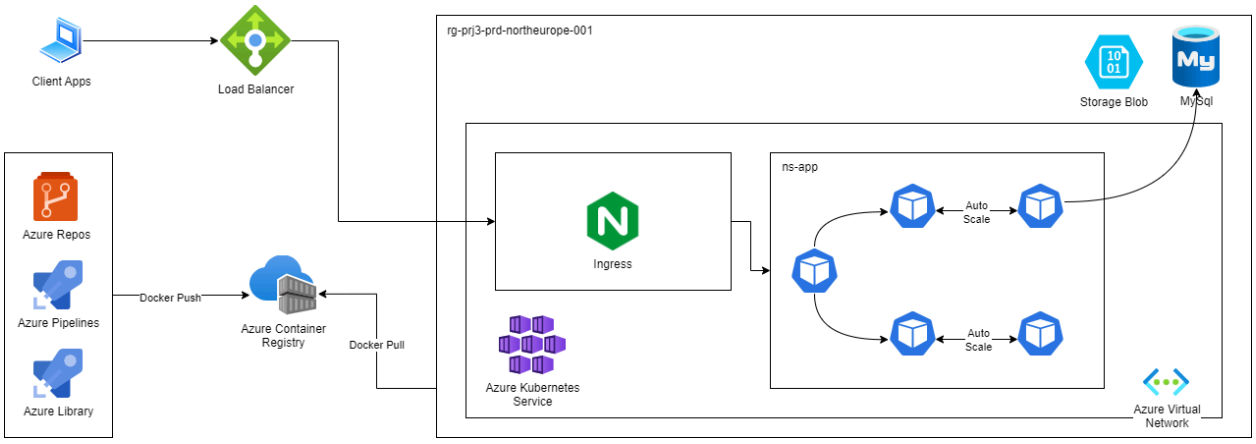

Figure 2 - Diagramme de l'Architecture Global

Cette architecture présente une implémentation en micro-services de nos applications. En effet, on a d'un côté toute la partie Azure DevOps avec nos repos, nos pipelines, la gestion des secrets et de l'autre une partie architecture d'AKS avec notamment nos fronts, nos services backend, et nos pods.

La partie AKS est provisionnée à partir de nos pipelines via Azure Container Registry qui sert de repository d'images. Après avoir créé un cluster AKS, le cluster est prêt à utiliser le load balancer. Ensuite, une fois le service NGINX déployé, le LoadBalancer sera configuré avec une nouvelle IP publique qui sera le point d'entrée de votre contrôleur Nginx. De cette manière, le LoadBalancer achemine le trafic internet vers le contrôleur d'entrée Nginx.

Ensuite le Ingress serveur expose les routes HTTP(S) aux services à l'intérieur du cluster, ainsi le trafic est acheminé vers les containers des pods qui délivrent le service.
Puis enfin, nos services sont connectés à la base MySQL.

Architecture Monitoring

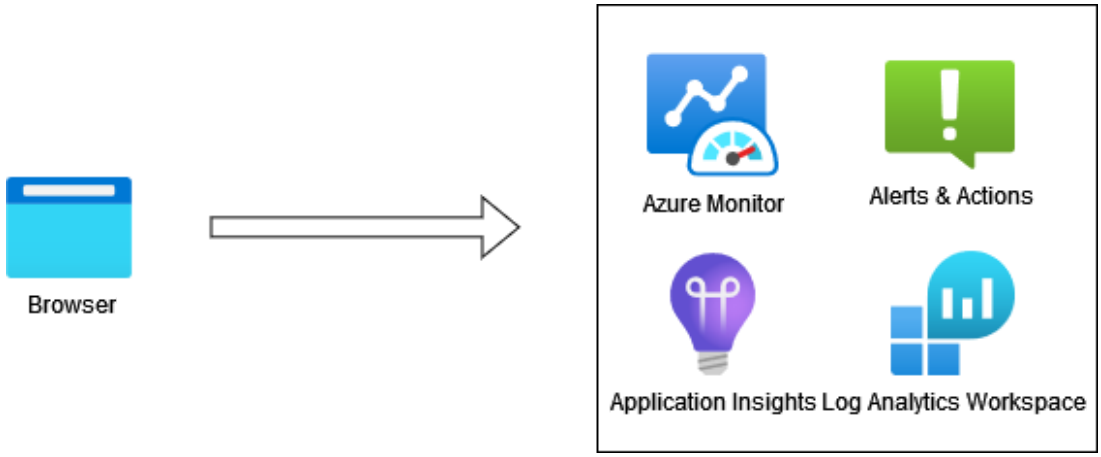

Figure 3 - Diagramme de l'Architecture Monitoring

Ici, nous avons une partie architecture de surveillance, d'outils d'observabilité, de log et d'alertes afin de surveiller notre architecture cloud.

