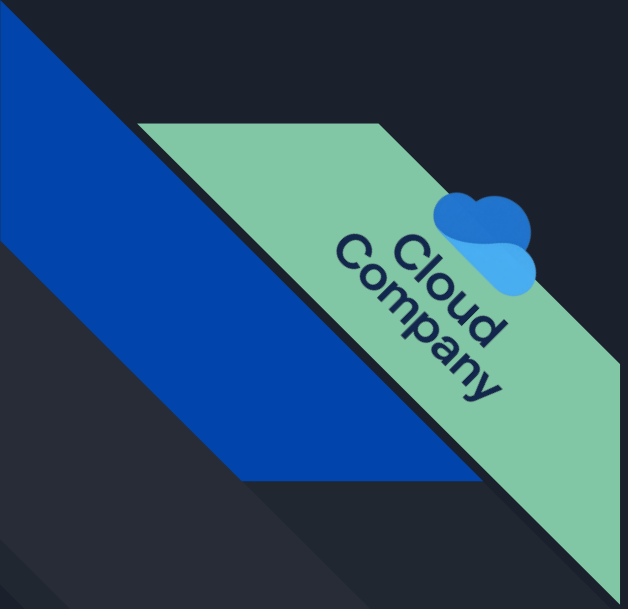


CAHIER D'ANALYSE

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

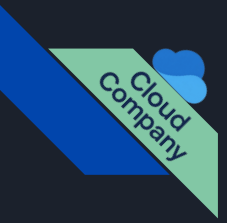
Rédiger par :

Ngouo Franck Leonel

MOUDIO ABEGA Laurent Stéphane

Supervisé par : M NGUIMBUS Emmanuel

PLAN DE PRÉSENTATION



1. Introduction
2. Contexte et Objectifs
3. Périmètre Fonctionnel et Exigences
4. Architecture des Templates Avancés et du Scaling
5. Modélisation UML (Conception de la Solution)
6. Conclusion

1. Introduction

La transformation numérique des organisations repose de plus en plus sur l'adoption des infrastructures cloud, capables d'offrir élasticité, automatisation et haute disponibilité. Dans ce contexte, les plateformes de cloud privé telles qu'OpenStack constituent une alternative stratégique pour les administrations publiques et les entreprises souhaitant conserver la maîtrise de leurs données et de leurs infrastructures.

OpenStack Heat est le service d'orchestration qui permet de décrire et de déployer des infrastructures cloud via des modèles déclaratifs appelés Heat Orchestration Templates (HOT). Toutefois, l'utilisation de Heat dans des scénarios complexes, impliquant plusieurs machines virtuelles, plusieurs réseaux et des mécanismes avancés de montée et de descente en charge, révèle un certain nombre de limites architecturales et opérationnelles.

Le présent cahier d'analyse vise à définir, de manière structurée et académique, les besoins, les exigences et la solution technique proposée pour une orchestration avancée des ressources cloud basée sur des templates Heat modulaires et un mécanisme de scaling automatique sécurisé et extensible.

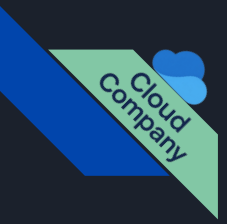
2. Contexte et Objectifs

2.1. 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.



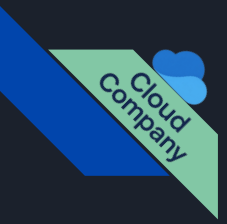
2. Contexte et Objectifs

2.2. Problématique identifié

L’analyse de l’existant met en évidence deux limitations majeures du service Heat :

- **L1 – Complexité de modélisation** : les architectures multi-VM et multi-réseaux sont souvent décrites à l’aide de templates monolithiques, longs, difficiles à maintenir et peu réutilisables.
- **L2 – Rigidité du scaling automatique** : les mécanismes de montée et de descente en charge reposent sur des seuils statiques et sont faiblement intégrés aux métriques applicatives réelles, ce qui limite leur efficacité et leur sécurité.

Ces limitations freinent l’adoption de Heat dans des environnements cloud complexes et nécessitent une approche architecturale plus modulaire et plus intelligente.



2. Contexte et Objectifs

2.3. Objectifs du Projet

L'objectif général du projet est de concevoir et de valider une solution d'orchestration avancée des ressources cloud OpenStack, reposant sur Heat, capable de gérer efficacement des architectures complexes et dynamiques.

Les objectifs spécifiques sont les suivants :

- concevoir des templates Heat modulaires basés sur des templates imbriqués ;
- automatiser le déploiement d'architectures multi-VM et multi-réseaux à partir d'un template racine unique
- mettre en place un mécanisme de scaling automatique basé sur des métriques réelles (CPU, mémoire, trafic) ;
- intégrer un système de monitoring externe pour la collecte et l'analyse des métriques ;
- développer un adaptateur de métriques sécurisé permettant l'interaction contrôlée avec le moteur Heat ;
- valider la solution dans un environnement de test basé sur DevStack.a

3. Périmètre Fonctionnel et Exigences

3.1. Besoins Fonctionnels (BF)

- **BF1 – Déploiement automatisé complet** : le système doit permettre le déploiement d'une infrastructure cloud complète à partir d'un seul template Heat.
- **BF2 – Scaling automatique intelligent** : la solution doit ajuster dynamiquement le nombre d'instances en fonction de la charge mesurée.
- **BF3 – Modularité des templates** : les templates doivent être organisés en sous-ensembles réutilisables afin d'améliorer la maintenabilité et l'évolutivité.
- **BF4 – Intégration du monitoring** : le système doit être capable de consommer des métriques issues d'un outil de monitoring externe.

3. Périmètre Fonctionnel et Exigences

3.2. Besoins Non Fonctionnels (BF)

- **BNF1 – Performance** : le temps de déploiement automatisé doit être inférieur à celui d'un déploiement manuel équivalent.
- **BNF2 – Sécurité** : l'architecture doit garantir l'isolation réseau, l'authentification des composants et la protection des mécanismes de scaling.
- **BNF3 – Maintenabilité** : les templates et modules développés doivent être documentés, lisibles et faciles à faire évoluer.

