UNIVERSITÉ SAINT JEAN

CAHIER DE CONCEPTION :

Orchestration de ressources multi-VM et multi-réseaux avec Heat et Auto-scaling

Rédiger par :

Ngouo Franck Leonel

MOUDIO ABEGA Laurent Stéphane

Supervisé par :

M. NGUIMBUS Emmanuel

Plan

1- INTRODUCTION

Suite à l'analyse des enjeux de la transformation numérique et de la souveraineté des données via OpenStack, ce cahier de conception formalise la réponse technique aux limites identifiées du service d'orchestration Heat. Alors que la phase précédente a défini les besoins en élasticité, ce document détaille l'architecture et les mécanismes nécessaires pour rendre cette automatisation opérationnelle.

L'objectif est de concevoir un système capable de dépasser la rigidité des modèles déclaratifs classiques face aux variations de charge. La solution repose sur l'articulation de trois composants clés : des modèles Heat modulaires, une surveillance granulaire par Prometheus et un module de décision intelligent en Python.

Ce cahier décrit donc la structure logicielle et les interactions dynamiques du système, garantissant une orchestration capable d'adapter non seulement le calcul, mais aussi l'infrastructure réseau de manière autonome et sécurisée.

2- Contexte Générale

OpenStack fournit un ensemble de services interconnectés pour la gestion des ressources de calcul (Nova), de stockage (Cinder, Swift) et de réseau (Neutron). Heat agit comme un orchestrateur transversal permettant de coordonner ces services afin de déployer des infrastructures complètes de manière automatisée.

Dans les environnements de production modernes, les architectures cloud sont rarement simples. Elles incluent généralement :

- ➔ plusieurs machines virtuelles ayant des rôles distincts ;
- ➔ des réseaux segmentés (frontend, backend, réseau d'administration) ;
- ➔ des exigences élevées en matière de disponibilité, de performance et de sécurité ;
- ➔ des besoins dynamiques en ressources liés à la charge applicative.

3. Architecture du système

3.1. Architecture Logique

L’architecture logique du système repose sur une **séparation stricte des responsabilités** entre les composants d’orchestration, de décision et de supervision. Elle vise à améliorer les capacités natives d’OpenStack Heat sans en modifier le cœur, en introduisant un **module personnalisé de gestion du scaling dynamique**, capable d’appliquer des politiques opérationnelles complexes adaptées aux besoins spécifiques des organisations.

Le système est structuré autour de **trois pôles fonctionnels principaux** :

- l’orchestration des ressources cloud géré par Heat
- la collecte et l’analyse des métriques, par un outil de monitoring externe
- la gouvernance et la prise de décision en matière de scaling géré par le module

3. Architecture du système

3.2. Interactions entre les composants

Le fonctionnement global de l'architecture repose sur une **chaîne d'interactions maîtrisée** :

1. **L'infrastructure déployée génère des métriques de fonctionnement.**
2. **Le système de métriques externe collecte et agrège ces données.**
3. **Les métriques sont transmises au module personnalisé.**
4. **Le module analyse les métriques selon les politiques définies.**
5. **En cas de dépassement de seuil, une alarme interne est déclenchée.**
6. **Le module décide d'une action de scale-out, de scale-down ou d'inaction.**
7. **L'ordre correspondant est transmis à OpenStack Heat.**
8. **Heat exécute l'action sur l'infrastructure cloud.**
9. **Le module journalise l'ensemble du processus à des fins d'audit et de traçabilité.**

3. Architecture du système

3.2. Architecture de communication

La communication au sein de l'architecture est **sécurisée, standardisée et découplée**, afin de garantir la fiabilité des échanges entre les composants et de supporter un scaling dynamique contrôlé.)

● APIs REST sécurisées (Heat & Module personnalisé

La communication entre le module personnalisé de scaling et OpenStack Heat s'effectue via les **API REST natives de Heat**. Le module envoie des requêtes HTTP (POST/PUT) pour déclencher des actions de scale-out ou de scale-down sur des stacks existants, et récupère en retour l'état d'exécution des opérations.

Technologies et procédures :

- API REST OpenStack Heat
- Protocole HTTPS
- Authentification via Keystone
- Contrôle des droits par RBAC

● Collecte de métriques via un système externe

Les métriques nécessaires au scaling sont collectées par un **outil de supervision externe** (par exemple Prometheus, Ceilometer/Gnocchi ou équivalent).Ces métriques sont transmises au module personnalisé via une **API REST dédiée** ou un mécanisme de push périodique.

Technologies et procédures :

- API REST de collecte de métriques
- Format JSON
- Transmission périodique ou événementielle
- Validation et filtrage des données côté module

3. Architecture du système

3.2. Architecture de communication

La communication au sein de l'architecture est **sécurisée, standardisée et découplée**, afin
garantir la fiabilité des échanges entre les composants et de supporter un scaling dynamiqu
contrôlé.

● **Gestion asynchrone des alarmes et événements (Bus de messages)**
Les événements liés au dépassement de seuils et les alarmes de scaling sont traités de ma
asynchrone afin d'éviter tout blocage du système lors des phases de charge élevée.
Un **bus de messages** est utilisé pour découpler la collecte des métriques, la prise de décis
l'exécution du scaling.

Technologies et procédures :

- Bus de messages (RabbitMQ – standard OpenStack)
- Files de messages internes
- Traitement événementiel
- Tolérance aux pics de charge
- Contrôle des droits par RBAC

● **Communication Administrateur ↔ Module personnalisé (Interface Web / API REST)**
L'administrateur cloud interagit avec le module personnalisé à travers une **interface web
sécurisée** ou une **API REST d'administration**. Ces échanges permettent de définir, modifie
superviser les politiques de scaling.

Technologies et procédures :

- Interface web HTTPS
- API REST d'administration
- Authentification forte
- Gestion des rôles et permissions

3. Architecture du système

3.2. Architecture de communication

La communication au sein de l'architecture est **sécurisée, standardisée et découplée**, afin
garantir la fiabilité des échanges entre les composants et de supporter un scaling dynamiqu
contrôlé.

● **Authentification et autorisation centralisées (Keystone)**
Toutes les communications impliquant OpenStack Heat sont sécurisées par une **authentifi
centralisée via Keystone**.
Chaque appel est vérifié avant exécution afin de garantir que seules les actions autorisées
réalisées.

Technologies et procédures :

- OpenStack Keystone
- Tokens d'authentification
- RBAC (Role-Based Access Control)

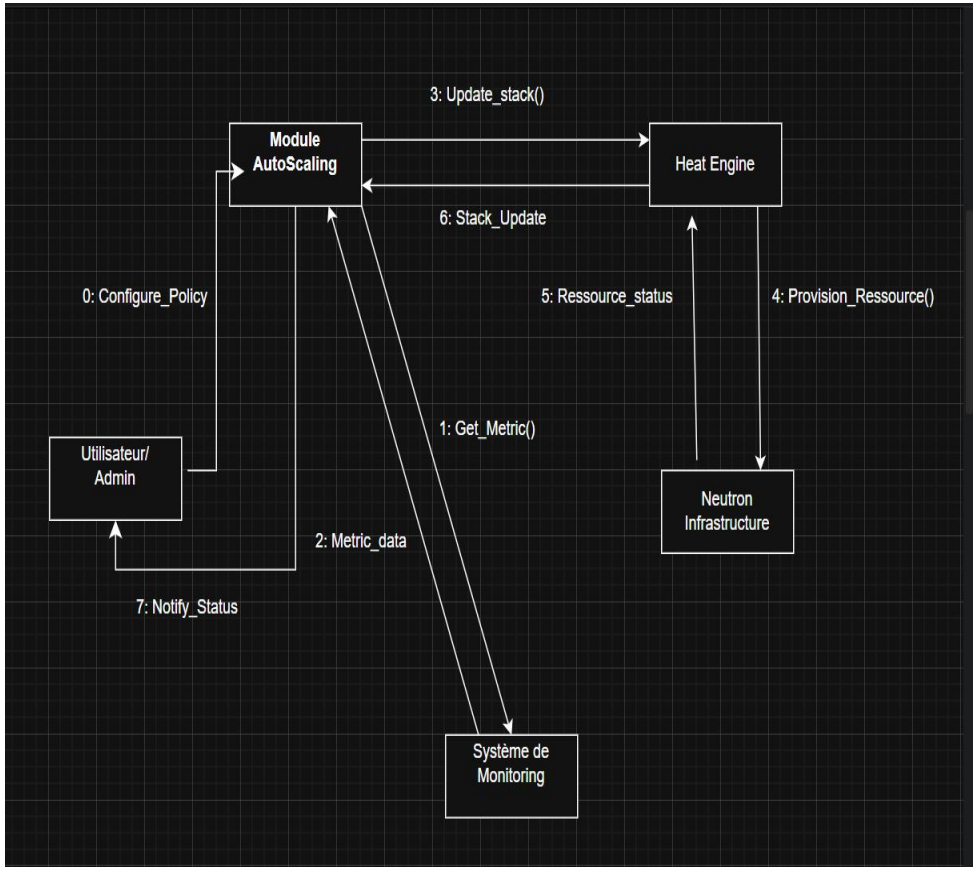
● **Journalisation et traçabilité des communications (Logs centralisés)**
Les échanges critiques, les décisions de scaling et les appels aux API sont **systématiquem
journalisés** afin de garantir la traçabilité et l'auditabilité du système.

