néosoft

SUPINFO International University

DOCUMENT D'ARCHITECTURE TECHNIQUE (DAT)
PRJ3_LYON2

DESCRIPTION

Ce Document d'Architecture Technique sert à expliciter l'architecture technique de notre solution. Il contient des détails sur les composants réseaux et logiciels, ainsi que sur leurs interactions.

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
01/12/2023	RIVALLAND Gregory	1.1	Création du document
12/12/2023	RIVALLAND Gregory	1.2	Correction et mise au propre
10/01/2024	RIVALLAND Gregory	1.3	Ajout des spécifications techniques et des estimations des coûts

néosoft

SUPINFO International University

SOMMAIRE

1. Introduction

a. Contexte du Projet

Le projet vise à mettre en place une plateforme d'orchestration Kubernetes pour héberger un site E-Commerce basé sur une architecture de microservices. Les services cloud d'Azure (AKS) seront utilisés, et l'automatisation sera intégrée avec Azure DevOps. La sécurité, la gouvernance des données et la gestion des coûts sont des aspects cruciaux.

b. Objectif du Document

Ce Document d'Architecture Technique (DAT) a pour objectif de détailler les choix technologiques ainsi que les spécifications techniques et les processus de mise en place.

2. Réponse Technique et Fonctionnement

a. Choix Technologiques

i. Fournisseur Cloud

Le fournisseur de services cloud choisi est Azure, en raison de son rapport qualité-prix optimal, de son intégration complète avec les outils souhaités (Azure DevOps) et de ses services managés (AKS).

ii. Outils DevOps

Azure DevOps sera l'outil central pour l'automatisation et la gestion du cycle de vie du projet. Les services clés de Repos, Pipelines, Release et Library seront utilisés pour assurer une intégration continue, un déploiement continu et la gestion des configurations.

- **Azure Repos:**
  - Azure Repos est utilisé pour héberger le code source de notre application et de notre infrastructure.
  - Cela permet un suivi complet des versions, une collaboration efficace entre les membres de l'équipe et une intégration transparente avec d'autres services Azure DevOps.

- **Azure Pipelines:**
  - Azure Pipelines automatise le processus d'intégration continue (CI) en générant des artefacts à partir du code source.
  - L'intégration continue garantit une qualité de code élevée, une détection précoce des erreurs et la création régulière d'artefacts prêts pour le déploiement.

- **Azure Release:**
  - Azure Release automatise les étapes de déploiement continu, notamment le provisionnement de l'infrastructure, la construction et l'envoi des images Docker, ainsi que la configuration d'Azure Kubernetes Service (AKS).
  - Cela garantit une livraison rapide et fiable des mises à jour, avec la possibilité de déployer plusieurs environnements de manière cohérente.

- **Azure Library:**
  - Azure Library est utilisé pour la gestion centralisée de la configuration, des variables et des secrets.
  - Cela renforce la sécurité en stockant de manière sécurisée les informations sensibles, tout en permettant un accès contrôlé aux autres services.

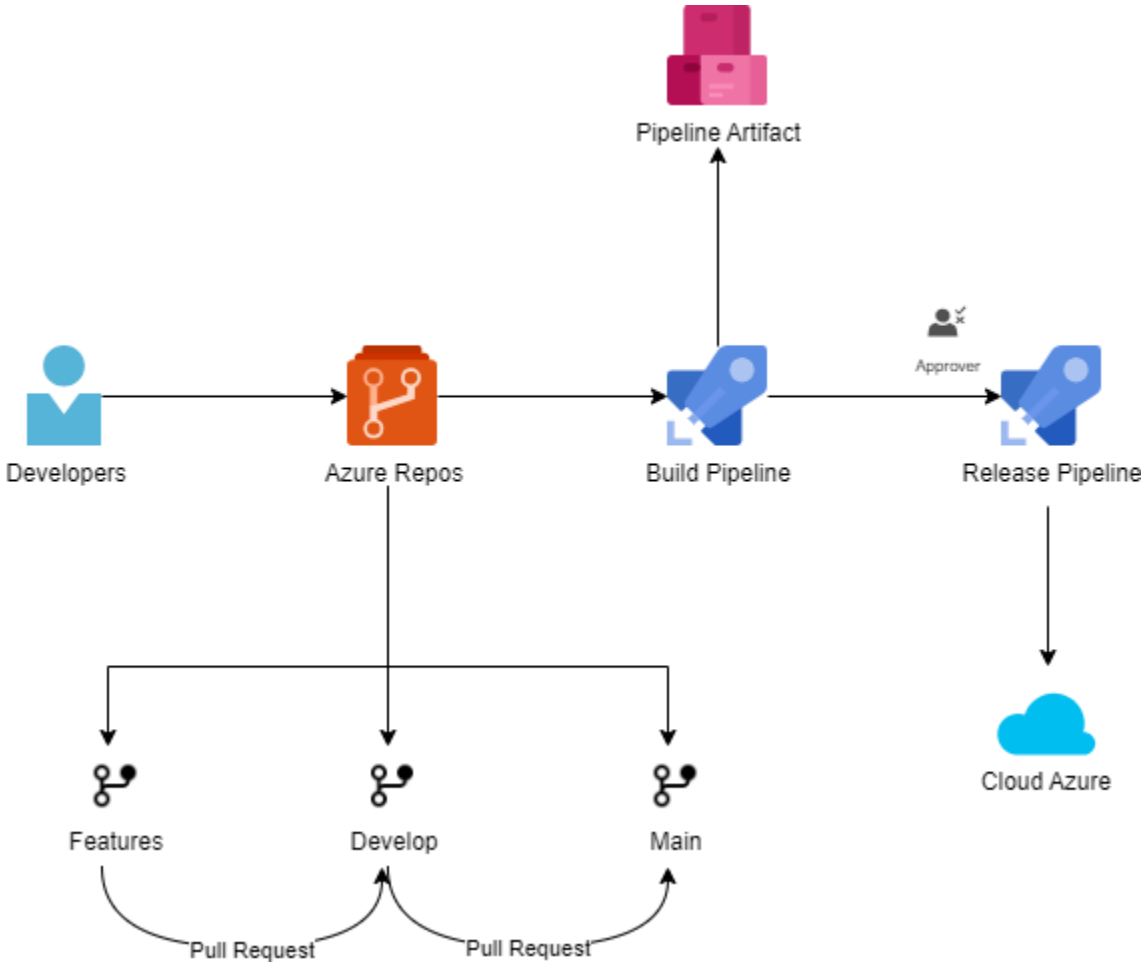


Figure 1 - Schéma du déploiement avec les outils DevOps

Nous utilisons également **Terraform** qui est un outil d'automatisation permettant la création, la modification et la versionnage de l'infrastructure.