5. Modélisation UML

Diagramme de Cas d'Utilisation :

Acteurs :

Acteur 1 : Administrateur Cloud

Rôle : Responsable de la conception des templates et de la configuration des politiques de scaling.

Responsabilités :

- Créer des templates Heat modulaires
- Configurer les politiques de scaling automatique
- Superviser les déploiements
- Gérer les bibliothèques de sous-templates

5. Modélisation UML

Diagramme de Cas d'Utilisation :

Acteurs :

Acteur 2 : Utilisateur Cloud

Rôle : Utilisateur final qui déploie des infrastructures à partir de templates.

Responsabilités :

- Déployer des stacks à partir de templates fournis
- Consulter l'état de ses déploiements
- Bénéficier du scaling automatique sans intervention manuelle

5. Modélisation UML

Diagramme de Cas d'Utilisation :

Acteurs :

Acteur 3 : Système Heat

Rôle : Moteur d'orchestration qui exécute les templates et gère le scaling.

Responsabilités :

- Interpréter et exécuter les templates HOT
- Déployer les ressources OpenStack
- Surveiller les métriques et déclencher le scaling
- Gérer le cycle de vie des stacks

5. Modélisation UML

Diagramme de Cas d'Utilisation :

Acteurs :

Acteur 4 : Système de Monitoring

Rôle : Système externe (Prometheus, Zabbix) qui collecte les métriques.

Responsabilités :

- Collecter les métriques des instances (CPU, RAM, réseau, métriques applicatives)
- Fournir les métriques au système Heat
- Stocker l'historique des métriques

5. Modélisation UML

Diagramme de Cas d'Utilisation :

CATALOGUE DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

- **CU1.1** : Créer un template Heat simple
- **CU1.2** : Créer un sous-template réutilisable
- **CU1.3** : Assembler des sous-templates dans un template principal
- **CU1.4** : Valider un template avant déploiement
- **CU1.5** : Organiser les templates dans une bibliothèque

5. Modélisation UML

Diagramme de Cas d'Utilisation :

CATALOGUE DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

- **CU2.1** : Déployer une architecture simple (mono-VM)
- **CU2.2** : Déployer une architecture complexe (multi-VM, multi-réseaux)
- **CU2.3** : Consulter l'état d'un déploiement
- **CU2.4** : Mettre à jour une infrastructure déployée
- **CU2.5** : Supprimer une infrastructure

5. Modélisation UML

Diagramme de Cas d'Utilisation :

CATALOGUE DES CAS D'UTILISATION

Groupe 3 : Configuration du Scaling Automatique

- **CU3.1** : Définir une politique de scale up basée sur les métriques
- **CU3.2** : Définir une politique de scale down basée sur les métriques
- **CU3.3** : Configurer les seuils de déclenchement du scaling
- **CU3.4** : Configurer l'intégration avec le système de monitoring

5. Modélisation UML

Diagramme de Cas d'Utilisation :

CATALOGUE DES CAS D'UTILISATION

Groupe 4 : Surveillance et Métriques

- **CU4.1** : Collecter les métriques des instances
- **CU4.2** : Visualiser les métriques en temps réel
- **CU4.3** : Consulter l'historique des actions de scaling
- **CU4.4** : Recevoir des alertes sur les événements de scaling

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

CU1.1 : Créer un template Heat simple

Acteur principal : Administrateur Cloud
Objectif : Créer un template Heat basique pour déployer une infrastructure simple.

Préconditions :
- L'administrateur est authentifié
- L'éditeur de templates est accessible

Scénario nominal :
1. L'administrateur accède à l'outil de création de templates
2. Il définit les informations de base (nom, description)
3. Il spécifie les paramètres d'entrée nécessaires (image, flavor, réseau)
4. Il définit les ressources à déployer (instances, réseaux, volumes)
5. Il définit les sorties du template (IPs, URLs d'accès)
6. Il sauvegarde le template
7. Le système valide la syntaxe du template

Post-conditions :
- Un template simple est créé et prêt à être utilisé
- Le template peut être déployé par les utilisateurs

Scénarios alternatifs :
A1 : Erreur de syntaxe
- À l'étape 7, le système détecte une erreur de syntaxe
- Un message d'erreur explicite est affiché
- L'administrateur corrige l'erreur
- Retour à l'étape 6

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

CU1.2 : Créer un sous-template réutilisable

Acteur principal : Administrateur Cloud
Objectif : Créer un composant modulaire réutilisable (ex : serveur web, base de données).

Préconditions :
- L'administrateur est authentifié
- Il a identifié un besoin de composant réutilisable

Scénario nominal :
1. L'administrateur accède à la création de sous-templates
2. Il définit l'objectif du composant (ex : serveur web Apache)
3. Il spécifie les paramètres d'entrée du sous-template
4. Il définit les ressources internes du composant
5. Il définit les sorties du composant (IP du serveur, port d'écoute)
6. Il documente l'utilisation du sous-template
7. Il sauvegarde le sous-template dans la bibliothèque

Post-conditions :
- Un sous-template réutilisable est créé
- Il est disponible dans la bibliothèque pour tous les administrateurs
- Il peut être intégré dans des templates principaux

Scénarios alternatifs :
A1 : Sous-template avec le même nom existe déjà
- Le système demande de choisir un autre nom ou de versionner
- L'administrateur choisit une option
- Le sous-template est sauvegardé

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

CU1.3 : Assembler des sous-templates dans un template principal

Objectif : Créer une architecture complexe en combinant des sous-templates existants.