Technologies et procédures :

- Logs applicatifs horodatés
- Centralisation des journaux
- Conservation des historiques de décisions

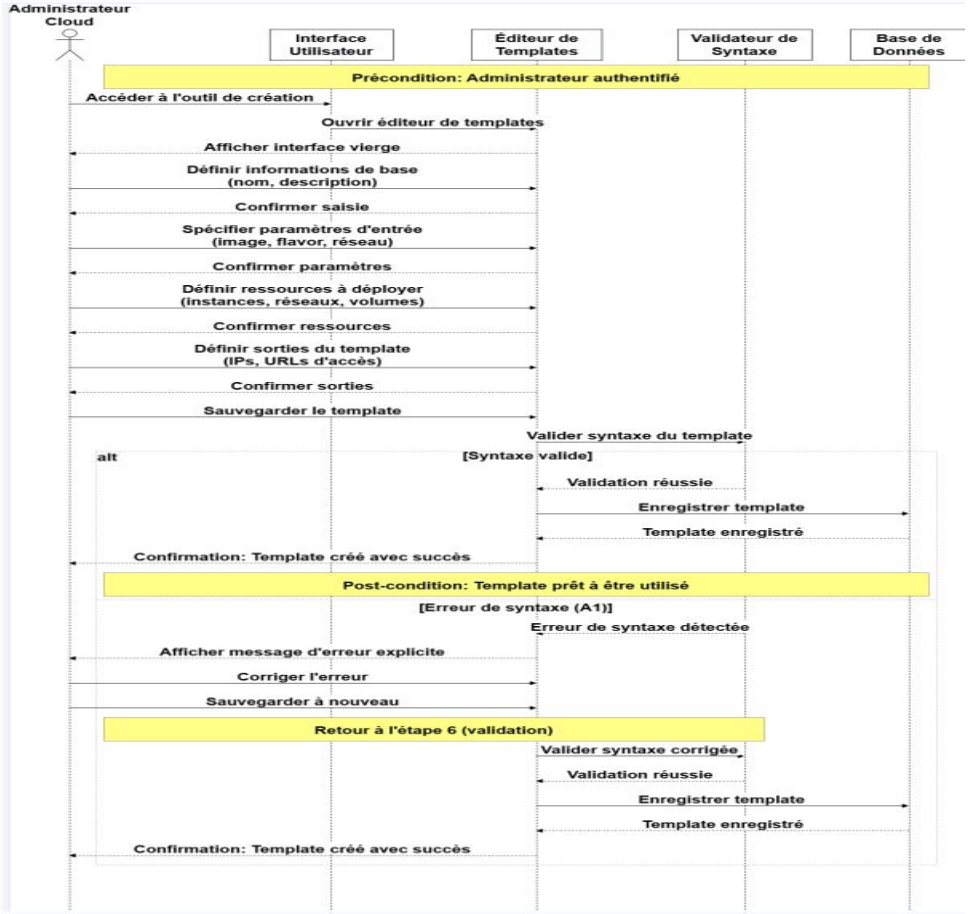
3. Architecture du système

Diagramme de Communication



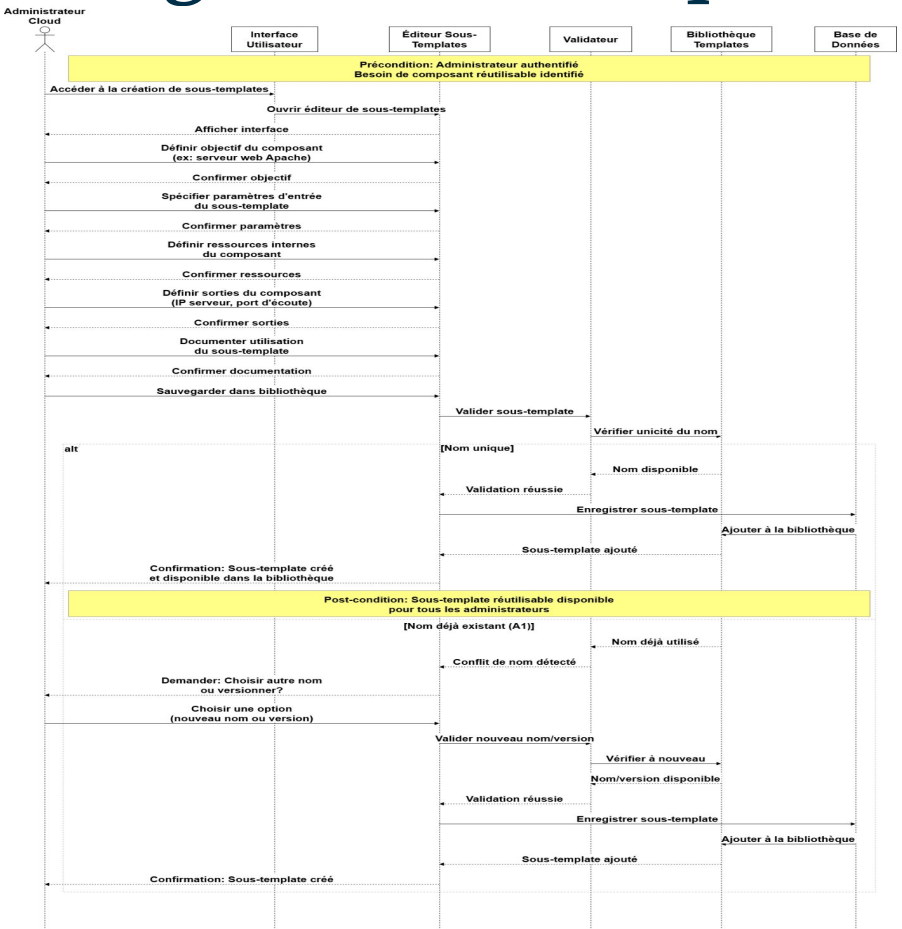
5. Diagramme de séquence

Creation
Template
Heat



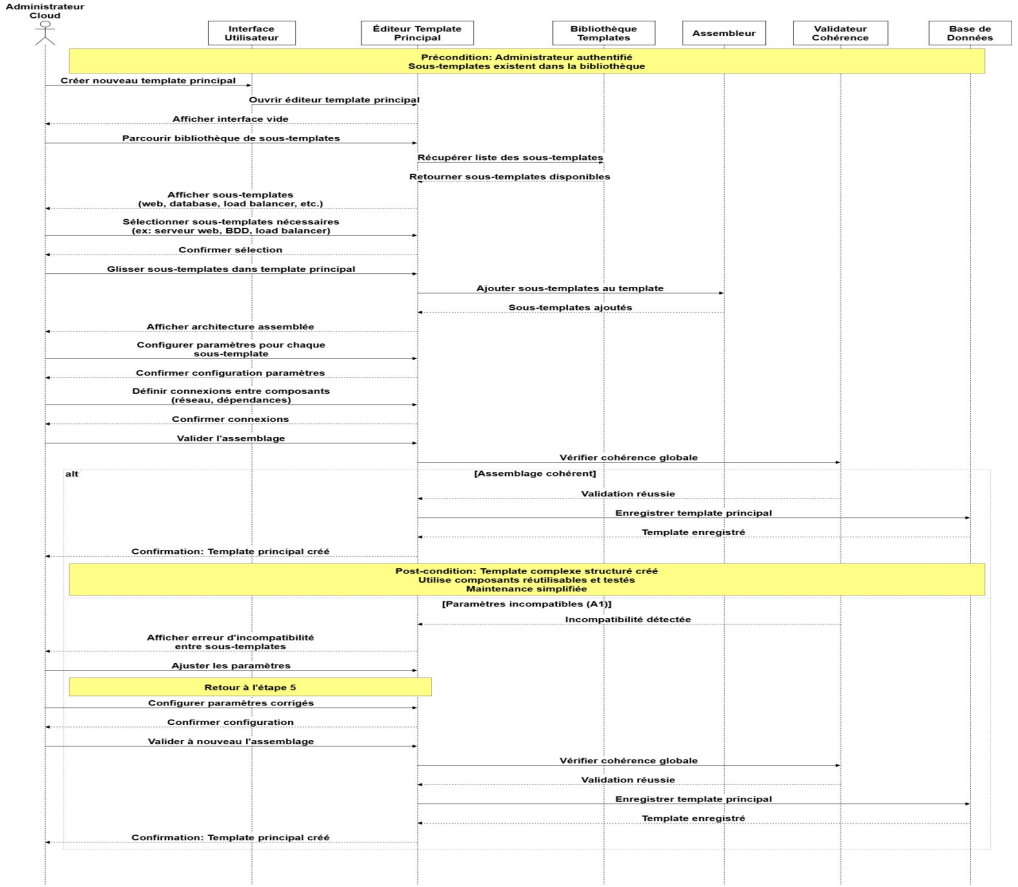
5. Diagramme de séquence

Creation Sous Template Heat



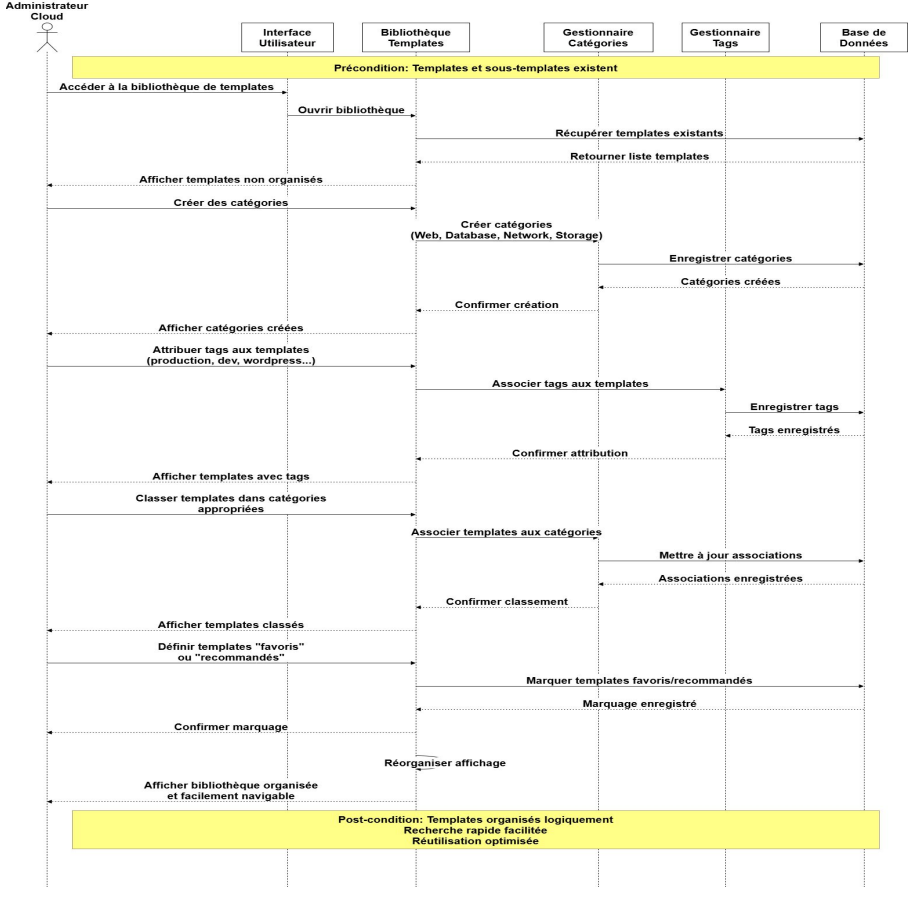
5. Diagramme de séquence

Assembler les Templates



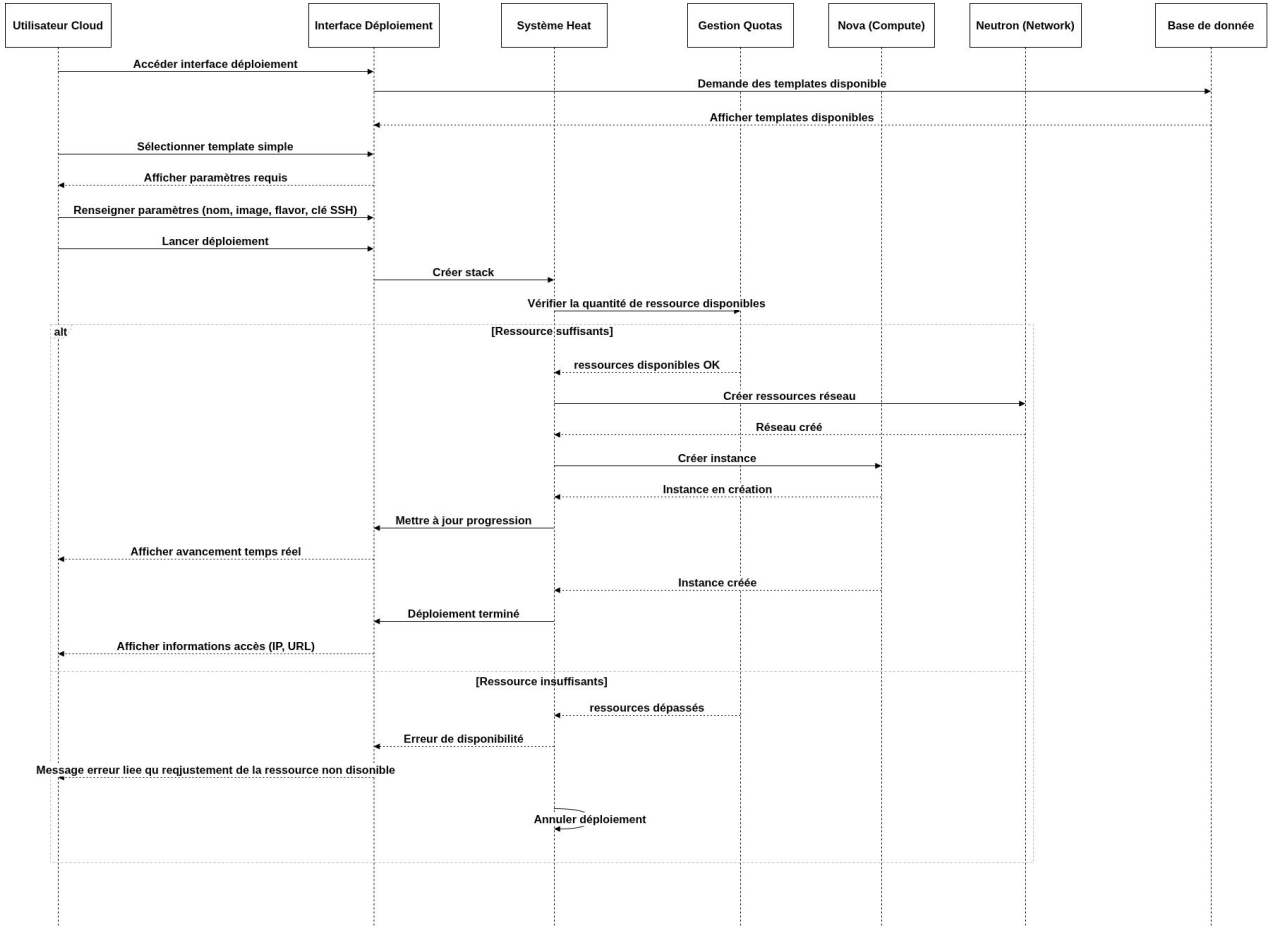
5. Diagramme de séquence

