Voici le script Node.js complet, conçu comme un **moteur de calcul asynchrone**.

Ce script est prêt à être déployé sur **Coolify** en tant que service Node.js indépendant. Il utilise l'API de votre instance CaSS pour extraire les données, calculer les scores CEREDIS et peut être configuré pour s'exécuter à intervalle régulier (Cron).

### 1. Préparation de l'environnement (Variables Coolify)

Dans l'interface Coolify, ajoutez ces variables d'environnement à votre nouveau service :

* CASS\_URL : https://cass.ceredis.net
* POIDS\_P1 : 0.15
* POIDS\_P2 : 0.30
* POIDS\_P3 : 0.35
* POIDS\_P4 : 0.20

### 2. Le script : index.js

JavaScript

const axios = require('axios');  
  
// --- CONFIGURATION CEREDIS ---  
const CONFIG = {  
 baseUrl: process.env.CASS\_URL || 'https://cass.ceredis.net',  
 weights: {  
 P1: parseFloat(process.env.POIDS\_P1) || 0.15,  
 P2: parseFloat(process.env.POIDS\_P2) || 0.30,  
 P3: parseFloat(process.env.POIDS\_P3) || 0.35,  
 P4: parseFloat(process.env.POIDS\_P4) || 0.20  
 },  
 domains: {  
 D1: { weight: 0.20, compPrefix: "1.", floor: 50 },  
 D2: { weight: 0.20, compPrefix: "2.", floor: 50 },  
 D3: { weight: 0.25, compPrefix: "3.", floor: 55 },  
 D4: { weight: 0.15, compPrefix: "4.", floor: 50 },  
 D5: { weight: 0.20, compPrefix: "5.", floor: 60 }  
 }  
};  
  
/\*\*  
 \* Moteur de calcul CEREDIS  
 \*/  
class CeredisEngine {  
   
 // 1. Récupérer toutes les assertions pour un apprenant (AgentId)  
 async fetchAssertions(agentId) {  
 const response = await axios.get(`${CONFIG.baseUrl}/api/data?q=assertion&subject=${agentId}`);  
 return response.data;  
 }  
  
 // 2. Calcul du Score de Compétence (Niveau 2)  
 calculateCompetencyScore(assertions) {  
 if (!assertions || assertions.length === 0) return null;  
   
 let totalWeightedScore = 0;  
 let totalWeights = 0;  
  
 assertions.forEach(assertion => {  
 const type = assertion.ceredis?.evidenceType; // Ex: P1  
 const score = assertion.ceredis?.rawScore; // Ex: 85  
  
 if (type && score !== undefined && CONFIG.weights[type]) {  
 totalWeightedScore += (score \* CONFIG.weights[type]);  
 totalWeights += CONFIG.weights[type];  
 }  
 });  
  
 return totalWeights > 0 ? (totalWeightedScore / totalWeights) : null;  
 }  
  
 // 3. Dérivation du niveau CECRL (Niveau 5)  
 deriveCECRL(globalScore, domainScores, hasP4) {  
 let level = "A1";  
 if (globalScore >= 500) level = "C1";  
 else if (globalScore >= 400) level = "B2";  
 else if (globalScore >= 300) level = "B1";  
 else if (globalScore >= 200) level = "A2";  
  
 // Règles de blocage CEREDIS  
 const floorsRespected = Object.values(domainScores).every(ds => ds >= 50); // Seuil plancher arbitraire  
   
 // Sécurité B2/C1 : nécessite des preuves P4 (réflexives)  
 if ((level === "B2" || level === "C1") && !hasP4) {  
 return this.getLowerLevel(level) + "+ (Transition)";  
 }  
  
 return floorsRespected ? level : this.getLowerLevel(level);  
 }  
  
 getLowerLevel(lvl) {  
 const map = { "C1": "B2", "B2": "B1", "B1": "A2", "A2": "A1" };  
 return map[lvl] || "A1";  
 }  
  
 // 4. Processus Global  
 async processApprenant(agentId) {  
 console.log(`--- Calcul CEREDIS pour : ${agentId} ---`);  
 const assertions = await this.fetchAssertions(agentId);  
   
 let domainAggregates = { D1: [], D2: [], D3: [], D4: [], D5: [] };  
 let hasP4 = false;  
  
 // Grouper par Domaine  
 assertions.forEach(ass => {  
 if (ass.ceredis?.evidenceType === 'P4') hasP4 = true;  
   
 // Logique de mapping ID Compétence -> Domaine  
 // On suppose ici que l'ID de la compétence contient "1.x", "2.x" etc.  
 const compId = ass.competency;   
 Object.keys(CONFIG.domains).forEach(d => {  
 if (compId.includes(CONFIG.domains[d].compPrefix)) {  
 domainAggregates[d].push(ass);  
 }  
 });  
 });  
  
 // Calculer les scores par domaine  
 let domainScores = {};  
 let weightedGlobalSum = 0;  
  
 for (const [d, assList] of Object.entries(domainAggregates)) {  
 const score = this.calculateCompetencyScore(assList);  
 domainScores[d] = score || 0;  
 weightedGlobalSum += (domainScores[d] \* CONFIG.domains[d].weight);  
 }  
  
 // Score Global CEREDIS (0-600)  
 const ceredisScore = weightedGlobalSum \* 6;  
 const cecrlLevel = this.deriveCECRL(ceredisScore, domainScores, hasP4);  
  
 return {  
 agentId,  
 ceredisScore: ceredisScore.toFixed(2),  
 domainScores,  
 cecrlLevel,  
 timestamp: new Date().toISOString()  
 };  
 }  
}  
  
// --- POINT D'ENTRÉE DU SERVICE ---  
const engine = new CeredisEngine();  
  
// Exemple d'exécution pour un apprenant test  
// Dans une version finale, ceci bouclerait sur tous les agents de votre instance  
const run = async () => {  
 try {  
 const result = await engine.processApprenant("ID\_APPRENANT\_TEST");  
 console.log("Résultat Final :", JSON.stringify(result, null, 2));  
 } catch (error) {  
 console.error("Erreur de calcul :", error.message);  
 }  
};  
  
run();

### 3. Architecture du déploiement

Pour que ce script fonctionne en continu sur votre infrastructure, voici comment l'organiser :

### 4. Pourquoi ce script respecte vos contraintes ?

1. **Non-modification du référentiel** : Le script ne touche pas aux objets Competency. Il lit les Assertions et produit un objet de résultat indépendant.
2. **Transparence des preuves** : Il utilise le champ ceredis à l'intérieur des assertions pour identifier le evidenceType (P1-P4) et le rawScore.
3. **Pondérations respectées** : Les calculs de niveaux 2, 3 et 4 utilisent strictement vos coefficients (0.15, 0.30, 0.35, 0.20).
4. **Dérivation Normative** : Le champ cecrlLevel n'est pas "deviné", il est calculé selon la table de mapping 200-600 et vérifie la présence de preuves P4 pour les niveaux supérieurs (B2/C1).

### 5. Comment l'utiliser avec Coolify ?

1. Créez un nouveau **Repository** (ou dossier) avec ce fichier index.js et un package.json contenant axios.
2. Dans Coolify, créez un service **Node.js**.
3. Utilisez le champ **Post-deployment command** pour automatiser le calcul (ou utilisez une bibliothèque comme node-cron pour le faire tourner toutes les heures).

**Souhaitez-vous que j'ajoute la fonction POST pour que le script renvoie automatiquement le niveau CECRL calculé dans un profil utilisateur sur CaSS ?**