- Utilisation de Terraform pour provisionner l'infrastructure, y compris le réseau, les nœuds AKS, et le stockage permanent.
- Terraform offre une approche codifiée pour gérer l'infrastructure, facilitant la reproductibilité, la gestion du cycle de vie des ressources, et assurant une configuration uniforme.

iii. Infrastructure as Code

L'Infrastructure as Code (IaC) est une approche permettant de gérer et de provisionner des infrastructures informatiques de manière automatisée à l'aide de fichiers de configuration plutôt que d'interventions manuelles. Terraform est l'un des outils populaires utilisés pour mettre en œuvre l'IaC. Voici quelques avantages de l'infrastructure as code, avec une focalisation sur Terraform :

Automatisation : L'IaC automatisée avec Terraform permet de déployer et de gérer des infrastructures de manière efficace. Cela réduit le temps et les efforts nécessaires pour effectuer des tâches de provisionnement, de configuration et de gestion des ressources.

Reproductibilité : Les fichiers de configuration Terraform peuvent être versionnés et partagés. Cela garantit la reproductibilité des infrastructures, ce qui signifie que vous pouvez recréer exactement la même infrastructure à tout moment, ce qui est essentiel pour le développement, les tests et la production.

Gestion de l'état : Terraform maintient un état de l'infrastructure déployée. Cela permet à l'outil de comprendre l'état actuel de l'infrastructure et de déterminer les changements nécessaires pour atteindre l'état souhaité. Cela facilite également la collaboration au sein d'une équipe en partageant et en verrouillant l'état de l'infrastructure.

Gestion des versions : Comme les fichiers de configuration Terraform peuvent être versionnés à l'aide de systèmes de contrôle de version tels que Git, il est facile de suivre les modifications apportées à l'infrastructure au fil du temps. Cela facilite la collaboration, la rétrogradation ou la mise à niveau de l'infrastructure en fonction des besoins.

Multi-cloud et hybride : Terraform prend en charge plusieurs fournisseurs de cloud, ce qui permet de créer des configurations d'infrastructure qui peuvent être déployées sur différents fournisseurs de services cloud ou même dans des environnements hybrides (cloud et infrastructure sur site).

Évolutivité : En utilisant Terraform, il est possible de gérer des infrastructures de petite à grande échelle de manière cohérente. Cela facilite la croissance et l'évolution de l'infrastructure en fonction des besoins de l'entreprise.

En résumé, l'utilisation de l'Infrastructure as Code avec Terraform offre des avantages significatifs en termes d'efficacité opérationnelle, de reproductibilité, de collaboration et de gestion automatisée des ressources informatiques.

iv. Services Clés

Les services clés incluent Azure Kubernetes Service (AKS) pour l'orchestration des conteneurs, Azure Container Registry pour le stockage sécurisé des images Docker, et Azure Monitor, Application Insights, Log Analytics pour le monitoring.

- **Azure Kubernetes Service (AKS):** AKS est le service managé de Kubernetes fourni par Microsoft Azure.
    - Responsabilité déléguée:
      Microsoft Azure est responsable de l'infrastructure en général:
        - Gestion du Cluster Kubernetes (Master) :
            - Avec AKS, Microsoft Azure gère le plan de contrôle Kubernetes, y compris la gestion du master Kubernetes. Cela inclut les mises à jour, la haute disponibilité, la sécurité et les opérations liées au plan de contrôle.
        - Mises à Jour du Système d'Exploitation du Nœud :
            - Microsoft Azure est responsable des mises à jour du système d'exploitation du nœud sous-jacent. Cela inclut la maintenance et les mises à jour de sécurité pour les machines virtuelles utilisées pour exécuter les nœuds du cluster.
        - Sécurité de Base :
            - Azure AKS gère la sécurité de base du cluster, y compris la gestion des certificats, le chiffrement des données en transit, et la configuration sécurisée par défaut du cluster.
        - Réseau et Adressage IP :
            - AKS gère le réseau et l'adressage IP du cluster Kubernetes, simplifiant la configuration réseau pour les utilisateurs. Cela inclut la gestion des adresses IP des nœuds et la configuration du réseau Kubernetes.
        - Haute Disponibilité :
            - Microsoft Azure prend en charge la haute disponibilité du plan de contrôle Kubernetes, garantissant la disponibilité continue du cluster même en cas de défaillance d'un nœud ou d'une zone.
        - Intégration avec d'Autres Services Azure :
            - AKS s'intègre facilement avec d'autres services Azure, tels que Azure Active Directory, Azure Monitor, Azure Policy, etc. La gestion de ces intégrations est simplifiée pour les utilisateurs.

En revanche, certaines responsabilités restent sous le contrôle de l'utilisateur ou du développeur, notamment :

- Configuration des Applications et Workloads :
  - L'utilisateur est responsable de la configuration et du déploiement des applications sur le cluster Kubernetes.
- Mises à Jour d'Application :
  - Les mises à jour des applications déployées sur le cluster restent de la responsabilité de l'utilisateur.

- Fonctionnalités et Utilisation:
  - Provisionnement d'un cluster AKS via Terraform.
  - Utilisation d'AKS pour orchestrer les conteneurs des micro-services et gérer la mise à l'échelle automatique.
- AKS simplifie la gestion des clusters Kubernetes, offre des fonctionnalités telles que l'évolutivité automatique, la gestion des mises à niveau, et s'intègre parfaitement avec d'autres services Azure.
- Une estimation des coûts en 'Pay as you go' pour AKS en North Europe, Tier Standard avec 1 cluster ($73.00) et des Nodes Linux ($78.11) est d'environ $151.11.
  En sachant que nous pouvons réduire ce coût mensuel en prenant par exemple:
  - 1 an de savings plan : $140.65
  - 3 ans de savings plan : $126.38
  - 1 an de reserved instances : $124.17
  - 3 ans de reserved instances : $106.75

- **Azure Container Registry:** Azure Container Registry (ACR) sert de référentiel d'images Docker privé.
  - Fonctionnalités et Utilisation:
    - Stockage sécurisé et privé des images Docker.
    - Authentification d'AKS auprès d'ACR pour accéder aux images.
  - Azure Container Registry garantit la confidentialité des images et simplifie le déploiement des conteneurs sur AKS en fournissant un accès sécurisé aux images Docker.