Préconditions :

- L'administrateur est authentifié
- Des sous-templates existent dans la bibliothèque

Scénario nominal :

1. L'administrateur crée un nouveau template principal
2. Il parcourt la bibliothèque de sous-templates
3. Il sélectionne les sous-templates nécessaires (ex : serveur web, base de données, load balancer)
4. Il les glisse dans le template principal
5. Il configure les paramètres à transmettre à chaque sous-template
6. Il définit les connexions entre les composants (réseau, dépendances)
7. Il valide l'assemblage
8. Le système vérifie la cohérence globale

Post-conditions :

- Un template principal complexe mais structuré est créé
- Il utilise des composants réutilisables et testés
- Le template est plus simple à maintenir qu'un template monolithique

Scénarios alternatifs :

A1 : Paramètres incompatibles entre sous-templates

- À l'étape 8, le système détecte une incompatibilité
- L'administrateur ajuste les paramètres
- Retour à l'étape 5

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

CU1.4 : Valider un template avant déploiement

Acteur principal : Administrateur Cloud, Utilisateur Cloud
Objectif : Vérifier qu'un template est correct avant de l'utiliser.
Préconditions :
● Un template existe ou vient d'être créé

Scénario nominal :
1. L'acteur sélectionne un template
2. Il lance la validation
3. Le système vérifie :
○ La syntaxe YAML
○ La structure HOT
○ L'existence des ressources référencées (images, flavors)
○ La cohérence des dépendances
4. Le système affiche un rapport de validation détaillé
5. Si le template est valide, l'acteur peut procéder au déploiement

Post-conditions :
● L'acteur sait si le template est utilisable
● Les erreurs potentielles sont identifiées avant déploiement

Scénarios alternatifs :
A1 : Template invalide
● À l'étape 4, des erreurs sont détectées
● Le rapport liste toutes les erreurs avec leur localisation
● L'acteur corrige les erreurs ou choisit un autre template

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 1 : Gestion des Templates Modulaires

CU1.5 : Organiser les templates dans une bibliothèque

Acteur principal : Administrateur Cloud
Objectif : Structurer et catégoriser les templates pour faciliter leur recherche.

Préconditions :
● Des templates et sous-templates existent
Scénario nominal :
1. L'administrateur accède à la bibliothèque de templates
2. Il crée des catégories (ex : Web, Database, Network, Storage)
3. Il attribue des tags aux templates (ex : production, dev, wordpress..)
4. Il classe les templates dans les catégories appropriées
5. Il peut définir des templates comme "favoris" ou "recommandés"
6. La bibliothèque devient facilement navigable

Post-conditions :
● Les templates sont organisés logiquement
● Les utilisateurs trouvent rapidement le template dont ils ont besoin
● La réutilisation est facilitée

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

CU2.1 : Déployer une architecture simple (mono-VM)

Acteur principal : Utilisateur Cloud
Objectif : Déployer rapidement une infrastructure basique.
Préconditions :
● L'utilisateur est authentifié
● Un template simple est disponible
● Les quotas de ressources sont suffisants

Scénario nominal :
1. L'utilisateur accède à l'interface de déploiement
2. Il sélectionne un template simple (ex : "Serveur Web Simple")
3. Le système affiche les paramètres à fournir
4. L'utilisateur renseigne :
○ Nom de l'infrastructure ○ Image à utiliser ○ Taille de l'instance (flavor) ○ Clé SSH pour l'accès
5. Il lance le déploiement
6. Le système Heat orchestre la création des ressources
7. Les ressources sont créées progressivement
8. Le système affiche l'avancement en temps réel
9. Le déploiement se termine avec succès
10. Les informations d'accès (IP, URL) sont affichées

Post-conditions :
● Une infrastructure fonctionnelle est déployée
● L'utilisateur peut accéder à ses ressources
● L'infrastructure est prête à être utilisée

Scénarios alternatifs :
A1 : Quotas insuffisants
● À l'étape 6, le système détecte un manque de ressources
● Un message explique les quotas dépassés
● Le déploiement est annulé
● L'utilisateur doit libérer des ressources ou demander une augmentation de quotas
A2 : Échec de création d'une ressource
● Une ressource ne peut être créée
● Le système annule automatiquement le déploiement
● Les ressources déjà créées sont supprimées
● Un message d'erreur explique le problème

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

CU2.2 : Déployer une architecture complexe (multi-VM, multi-réseaux)
Acteur principal : Utilisateur Cloud, Administrateur Cloud

Objectif : Déployer une infrastructure complète multi-couches en une seule opération.
Préconditions :
● L'acteur est authentifié
● Un template complexe (utilisant des sous-templates) est disponible
● Les quotas sont suffisants

Scénario nominal :
1. L'acteur sélectionne un template complexe (ex : "Application Web 3-tiers")
2. Le système affiche une description de l'architecture :
○ Couche Web (2-3 serveurs web + load balancer) ○ Couche Application (2 serveurs applicatifs)
○ Couche Database (1 serveur database master + 1 slave) ○ Réseaux privés écurisés
3. L'acteur fournit les paramètres globaux
4. Il lance le déploiement
5. Le système Heat orchestre la création dans l'ordre :
○ Création des réseaux ○ Création des groupes de sécurité ○ Déploiement des instances par couche
○ Configuration des interconnexions
6. L'acteur peut suivre la progression en temps réel
7. Le déploiement se termine avec succès
8. Les informations d'accès et la topologie sont affichées

Post-conditions :
● Une infrastructure complète multi-niveaux est opérationnelle
● Toutes les couches peuvent communiquer correctement
● L'application peut être utilisée

Scénarios alternatifs :
A1 : Échec sur une couche
● Si une couche échoue, le système annule tout
● Toutes les ressources sont supprimées
● L'acteur reçoit un rapport d'erreur détaillé

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

CU2.3 : Déployer une architecture complexe (multi-VM, multi-réseaux)
Acteur principal : Utilisateur Cloud, Administrateur Cloud

Objectif : Déployer une infrastructure complète multi-couches en une seule opération.
Préconditions :
● L'acteur est authentifié
● Un template complexe (utilisant des sous-templates) est disponible
● Les quotas sont suffisants

Scénario nominal :
1. L'acteur sélectionne un template complexe (ex : "Application Web 3-tiers")
2. Le système affiche une description de l'architecture :
○ Couche Web (2-3 serveurs web + load balancer) ○ Couche Application (2 serveurs applicatifs)
○ Couche Database (1 serveur database master + 1 slave) ○ Réseaux privés écurisés
3. L'acteur fournit les paramètres globaux
4. Il lance le déploiement
5. Le système Heat orchestre la création dans l'ordre :
○ Création des réseaux ○ Création des groupes de sécurité ○ Déploiement des instances par couche
○ Configuration des interconnexions
6. L'acteur peut suivre la progression en temps réel
7. Le déploiement se termine avec succès
8. Les informations d'accès et la topologie sont affichées