Azure Monitor est un service de surveillance complet qui permet de collecter, d'analyser et de réagir aux données de performance, de disponibilité des applications et des ressources dans Azure. On pourra ici faire un suivi des métriques, voir les journaux d'activité, et les diagnostics des ressources.

Azure Alert permet de configurer et de gérer des alertes basées sur des métriques, des journaux et d'autres données de surveillance. Grâce à ce service, les utilisateurs peuvent définir des conditions personnalisées et recevoir des notifications en temps réel lorsqu'une activité anormale ou des événements prédéfinis se produisent dans leur infrastructure Azure.

Azure Application Insights est une plateforme de surveillance des performances des applications qui permet de détecter, diagnostiquer et résoudre les problèmes de performance et de disponibilité des applications. Il offre des informations détaillées sur les performances, les dépendances et les utilisateurs des applications.

Azure Log Analytics Workspace est un service qui collecte, stocke et analyse les données de télémétrie et de journalisation à grande échelle. Il offre des fonctionnalités avancées pour interroger et visualiser les données, créer des tableaux de bord personnalisés, détecter des tendances et effectuer des analyses approfondies pour prendre des décisions informées en matière de gestion et d'optimisation des ressources Azure.

4. Architecture AKS

a. Organisation des namespaces

Voir partie **3. Architecture globale**, sous partie **a. Convention et bonne pratiques**, sous partie **Convention de nommage**

Pour les namespaces, nous allons respecter les bonnes pratiques de nommage c'est-à-dire l'utilisation de "-" entre chaque mots.

b. Provisioning d'AKS via Terraform

Le service AKS d'Azure doit être déployé via Terraform, en configurant la version de Kubernetes, la région, la zone de disponibilité, le type de mise à niveau, la taille des nœuds et le nombre de nœuds. Respectant un nombre de nœuds minimum à 1 et maximum à 3, et configurer le seuil de consommation à 70%.

Azure Services prévoit la gestion des services du core d'AKS, ainsi la gestion du node master se fait par Azure. Il n'y a donc pas besoin de masquer le node master, ou encore de prévoir la gestion des nodes affinity, du scaling des nodes.

Azure gère l'ensemble du plan de contrôle pour nous. Cela comprend :

- l'approvisionnement et la mise à niveau des ressources du plan de contrôle
- La mise à l'échelle du plan de contrôle pour assurer la haute disponibilité.
- Fonctionnalité de sauvegarde et de restauration pour etcd, y compris la gestion du cluster.
- Surveillance et diagnostic des composants du plan de contrôle.
- Les services de réseau, tels que l'équilibrage des charges et les stratégies de réseau.
- Paramètres de sécurité, y compris le contrôle d'accès basé sur les rôles (RBAC) et l'intégration d'Azure Active Directory.

En résumé, Azure fait abstraction de la complexité de la gestion du plan de contrôle (Control Plane), ce qui retire une part de responsabilité et permet au client de se concentrer sur le déploiement et l'exécution des applications. On ne paye que pour les nœuds de travail qui exécutent nos applications, et non pour la gestion fournie par Azure. Cela fait d'AKS une solution rentable pour le déploiement d'applications conteneurisées à grande échelle.