- **Azure Monitor, Application Insights, Log Analytics:** Ces services offrent des fonctionnalités de surveillance, de journalisation et d'analyse des performances.
  - Fonctionnalités et Utilisation:
    - Azure Monitor collecte les métriques et les journaux.
    - Application Insights offre une surveillance des performances des applications.
    - Log Analytics permet d'exécuter des requêtes interactives sur les données de journal.
  - Ces services assurent une observabilité approfondie de l'infrastructure et des applications, facilitant la détection précoce des problèmes et l'optimisation des performances.

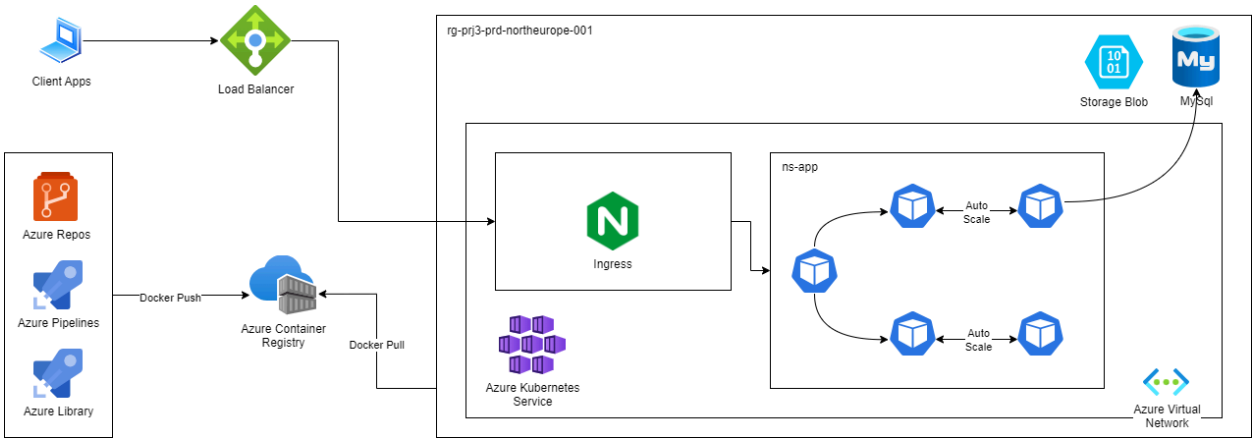


Figure 2 - Diagramme d'architecture AKS

v. Les ressources implémentées et spécifications techniques

Ressource "random_pet" "prefix"

La ressource "random_pet" joue un rôle essentiel en générant un préfixe aléatoire qui est utilisé pour nommer de manière unique les ressources déployées ultérieurement. Ce préfixe aléatoire renforce l'unicité des noms de ressources, contribuant ainsi à éviter tout risque de conflit de noms. Cette mesure de prévention est particulièrement cruciale dans des environnements partagés, où plusieurs projets coexistent, assurant ainsi une isolation efficace entre les différents projets.

Ressource "azurerm_container_registry" "acr"

La ressource "azurerm_container_registry" est responsable de la création d'un Azure Container Registry (ACR) qui offre un espace sécurisé pour stocker les images Docker. Dans le souci d'optimiser les coûts, le mode "Basic" est sélectionné pour les environnements de développement et de test. La désactivation de l'option "admin_enabled" renforce la sécurité en limitant l'accès aux seuls comptes Azure autorisés. De plus, l'attribution du rôle "Acr Pull" au service principal du cluster Kubernetes assure un accès contrôlé aux images, garantissant ainsi une sécurité et une cohérence dans le déploiement des conteneurs.

En terme de spécification technique:

- **sku:** Nous avons choisi de mettre notre SKU en 'Basic' en sachant qu'il a les mêmes fonctionnalités que le SKU 'Standard' mais pour un coût inférieur.
- **admin_enabled:** Ce paramètre détermine si l'authentification de l'administrateur est activée pour le registre. En le définissant sur 'false', l'accès administratif direct est désactivé, ce qui peut être une bonne pratique pour renforcer la sécurité, limitant l'accès direct aux informations d'identification administratives.
- **identity:** Configure l'identité managée associée au registre de conteneurs. Dans ce cas, l'identité est de type 'SystemAssigned', ce qui signifie qu'Azure génère et gère automatiquement les informations d'identification nécessaires. Cela peut simplifier l'intégration avec d'autres services Azure et renforcer la sécurité en évitant l'utilisation explicite de clés d'authentification.

- **tags:** Les tags sont des métadonnées qui peuvent être associées à la ressource. Dans ce code, les tags incluent des informations telles que l'environnement (var.ENVIRONMENT) et le projet (var.PROJECT_NAME). Les tags facilitent l'organisation, la gestion et la recherche de ressources dans Azure en fournissant des informations supplémentaires sur la ressource.

Ressource "azurerm_role_assignment" "acr_pull"

La ressource "azurerm_role_assignment" a pour objectif d'attribuer le rôle "Acr Pull" au service principal du cluster Kubernetes, permettant ainsi un accès autorisé au registre ACR. Cette attribution de rôle garantit que le cluster AKS peut tirer en toute sécurité les images Docker stockées dans le registre ACR, favorisant un déploiement cohérent et sécurisé des conteneurs dans le cluster.