Post-conditions :
● Une infrastructure complète multi-niveaux est opérationnelle
● Toutes les couches peuvent communiquer correctement
● L'application peut être utilisée

Scénarios alternatifs :
A1 : Échec sur une couche
● Si une couche échoue, le système annule tout
● Toutes les ressources sont supprimées
● L'acteur reçoit un rapport d'erreur détaillé

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

CU2.4 : Mettre à jour une infrastructure déployée
Acteur principal : Administrateur Cloud
Objectif : Modifier une infrastructure existante sans la redéployer complètement.
Préconditions :
● Une infrastructure est déployée et opérationnelle
● L'administrateur souhaite modifier certains paramètres

Scénario nominal :
1. L'administrateur sélectionne une infrastructure
2. Il choisit "Mettre à jour"
3. Il peut modifier :
○ Des paramètres (taille des instances, nombre de serveurs)
○ Ajouter de nouveaux composants ○ Retirer des composants obsolètes
4. Le système calcule les changements nécessaires
5. Il affiche un aperçu des modifications
6. L'administrateur confirme
7. Le système applique les changements progressivement
8. L'infrastructure est mise à jour sans interruption de service

Post-conditions :
● L'infrastructure est modifiée selon les nouveaux besoins
● Les changements sont appliqués avec un minimum d'interruption

Scénarios alternatifs :

A1 : Modification nécessitant un redémarrage
● Le système avertit que certaines instances devront être redémarrées
● L'administrateur confirme ou annule

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 2 : Déploiement d'Infrastructures

CU2.5 : Supprimer une infrastructure

Acteur principal : Utilisateur Cloud, Administrateur Cloud
Objectif : Retirer une infrastructure devenue inutile.
Préconditions :
- Une infrastructure existe
- L'acteur a les droits de suppression

Scénario nominal :
1. L'acteur sélectionne une infrastructure
2. Il choisit "Supprimer"
3. Le système affiche toutes les ressources qui seront supprimées
4. Une confirmation est demandée
5. L'acteur confirme
6. Le système supprime toutes les ressources dans l'ordre approprié
7. Les quotas sont libérés
8. L'infrastructure disparaît de la liste

Post-conditions :
- Toutes les ressources sont supprimées
- Les coûts associés cessent
- Les quotas sont disponibles pour d'autres usages

Scénarios alternatifs :
A1 : Annulation
- L'acteur annule à l'étape 5
- Aucune ressource n'est supprimée

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 3 : Configuration du Scaling Automatique

CU3.1 : Définir une politique de scale up basée sur les métriques

Acteur principal : Administrateur Cloud
Objectif : Configurer l'augmentation automatique des ressources en cas de surcharge.
Préconditions :
● Une infrastructure avec un groupe d'instances est déployée
● Le système de monitoring est opérationnel

Scénario nominal :
1. L'administrateur accède à la configuration du scaling
2. Il sélectionne le groupe d'instances à surveiller
3. Il définit les conditions de scale up :
○ Métrique à surveiller (CPU, RAM, Trafic réseau, métrique applicative)○ Seuil de déclenchement (ex : CPU > 75%)
○ Durée d'observation (ex : pendant 5 minutes)
4. Il définit l'action à effectuer :
○ Ou pourcentage d'augmentation (ex : +50%)
5. Il définit le cooldown (temps d'attente avant une nouvelle action)
6. Il sauvegarde la politique
7. Le système active la surveillance

Post-conditions :
● La politique de scale up est active
● Le système surveille en continu les métriques
● L'infrastructure s'adaptera automatiquement à la charge

Scénarios alternatifs :
A1 : Métrique non disponible
● Le système avertit que la métrique choisie n'est pas collectée
● L'administrateur configure le monitoring ou choisit une autre métrique

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 3 : Configuration du Scaling Automatique

CU3.2 : Définir une politique de scale down basée sur les métriques

Acteur principal : Administrateur Cloud
Objectif : Configurer la réduction automatique des ressources en cas de sous-charge.
Préconditions :
● Une infrastructure avec un groupe d'instances est déployée
● Une politique de scale up existe (recommandé)

Scénario nominal :
1. L'administrateur accède à la configuration du scaling
2. Il sélectionne le groupe d'instances
3. Il définit les conditions de scale down :
○ Métrique à surveiller
○ Seuil bas (ex : CPU < 30%)
○ Durée d'observation (généralement plus longue que pour le scale up)
4. Il définit l'action :
○ Pourcentage des ressources à Retirer (ex : -30 %)
5. Il définit le cooldown (généralement
plus long que pour le scale up)
6. Il sauvegarde la politique
7. Le système active la surveillance

Post-conditions :
● La politique de scale down est active
● L'infrastructure pourra réduire ses coûts automatiquement lors des périodes de faible
charge

Scénarios alternatifs :

A1 : Risque de suppression excessive
● Le système avertit si le nombre minimum de % de Ressources n'est pas défini
● L'administrateur définit un minimum pour garantir la disponibilité

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 3 : Configuration du Scaling Automatique

CU3.3 : Configurer les seuils de déclenchement du scaling
Acteur principal : Administrateur Cloud
Objectif : Ajuster finement les conditions qui déclenchent le scaling.
Préconditions :
● Une politique de scaling est en cours de configuration

Scénario nominal :
1. L'administrateur sélectionne une métrique (CPU, RAM, réseau, métrique custom)
2. Il définit :
○ Le seuil (valeur numérique ou pourcentage)
○ L'opérateur (>, <, >=, <=)
○ La durée d'évaluation (fenêtre temporelle)
3. Il peut combiner plusieurs conditions (ex : CPU > 70% ET RAM > 80%)
4. Il valide la configuration

Post-conditions :
● Les seuils sont configurés de manière pertinente
● Le scaling se déclenchera au bon moment

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 3 : Configuration du Scaling Automatique

