CLO5 - My Booking Service

I - Introduction

A. Organisation

Pour répondre aux besoins du client, nous avons choisi de suivre une méthodologie Agile avec des sprints de 3 semaines. Les cérémonies retenues sont les daily un jour sur deux, un weekly chaque jeudi et un monthly tous les 3 vendredis. Les journées seront ponctuées de séances de peer-programming entre les profils concernés par la tâche en cours (entre DevOps, entre Devs ou entre DevOps et Devs).

Lors des monthly, nous ferons le point sur l'évolution du projet, le respect des deadlines et des objectifs, la bonne répartition des tâches et les nouveaux objectifs si besoin.

B. Profils

Pour la réparation des tâches nous avons fait en fonction des expertises de chacun en effet :

Alan est DevOps, il pourra apporter son expérience et ses connaissances sur la CI/CD et apporter du soutien sur le maintien de l'infrastructure.

Maxime a un profil multi-casquette (Dev/DevOps/Ops). Il se concentrera sur la configuration du cluster Kubernetes et d'Istio, ainsi que du déploiement des API sur le cluster. L'accent sera mis sur les stratégies de déploiement. Il accompagnera aussi les développeurs sur les bonnes pratiques de développement pour ce genre d'environnement.

Randy est développeur Full-Stack spécialisé dans les projets Symfony, il pourra apporter un jugement sur le développement des API et des tests.

Corentin aussi est développeur Full-Stack, il a l'habitude de travailler aussi bien sur le front-end que sur le back-end. Ainsi, il a su développer une réflexion pertinente sur le jugement des fonctionnalités à développer. Il pourra facilement détecter et remonter les anomalies et bugs d'où son rôle de lead test.

C. Technologies

Nous utiliserons les outils suivants :

- Ansible pour la configuration de l'infrastructure (Kubernetes + Istio)
- Ansible Vault pour chiffrer les données sensibles
- GitLab pour la gestion du pipeline de déploiement
- Containerd pour le runtime de nos containers
- Kubernetes pour l'orchestration de conteneurs
- Ingress Nginx pour la gestion de points d'entrés

- Calico pour la gestion du trafic réseau
- la stack TICK pour le monitoring
- Gitlab Runner (Kubernetes) pour la CI/CD

Pour développer nos API, FastAPI est la technologie la plus simple pour faire des API minimalistes de plus c'est un langage universel : le python ce choix de technologies a été vu par le Lead Dev.
Les tests unitaires seront dans le même langage que leur API, il conviendra au Test Lead de s'adapter au choix qui aura été retenu. Nos données seront mockées afin de rendre les tests unitaires indépendants des autres API.
Nous privilégions une documentation Swagger pour fournir une interface graphique permettant d'expérimenter avec nos API.

II - Découpage en micro-services

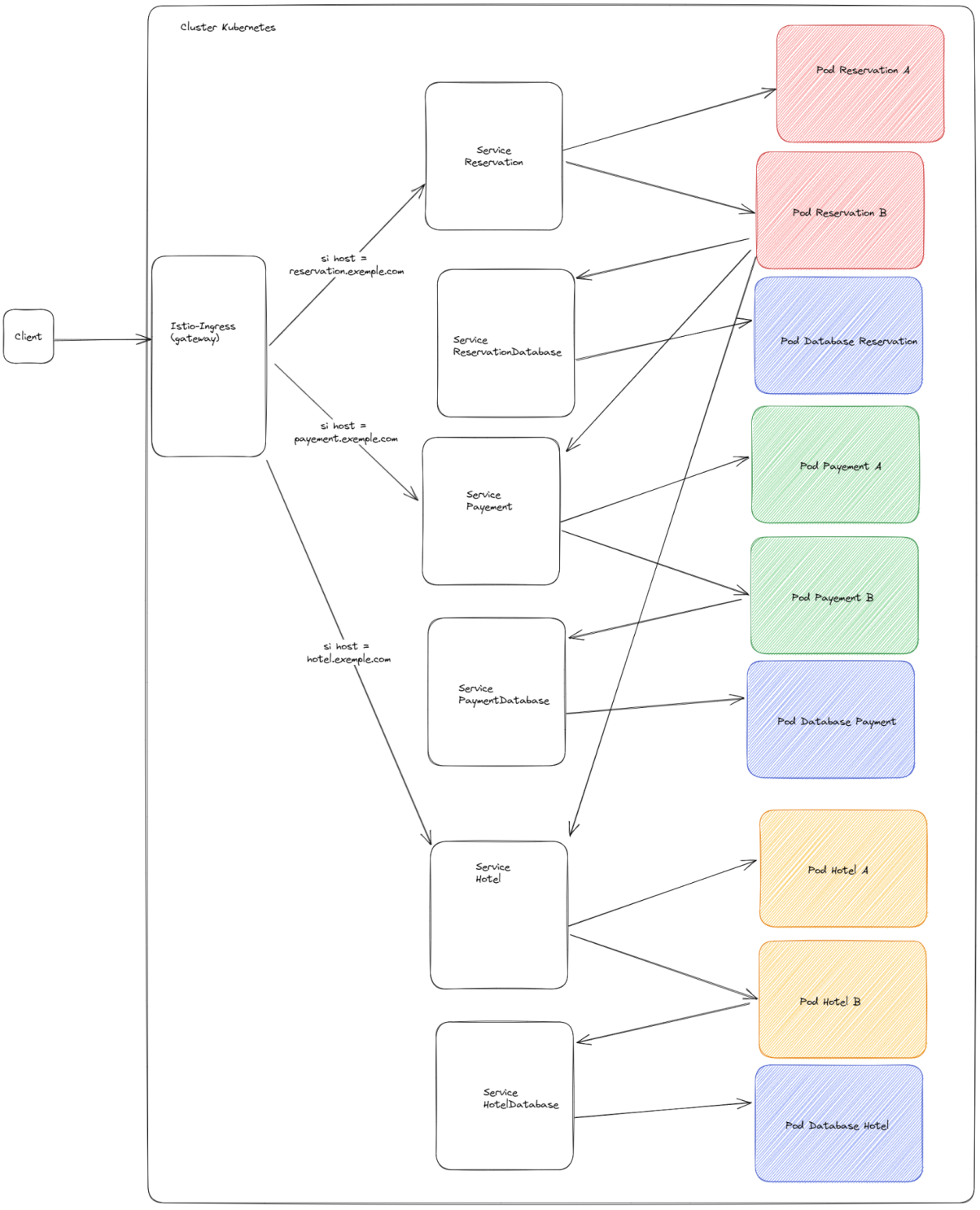
Le découpage du besoin en microservices que nous avons réalisé est comme suit :