Organiser Template



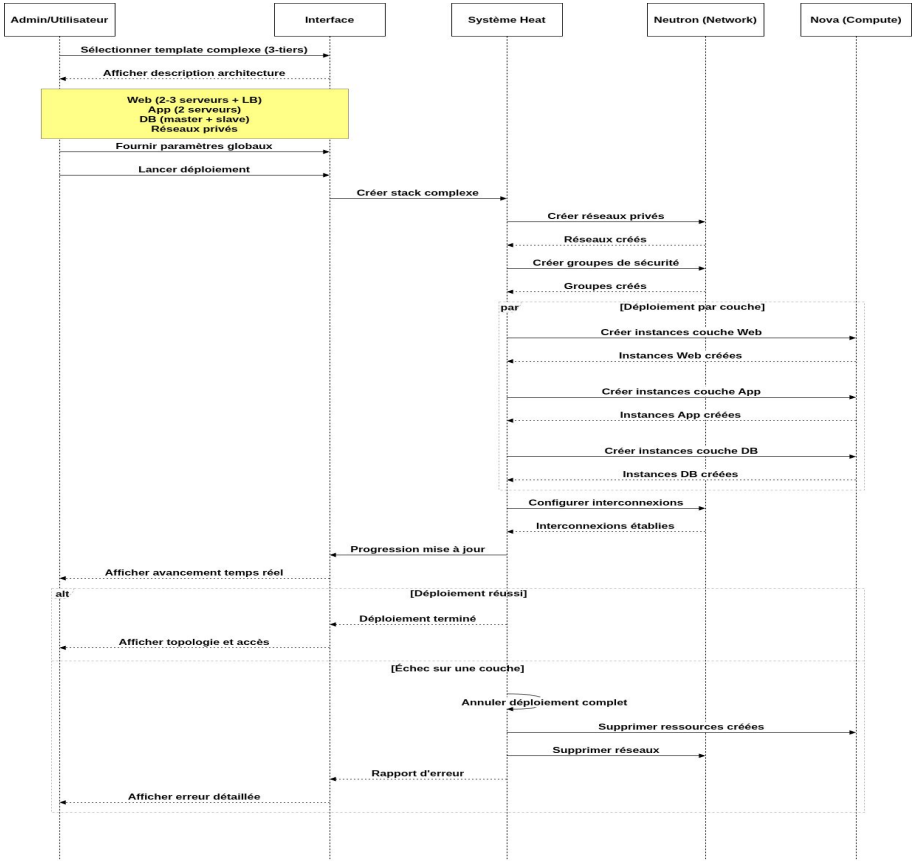
5. Diagramme de séquence

Déploiement
Mono-vm



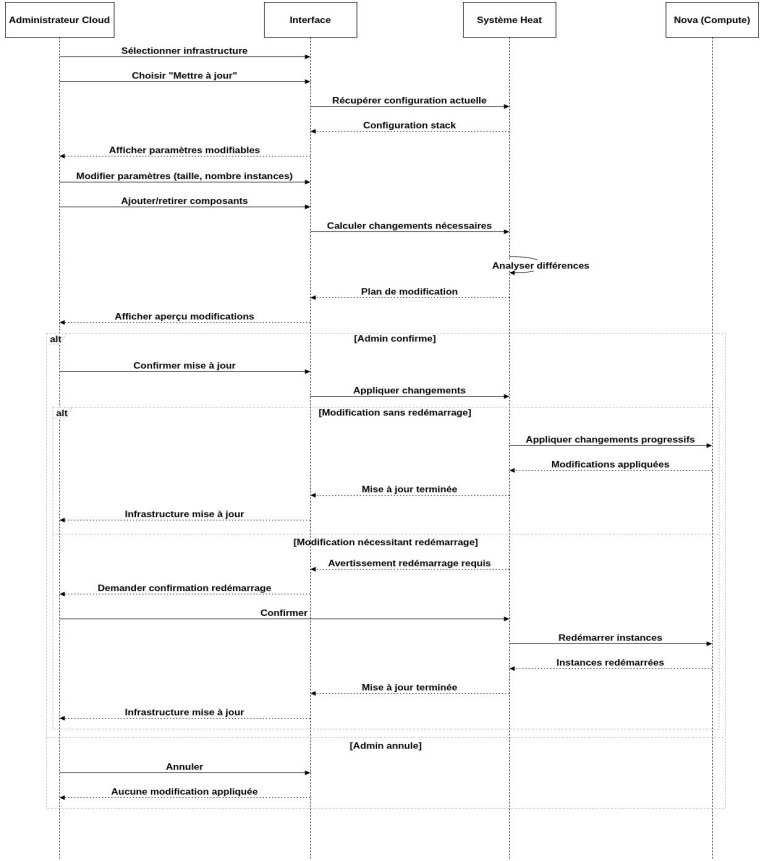
5. Diagramme de séquence

Déploiement
Multi vm et
multi réseau



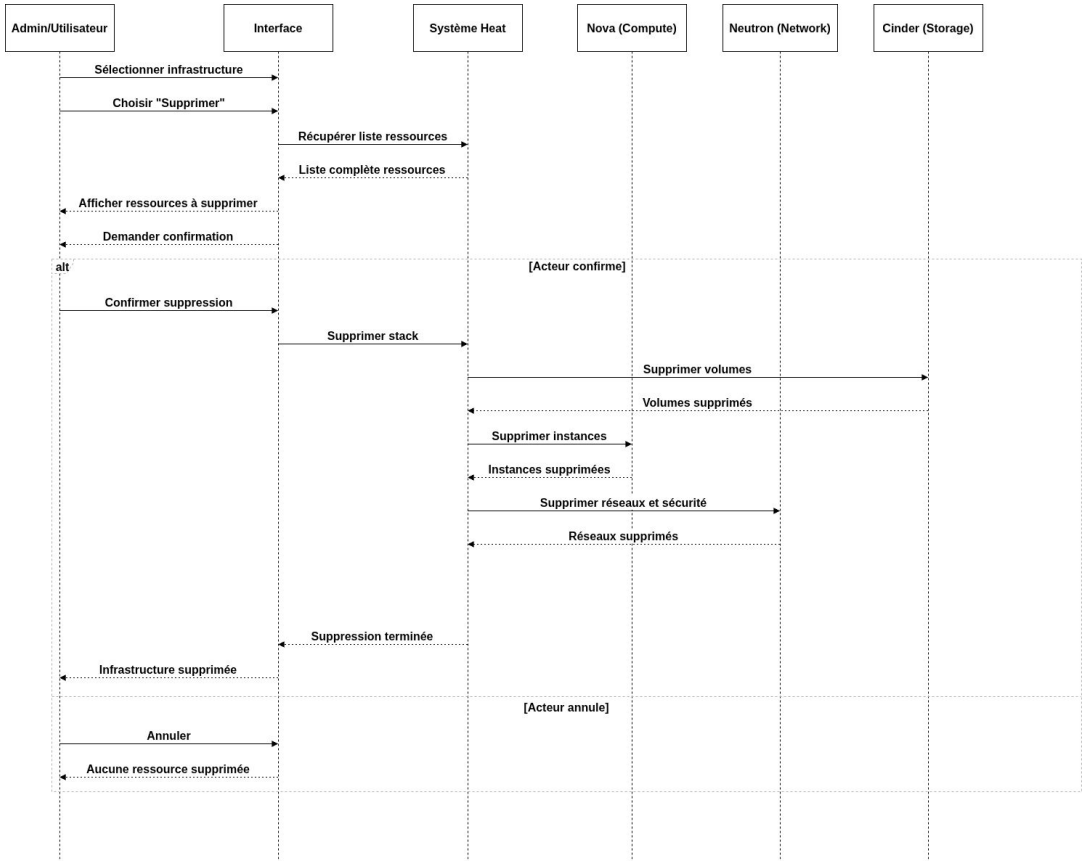
5. Diagramme de séquence

Modifier l'infrastructure



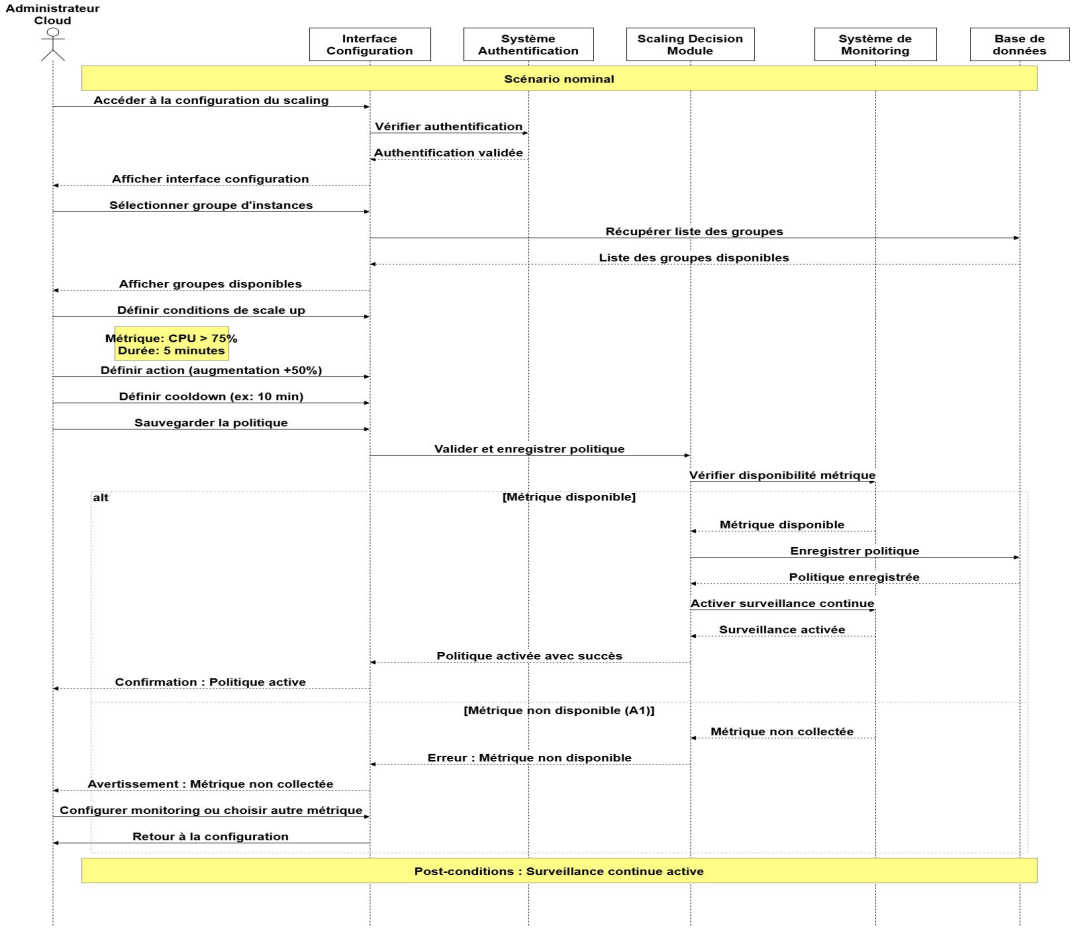
5. Diagramme de séquence

Supprimer l'infrastructure



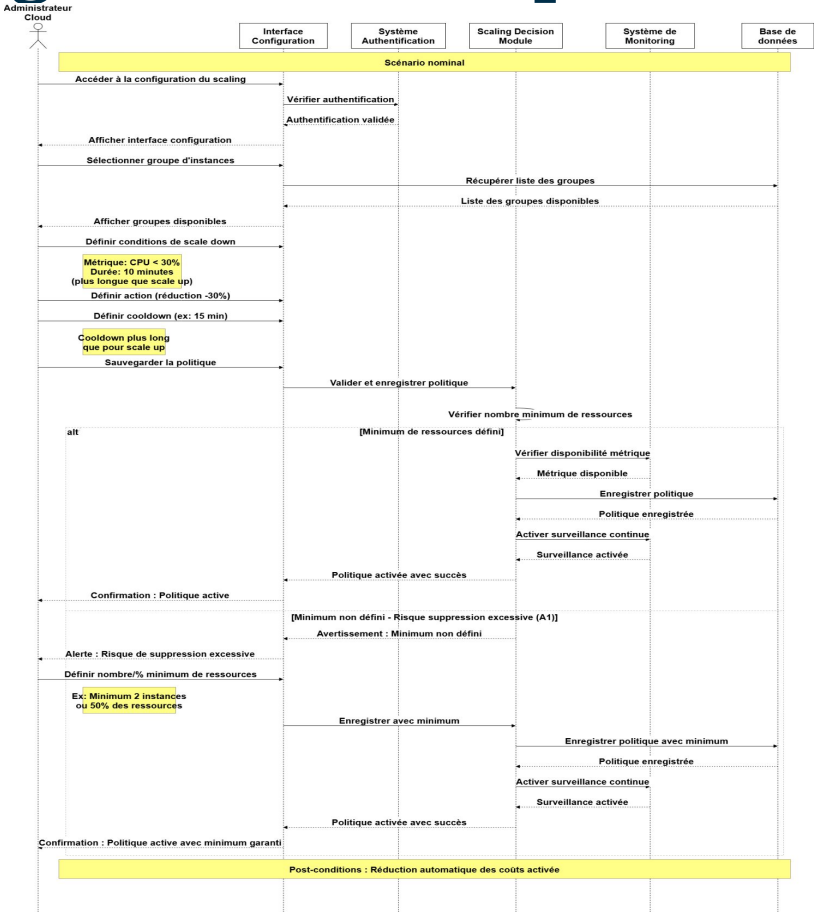
5. Diagramme de séquence

Définir une politique de scale up basée sur les métriques



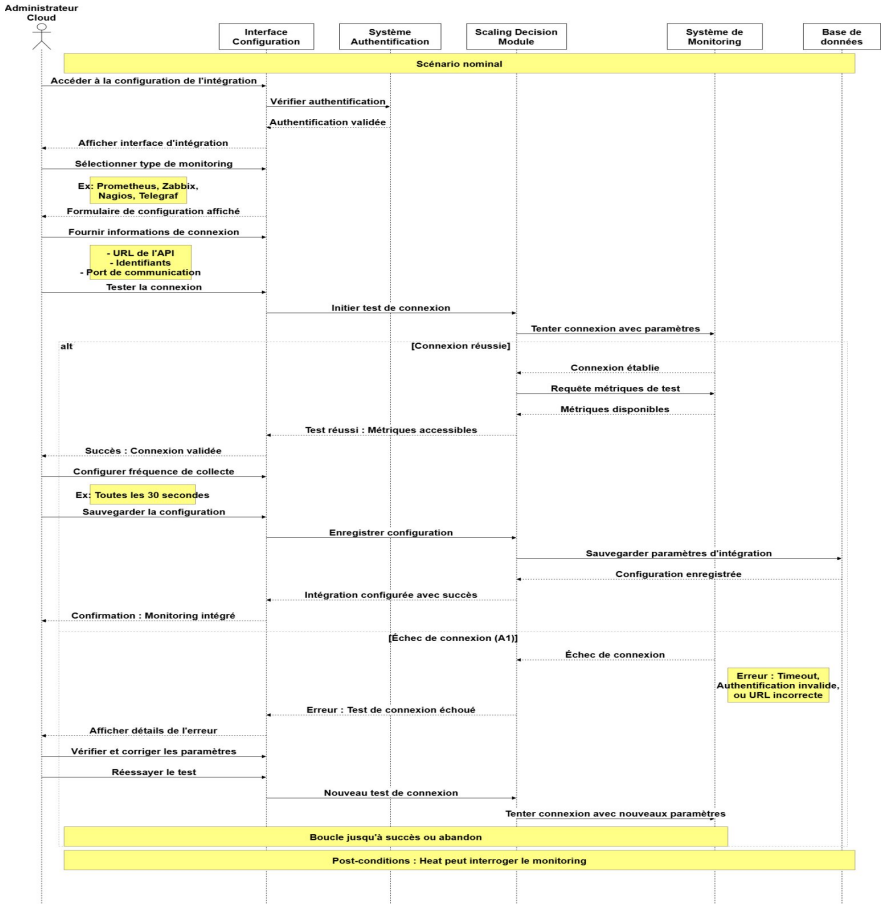
