CAHIER DE CONCEPTION

Conteneurisation et gestion des workloads avec Magnum

Rédigé par:

- ESSONO EFFA Etienne Florian
- NGNINTEDEM FEUPI Vaillant

Sous la supervision : M. NGUIMBUS Emmanuel

Année Académique 2025 - 2026

PLAN

INTRODUCTION

1. CONTEXTE DU PROJET ET OBJECTIFS
2. ARCHITECTURE DU SYSTÈME
3. DIAGRAMMES DE SÉQUENCES
4. DIAGRAMME DE CLASSE
5. DIAGRAMME DE COMPOSANTS
6. DIAGRAMME DE DÉPLOIEMENT

CONCLUSION

INTRODUCTION

Le présent dossier de conception fait suite au Cahier d'Analyse portant sur la « Conteneurisation et gestion des workloads avec Magnum ». Alors que la phase d'analyse a permis d'identifier les besoins métiers et les cas d'utilisation, ce document a pour objectif de définir la réponse technique permettant d'implémenter ces services au sein de l'écosystème OpenStack.

L'enjeu majeur de cette conception est de modéliser une infrastructure capable de supporter à la fois la gestion automatisée de clusters Kubernetes (via **Magnum**) et l'orchestration de ressources mixtes (via **Heat**). Nous détaillerons ici :

- **La dynamique du système :** À travers les diagrammes de séquence illustrant les interactions entre les services Keystone, Magnum, Heat et Nova.
- **La structure des données :** Via un diagramme de classe intégrant les notions de projets, de modèles de clusters et de ressources d'infrastructure.
- **L'architecture logicielle :** Par une vue en composants montrant l'interopérabilité des APIs.
- **Le déploiement physique :** Précisant la répartition des rôles entre les nœuds de contrôle et les nœuds de calcul.

Cette étape de conception garantit la robustesse du système avant sa phase de déploiement effectif sur notre plateforme DevStack.

1. CONTEXTE DU PROJET ET OBJECTIFS

A. CONTEXTE DU PROJET

L’évolution rapide des architectures logicielles vers le **Cloud Native** et les **micro-services** a transformé la manière dont les infrastructures sont gérées. Aujourd'hui, les organisations ne se contentent plus de simples machines virtuelles ; elles exigent des plateformes capables de gérer des **workloads distribués, dynamiques et conteneurisés**.

Dans ce cadre, la gestion manuelle des ressources (réseaux, serveurs, stockage) devient un frein à l'innovation, car elle est :

- **Complexe :** Multiplicité des interfaces et des configurations.
- **Coûteuse en temps :** Déploiements longs et répétitifs.
- **Source d'erreurs :** Risques d'incohérence entre les environnements de test et de production.

Le projet s'inscrit donc dans une volonté de modernisation au sein de l'**Université Saint Jean**, en utilisant la puissance d'**OpenStack** pour offrir un environnement d'infrastructure agile et automatisé.

1. CONTEXTE DU PROJET ET OBJECTIFS

B. OBJECTIFS DU PROJET

L'objectif principal est de concevoir et de mettre en œuvre une plateforme d'orchestration robuste capable de répondre à trois défis majeurs :

1. Automatisation de l'Orchestration de Conteneurs (Magnum)

Fournir une capacité de déploiement "en un clic" de clusters **Kubernetes** ou **Docker Swarm**. L'objectif est de masquer la complexité de l'infrastructure sous-jacente pour permettre aux développeurs de se concentrer sur leurs applications.

2. Orchestration Hybride Mixte (Heat)

Permettre le déploiement simultané, au sein d'une même pile (stack), de **machines virtuelles classiques** et de **clusters de conteneurs**. Ce modèle hybride est crucial pour supporter des applications héritées (bases de données sur VM) communiquant avec des services modernes conteneurisés.