En terme de spécification technique:

- **principal_id:** Ce paramètre spécifie l'identifiant principal (principal ID) de l'entité de sécurité (principale) à laquelle le rôle doit être assigné. Dans ce cas, il s'agit de l'identifiant principal de l'entité de sécurité associée au cluster Kubernetes, récupéré à partir de azurerm_kubernetes_cluster.aks.identity[0].principal_id. Cela garantit que le rôle est attribué à l'entité de sécurité appropriée.
- **scope:** Le paramètre scope spécifie la portée à laquelle le rôle est attribué. Ici, il est défini sur l'ID du registre de conteneurs (azurerm_container_registry.acr.id). Cela signifie que le rôle "AcrPull" est attribué au registre de conteneurs spécifié, limitant l'étendue de l'assignation à ce registre.
- **role_definition_name:** Ce paramètre indique le nom du rôle à assigner. Dans ce cas, le rôle "AcrPull" est attribué, ce qui signifie que l'entité de sécurité (le cluster Kubernetes) aura le droit de tirer des images du registre de conteneurs. Cela est souvent utilisé pour permettre aux clusters Kubernetes d'accéder aux images stockées dans le registre.
- **skip_service_principal_aad_check:** Lorsque ce paramètre est défini sur true, cela indique à Terraform de ne pas effectuer de vérification AAD (Azure Active Directory) sur l'entité de sécurité associée au service principal. Cela peut être utile dans des scénarios où l'entité de sécurité n'est pas inscrite dans Azure AD ou lorsque des configurations spécifiques sont en place.
- **depends_on:** La clause depends_on spécifie les ressources sur lesquelles cette ressource dépend. Dans ce cas, la ressource de rôle d'assignation dépend du déploiement réussi du registre de conteneurs (azurerm_container_registry.acr) et du cluster Kubernetes (azurerm_kubernetes_cluster.aks). Cela assure que les ressources nécessaires sont créées avant l'assignation du rôle.

Ressource "azurerm_kubernetes_cluster" "aks"

La ressource "azurerm_kubernetes_cluster" facilite le déploiement et la configuration d'un cluster Kubernetes, assurant ainsi l'orchestration efficace des conteneurs de l'application. La configuration englobe des paramètres spécifiques tels que la taille des nœuds, l'auto-scaling, et l'utilisation d'un équilibreur de charge pour optimiser la distribution de la charge. L'activation de l'identité système facilite l'intégration transparente avec d'autres services Azure, tandis que l'utilisation de tags permet une gestion organisationnelle efficiente.

En terme de spécification technique:

- **sku_tier:** Définit le niveau de service du cluster Kubernetes. Dans ce cas, le niveau "Standard" est spécifié. Le choix du niveau de service dépend des besoins en termes de fonctionnalités et de performances. "Standard" est souvent utilisé pour les charges de travail de production.
- **dns_prefix:** Ce paramètre spécifie le préfixe DNS pour le cluster. Il est généré en utilisant un identifiant aléatoire, ce qui garantit l'unicité du préfixe DNS.
- **default_node_pool:** Configure le pool de nœuds par défaut du cluster Kubernetes, spécifiant des détails tels que la taille de la machine virtuelle, la mise à l'échelle automatique et la taille du disque OS. Cela permet de définir les caractéristiques de base du pool de nœuds.
- **auto_scaler_profile:** Configure le profil de l'auto-scaler, définissant des paramètres tels que l'intervalle de balayage, l'équilibre des groupes de nœuds similaires, et l'expander. L'auto-scaling est une fonctionnalité cruciale pour ajuster automatiquement le nombre de nœuds en fonction de la charge de travail.
- **identity:** Configure l'identité managée associée au cluster Kubernetes. Ici, l'identité est de type "SystemAssigned", ce qui signifie qu'Azure génère et gère automatiquement les informations d'identification nécessaires. Cela peut simplifier l'intégration avec d'autres services Azure et renforcer la sécurité en évitant l'utilisation explicite de clés d'authentification.
- **tags:** Les tags sont des métadonnées qui peuvent être associées à la ressource. Dans ce code, les tags incluent des informations telles que l'environnement (var.ENVIRONMENT) et le projet (var.PROJECT_NAME). Les tags facilitent l'organisation, la gestion et la recherche de ressources dans Azure en fournissant des informations supplémentaires sur la ressource.

Ressource "azurerm_mysql_server" "mysql_server"

La ressource "azurerm_mysql_server" est responsable du déploiement d'un serveur MySQL destiné au stockage des données de l'application PrestaShop. La configuration englobe des paramètres tels que la version MySQL, la taille de stockage, et la désactivation de certaines fonctionnalités, telles que SSL, pour optimiser les coûts. L'activation de l'identité système renforce la sécurité, et l'accès à la base de données est soigneusement restreint aux services Azure autorisés grâce à des règles de pare-feu.

En terme de spécification technique:

- **administrator_login && administrator_login_password:** Ces paramètres définissent le nom de l'administrateur et le mot de passe pour le serveur MySQL. Ils sont généralement fournis en tant que variables (var.MYSQL_ADMIN_LOGIN et var.MYSQL_ADMIN_PASSWORD dans ce cas). Ils sont essentiels pour l'accès et la gestion du serveur.
- **sku_name:** Définit le nom du niveau de service du serveur MySQL. Dans ce cas, le choix du niveau de service dépend de l'environnement, avec des valeurs différentes pour la production ("B_Gen5_1") et le développement ("B_Gen5_2").
- **storage_mb:** Ce paramètre spécifie la taille de stockage en mégaoctets pour le serveur MySQL. Il est fixé à 5120 Mo dans ce cas, mais peut être ajusté en fonction des besoins de l'application.
- **version:** Indique la version de MySQL à déployer. Dans ce cas, la version 5.7 est spécifiée.
- **auto_grow_enabled:** Désactive l'option de croissance automatique pour le stockage. Cela peut être utile pour contrôler les coûts dans un environnement de développement ou si des limites strictes de ressources sont nécessaires.

- **backup_retention_days:** Définit le nombre de jours pendant lesquels les sauvegardes sont conservées. Ici, il est fixé à 7 jours, mais cela peut être ajusté en fonction des politiques de sauvegarde spécifiques.
- **geo_redundant_backup_enabled:** Désactive la sauvegarde géo-redondante. Cela peut être ajusté en fonction des exigences de reprise après sinistre et de continuité des activités.
- **infrastructure_encryption_enabled:** Désactive le chiffrement de l'infrastructure pour le stockage. Cela peut être ajusté en fonction des besoins en matière de sécurité.
- **punlic_network_access_enabled:** Autorise ou interdit l'accès au serveur MySQL depuis Internet. Dans ce cas, l'accès depuis Internet est activé.
- **ssl_enforcement_enabled && ssl_minimal_tls_version_enforced:** Désactive l'application de l'authentification SSL. Dans ce projet, il est spécifié que SSL n'est pas activé en raison de l'absence de domaine personnalisé et de certificat.
- **identity:** Configure l'identité managée associée au serveur MySQL. Ici, l'identité est de type "SystemAssigned", ce qui signifie qu'Azure génère et gère automatiquement les informations d'identification nécessaires. Cela peut simplifier l'intégration avec d'autres services Azure et renforcer la sécurité en évitant l'utilisation explicite de clés d'authentification.
- **tags:** Les tags sont des métadonnées qui peuvent être associées à la ressource. Dans ce code, les tags incluent des informations telles que l'environnement (var.ENVIRONMENT) et le projet (var.PROJECT_NAME). Les tags facilitent l'organisation, la gestion et la recherche de ressources dans Azure en fournissant des informations supplémentaires sur la ressource.

