FR

Configurer ses premiers agents OpenClaw

JARVIS, HERMÈS et CHRONOS

Série OpenClaw — Billet 5/7
BOTUM — Mars 2026

1. Pourquoi spécialiser ses agents

Un agent IA généraliste, c'est comme demander à un seul employé de gérer l'infrastructure, les emails, le calendrier et la rédaction simultanément. Ça fonctionne… jusqu'à ce que ça ne fonctionne plus.

Un agent généraliste accumule des instructions qui finissent par se contredire. Il jongle entre des contextes hétérogènes dans une fenêtre de contexte qui se sature. Quand la compaction arrive, des informations importantes tombent. Le comportement devient imprévisible.

La spécialisation répond à trois problèmes concrets :

- **Contexte ciblé** — Un agent email ne charge que les règles et l'historique liés à l'email. Son contexte reste léger, pertinent, prévisible.
- **Périmètre défini** — L'agent ne peut pas — par conception — faire des actions hors de son domaine. Le risque d'effet de bord est réduit structurellement.
- **Maintenance indépendante** — On peut modifier les règles de l'agent calendrier sans toucher à l'agent email. Les changements sont isolés et traçables.

Selon une analyse interne BOTUM menée sur 6 mois d'exploitation, un réseau de 5 agents spécialisés traite 3,4× plus de tâches par jour qu'un agent généraliste équivalent, avec un taux d'erreur divisé par 2,7.

2. L'architecture multi-agents

Tous les agents OpenClaw d'un même déploiement partagent un **workspace commun** — un répertoire Git versionné. C'est leur terrain commun : ils y lisent, y écrivent, y communiquent. Mais chaque agent a une identité distincte.

AGENTS.md — la carte d'identité collective

Décrit le réseau complet : qui fait quoi, quels agents existent, leurs responsabilités. Chaque agent le lit en début de session.

SOUL.md — la personnalité partagée

Définit la voix, le ton et les valeurs communes à tous les agents. C'est le contrat de base du réseau.

Fichiers de rôle spécifiques

Chaque agent a son propre répertoire dans agents/. Il y stocke ses règles métier, son historique spécifique, ses fichiers de configuration.

Mémoire partagée vs isolée

MEMORY.md et memory/YYYY-MM-DD.md sont accessibles à tous. Chaque agent maintient aussi une mémoire isolée dans son répertoire.

3. Agent JARVIS — Le gardien de l'infrastructure

JARVIS est l'agent système. Son périmètre : tout ce qui touche à la santé de l'infrastructure, à la gestion du workspace et à la continuité opérationnelle du réseau d'agents.

Responsabilités

- Monitoring santé — vérification périodique des services Docker, processus critiques, espace disque.
- Git commits automatiques — commit régulier des changements pour garantir la traçabilité.
- Mémoire long terme — consolidation de memory/ vers MEMORY.md. Identifie ce qui mérite d'être retenu.
- Compactions de contexte — quand la fenêtre d'un agent approche sa limite, JARVIS déclenche une compaction propre.
- Alertes proactives — notifications messagerie si seuil critique atteint.

Configuration type (agents/jarvis/ROLE.md)

```
# JARVIS — Agent Système ## Périmètre - Infrastructure Docker - Workspace Git (read/write) - Monitoring santé - Compactions de contexte ## Règles - Commit git toutes les 4h si changements - Alerte messagerie si service down > 5 min - Consolider memory/ → MEMORY.md chaque dimanche 23h - Ne jamais modifier les fichiers de rôle des autres agents ## Crons - 08:00 : santé infra complète - 20:00 : commit workspace + log quotidien - Dimanche 23:00 : consolidation mémoire
```

4. Agent HERMÈS — Le pipeline email

HERMÈS est l'agent email. Son périmètre : tout ce qui traverse l'inbox — lecture, classification, digests, réponses standardisées, transferts conditionnels.

Responsabilités

- Lecture inbox IMAP — connexion sécurisée via IMAP/SSL.
- Classification intelligente — client, prospect, partenaire, newsletter, spam, urgence.
- Digests quotidiens — à 8h, résumé structuré : priorités, nb emails, actions requises.
- Règles de filtrage métier — certains emails déclenchent des actions automatiques.
- Réponses standardisées — accusés de réception, confirmations via SMTP.

Configuration type (agents/hermes/ROLE.md)

```
## Connexions - IMAP : récupéré depuis vault (clé: imap-botum) - SMTP : récupéré depuis vault (clé: smtp-botum) ## Classification - Objet contient "URGENT" → priorité P1, notification messagerie - Expéditeur VIP → digest prioritaire - Newsletter/promo → archive automatique - Facture/paiement → notifier LEDGER via queue ## Digest — 08:00 quotidien
```

Règle fondamentale : *HERMÈS classe et délègue — il n'usurpe jamais le périmètre des autres agents. Une facture → tâche pour LEDGER. Une demande de publication → notification à CYRANO.*

5. Agent CHRONOS — L'architecte du temps

CHRONOS est l'agent calendrier. Son périmètre : consultation et mise à jour de Google Calendar, génération de briefings, coordination des rappels entre agents.