1. CONTEXTE DU PROJET ET OBJECTIFS

B. OBJECTIFS DU PROJET

3. Gestion du Cycle de Vie et Scalabilité

Garantir que l'infrastructure peut s'adapter en temps réel à la charge (Scaling) :

- **Scalabilité :** Ajouter ou supprimer des nœuds de manière fluide (UC03).
- **Résilience :** Assurer la disponibilité des services même en cas de défaillance d'un nœud.
- **Infrastructure as Code (IaC) :** Utiliser des scripts et des APIs pour rendre l'infrastructure reproductible et versionable (UC04).

4. Optimisation des Ressources

En utilisant un environnement comme **DevStack**, l'objectif est également de maximiser l'utilisation des ressources matérielles disponibles par une gestion fine des quotas et un nettoyage systématique des ressources inutilisées(UC05).

3. ARCHITECTURE DU SYSTÈME

L'architecture retenue repose sur une approche **modulaire et orientée services (SOA)**. Elle utilise l'écosystème OpenStack pour transformer des ressources matérielles brutes en une plateforme de services (PaaS) capable de gérer des conteneurs.

1. Architecture Logique (Services Core)

Le système s'articule autour de trois piliers principaux qui collaborent de manière asynchrone :

- **Le Pilote d'Orchestration de Conteneurs (Magnum) :** Il agit comme la couche supérieure. Sa responsabilité est de traduire les intentions de l'utilisateur (ex: "Je veux un cluster Kubernetes de 3 nœuds") en un plan d'infrastructure.
- **Le Moteur d'Orchestration Générique (Heat) :** C'est le pivot central. Il reçoit les plans (templates HOT) de Magnum ou de l'utilisateur. Il gère l'ordre de création des ressources, le rollback en cas d'erreur, et les dépendances entre les VMs et le réseau.
- **La Couche d'Infrastructure (Nova, Neutron, Cinder) :** Ces services fournissent les briques élémentaires (Calcul, Réseau, Stockage). Ils sont pilotés par Heat mais restent transparents pour l'utilisateur final de Magnum.

3. ARCHITECTURE DU SYSTÈME

2. Architecture de Communication (Inter-services)

La communication au sein de l'architecture est sécurisée et standardisée :

- **APIs RESTful :** Tous les échanges entre l'utilisateur et le système (ou entre les services) passent par des interfaces de programmation REST.
- **Bus de Messages (RabbitMQ) :** Pour assurer la fluidité des opérations longues (comme la création d'un cluster), les composants communiquent via un bus de messages asynchrone, permettant au système de rester réactif même pendant les phases de calcul intensif.
- **Identité Centralisée (Keystone) :** Chaque composant de l'architecture vérifie systématiquement l'identité et les droits (RBAC) avant d'exécuter une tâche.

3. ARCHITECTURE DU SYSTÈME

3. Architecture Réseau du Cluster

Le système met en place une topologie réseau spécifique pour les workloads :

- **Réseau de Management :** Pour la communication interne entre les services OpenStack.
- **Réseau de Données (Tenant Network) :** Un réseau privé isolé pour chaque projet, où les nœuds du cluster (Masters et Workers) communiquent entre eux.