Ressource "azurerm_mysql_database" "mysql_database"

La ressource "azurerm_mysql_database" vise la création d'une base de données MySQL dédiée au stockage des données de l'application PrestaShop. Cette ressource configure des paramètres spécifiques à la base de données, tels que le jeu de caractères et le mode de comparaison. Elle est étroitement liée au serveur MySQL créé précédemment et représente la base de données utilisée par l'application PrestaShop, contribuant ainsi à une gestion centralisée et cohérente des données.

En terme de spécification technique:

- **server_name:** Ce paramètre spécifie le nom du serveur MySQL auquel la base de données doit être associée. Il utilise la référence à la ressource du serveur MySQL (azurerm_mysql_server.mysql_server.name), assurant ainsi que la base de données est liée au serveur approprié.
- **charset && collation:** Ces paramètres définissent respectivement le jeu de caractères et la collation de la base de données MySQL. Ici, ils sont définis sur "utf8" et "utf8_unicode_ci", respectivement, assurant un encodage approprié et la prise en charge des caractères Unicode.

Ressource "azurerm_mysql_firewall_rule" "firewall_rule_allow_azure_services"

La ressource "azurerm_mysql_firewall_rule" a pour objectif d'autoriser l'accès à la base de données MySQL depuis tous les services Azure. Cette règle de pare-feu spécifique permet à tous les services

Azure d'accéder à la base de données MySQL, assurant ainsi une connectivité sans entrave avec d'autres services au sein de l'écosystème Azure.
En terme de spécification technique:

- **server_name:** Ce paramètre spécifie le nom du serveur MySQL auquel la règle de pare-feu doit être associée. Il utilise la référence à la ressource du serveur MySQL (azurerm_mysql_server.mysql_server.name), assurant ainsi que la règle est liée au serveur approprié.
- **start_ip_address et end_ip_address:** Ces paramètres définissent la plage d'adresses IP pour laquelle la règle de pare-feu est applicable. Dans ce cas, "0.0.0.0" est spécifié pour les deux, ce qui signifie que la règle permet l'accès depuis toutes les adresses IP. Cela correspond à l'autorisation de tous les services internes d'Azure d' accéder à la base de données MySQL.
- **depends_on:** La clause depends_on spécifie les ressources sur lesquelles cette ressource dépend. Dans ce cas, la règle de pare-feu dépend du déploiement réussi du serveur MySQL (azurerm_mysql_server.mysql_server) et de la base de données MySQL (azurerm_mysql_database.mysql_database). Cela assure que les ressources nécessaires sont créées avant la configuration de la règle de pare-feu.

b. Processus de Mise en Place

i. Environnements: Dev & Prod

Dans notre projet, nous avons deux environnements differents, le Développement (DEV) pour les tests et la Production (PROD) pour les utilisateurs finaux..

Variables d'Environnement
L'utilisation de variables d'environnement dans notre configuration Terraform joue un rôle central dans la création de ressources adaptées à chaque environnement. Nous avons défini une variable spécifique, appelée ENVIRONMENT, qui prend la valeur "dev" pour l'environnement de développement et "prd" pour l'environnement de production.

Adaptation Dynamique des Ressources
Lorsque Terraform est exécuté, il détecte la valeur de la variable d'environnement et ajuste dynamiquement la configuration en conséquence. Cela se traduit par la création de ressources avec des spécificités propres à chaque environnement, assurant ainsi une cohérence tout en permettant des configurations distinctes.

Maintenabilité
Cette approche simplifie la maintenance en évitant la duplication de code. Les ajustements nécessaires pour chaque environnement sont centralisés grâce aux variables d'environnements. Les composantes du système comme Terraform, la configuration Kubernetes et les différents scripts adaptant leurs actions en fonction de celles-ci.

ii. Infrastructure

1. **Provisionnement du Stockage Partagé (Azure CLI):**

Le stockage partagé constitue un élément essentiel pour garantir la stabilité et la cohérence de l'infrastructure. Afin de répondre à cette nécessité, le processus débute par la création d'un stockage partagé avec Azure CLI. Ce stockage hébergera le fichier "tfstate" de Terraform, assurant ainsi la gestion optimale de l'état et du cycle de vie de l'infrastructure. Cette approche évite la modification simultanée, prévient toute perte de données et permet des opérations sécurisées grâce aux verrous opérés par Terraform sur le fichier "tfstate" lors des opérations.

Methodologie:

- Utilisation de commandes et scripts Azure CLI dans une pipeline Azure DevOps.
- Création du compte de stockage, du conteneur, et configuration des paramètres nécessaires.

2. **Provisionnement de l'Infrastructure avec Terraform:**

L'étape suivante du processus consiste à utiliser Terraform pour provisionner l'ensemble de l'infrastructure, en commençant par les composantes réseau. Terraform, en tant qu'outil d'automatisation, permet une gestion efficace du cycle de vie des ressources, garantissant une configuration reproductible et conforme aux besoins du projet.

Methodologie:

- Intégration de scripts Terraform dans le référentiel Azure DevOps.
- Configuration de l'environnement Terraform avec les fichiers de configuration et l'accès au stockage partagé.
- Exécution d'une tâche Terraform dans un pipeline Azure DevOps pour déployer les ressources réseau et autres composantes de l'infrastructure.

3. **Construction des Images Docker et Déploiement dans Azure Container Registry:**

La phase suivante du processus concerne la construction des images Docker à partir du code source de l'application, suivi du déploiement sécurisé dans Azure Container Registry (ACR). Cette étape est cruciale pour assurer la gestion efficace des images et la préparation optimale des conteneurs pour le déploiement ultérieur.