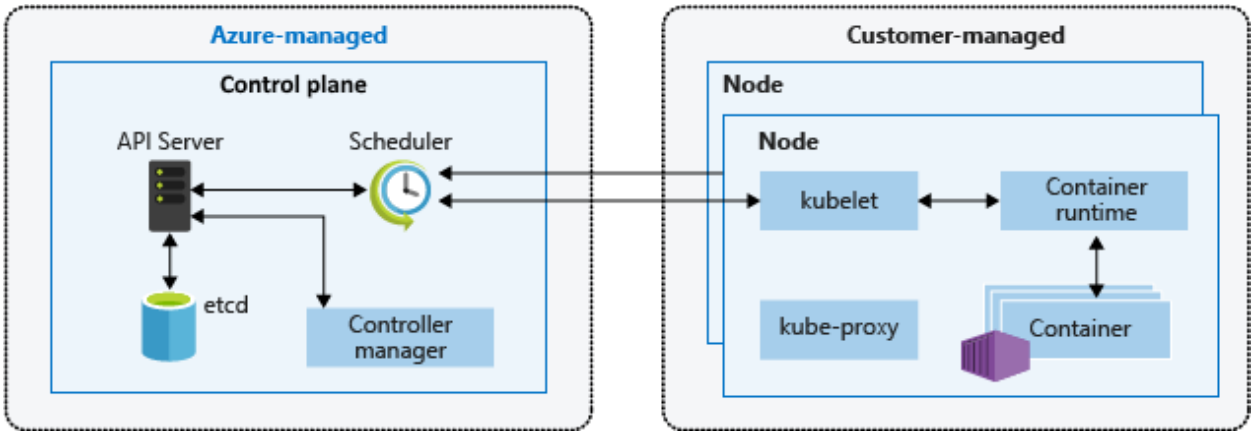

Figure 4 - Responsabilité d'Azure et du client

c. Provisioning d'Nginx Controller

Le déploiement de Nginx Controller est réalisé via un job de pipeline après le déploiement des clusters AKS. Une fois le Nginx Ingress Controller ajouté dans le cluster AKS, la configuration sera réalisée avec des règles d'acheminement pour le site e-commerce en utilisant des objets Kubernetes tels que Ingress et IngressRoute.

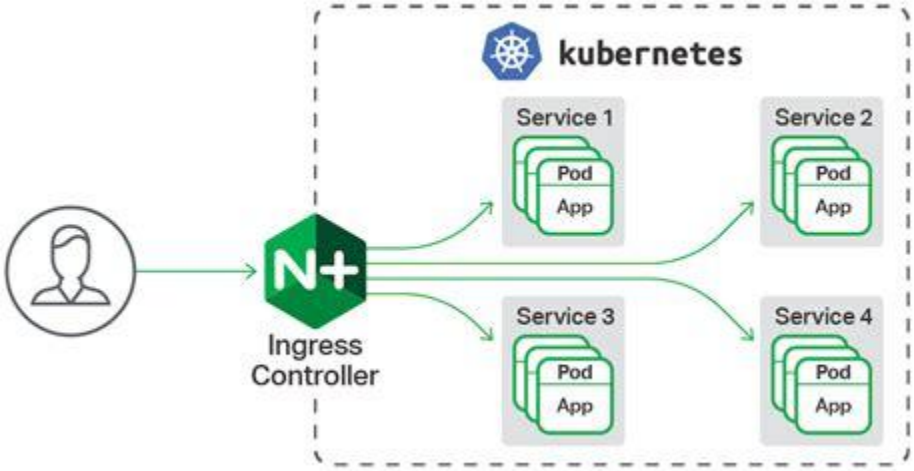

Figure 5 - Ingress Controller

Conclusion

En conclusion de ce HLD, on peut résumer l'architecture "haut niveau", par un environnement Azure Cloud avec Azure Portal, Azure Repo connecté à l'environnement Azure AKS via un container Registry. Azure gère AKS avec son loadbalancer intégré, et nous déployons l'infrastructure via Terraform, des différents pods le tout en AutoScaling. Cette architecture utilise Azure Storage Account et se connecte à PostgreSQL, avec une gestion des secrets par Azure Library Secrets Storage, et du réseau par Azure Virtual Network.

Dans l'ensemble, notre architecture basée sur Azure nous permet de bénéficier d'une infrastructure évolutive, hautement disponible et sécurisée pour notre site e-commerce. Elle offre une gestion simplifiée des conteneurs, un déploiement automatisé, une gestion des accès et des autorisations, ainsi qu'une connectivité sécurisée avec la base de données. Les bonnes pratiques en termes de conventions de nommage, mais aussi d'accès, permettent à la plateforme d'être robuste et fiable, ce qui peut être un gage de confiance pour nos clients.