4. DIAGRAMMES DE SÉQUENCES

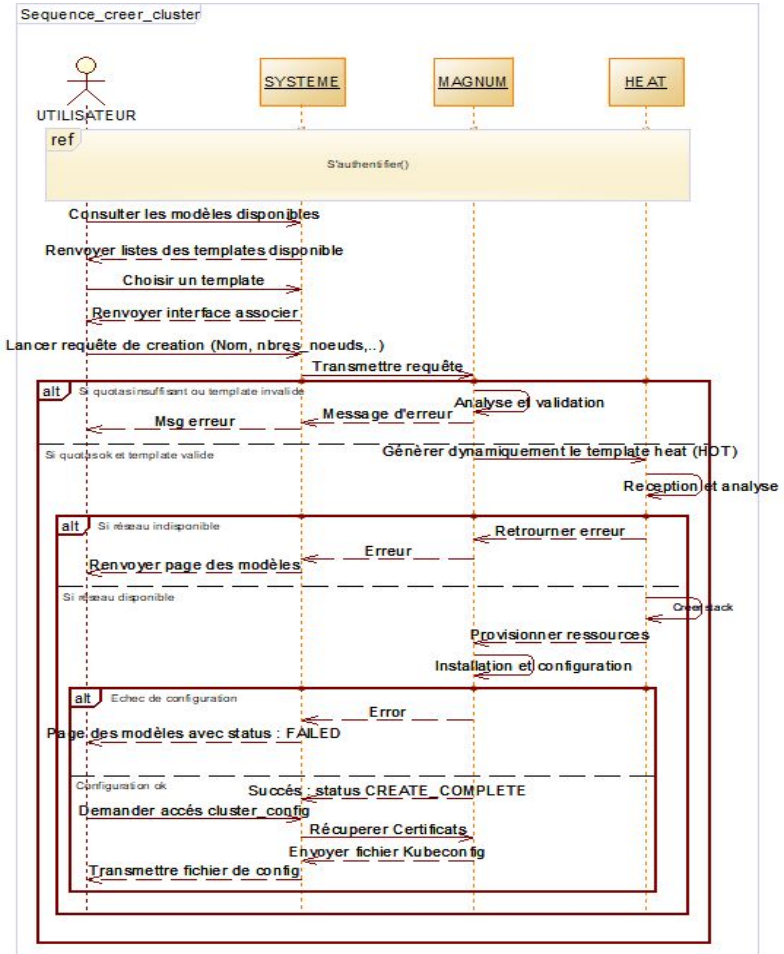
a. Présentation

Les diagrammes des séquences documentent les interactions à mettre en œuvre entre les classes pour réaliser un résultat, tel qu'un cas d'utilisation. UML étant conçu pour la programmation orientée objet, ces communications entre les classes sont reconnues comme des messages. Le diagramme des séquences énumère des objets horizontalement, et le temps verticalement. Il modélise l'exécution des différents messages en fonction du temps. Dans un diagramme des séquences, les classes et les acteurs sont énumérés en colonnes, avec leurs lignes de vie verticales indiquant la durée de vie de l'objet :

4. DIAGRAMMES DE SÉQUENCES

b. Quelques diagrammes de séquences

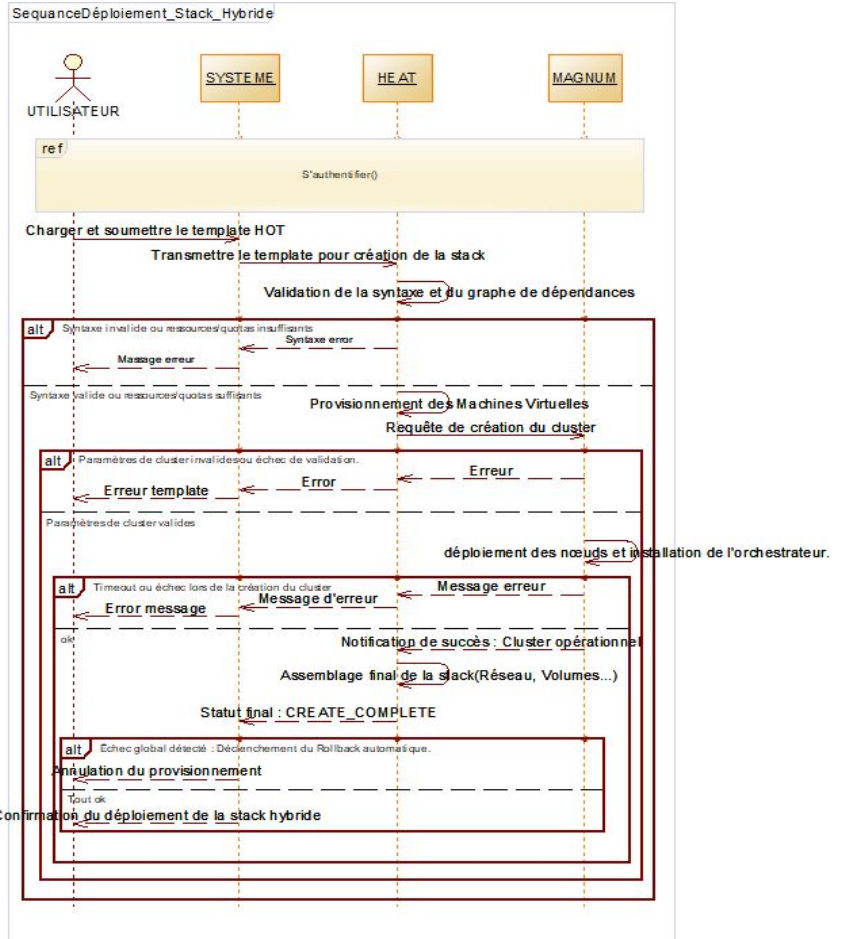
b.1. Diagramme de séquence du cas CRÉER UN CLUSTER KUBERNETES VIA MAGNUM