Methodologie:

- Utilisation du service Azure DevOps Pipelines.
- Configuration d'une tâche de build pour exécuter les étapes nécessaires, telles que la compilation du code, la construction des images Docker, etc.
- Envoi sécurisé des images vers Azure Container Registry, en utilisant les informations d'authentification stockées de manière sécurisée dans Azure DevOps Library.

4. **Configuration du Service AKS pour Ordonnancer les Conteneurs:**

La dernière étape du processus de mise en place concerne la configuration du service Azure Kubernetes Service (AKS) pour orchestrer les conteneurs. AKS simplifie le déploiement d'un cluster Kubernetes géré

dans Azure, offrant une mise à l'échelle automatique, une gestion simplifiée, et une optimisation des coûts.
Methodologie:

- Préparation de l'environnement Kubernetes avec des scripts et fichiers de configuration stockés dans le référentiel Azure DevOps.
- Utilisation d'une tâche Azure DevOps pour injecter la configuration Kubernetes dans AKS.
- AKS prend en charge la configuration, créant les ressources et réalisant les actions spécifiées, orchestrant ainsi les conteneurs de manière optimale.

iii. Automatisation avec Azure DevOps

L'intégration d'Azure DevOps dans notre processus de développement garantit une automatisation complète du cycle de vie, allant de la gestion du code source à la livraison continue. Voici comment chaque service de DevOps est utilisé pour optimiser et simplifier le processus.

Utilisation des Services DevOps:

- **Azure Repos (Stockage du Code) :**
  - Azure Repos est utilisé comme système de contrôle de version pour stocker de manière sécurisée le code source.
- **Azure Pipelines (Intégration Continue) :**
  - Les pipelines Azure DevOps sont configurés pour l'intégration continue, déclenchés automatiquement à chaque commit.
  - Le pipeline intègre les tests automatisés, garantissant la qualité du code avant le déploiement.
- **Azure Release (Déploiement Continu) :**
  - Azure Release est utilisé pour le déploiement continu des infrastructures Terraform dans les environnements cibles (Dev, Prd).
  - Les dépendances entre les différentes étapes de déploiement sont gérées de manière à assurer une cohérence dans les déploiements.
- **Azure Library (Gestion des Secrets) :**
  - La Library d'Azure DevOps est utilisée pour stocker de manière centralisée les secrets et configurations sensibles, assurant un accès sécurisé.
  - Les variables de la Library sont intégrées dans les pipelines, permettant une gestion centralisée des paramètres sensibles.

Stockage Partagé de "tfstate" : Le fichier "tfstate" essentiel pour maintenir l'état de l'infrastructure Terraform, est stocké de manière partagée pour éviter des modifications concurrentes, garantissant la stabilité de l'infrastructure.

Création d'Artéfacts via Pipelines : Les pipelines Azure DevOps créeront des artefacts à partir du code source, assurant une intégration continue fluide.

Gestion des Secrets et Configurations avec Library : La gestion centralisée des secrets et configurations sensibles permet de garantir la sécurité des informations sensibles et sera assurée par le service Library.

3. Architecture Globale

a. Architecture Microservices

Le système sera conçu comme une plateforme Kubernetes en mode multi-couche, intégrant différentes couches pour répondre aux besoins spécifiques du projet.

1. **Couche Applicative avec des microservices :** La couche applicative reposera sur une architecture de microservices, permettant une modularité accrue, une évolutivité simplifiée et une maintenance facilitée. Chaque microservice sera responsable d'une fonctionnalité spécifique du système, favorisant la séparation des préoccupations et la résilience.

2. **Couche de Services:** Cette couche comprendra une base de données PostgreSQL pour le stockage persistant des données, garantissant la fiabilité et la cohérence des informations. Un service de cache Redis sera également intégré pour améliorer les performances en stockant temporairement les données fréquemment utilisées.

3. **Couche d'Infrastructure avec AKS et Azure Container Registry :**

    a. **Azure Kubernetes Service (AKS) :** Utilisé pour orchestrer et gérer les conteneurs des microservices. AKS offre une mise à l'échelle automatique, une gestion des versions simplifiée et une disponibilité élevée.
    b. **Azure Container Registry (ACR) :** Servira de référentiel sécurisé pour stocker les images Docker des microservices. ACR facilite la gestion des versions, le suivi des modifications et garantit un déploiement fiable.

4. **Couche Réseau pour l'Isolation et la Gestion du Trafic :** La couche réseau sera configurée pour assurer une isolation efficace entre les différentes composantes du système. Les services seront exposés via des points d'entrée bien définis. Des politiques de réseau strictes seront mises en place pour contrôler le flux du trafic et garantir la sécurité.

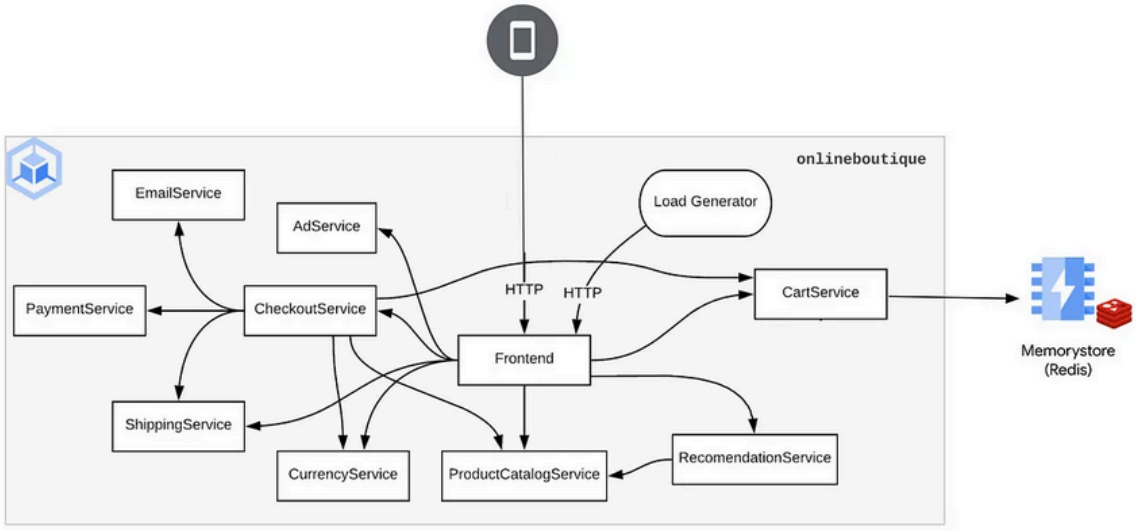


Figure 3 - Schéma de l'architecture en microservices

Cette architecture microservices, combinée à une plateforme Kubernetes, permettra une évolutivité horizontale facile, une gestion efficace des ressources, et une résilience accrue du système. Chaque couche est soigneusement conçue pour répondre aux exigences spécifiques tout en favorisant la cohérence globale de l'architecture.

b. Technologies et Outils

L'architecture globale du système s'appuie sur un ensemble de technologies et d'outils puissants, soigneusement sélectionnés pour assurer une performance optimale, une évolutivité maximale et une gestion efficace du cycle de vie du développement.

