Pause / reprise après quota Claude

Pause / reprise après quota Claude

> *Dernière mise à jour : 2026-05-10*

Le compte Claude utilisé par la VM oracle a un quota Pro/Team (fenêtre 5h glissante). Quand il sature, on veut **mettre en pause les jobs en cours** et **reprendre auto** à l'heure de reset, sans perdre le travail déjà fait.

Détection (`services/claude-quota.ts`)

Le service parse la sortie du CLI Claude pour détecter le message de quota.

Marqueurs supportés

```
QUOTA_MARKERS = [
  'usage limit reached',
  "you've hit",
  'hit your usage limit',
  'hit your session',
  'rate limit reached',
```

```
  // …
]
```

⚠ Le scan se fait sur **toute** la sortie du CLI, **même quand Claude a émis du texte assistant** — certaines versions du CLI renvoient le message de quota comme une réponse assistant normale au lieu d'exiter en erreur.

Patterns d'extraction de l'heure de reset

Par ordre de priorité :

1. `|<unix_seconds>` — timestamp Unix direct (le plus fiable)
2. `reset at HH:MM AM/PM` (12h)
3. `resets at HH:MM AM/PM`
4. `try again at HH:MM AM/PM`
5. `available again at HH:MM AM/PM`
6. `reset at HH:MM` (24h, sans AM/PM)
7. **Fallback** : `+1h` si rien ne match

→ `ClaudeQuotaError` avec champ `resetAt: Date`.

Pattern à respecter partout où Claude est appelé pendant un job long

Émetteurs (`claude-ssh.ts`, `claude-local.ts`)

En fin de stream, si rien n'a été généré, appeler `throwIfQuota(stderr + streamOutput)` **avant** de yield l'erreur générique. Sinon le quota se transforme en "answer empty" silencieux.

Workers parallèles (pool SSH)

Pour `contextual-llm.ts`, `conflict-detect.ts`, etc. :

```
let quotaError: ClaudeQuotaError | null = null;

while (todoIndex < todo.length && !quotaError) {
  const task = todo[todoIndex++];
  try {
    await processTask(task);
  } catch (err) {
    if (err instanceof ClaudeQuotaError) {
      quotaError = err;
      return;  // sans compter l'échec comme un fallback
                // (le chunk n'est pas en cache, sera retenté à la reprise)
    }
    throw err;  // autres erreurs : propagation normale
  }
}

await Promise.all(workers);
saveCache(...);  // ⚠ TOUJOURS flusher AVANT de throw
if (quotaError) throw quotaError;
```

L'ordre `flush THEN throw` est critique : si on throw avant le flush, la reprise repart des derniers ~20 chunks au lieu de pile où on s'est arrêté.

Coordinator (`ingest/coordinator.ts`)

Dans le `catch` global :

```
catch (err) {
  if (err instanceof ClaudeQuotaError) {
    const resetAt = err.resetAt;
    const retryAt = new Date(resetAt.getTime() + 2 * 60 * 1000);  // grace 2 min

    await gamesRepo.update(gameId, {
      ingestStatus: 'scheduled',
      ingestScheduledAt: retryAt.toISOString(),
    });

    scheduleIngestStart(gameId, retryAt);
```

```
    push('quota_pause', { resetAt, retryAt });

    // ⚠️ NE PAS passer le jeu en 'error' — ce n'est pas un échec, juste une pause
    return;
  }
  // ...erreur générique
}
```

Frontend

L'event SSE `quota_pause` bascule le wizard d'ingestion sur le panneau `step === 'scheduled'` avec `reason: 'quota'` (au lieu de `'user'` pour la planification volontaire). La copie est dédiée :

> "L'oracle se repose. Reprise prévue à HH:MM."

L'utilisateur n'a rien à faire — le scheduler relance automatiquement.

Ajouter un nouveau marker

Si Claude change le wording du message de quota et que la détection rate :

1. Vérifier les logs serveur (`/app/data/logs/server.log`) pour voir le wording exact
2. Étendre `QUOTA_MARKERS` dans `claude-quota.ts`
3. Si le format de l'heure change, étendre aussi `detectQuotaResetTime` (les patterns regex)
4. Ajouter un test unitaire dans `claude-quota.test.ts` (ce service a déjà des tests pour les patterns existants)

Pas dans `'error'`, jamais

Le quota n'est pas un échec applicatif : c'est une pause normale prévue. Toute logique downstream (UI, monitoring, alerting) doit traiter `'scheduled' + reason='quota'` ≠ `'error'`.

Revision #1
Created 2026-05-10 15:20:16 UTC by thymon
Updated 2026-05-10 15:20:16 UTC by thymon