4. DIAGRAMMES DE SÉQUENCES

b. Quelques diagrammes de séquences

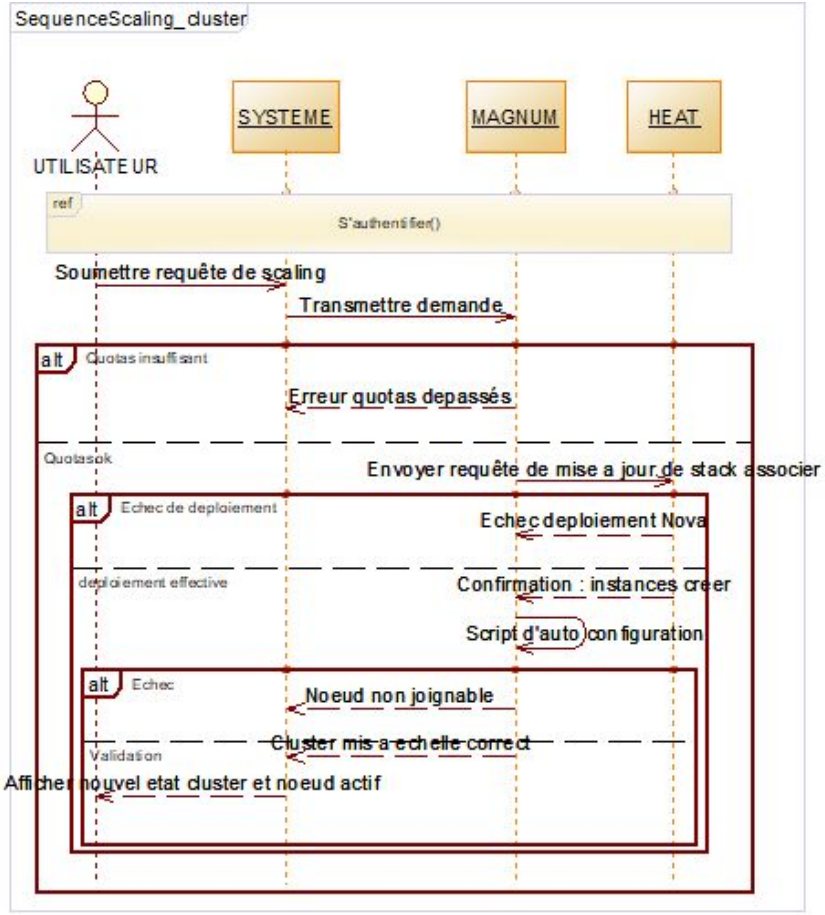
b.1. Diagramme de séquence du cas DÉPLOIEMENT D'UNE STACK HYBRIDE (VM + CLUSTER) VIA HEAT