- Service de réservation et de paiement : ce microservice serait responsable de gérer les réservations et les annulations de réservations. Il s'occuperait de la vérification de la disponibilité des chambres, du calcul du prix de la réservation, de l'envoi de la confirmation de réservation par e-mail, et de la gestion des annulations de réservations.

- Service de gestion des hôtels : ce microservice serait responsable de gérer les informations sur les différents hôtels, tels que les informations sur les chambres disponibles, les tarifs, les services proposés, etc. Il fournira ces informations au service de réservation pour aider à la prise de décision sur la disponibilité des chambres et le calcul des prix.

- Service de paiement : ce microservice serait responsable de la gestion des paiements pour les réservations de chambres d'hôtel. Il s'occuperait de l'autorisation des paiements par carte de crédit, de la collecte des paiements et de la notification du service de réservation une fois le paiement accepté.

Ce schéma ci-dessous représente le fonctionnement simplifié d'un appel HTTP vers un applicatif hébergé sur un cluster Kubernetes utilisant Istio pour faire du service-meshing. Les pods applicatifs seront stateless et les bases de données seront stateful avec des volumes de stockage réservés.


Cluster Kubernetes
Client
Istio-Ingress
(gateway)
si host =
reservation.exemple.com
si host =
payement.exemple.com
si host =
hotel.exemple.com
Service
Reservation
Service
ReservationDatabase
Service
Payement
Service
PaymentDatabase
Service
Hotel
Service
HotelDatabase
Pod Reservation A
Pod Reservation B
Pod Database Reservation
Pod Payement A
Pod Payement B
Pod Database Payment
Pod Hotel A
Pod Hotel B
Pod Database Hotel

III - Fonctionnement et intégration du Git Flow

A. Environnements

Nous disposons actuellement d'un cluster Kubernetes. Idéalement, nous aurions un cluster par environnement. Afin de respecter les ressources attribuées, nous simulons un environnement de Production et de Pré-production en les isolant par namespace et par sous-domaine.

Par exemple :

- Production : hotel-api.example.local
- Préproduction : hotel-api-preprod.example.local

