# CAHIER DE CHARGE

## THÈME: Orchestration des ressources avec Heat

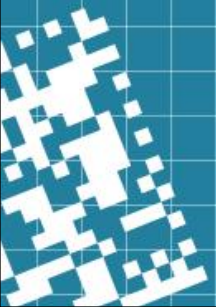
**RÉDIGÉ PAR:**
**Moudio Abega Laurent Stéphane**
**NGOUO Franck Leonel**

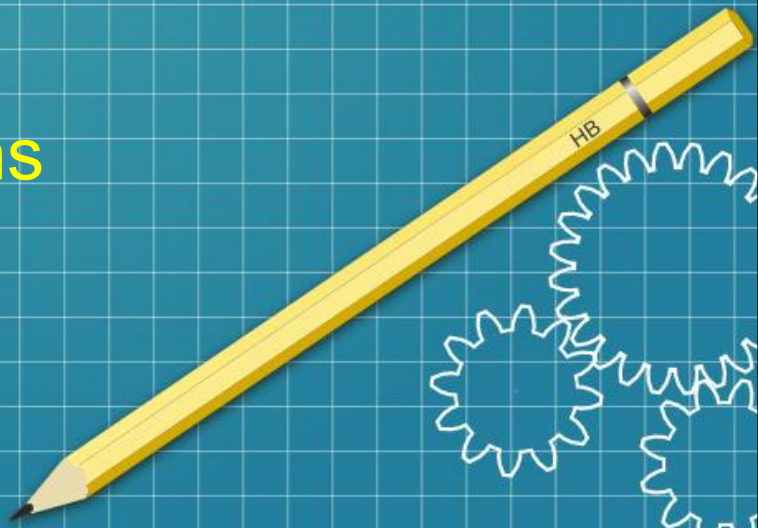
**Sous la supervision de : M NGUIMBUS Emmanuel**

**Master 2 SSI, 2025-2026**

# SOMMAIRE







# Introduction

Dans un contexte où les infrastructures informatiques deviennent de plus en plus dynamiques, l'automatisation du déploiement et de la gestion des ressources cloud est devenue essentielle.

OpenStack, solution open source de cloud computing, fournit un ensemble de services permettant de gérer les ressources de calcul, de stockage et de réseau.

Parmi ces services, Heat joue un rôle clé : il permet d'orchestrer et d'automatiser le déploiement d'environnements complets à l'aide de modèles déclaratifs (templates HOT).

Cependant, dans sa version actuelle, Heat présente certaines limites dans la modélisation d'architectures complexes et dans la gestion automatique de la scalabilité.

Ce projet vise donc à améliorer l'orchestration des ressources en développant des templates Heat avancés et en ajoutant des modules de scaling automatique personnalisés.

# I. Présentation de l'existant

OpenStack est une plateforme open source de gestion de cloud privé et public. Elle permet  d'administrer des ressources telles que :

Nova (Compute), Neutron (Networking), Cinder (Block Storage), Glance (Images), Keystone (Identity) et Horizon (Dashboard). OpenStack offre ainsi un cadre complet pour le déploiement, la supervision et la gestion des infrastructures cloud. Cependant, la coordination entre ces services nécessite un moteur d'orchestration efficace : Heat.

Heat est le service d'orchestration d'OpenStack. Il permet de décrire, déployer et gérer des ensembles de ressources (VM, réseaux, volumes, etc.) via des templates écrits en YAML (HOT).Grâce à Heat, les administrateurs peuvent automatiser la création d'environnements complets, gérer les dépendances entre ressources et appliquer des politiques de scaling et de haute disponibilité. Heat est donc le chef d'orchestre d'OpenStack : il transforme un modèle descriptif enune infrastructure opérationnelle.

# II. Contexte et Objectifs

## 1. Contexte

Les environnements cloud modernes nécessitent une orchestration capable de gérer automatiquement la montée ou la descente en charge et de déployer des architectures multi-niveaux sans intervention humaine. Bien que Heat remplisse  ces fonctions, il présente plusieurs limites qui freinent son adoption dans des scénarios complexes.