4. DIAGRAMMES DE SÉQUENCES

b. Quelques diagrammes de séquences

b.1. Diagramme de séquence du cas MISE À L'ÉCHELLE (SCALING) D'UN CLUSTER



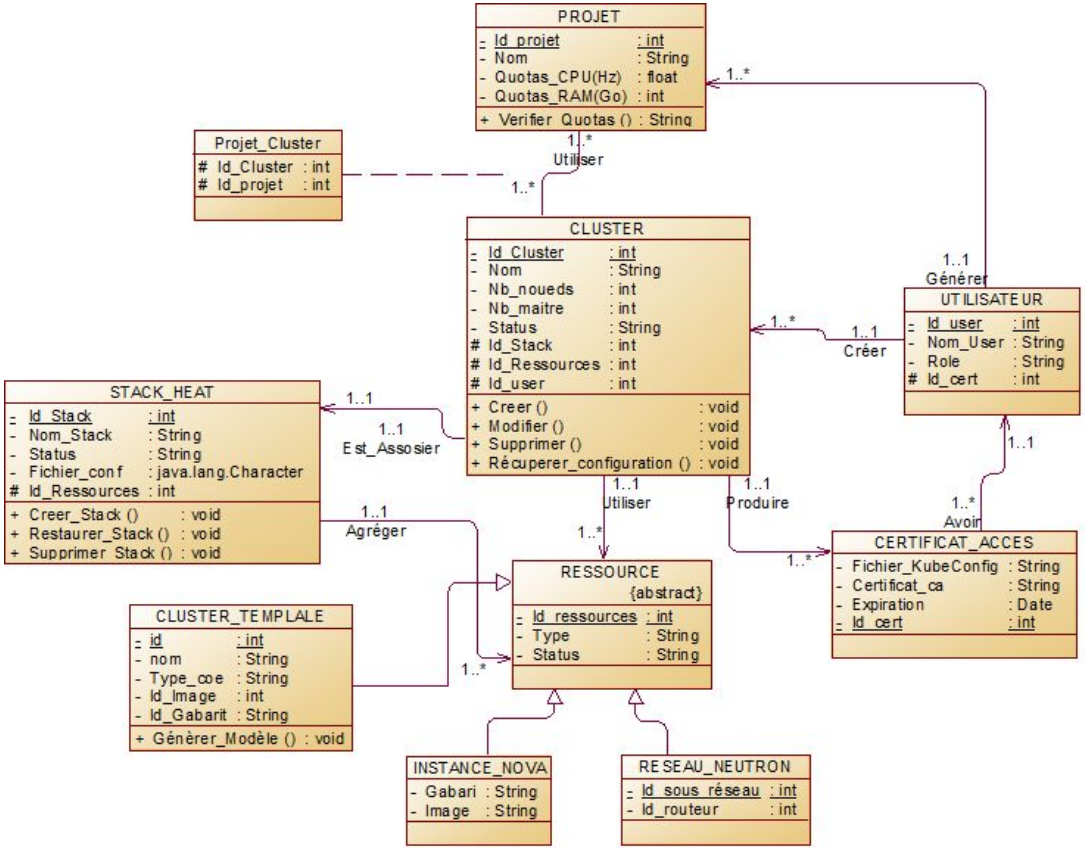
4. DIAGRAMMES DE CLASSE

a. **Présentation**

Il est essentiel car il définit la structure statique du système, les données manipulées et les relations entre les différents services d'OpenStack que nous allons orchestrer.