5. Diagramme de séquence

Définir une politique de scale down basée sur les métriques



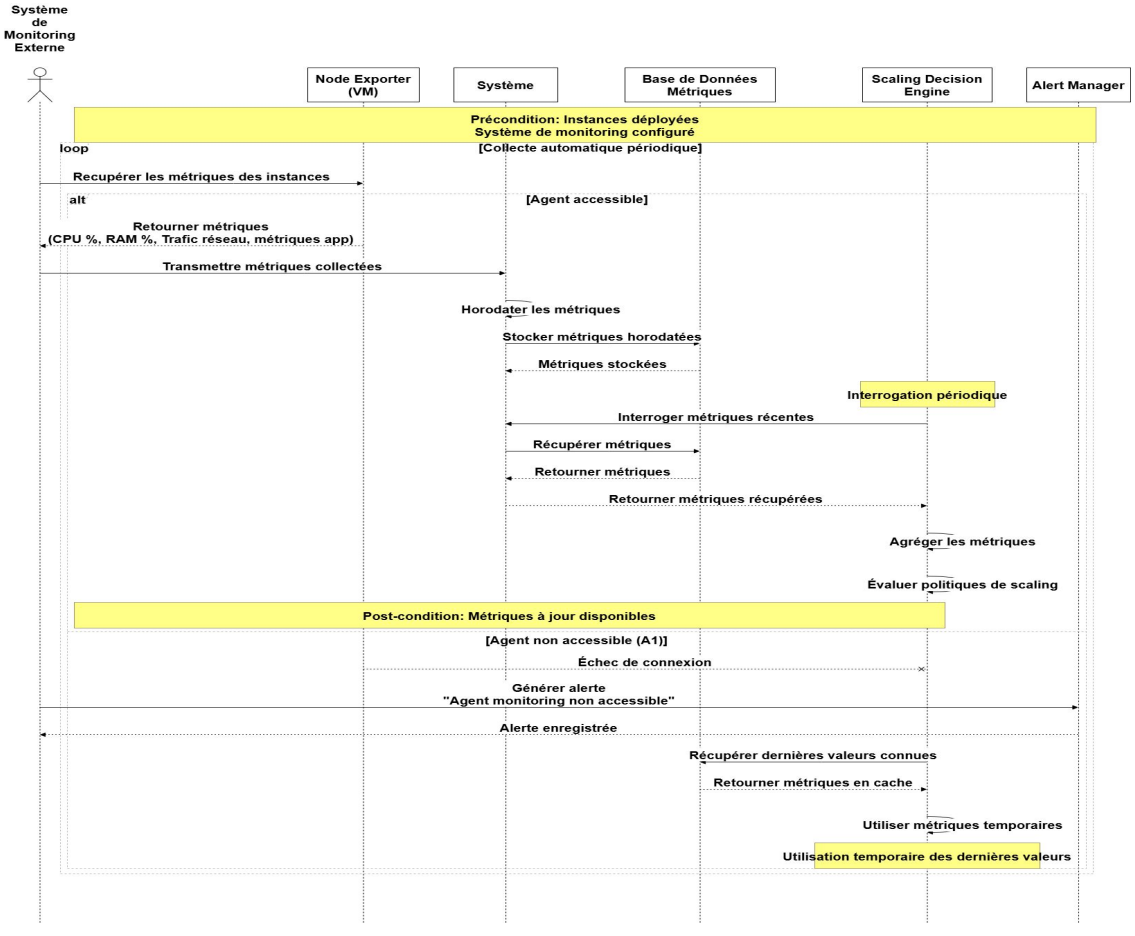
5. Diagramme de séquence

Configurer l'intégration avec le système de monitoring



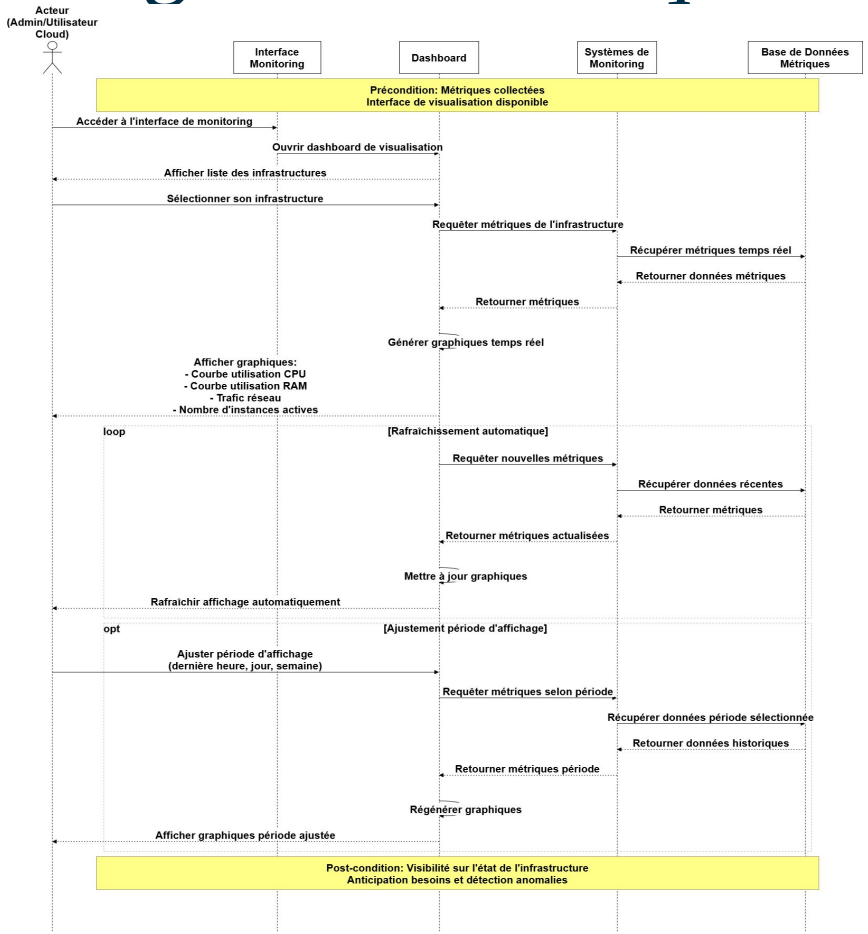
5. Diagramme de séquence

Collecter les métriques des instances



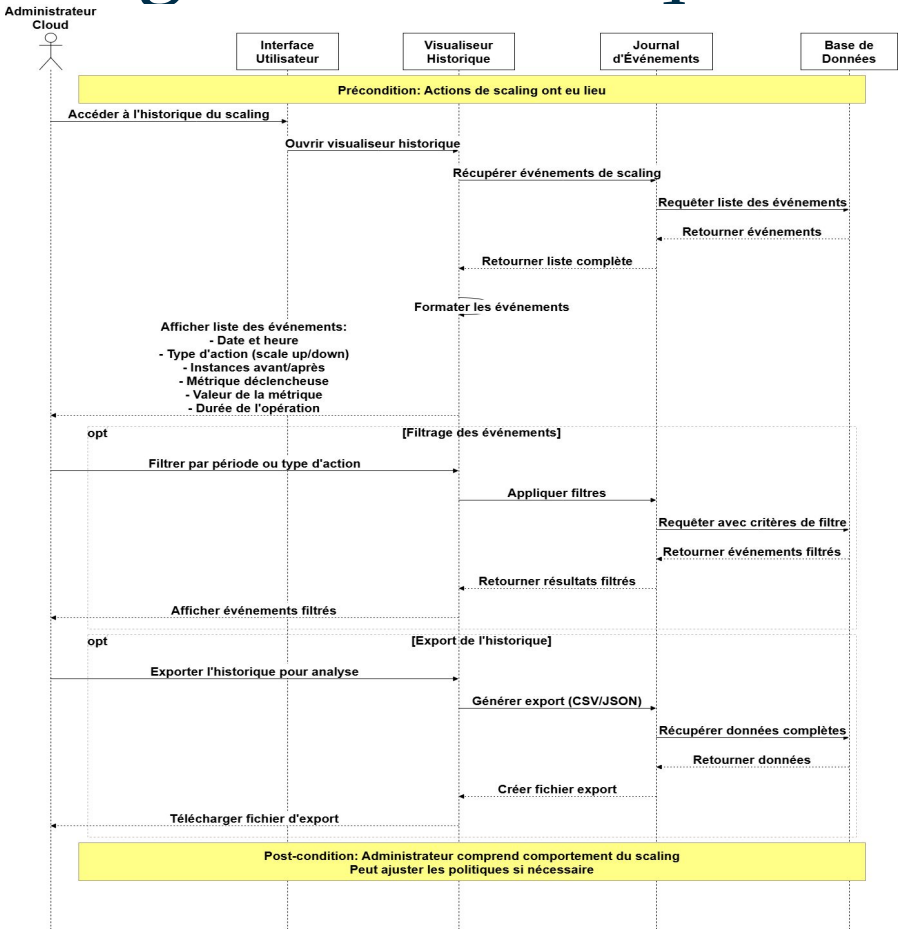
5. Diagramme de séquence

Visualiser les métriques en temps réel



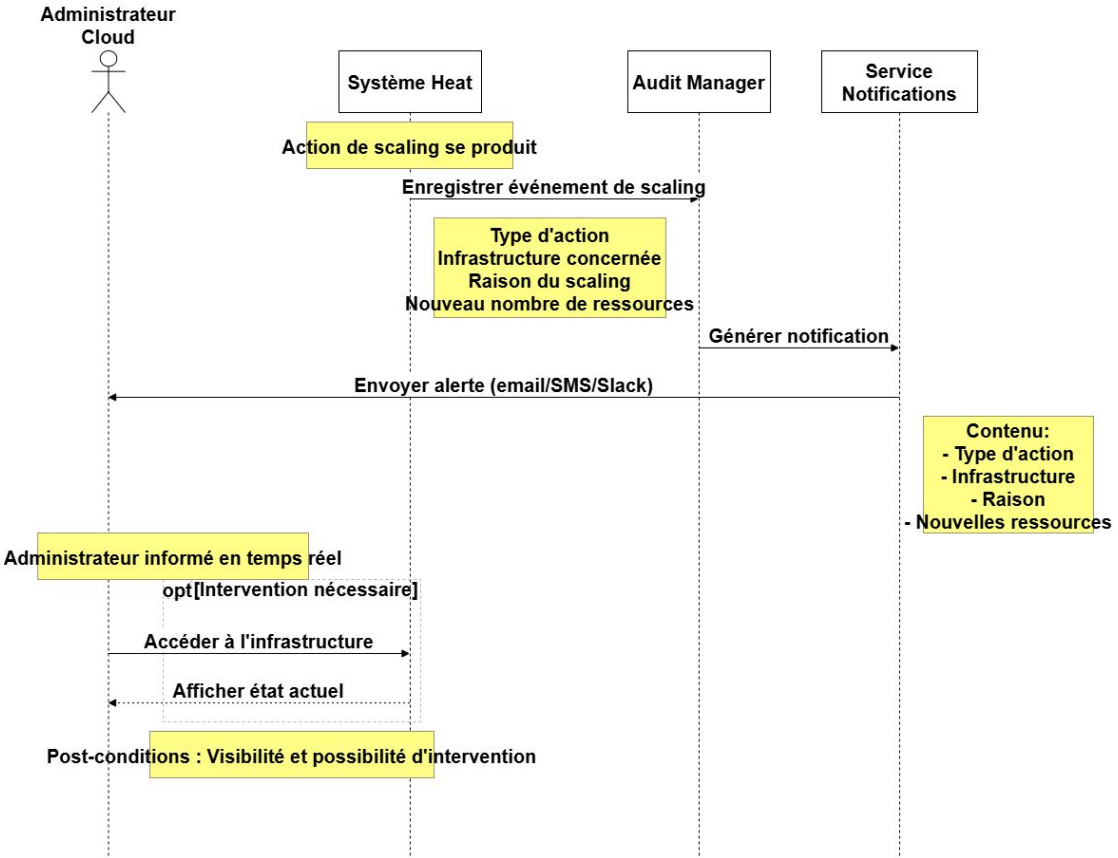
5. Diagramme de séquence

consulter l'historique



5. Diagramme de séquence

Recevoir les Alertes



