MÓDULO 09

Sistema Imunológico *Manter Tudo Funcionando*

Watchdog, feedback loops, monitoramento e resiliência — o que separa hobby de produção

DURAÇÃO	FORMATO	KIT
20 min	Demo + slides	2 arquivos

A Verdade Que Ninguém Conta

> **"Agents are 30% of the work. The other 70% is the immune system."**
> — Eric Siu

Você configurou o agente. Ele tá funcionando. Mas quanto tempo até ele quebrar silenciosamente?

O Problema Real

Falhas Silenciosas

Crons param de executar. Sub-agents travam no limbo. APIs mudam e você descobre dias depois quando alguém reclama.

⚠ Runaway Costs

Um bug faz o agente chamar a API 1000x por hora. Você acorda com uma fatura de $500.

Repetição de Erros

Agente sugere algo ruim, você rejeita. Próxima sessão: mesma sugestão. Sem memória de feedback = sem aprendizado.

A Solução: Sistema Imunológico

Um sistema que monitora, detecta, recupera e aprende automaticamente. 5 camadas:

1. **Watchdog** — monitora crons e auto-retry
2. **Feedback Loops** — aprende com approve/reject
3. **Cost Monitoring** — previne runaway billing
4. **Security Audits** — hardening periódico
5. **Backup & Rollback** — reverter quando der ruim

Este módulo separa "tô brincando" de "tô em produção"

Se você quer um agente confiável rodando 24/7, você PRECISA de imunidade.

1. Watchdog: Monitorar e Auto-Recuperar

Um cron que monitora os outros crons. Se algo falhar, ele tenta consertar sozinho antes de te alertar.

Lógica

1. Listar todos os crons ativos
2. Checar último run de cada um
3. Se algum falhou → retry automático (até 3x)
4. Se falhou 3x → alertar no Telegram

```JSON
// Cron Watchdog — roda 1x/dia
{
  "name": "Watchdog - Monitor de Crons",
  "schedule": {
    "kind": "cron",
    "expr": "0 8 * * *",
    "tz": "America/Sao_Paulo"
  },
  "sessionTarget": "isolated",
  "payload": {
    "kind": "agentTurn",
    "message": "Checar saúde de todos os crons. Listar os que falharam nas últimas
  },
  "delivery": { "mode": "announce" }
}
```

Por Que Funciona

- `isolated` + `agentTurn` = executa de verdade (não só dispara)
- Retry automático antes de alertar = menos ruído
- Roda 1x/dia = pega problemas cedo sem gastar muito

2. Feedback Loops: Aprender com Decisões

O agente precisa lembrar o que você aprovou e rejeitou. Sem isso, ele vai repetir os mesmos erros toda sessão.

Setup

```BASH
# Criar estrutura de feedback
mkdir -p memory/feedback

# Arquivos por domínio
touch memory/feedback/content.json
```

```
touch memory/feedback/tasks.json
touch memory/feedback/recommendations.json
```

Formato de Feedback

```json
{
  "entries": [
    {
      "date": "2026-02-13",
      "context": "Sugeri thread sobre X para LinkedIn",
      "decision": "approve",
      "reason": "Tom certeiro, dados específicos",
      "tags": ["linkedin", "thread", "tom"]
    },
    {
      "date": "2026-02-14",
      "context": "Sugeri usar skill X pra tarefa Y",
      "decision": "reject",
      "reason": "Skill não é confiável, prefiro fazer manualmente",
      "tags": ["skills", "tools"]
    }
  ]
}
```

Regras CRÍTICAS

⚠ **Max 30 entradas por arquivo**

Quando chegar em 30, remove as mais antigas (FIFO). Senão o contexto explode.

Agente DEVE consultar antes de sugerir

Adicione ao AGENTS.md: "Antes de sugerir algo (conteúdo, task, tool), consultar memory/feedback/ pra evitar repetir erros."

Ciclo de Consolidação

Feedback não substitui memória permanente. A cada 30 dias:

1. **Feedback (granular, JSON)** → raw data de decisões
2. **Lessons (curado, prose)** → padrões extraídos
3. **Decisions (permanente)** → políticas consolidadas

Exemplo

5 rejeições de skills → lesson: "Prefere fazer X manualmente" → decision: "Nunca sugerir skill Y pra tarefa X"

3. Monitoramento de Custos

Prevenir runaway billing ANTES de virar problema.

Split de Modelos por Uso

USO	MODELO	CUSTO RELATIVO
Interação direta	Opus	$$$
Crons e automação	Sonnet	$
Heartbeats	Haiku	¢

Regra INVIOLÁVEL

TODOS os crons devem rodar em Sonnet (nunca Opus). Heartbeats em Haiku. Só a interação direta usa Opus.

Rate Limits e Budgets

JSON

```
// Em openclaw.json
{
  "model": {
    "default": "anthropic/claude-opus-4-6",
    "budget": {
      "daily_max_usd": 5.0,
      "alert_threshold_usd": 3.0
    },
    "rate_limit": {
      "max_requests_per_minute": 10
    }
  }
}
```

Isso previne loops infinitos e bugs que drenam a API.

Economia Real

Tudo em Opus

17 crons/dia × $0.10 = $1.70/dia

Heartbeats 4x/dia × $0.10 = $0.40/dia

Interação = ~$0.50/dia

Split otimizado

17 crons/dia × $0.01 = $0.17/dia

Heartbeats 4x/dia × $0.005 = $0.02/dia

Interação = ~$0.50/dia

Total: ~$2.60/dia

Total: ~$0.70/dia

73% de economia só com modelo certo no lugar certo.