b. **Présentation du Diagramme de classe Globale du Système**

4. DIAGRAMMES DE CLASSE



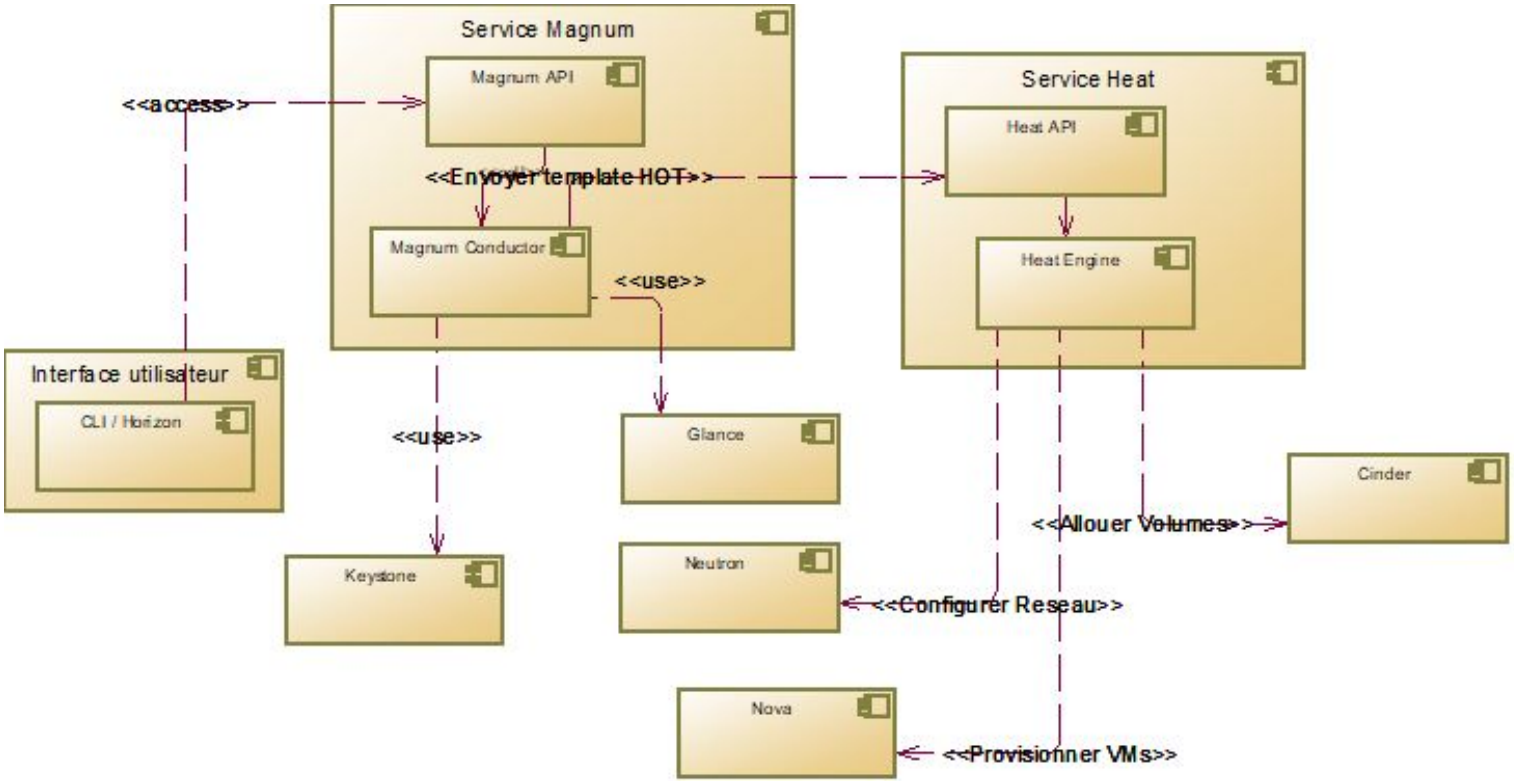
4. DIAGRAMMES DE COMPOSANT

a. Présentation

Le **Diagramme de Composants** est une vue de conception qui montre l'organisation logique du système et les dépendances entre les modules logiciels. Pour notre projet OpenStack, il permet de visualiser comment Magnum et Heat s'appuient sur les services de base pour livrer un cluster.