CU3.4 : Configurer l'intégration avec le système de monitoring

Acteur principal : Administrateur Cloud
Objectif : Connecter Heat à l'outil de monitoring externe pour obtenir des métriques fiables.
Préconditions :
● Un système de monitoring est déployé
● L'administrateur a les informations de connexion

Scénario nominal :
1. L'administrateur accède à la configuration de l'intégration
2. Il sélectionne le type de système de monitoring
3. Il fournit les informations de connexion :○ URL de l'API
○ Identifiants d'authentification ○ Port de communication
4. Il teste la connexion
5. Le système valide que les métriques sont accessibles
6. Il configure la fréquence de collecte des métriques
7. Il sauvegarde la configuration

Post-conditions :
● Heat peut interroger le système de monitoring
● Les métriques sont disponibles pour les décisions de scaling
● Le système est prêt pour le scaling automatique

Scénarios alternatifs :
A1 : Échec de connexion
● Le test de connexion échoue
● L'administrateur vérifie les paramètres et réessaye

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 4 : Surveillance et Métriques

CU4.1 : Collecter les métriques des instances

Acteur principal : Système de Monitoring
Objectif : Récupérer automatiquement les métriques nécessaires au scaling.
Préconditions :
● Des instances sont déployées
● Le système de monitoring est configuré

Scénario nominal :
1. Le système de monitoring collecte automatiquement :
○ Utilisation CPU (%) ○ Utilisation RAM (%) ○ Trafic réseau (Mbps)
○ Métriques applicatives si disponibles
2. Les métriques sont horodatées et stockées par le système
3. le système interroge périodiquement le système de monitoring
4.les métriques sont Récupérés et agrégé
5. Les métriques sont utilisées pour évaluer les politiques de scaling

Post-conditions :
● Des métriques à jour sont disponibles

Scénarios alternatifs :
A1 : Agent de monitoring non accessible
● Le système génère une alerte
● Les dernières valeurs connues sont utilisées temporairement

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 4 : Surveillance et Métriques

CU4.2 : Visualiser les métriques en temps réel

Acteur principal : Administrateur Cloud, Utilisateur Cloud
Objectif : Observer l'état de charge de l'infrastructure.
Préconditions :
● Des métriques sont collectées
● Une interface de visualisation est disponible

Scénario nominal :
1. L'acteur accède à l'interface de monitoring
2. Il sélectionne son infrastructure
3. Le système affiche des graphiques en temps réel :
○ Courbe d'utilisation CPU
○ Courbe d'utilisation RAM
○ Trafic réseau
○ Nombre d'instances actives
4. Les graphiques se rafraîchissent automatiquement
5. L'acteur peut ajuster la période d'affichage (dernière heure, jour, semaine

Post-conditions :
● L'acteur a une visibilité sur l'état de son infrastructure
● Il peut anticiper les besoins ou détecter les anomalies

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 4 : Surveillance et Métriques

CU4.3 : Consulter l'historique des actions de scaling

Acteur principal : Administrateur Cloud
Objectif : Analyser les actions de scaling passées pour optimiser les politiques.
Préconditions :
● Des actions de scaling ont eu lieu

Scénario nominal :
1. L'administrateur accède à l'historique du scaling
2. Le système affiche la liste des événements :
○ Date et heure
○ Type d'action (scale up / scale down)
○ Nombre d'instances avant/après
○ Métrique qui a déclenché l'action
○ Valeur de la métrique
○ Durée de l'opération
3. L'administrateur peut filtrer par période ou type d'action
4. Il peut exporter l'historique pour analyse

Post-conditions :
● L'administrateur comprend comment le scaling se comporte
● Il peut ajuster les politiques si nécessaire

DESCRIPTIONS DÉTAILLÉES DES CAS D'UTILISATION

Groupe 4 : Surveillance et Métriques

CU4.4 : Recevoir des alertes sur les événements de scaling

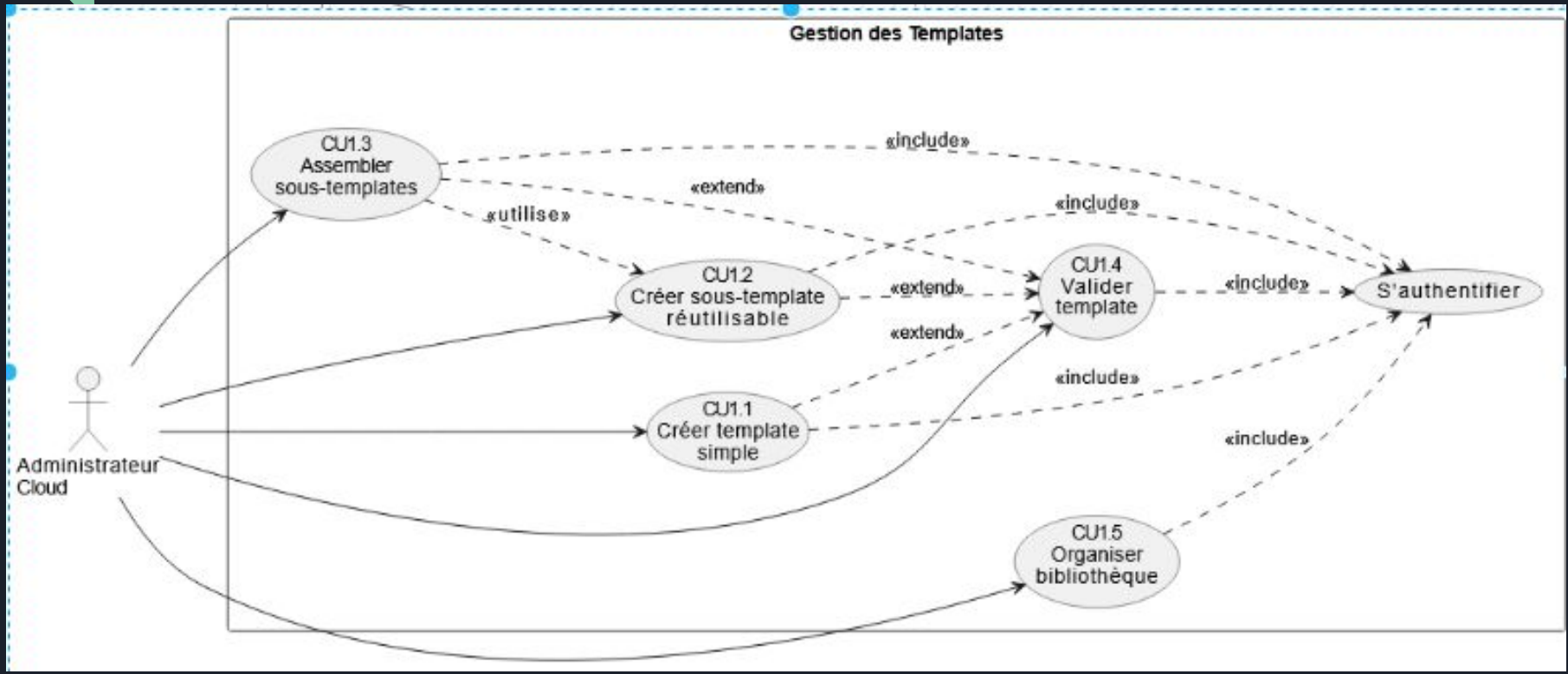
Acteur principal : Administrateur Cloud
Objectif : Être informé des actions de scaling importantes.
Préconditions :
● Les notifications sont configurées