Responsabilités

- Briefing matinal (8h15) — calendrier des 48 prochaines heures, réunions, deadlines, conflits.
- Consultation à la demande — disponibilités, créneaux libres, événements à venir.
- Mise à jour calendrier — créer, modifier, supprimer sur instruction explicite.
- Coordination des rappels — informe les autres agents des événements qui les concernent.

Configuration type (agents/chronos/ROLE.md)

```
## Calendrier cible - ID calendrier : récupéré depuis vault (clé: gcal-primary) ## Briefing matinal — 08:15 quotidien - Fenêtre : J+0 à J+2 - Format : réunions, deadlines, conflits signalés ## Coordination inter-agents - Deadline publication → notifier CYRANO (J-1) - Réunion client → notifier NEXUS (J-1) - Facture due → notifier LEDGER (J-2)
```

6. Protocoles de communication inter-agents

Les agents ne communiquent pas en temps réel. Ils communiquent via des **artefacts persistants dans le workspace** — fichiers JSON, files de tâches, états partagés.

La queue de tâches

Le fichier queue/tasks.json est le canal principal. Un agent producteur y dépose une tâche, un agent consommateur la lit à sa prochaine exécution. Chaque tâche a un statut explicite (pending, done, failed, skipped).

Les artefacts partagés

Pour les échanges volumineux, le dossier artifacts/ est utilisé. HERMÈS y dépose un export structuré de l'inbox. CHRONOS y dépose son briefing JSON.

Les triggers

Certains événements déclenchent des actions en cascade. Un trigger dans triggers/ contient les métadonnées nécessaires (slug, langue, deadline) sans nécessiter d'interaction humaine.

Exemple de tâche dans queue/tasks.json

```
{ "id": "task-2026-03-14-001", "from": "hermes", "to": "ledger", "type": "invoice_received", "priority": "normal", "status": "pending" }
```

7. Erreurs à éviter

Périmètres qui se chevauchent

Si deux agents ont tous les deux la responsabilité de "gérer les emails urgents", ils produiront deux réponses pour le même email. **Règle : un seul agent responsable par domaine, sans exception.** Les zones grises doivent être résolues dans AGENTS.md.

Conflits de mémoire

Si HERMÈS et JARVIS écrivent tous les deux dans MEMORY.md sans coordination, on obtient des entrées contradictoires. **Règle : un seul agent propriétaire de la mémoire long terme (JARVIS).** Les autres déposent dans memory/YYYY-MM-DD.md.

Boucles infinies de tâches

HERMÈS dépose une tâche pour CHRONOS, CHRONOS redépose pour HERMÈS qui recrée la même tâche. **Règle : chaque tâche doit avoir un statut final explicite** et les agents vérifient qu'une tâche similaire n'existe pas déjà.

8. Une journée type avec 3 agents actifs

- **07:43** — HERMÈS lit l'inbox nocturne (12 emails). 1 urgence → tâche LEDGER. Archives newsletters.
- **08:00** — JARVIS vérifie la santé infra. Tous services nominaux. Commit workspace Git.
- **08:15** — CHRONOS génère le briefing : 2 réunions, deadline blog demain. Trigger pour CYRANO.
- **09:30** — ARGUS dépose rapport hebdo dans artifacts/. JARVIS notifie via messagerie.
- **12:00** — HERMÈS traite batch matinée (5 emails). 1 réponse standardisée. 1 tâche pour NEXUS.
- **16:00** — CYRANO, alerté par trigger CHRONOS, rédige le billet. Draft dans artifacts/drafts/.
- **20:00** — JARVIS commit final. Consolide memory/2026-03-14.md. Rapport opérationnel quotidien.

Zéro intervention humaine sur ces 12 heures. Chaque agent a opéré dans son périmètre, communiqué via les canaux définis, laissé des traces auditables dans le workspace Git.

Conclusion

La configuration de JARVIS, HERMÈS et CHRONOS représente le socle opérationnel minimum d'un réseau d'agents OpenClaw. Ces trois agents couvrent les domaines fondamentaux de toute organisation : l'infrastructure, la communication, et le temps.

La vraie difficulté n'est pas technique — c'est organisationnelle. Définir des périmètres clairs, établir des protocoles sans ambiguïté, et résister à la tentation d'ajouter des responsabilités "temporaires".

→ Billet 6 : OpenClaw vs ChatGPT vs Claude API — quel outil pour quel usage en enterprise ?

■ Lire la version complète en ligne

Ce billet fait partie d'une série de 7 articles sur le déploiement OpenClaw en entreprise.

blog.botum.ca/openclaw-configurer-premiers-agents-jarvis-hermes-chronos/