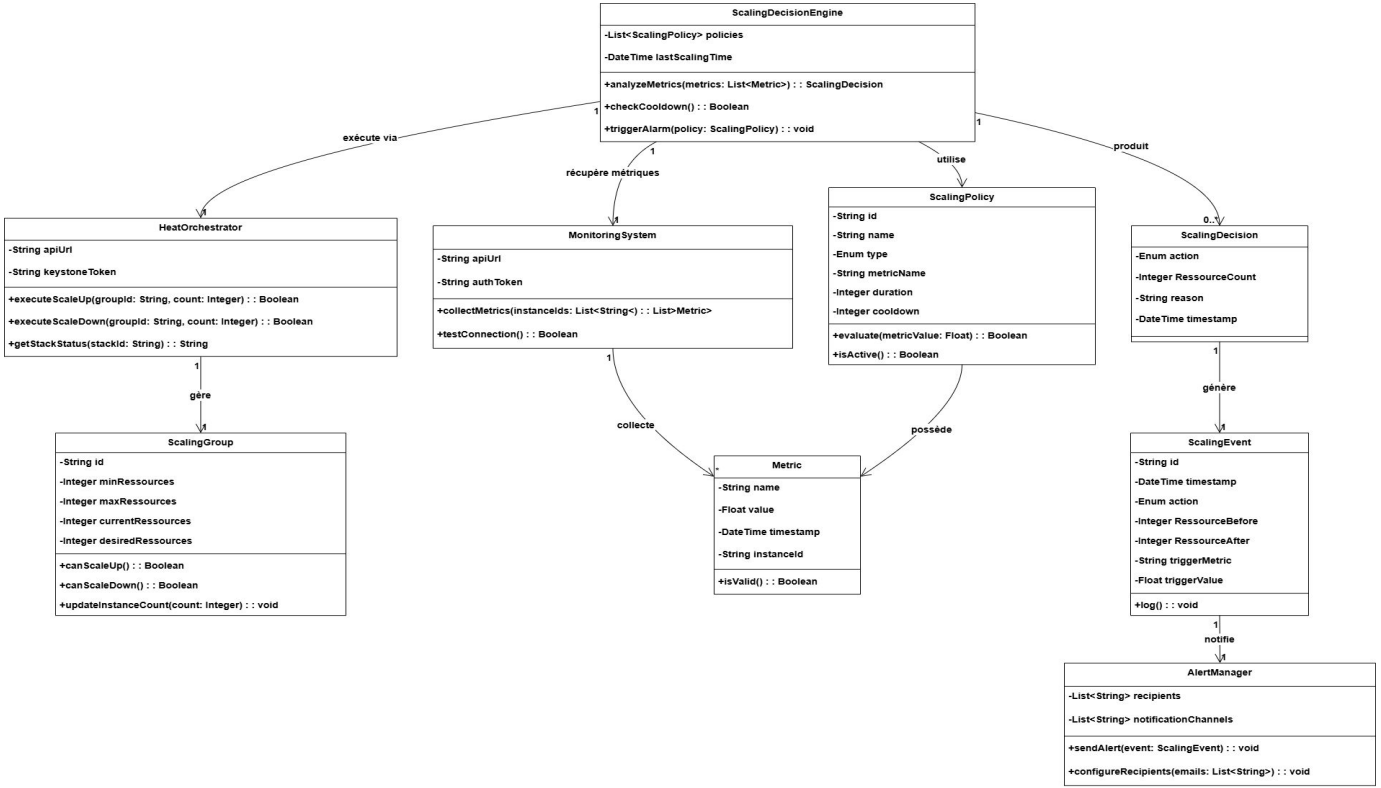
6. Diagramme de classe

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

6. Diagramme de classe



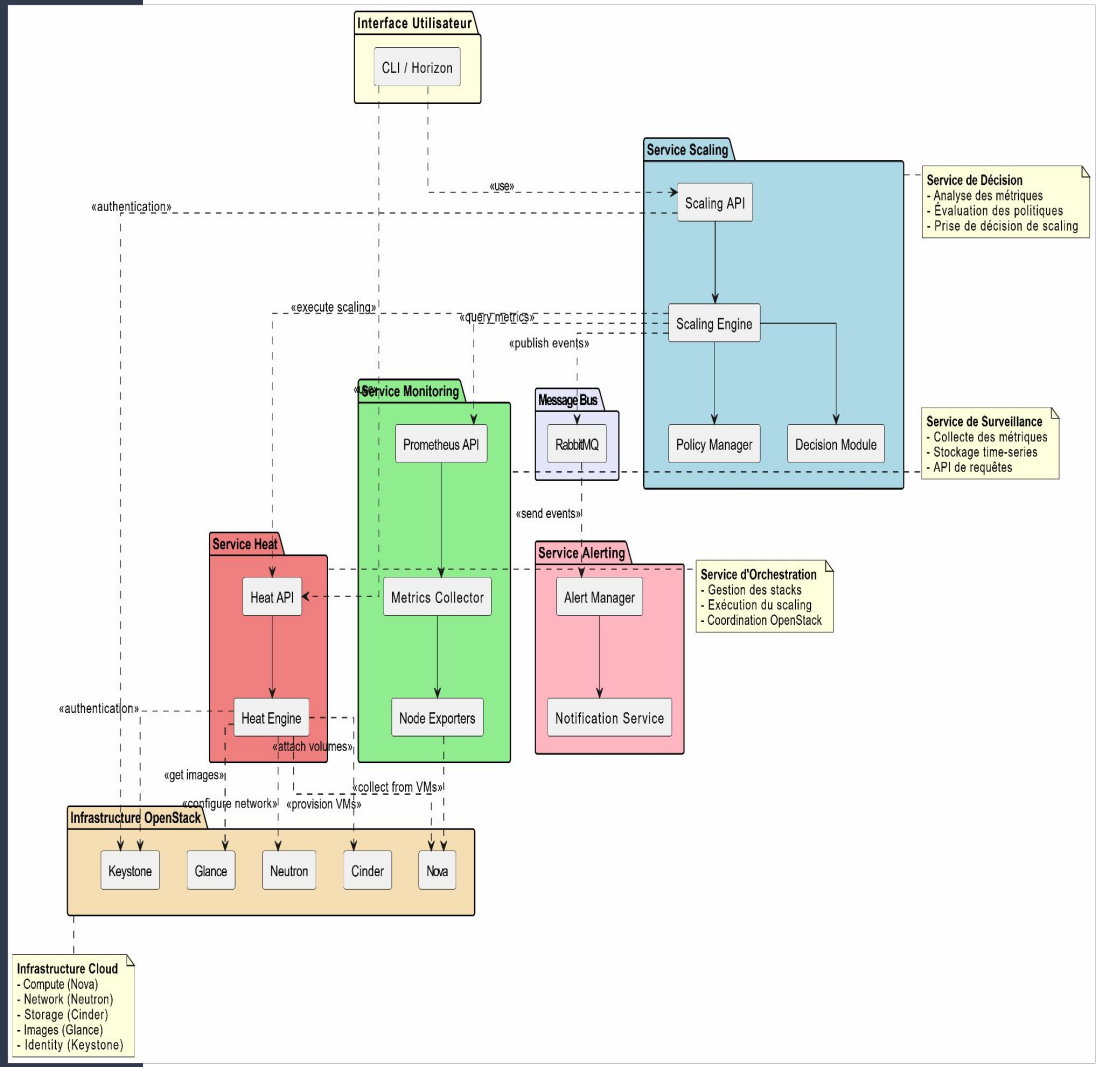
7. Diagramme de composant

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

7. Diagramme de composant



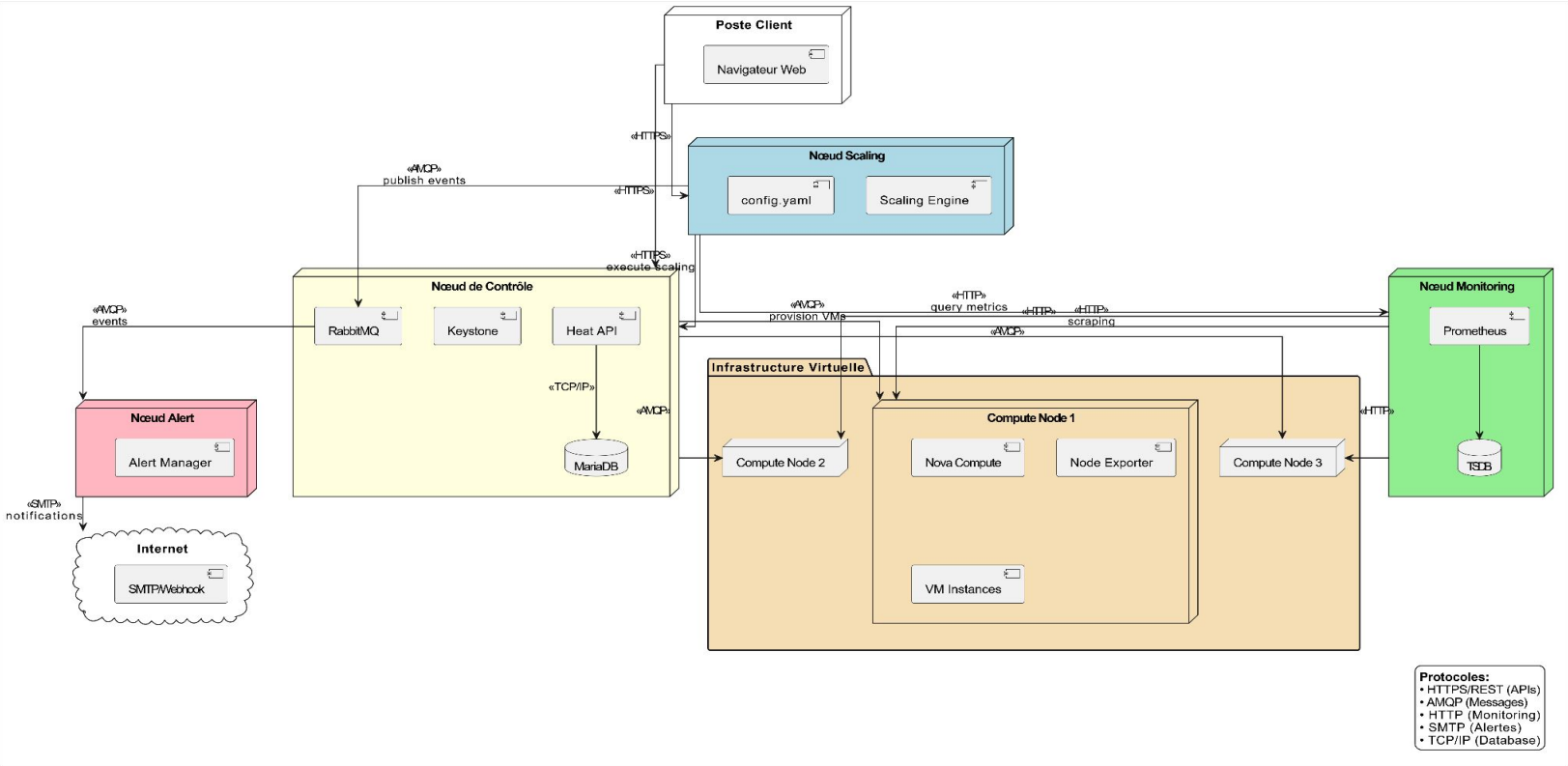
8. Diagramme Déploiement

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

8. Diagramme de déploiment



9- CONCLUSION

La conception détaillée dans ce dossier confirme la faisabilité technique des objectifs fixés. En s'appuyant sur le couplage