Scénario nominal :
1. Une action de scaling se produit
2. Le système génère une notification3. L'administrateur reçoit un message (email, SMS, Slack) contenant :
○ Type d'action
○ Infrastructure concernée
○ Raison du scaling
○ Nouveau ù de ressources
4. L'administrateur est informé en temps réel

Post-conditions :
● L'administrateur a une visibilité sur les changements automatiques
● Il peut intervenir si nécessaire

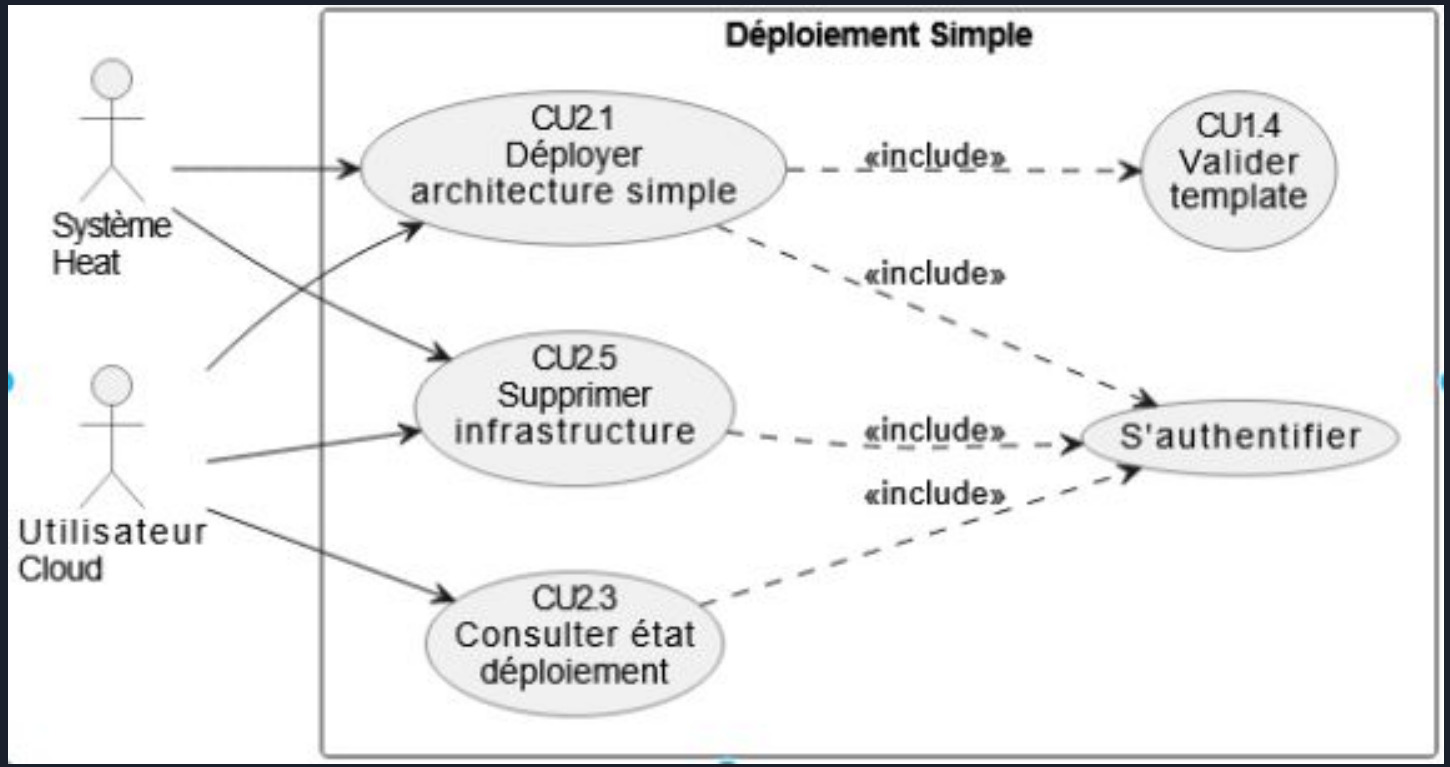
5. Modélisation UML

Diagramme de Cas d'Utilisation : Gestion Des Templates



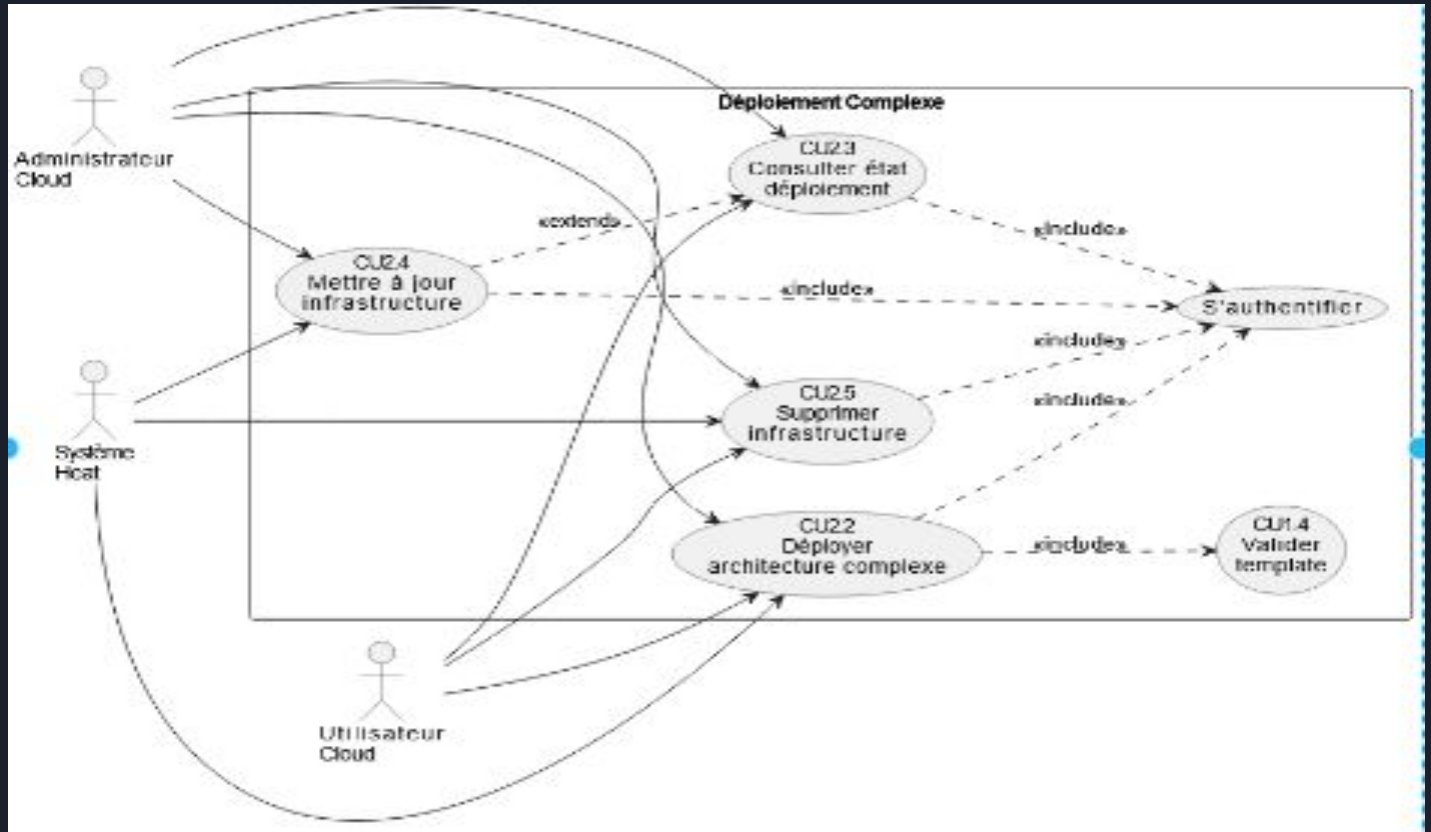
5. Modélisation UML

Diagramme de Cas d'Utilisation : Deploiement Simple