4. Security Audit Periódico

Segurança degrada com o tempo. Portas que fecharam podem abrir. Fail2ban pode desativar. SSH pode degradar.

Cron de Auditoria Semanal

```json
{
  "name": "Security Audit - Semanal",
  "schedule": {
    "kind": "cron",
    "expr": "0 6 * * 0",
    "tz": "America/Sao_Paulo"
  },
  "sessionTarget": "isolated",
  "payload": {
    "kind": "agentTurn",
    "message": "Rodar audit de segurança completo. Checar: UFW ativo, Fail2ban roda
  },
  "delivery": { "mode": "announce" }
}
```

Checklist de Auditoria

- □ UFW ativo e configurado corretamente
- □ Fail2ban rodando com regras SSH
- □ Portas expostas = mínimo necessário
- □ SSH: PasswordAuthentication no, PermitRootLogin no
- □ Credenciais no 1Password (NUNCA hardcoded)
- □ Certificados SSL válidos (se aplicável)
- □ Packages atualizados (apt update + upgrade)

Auto-Fix vs Manual

> ⚠ **Nunca auto-fix security sem review**
>
> O agente deve RELATAR issues e SUGERIR correções. Você aprova. Senão, risco de lockout.

5. Backup e Rollback

Antes de qualquer mudança estrutural (config, novos agentes, reorganização), salve um backup + instruções de rollback.

```bash
# Backup antes de mudança
mkdir -p backups/$(date +%Y-%m-%d)
cp /root/.openclaw/openclaw.json backups/$(date +%Y-%m-%d)/
cp -r memory/ backups/$(date +%Y-%m-%d)/
```

ROLLBACK.md

Sempre deixe instruções de como reverter:

```markdown
# ROLLBACK.md

## Backup Date: 2026-02-16

## O Que Mudou:
- Criado agente "content-curator"
- Modificado split de modelos (Opus → Sonnet nos crons)

## Como Reverter:

1. Restaurar config:
   cp backups/2026-02-16/openclaw.json /root/.openclaw/
   openclaw gateway restart

2. Deletar agente criado:
   rm -rf agents/content-curator/

3. Verificar:
   openclaw gateway status
```

Sub-Agents: Nunca "Fire and Forget"

Todo sub-agent precisa de follow-up. Sem isso, ele pode travar no limbo e você nunca sabe.

Regra Obrigatória

1. **Ao spawnar:** informar o que vai fazer
2. **Follow-up em 15-30min:** checar status
3. **Sucesso:** resumir resultado
4. **Falha:** retry imediato → se falhar 2x → avisar
5. **NUNCA** deixar cair no limbo silencioso

> **Caso Real: Sub-Agent no Limbo**
>
> Spawnei um agente pra gerar relatório. Ele falhou silenciosamente. Descobri 3 dias depois quando procurei o resultado. Solução: sempre follow-up.

Ralph Loop vs Feedback Loop

Dois conceitos diferentes:

LOOP	PROPÓSITO
Ralph Loop	Coding loop: escreve código → testa → corrige → repete (1 sessão)
Feedback Loop	Aprendizado entre sessões: você aprova/rejeita → agente lembra pra sempre

✅ Checkpoint do Módulo 9

- [ ] Watchdog de crons configurado e ativo
- [ ] Feedback loops criados (pelo menos 1 domínio)
- [ ] Split de modelos aplicado (Opus/Sonnet/Haiku)
- [ ] Rate limits e budgets configurados
- [ ] Security audit semanal agendado
- [ ] Regra de backup antes de mudanças documentada
- [ ] Sub-agents com follow-up obrigatório no AGENTS.md

📋 Prompt para o Agente

Cole este prompt no chat do seu OpenClaw depois de concluir o módulo:

Acabei de assistir o Módulo 9 sobre o sistema imunológico. "Agents are 30% of the work. The other 70% is the immune system." Me guie. **O que preciso que você faça:** 1. **Watchdog de crons** — Configure um cron que monitora se os outros crons estão executando. Se algum falhar, retry automático até 3x. Se falhar 3x, me avisa. 2. **Feedback Loops** — Me ajude a configurar um sistema de approve/reject. Quando você me sugerir algo e eu rejeitar, anote o motivo. Consulte essas anotações antes de sugerir novamente. Me mostre como isso funciona na prática. 3. **Monitoramento de custos** — Configure o split de modelos (Haiku pra heartbeats, Sonnet pra crons, Opus pra interação). Configure rate limits e budgets pra prevenir runaway. Me mostre quanto estou gastando por dia/semana. Meta: de ~$2-3/dia pra ~$0.10/dia com otimizações. 4. **Audit de segurança periódico** — Configure um cron semanal de security audit. Rode um agora pra eu ver o resultado. 5. **Backup antes de mudanças** — Crie uma regra: antes de qualquer mudança estrutural, salvar backup + ROLLBACK.md. Me mostre como reverter se algo der errado. **Regras:** - Esse módulo é o que separa "tô brincando" de "tô em produção" - Me explique cada proteção e qual problema ela previne - No final, rode um health check completo e me dê o score Vamos construir a imunidade?

PRÓXIMO MÓDULO

Módulo 10 — Mission Control: Seu Painel Operacional