entre Heat, Prometheus et un module de décision intelligent, nous avons conçu une architecture répondant aux critères de modularité, d'élasticité et d'automatisation intelligente.

Le passage d'une vision fonctionnelle à une vision technique a permis de mettre en lumière des points critiques, notamment la gestion sécurisée des communications inter-composants via l'authentification Keystone, la nécessité d'un mécanisme de cooldown pour éviter les oscillations de scaling, et l'importance de la traçabilité des décisions pour l'audit et l'optimisation des politiques. La structure modulaire choisie, avec la séparation stricte entre collecte de métriques, prise de décision et exécution, assure que le système pourra évoluer sans remettre en cause les fondations de l'orchestration.

Les diagrammes de séquence, de classes, de composants et de déploiement établissent une base solide pour l'implémentation, en garantissant que chaque composant dispose d'une responsabilité clairement définie et d'interfaces de communication standardisées. L'architecture proposée permet non seulement de dépasser les limitations identifiées du service Heat natif, mais aussi d'ouvrir la voie à des extensions futures telles que le scaling multi-métriques combinées ou l'intégration de l'intelligence artificielle pour la prédiction de charge.

Prochaines étapes : La validation de cette conception marque le début de la phase de Réalisation, qui comprendra l'implémentation du module de décision en Python, la configuration de Prometheus avec ses exporteurs, l'écriture des templates Heat modulaires, et la mise en place de l'environnement de test sur DevStack pour valider le comportement du système dans des scénarios réels de montée et de descente en charge.