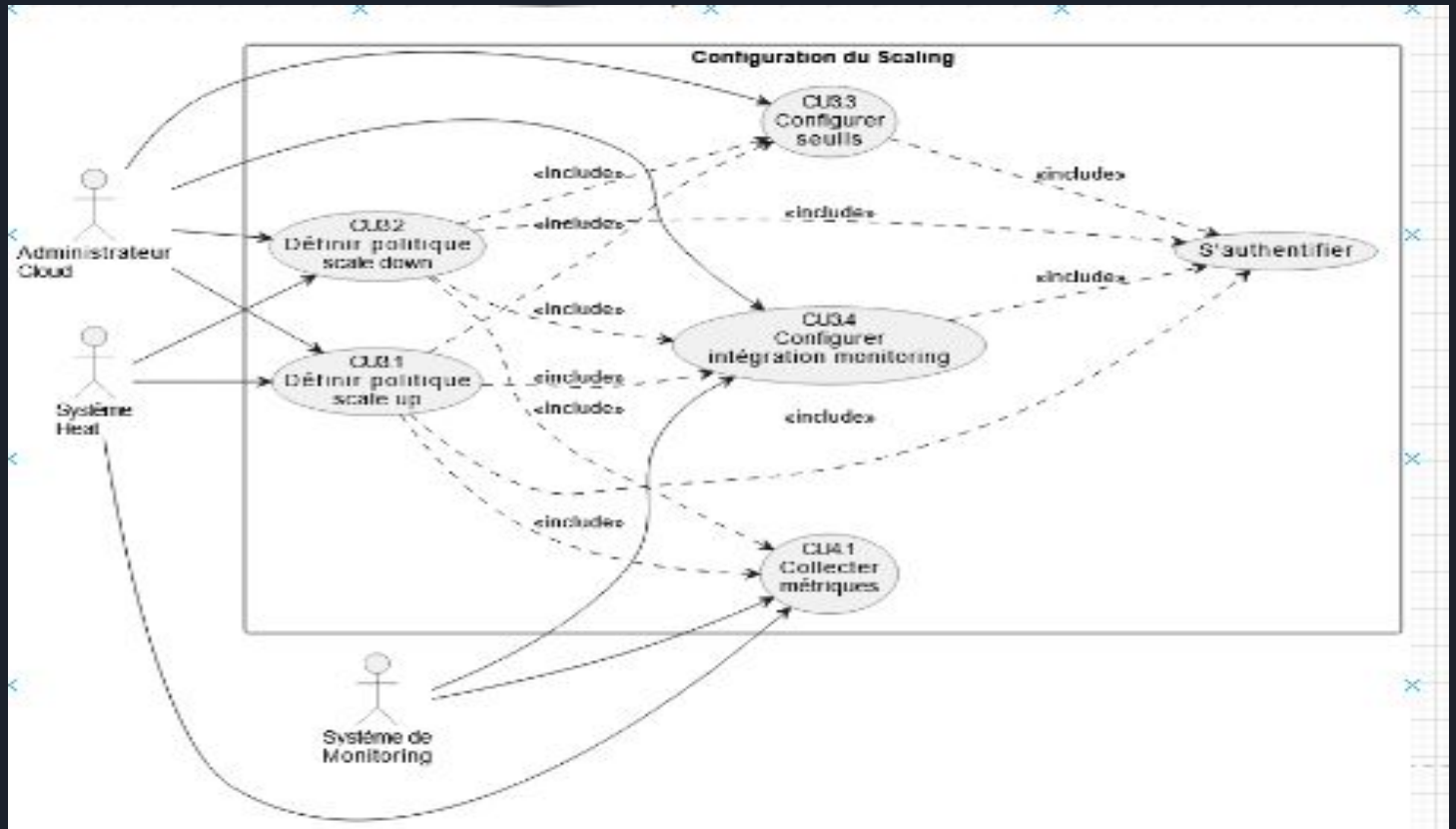
5. Modélisation UML

Diagramme de Cas d'Utilisation : Déploiement Complexe



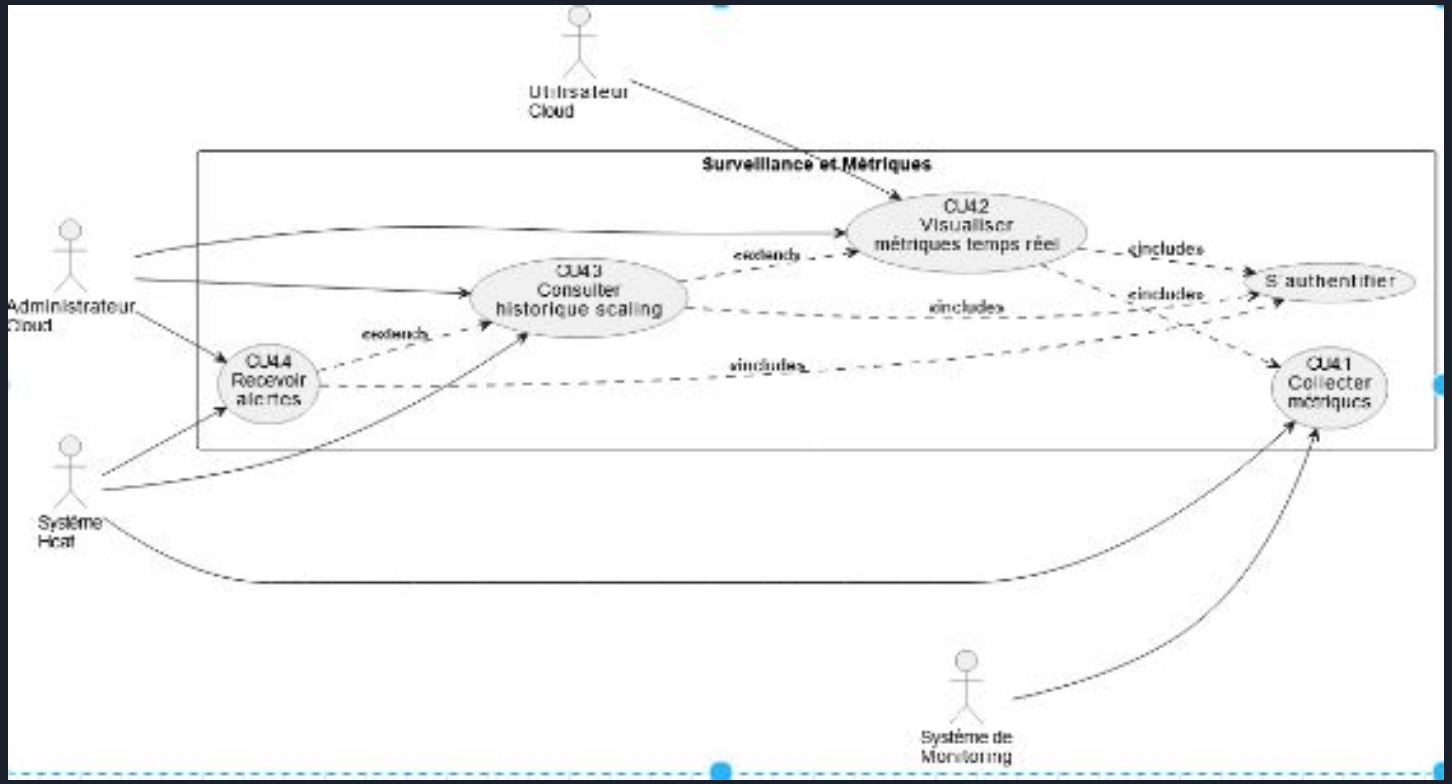
5. Modélisation UML

Diagramme de Cas d'Utilisation : ConfiguRation Scaling



5. Modélisation UML

Diagramme de Cas d'Utilisation : Surveillance Métrique



5. Conclusion

Ce cahier d'analyse présente une spécification complète des besoins utilisateurs pour l'amélioration du service Heat d'OpenStack. Les cas d'utilisation identifiés couvrent deux axes principaux : **Simplification de la modélisation** : création de templates modulaires et réutilisables pour faciliter le déploiement d'architectures complex et **Amélioration du scaling automatique** : développement d'un système de scaling intelligent basé sur des métriques réelles provenant d'outils de monitoring modernes L'ensemble des cas d'utilisation décrits constitue une base solide pour la phase de conception et d'implémentation du projet. Ils répondent aux objectifs fixés dans le cahier des charges en se concentrant sur les besoins réels des utilisateurs (administrateurs et utilisateurs cloud) sans entrer dans les détails techniques d'implémentation. La solution proposée permettra de rendre Heat plus accessible, plus performant et mieux adapté aux besoins des environnements cloud modernes.