# II. Contexte et Objectifs

## 2. Limites

- **Complexité dans la modélisation des architectures multi-VM / multi-réseaux**
  - Les templates Heat deviennent rapidement longs et difficiles à maintenir dès qu'ils incluent plusieurs couches (réseau, sécurité, VM, stockage).
  - Les dépendances explicites entre ressources rendent la conception fastidieuse.
  - La réutilisation de portions de modèles est limitée, ce qui réduit la flexibilité.

**Conséquence : difficile de déployer des topologies complexes en une seule commande.**

# II. Contexte et Objectifs

## 2. Limites

- **Scaling automatique rigide et dépendant d'autres services**
  - Le scaling Heat repose sur Ceilometer et Aodh, deux services souvent difficiles à configurer et peu flexibles.Les dépendances explicites entre ressources rendent la conception fastidieuse.
  - Les politiques de scaling utilisent des seuils statiques (ex. : CPU > 70 %) qui ne reflètent pas toujours la charge réelle.
  - L'absence d'intégration avec des outils de monitoring modernes limite la réactivité.

**Conséquence : Le système ne s'adapte pas efficacement aux variations de charge.**

# II. Contexte et Objectifs

## 3. objectifs

- **Objectif Generale: Concevoir et implémenter une solution d'orchestration avancée avec OpenStack Heat permettant :**
  - **Le déploiement automatique d'architectures complexes (multi-VM et multi-réseaux)**
  - **La mise à l'échelle dynamique des ressources en fonction de la charge réelle.**

# II. Contexte et Objectifs

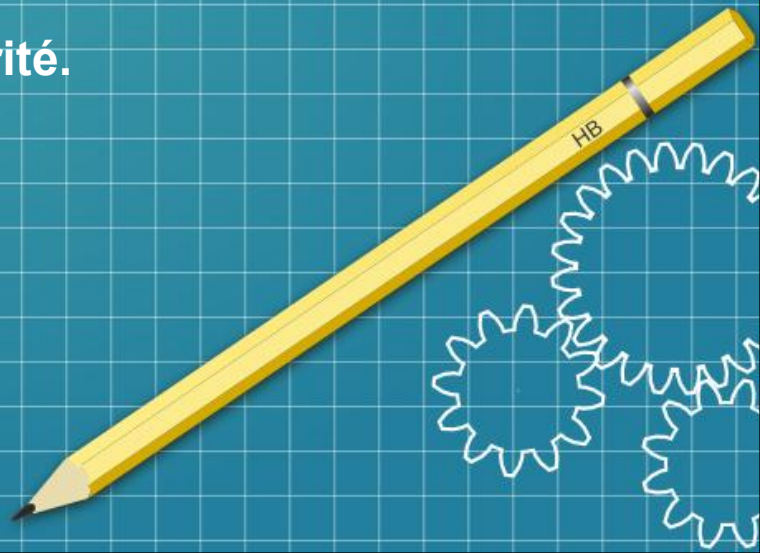
## 3. objectifs

❖ **Objectif Spécifiques :**

- **Concevoir des templates Heat complexes capables de décrire et déployer des topologies multi-VM et multi-réseaux.**
- **Développer des modules Heat personnalisés pour gérer le scaling automatique basé sur la charge (CPU, mémoire, trafic).**
- **Mettre en place un environnement de test basé sur DevStack pour valider les déploiements.**
- **Mesurer les performances et l'efficacité de la solution proposée.**

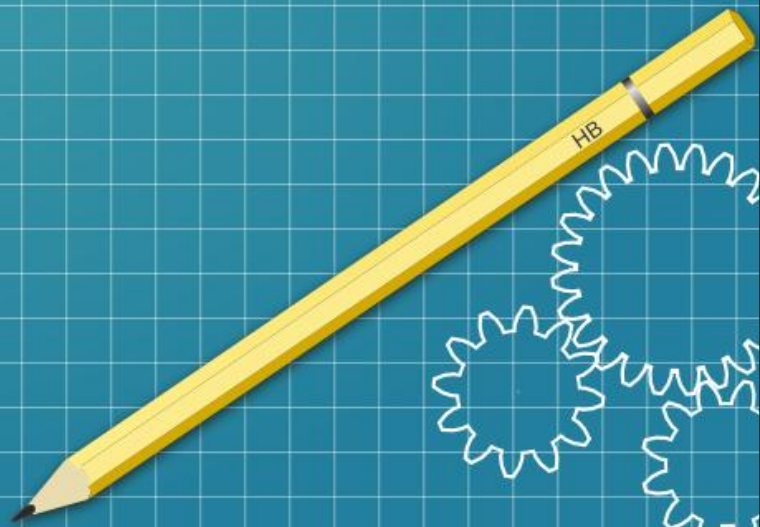
# III. Expression des Besoins

- **Besoins Fonctionnels:**
  - Création automatisée d’architectures complètes via un seul template.
  - Scaling automatique des instances en fonction de la charge mesurée.
  - Réutilisation des sous-templates pour la modularité.
  - Intégration avec un outil de monitoring externe .