b. Illustration du Diagramme de composant

4. DIAGRAMMES DE COMPOSANT



4. DIAGRAMMES DE DÉPLOIEMENT

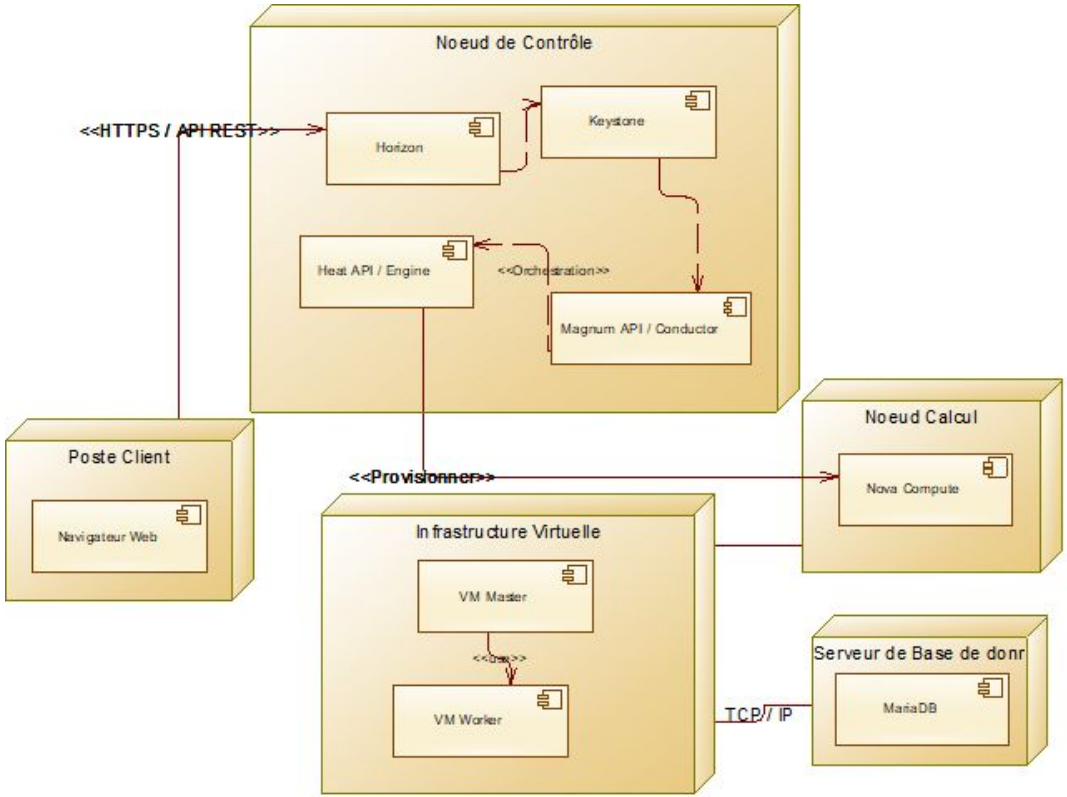
a. Présentation

Le **Diagramme de Déploiement** est l'étape finale de notre dossier de conception. Il permet de visualiser comment les composants logiciels (Magnum, Heat, Nova, etc.) sont physiquement répartis sur l'infrastructure matérielle ou virtuelle, ainsi que les liens réseau entre eux.

Dans le cadre d'un environnement DevStack (souvent utilisé pour les tests et le développement), nous avons modélisées une architecture à cinq nœuds principaux.

b. Illustration

4. DIAGRAMMES DE DÉPLOIEMENT



CONCLUSION

La conception détaillée dans ce dossier confirme la faisabilité technique des objectifs fixés. En s'appuyant sur le couplage entre **Magnum** et **Heat**, nous avons conçu une architecture répondant aux critères de flexibilité, de scalabilité et d'automatisation.

Le passage d'une vision fonctionnelle à une vision technique a permis de mettre en lumière des points critiques, notamment la gestion des quotas par projet et la nécessité d'un rollback automatique en cas d'échec de stack hybride. La structure modulaire choisie assure que l'infrastructure pourra évoluer sans remettre en cause les fondations du système.

Prochaines étapes : La validation de cette conception marque le début de la phase de **Réalisation**.