1. **Azure Kubernetes Service (AKS) - Services Kubernetes :** AKS sera utilisé comme service d'orchestration des conteneurs. Il simplifie le déploiement, la gestion et l'opération des applications conteneurisées à l'échelle, tout en garantissant une haute disponibilité et une élasticité automatique.
2. **Azure DevOps - Plateforme DevOps Complète :** Azure DevOps englobe l'intégralité du cycle de vie DevOps, fournissant des services tels que Repos, Pipelines, Release et Library. Cette plateforme complète facilite la collaboration, l'automatisation des déploiements, et la gestion centralisée des artefacts et des configurations.

3. **Docker - Technologie de Conteneurisation :** Docker sera utilisé pour encapsuler chaque microservice et ses dépendances en conteneurs. Cette approche offre une portabilité accrue, une gestion simplifiée des dépendances, et une isolation efficace des applications.
4. **Terraform - Automatisation de la Création des Ressources :** Terraform sera le moteur d'automatisation pour la création de l'infrastructure. Les fichiers de configuration Terraform définiront de manière déclarative l'ensemble des ressources nécessaires, offrant ainsi une gestion efficace et reproductible des infrastructures en tant que code.
5. **PostgreSQL - Système de Base de Données Relationnelles SQL :** PostgreSQL sera le système de gestion de base de données relationnelles utilisé pour la couche de services. Il offre des performances élevées, une extensibilité robuste et une compatibilité avec les normes SQL.

Ces technologies et outils sont intégrés de manière cohérente pour former une architecture solide, répondant aux exigences spécifiques du projet. Chacun contribue à l'efficacité opérationnelle, à la sécurité et à la qualité globale du système.

4. Convention et Bonnes Pratiques

a. Sécurité

Adoption de l'approche "Security by Design": L'ensemble du système est conçu en suivant les principes de "Security by Design". Cela signifie que la sécurité est intégrée dès la phase de conception du système, garantissant ainsi une protection intrinsèque contre les menaces potentielles.

Respect des principes du RGPD et du triptyque CIA: Le projet se conforme rigoureusement aux principes du Règlement Général sur la Protection des Données (RGPD). De plus, les principes du triptyque CIA (Confidentialité, Intégrité, Disponibilité) sont respectés à chaque étape du cycle de vie des données, assurant ainsi une gestion robuste de la sécurité.

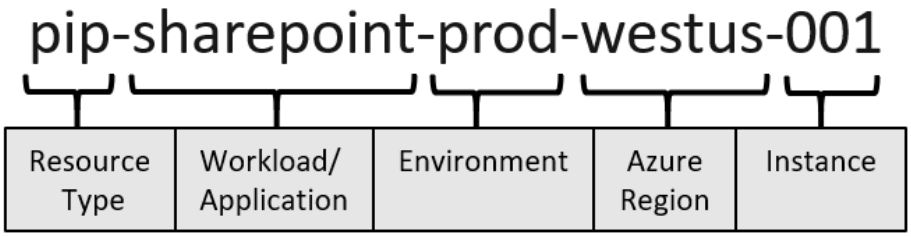
b. Gouvernance des Données

Respect des régulations sur la protection des données: La gestion des données est conforme aux régulations en vigueur sur la protection des données. Les politiques de gouvernance des données sont élaborées et mises en œuvre en tenant compte des normes légales et industrielles.

Responsabilités partagées entre Azure et l'équipe du projet: Les responsabilités en matière de sécurité des données sont clairement définies, avec une collaboration étroite entre Azure en tant que fournisseur de services cloud et l'équipe du projet. Cette approche garantit une sécurité complète de bout en bout.

c. Convention de Nommage

Utilisation de conventions claires et cohérentes pour les ressources Azure: Toutes les ressources déployées sur la plateforme Azure suivent des conventions de nommage claires et cohérentes. Cela facilite la gestion, la recherche et la compréhension des ressources, contribuant ainsi à une gestion efficace de l'infrastructure.



d. Terraform

Partage du fichier "tfstate" pour éviter les modifications simultanées: Le fichier "tfstate" utilisé par Terraform pour suivre l'état de l'infrastructure est partagé de manière centralisée. Cela évite les conflits liés aux modifications simultanées et assure une gestion collaborative et sécurisée de l'infrastructure en tant que code.

e. Monitoring

Mise en place d'un système de monitoring pour une surveillance en temps réel: Un système de monitoring robuste est mis en place pour assurer une surveillance continue en temps réel. Cela permet une détection proactive des incidents, garantissant une réactivité immédiate en cas de problème.

5. Gestion des Coûts

Le suivi des alertes de coûts est une composante essentielle de la gestion financière d'une infrastructure cloud. Il permet une réactivité accrue face aux variations budgétaires et assure une utilisation optimale des ressources tout en évitant les dépassements imprévus. Voici une approche détaillée pour le suivi des alertes de coûts :

Configuration des Alertes:

- **Définition des Seuils:** Établir des seuils de coûts préventifs en fonction des budgets alloués à chaque service ou projet.

- **Granularité des Alertes:** Configurer des alertes à différents niveaux de granularité, que ce soit par ressource individuelle, groupe de ressources, service, ou projet.

Utilisation des Services Cloud:

- **Tableau de bord de coûts:** Exploiter les tableaux de bord de coûts fournis par le fournisseur cloud pour une visualisation claire et détaillée des dépenses.
- **Alertes personnalisées:** Configurer des alertes personnalisées basées sur des métriques spécifiques, telles que le coût quotidien ou hebdomadaire.