# III. Expression des Besoins

- **Besoins Non Fonctionnels:**
  - **Performance : déploiement d'une infrastructure complexe plus rapidement .**
  - **Sécurité : isolation réseau et authentification.**
  - **Maintenabilité : code modulaire et documenté.**



# IV. Planification

| Phase | Livrable | Durée | Début | Fin |
|---|---|---|---|---|
| Étude et cahier des charges | Cahier de Charge | 1 semaine | 18/10/2025 | 25/10/2025 |
| Analyse | Cahier d'analyse | 1 semaines | 25/10/2025 | 01/11/2025 |
| Conception | Cahier de conception | 1 semaines | 01/11/2025 | 08/11/2025 |
| Mise en oeuvre | Développement | 4 semaines | 08/11/2025 | 06/12/2025 |
| Rédaction et soutenance | Rapport + présentation | 1 semaine | 06/12/2025 | 13/01/2026 |

# V. Budget

| Poste | Description | Coût unitaire (FCFA) | Quantité | Coût total (FCFA) |
|---|---|---|---|---|
| **Ingénieur Cloud OpenStack** | Conception des templates Heat complexes | 15 000 | 160h | **2 400 000** |
| **Développeur DevOps** | Développement du module de scaling adaptatif | 15 000 | 160h | **2 400 000** |
| **Total estimé** | | | | **4 800 000** |

# VI. Resume

| Catégorie | Fonctionnalités initiales de Heat | Fonctionnalités à ajouter dans le projet | Valeur ajoutée attendue |
|---|---|---|---|
| Orchestration de essources | Heat permet de déployer des ressources OpenStack (VM, réseaux, volumes, etc.) à partir de fichiers YAML appelés *Heat Orchestration Templates (HOT)* . | Extension de la logique d'orchestration avec des **templates imbriqués et modulaire;** pour gérer des topologies multi-VM, multi-réseaux et multi-niveaux. | Meilleure modularité, réutilisation du code et simplification de la maintenance des déploiements complexes. |
| iestion des épendances | Heat gère les dépendances entre ressources via la directive **depends_on** dans les templates. | Intégration de **modèles hiérarchiques** où les dépendances sont encapsulées dans des sous-templates spécialisés. | Diminution des erreurs humaines et amélioration de la lisibilité du code YAML. |
| caling utomatique | Basé sur Ceilometer et Aodh (services de télémétrie et d'alarmes) pour déclencher des actions de scaling selon des seuils prédéfinis. | Développement d'un **module Heat personnalisé** interfacé avec d'autre outils de monitoring pour un **scaling adaptatif** selon la charge réelle. | Scaling dynamique et intelligent, ajustement en temps réel selon les métriques système. |
| Monitoring | Limité aux outils internes OpenStack, nécessite une configuration complexe de Ceilometer. | Intégration native d'un **système de supervision externe** pour collecter et exploiter des métriques CPU, RAM, réseau, etc. | Visibilité accrue sur les performances et meilleure réactivité du système. |
| Déploiement | Exécution manuelle ou automatisée via Horizon (Dashboard) ou CLI, mais limitée à un seul template principal. | Automatisation complète via un **ensemble de templates orchestrés** en chaîne, déployant l'architecture complète en une seule commande. | Gain de temps et cohérence dans les déploiements. |

# Conclusion

Ce projet vise à renforcer les capacités du service Heat d'OpenStack en proposant une orchestration avancée et un scaling intelligent.
L'étude partira du constat des limites actuelles de Heat pour construire une solution concrète et mesurable :
des templates modulaires, faciles à maintenir, et un mécanisme de scaling automatique qui réagit à la charge réelle.
L'ensemble contribuera à rendre l'orchestration cloud plus performante, adaptable et alignée sur les besoins métiers actuels.