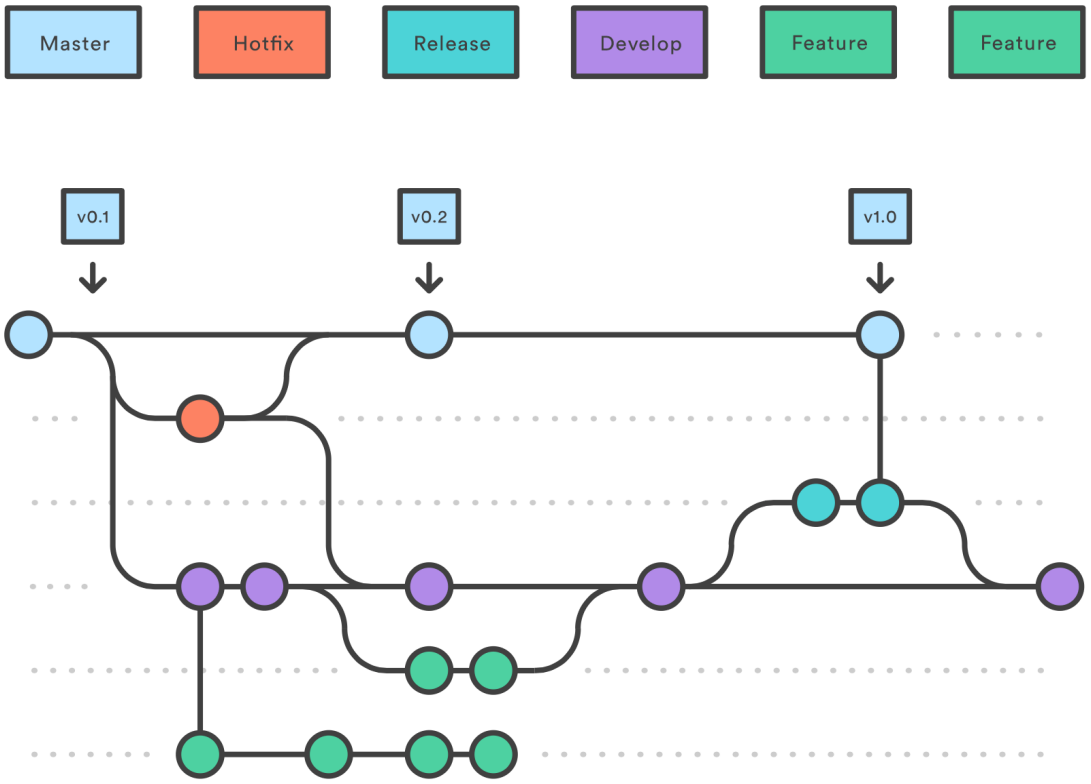
B. Politique de branche

La mise en place d'une politique de branche pour empêcher de pousser des modifications sur la branche main est primordiale afin de s'assurer de la cohérence des modifications et de la qualité du code.

Pour pousser des modifications sur la branche de développement, il faudra passer par une Merge Request, qui déclenche le lancement des tests unitaires sur la branche à merger pour mettre en avant la couverture des tests et empêcher les régressions.

Les Merge Request auront par ailleurs l'obligation d'être approuvées par un pair afin d'éviter au maximum les oublis et mauvaises pratiques : Les développeurs et devOps auront des branches de feature respectives à leurs organisations. Concernant l'intégration des modifications dans la branche main via la branche develop, celle-ci nécessitera un déploiement en préprod avec tests et l'autorisation d'un devOps et d'un développeur.

Nous définirons des Tags sur les commits au fur et à mesure de l'ajout des features, cela nous permettra de pouvoir revenir sur une version plus facilement si nécessaire.
Au fil du développement du projet, nous demanderons aux devops et développeurs d'informer par l'intermédiaire des issues Git les informations importantes à signaler sur le nouveau commit : Détailler légèrement le code apporté, éventuellement alerter si un problème est présent, s'il nécessite d'être revu ou testé par quelqu'un d'autres plus en détail, ou tout simplement signaler que tout est OK. Les issues pourront aussi permettre de créer une nouvelle tâche au projet avec une description du besoin, si un développeur exige un environnement particulier de la part des DevOps par exemple.



C. Tests de CI

Pour la lisibilité et la cohérence de notre code entre les développeurs, nous mettrons une norme à notre code qui sera analysé via Pylint.
Pour finir nous devrons mettre en œuvre, bien entendu, le SAST Gitlab afin de vérifier les possibles failles et vulnérabilités dans notre code ou les librairies utilisées. Il faut néanmoins savoir que le SAST est un outil permettant de trouver une faille de manière théorique, afin d'avoir un retour pertinent sur la sécurité de notre application en pratique nous instaurons un DAST qui analysera dynamiquement notre application et trouvera en les failles dû à notre code ou notre environnement.

D. Mise en production

Concernant la CI/CD, nous utiliserons des agents Gitlab hébergés sur notre infrastructure. La pipeline consistera à construire les images Docker, les publier sur un registre de conteneurs puis de les déployer avec des charts Helm sur notre cluster Kubernetes. Une fois les applications déployées, des tests d'intégrations seront lancés à l'aide d'un Runner Postman pour s'assurer de la disponibilité et du bon fonctionnement individuel de nos API. Enfin, une fois que les API seront toutes déployées, il faudra lancer des tests End-To-End. Ces derniers seront aussi lancés avec un Runner Postman.