Réaction aux Alertes:

- **Plan d'Action Préétabli:** Définir un plan d'action clair en cas de déclenchement d'une alerte, spécifiant les étapes à suivre pour réduire les coûts.
- **Responsabilités assignées:** Attribuer des responsabilités claires pour chaque type d'alerte, assurant ainsi une réponse rapide et organisée.

Analyse des Alertes:

- **Revues Régulières:** Effectuer des revues régulières des alertes pour identifier les tendances, les variations saisonnières, et les anomalies.
- **Analyse des Causes Racines:** En cas de dépassement des seuils, mener des analyses approfondies pour comprendre les causes sous-jacentes et éviter les récidives.

Optimisation Continue:

- **Réajustement des Seuils:** Ajuster périodiquement les seuils d'alerte en fonction des changements dans les besoins du projet ou des fluctuations du marché.
- **Rétroaction et amélioration continue:** Capitaliser sur les enseignements tirés des alertes précédentes pour améliorer continuellement les processus et les prévisions budgétaires.

En suivant ces bonnes pratiques, l'équipe peut s'assurer que le suivi des alertes de coûts devient un processus agile et proactif, contribuant ainsi à la maîtrise des dépenses cloud et à l'efficacité opérationnelle.

6. Gestion de Kubernetes avec Terraform

La gestion de Kubernetes avec Terraform offre une approche puissante et cohérente pour provisionner, configurer et gérer des clusters Kubernetes. Voici comment cette gestion peut être mise en œuvre de manière efficace :

Provisionner le Cluster Kubernetes:

- **Définir l'Infrastructure:** L'infrastructure sous-jacente nécessaire au cluster Kubernetes sera définie en utilisant Terraform. Cela inclura la création de machines virtuelles, la configuration du réseau, et d'autres ressources essentielles pour le bon fonctionnement du cluster.

Configurer le Cluster Kubernetes:

- **Déclarer les Ressources Kubernetes:** Terraform sera utilisé pour déclarer de manière déclarative les ressources Kubernetes essentielles. Cela inclut la définition des déploiements pour déployer des applications, la création de services pour permettre la communication entre les différents composants, ainsi que la gestion des secrets pour sécuriser les informations sensibles.

- **Gérer les Configurations Sensibles:** Pour renforcer la sécurité, des pratiques spécifiques seront mises en œuvre concernant la gestion des configurations sensibles. Terraform propose des fonctionnalités telles que la gestion des secrets, permettant de stocker en toute sécurité des informations confidentielles. De plus, des techniques de gestion externe des secrets seront intégrées pour garantir la protection des données sensibles. Ces mesures visent à réduire les risques potentiels liés à la manipulation d'informations confidentielles au sein du cluster Kubernetes.

Gérer les Mises à Jour:

- **Faire des Mises à Jour Déclaratives:** Les mises à jour du cluster Kubernetes seront réalisées de manière déclarative en utilisant Terraform. Cette approche implique la spécification des versions de Kubernetes et d'autres composants directement dans le code Terraform, permettant ainsi une gestion centralisée et traçable des mises à jour. En définissant ces informations de manière explicite, le processus de mise à jour devient transparent et reproductible.

- **Effectuer des Tests Post-Mise à Jour:** Un aspect critique du processus de mise à jour consistera à intégrer des tests post-mise à jour. Ces tests visent à évaluer la stabilité et la performance du cluster après l'application des mises à jour. Les tests post-mise à jour sont essentiels pour détecter rapidement tout problème éventuel et garantir que le cluster reste opérationnel et conforme aux exigences du projet. Cette approche proactive contribue à assurer la fiabilité et la continuité des opérations du cluster Kubernetes suite aux mises à jour effectuées.

Intégrer avec les Pipelines CI/CD:

- **Automatiser les Déploiements:** L'automatisation des déploiements du cluster Kubernetes sera intégrée dans les pipelines CI/CD à l'aide d'Azure DevOps. Cette approche vise à garantir une livraison continue et fiable en automatisant le provisionnement initial du cluster ainsi que les mises à jour ultérieures. En incluant ces étapes cruciales dans le pipeline CI/CD, le processus devient transparent, reproductible, et répond aux principes fondamentaux de la méthodologie DevOps.

- **Réaliser des Tests Automatisés:** Dans le cadre du processus CI/CD, des tests automatisés seront mis en place pour valider la configuration du cluster Kubernetes et s'assurer du bon déroulement des mises à jour. Ces tests automatisés couvrent divers aspects tels que la stabilité, la performance, et la conformité aux spécifications du projet. L'intégration de tests automatisés contribue à identifier rapidement les éventuels problèmes, garantissant ainsi la qualité du cluster tout au long du cycle de vie du développement. Cette approche renforce la confiance dans les déploiements automatisés et accélère la livraison des nouvelles fonctionnalités.

Monitoring et Troubleshooting:

- **Intégrer des Outils de Surveillance:** L'intégration d'outils de surveillance compatibles avec Kubernetes sera une étape cruciale pour garantir la détection proactive des problèmes de performance ou des pannes. Ces outils permettront une surveillance en temps réel de l'état du cluster, des applications, et des ressources. En détectant rapidement les anomalies, cette approche permet une intervention précoce et contribue à maintenir la disponibilité et la fiabilité du système.

- **Collecter les Logs et Traces:** Une gestion efficace des logs et des traces sera assurée en intégrant des solutions dédiées dans l'environnement Kubernetes. Cette pratique facilite le dépannage en fournissant une visibilité détaillée sur le comportement des applications et des composants du cluster. L'intégration de solutions de logs et de traces contribue à accélérer la résolution des problèmes en identifiant rapidement les sources d'erreurs et en fournissant des informations précieuses pour comprendre les flux d'exécution. En résumé, cette approche renforce la capacité à assurer un fonctionnement stable et réactif du cluster Kubernetes.

Conclusion

L'architecture proposée, basée sur Microsoft Azure, offre une plateforme évolutive, hautement disponible et sécurisée pour le site E-Commerce en utilisant les bonnes pratiques DevOps. Les choix technologiques, les processus de mise en place, les recommandations de sécurité, et les conventions définies dans ce DAT constituent un guide complet pour l'équipe technique tout au long du projet. La sécurité, la gouvernance des données, la convention de nommage, et la surveillance sont au cœur de la conception pour assurer la robustesse et la fiabilité de la plateforme.