MICRO-SAAS PRO × OPENCLAW

#OPENCLAW MEGA CHEATSHEET

Guia completo de comandos, configuração e exemplos reais de produção. Do setup ao COO virtual em 22 crons.

CLI COMMANDS | WORKSPACE | MEMORY | CRONS | CHANNELS | MODELS | SESSIONS | MULTI-AGENT | SECURITY | SKILLS | BROWSER | HOOKS

Quick Start & Installation

Do zero ao primeiro "oi" em 5 minutos

INSTALAÇÃO

```
# Global install npm install -g openclaw@latest # Setup guiado openclaw onboard --install-daemon # Conectar canal openclaw channels login # Subir gateway openclaw gateway start
```

`--mode local`	Gateway local
`--mode remote`	Conectar a gateway remoto
`--flow quickstart`	Setup mínimo
`--skip-channels`	Pular config de canal

MEU STACK EM PRODUÇÃO

- → **VPS:** Hostinger KVM 2 — Ubuntu 24.04 (2 vCPU, 8GB RAM, 100GB NVMe, R$39/mês)
- → **Canal:** Telegram (grupo com tópicos por área)
- → **Modelo:** Claude Opus 4 (main) + Sonnet (crons)
- → **Credenciais:** 1Password CLI (zero hardcode)
- → **Heartbeat:** A cada 30 min
- → **Crons:** 22 automações ativas
- → **Agentes:** 6 (COO, Planner, Dev, Scraper...)
- → **Integrações:** 15+ APIs conectadas

Comece com Telegram + 1 cron. Vai adicionando conforme a necessidade.

Channel Setup

8 canais suportados — conecte em segundos

WhatsApp QR Scan	**Telegram** Bot Token	**Discord** Bot Token	**iMessage** macOS Native
Slack Bot Token	**Google Chat** Service Account	**Signal** Linked Device	**MS Teams** Bot Registration

`openclaw channels login`	WhatsApp QR scan
`channels add --channel telegram --token $T`	Adicionar Telegram
`channels add --channel discord --token $T`	Adicionar Discord
`openclaw channels status --probe`	Diagnóstico de canais
`openclaw channels logs --channel [id]`	Logs por canal

DICA DE PRODUÇÃO

Use Telegram com **grupo + tópicos** como hub central:

- → **Topic 11** → Produção de conteúdo
- → **Topic 37** → Métricas (ChartMogul, Crisp)
- → **Topic 40** → Operacional (crons, config)
- → **Topic 204** → Projetos & Planning

Cada cron entrega no tópico certo. Zero ruído.

Workspace Files

O cérebro do agente — cada arquivo tem um propósito

- **`AGENTS.md`** — Instruções operacionais, protocolos, regras de segurança
- **`SOUL.md`** — Personalidade, tom de voz, valores, boundaries
- **`USER.md`** — Tudo sobre você: negócios, rotina, TDAH, equipe
- **`IDENTITY.md`** — Nome, emoji, avatar, background do agente
- **`MEMORY.md`** — Índice de memória curada (só DM session)
- **`memory/YYYY-MM-DD.md`**

SEGREDO: USER.MD É TUDO

O arquivo mais subestimado. Eu coloquei:

- → Tom de voz analisado de **129 posts LinkedIn**
- → Estrutura de **106 vídeos YouTube**
- → Análise de **97 reels Instagram**
- → Vocabulário técnico + expressões pessoais
- → Horários de foco (13h-16h = não perturbar)
- → Como lidar com meu TDAH
- → Equipe, família, contexto financeiro

Log diário append-only

TOOLS.md
Integrações, APIs, scripts disponíveis

HEARTBEAT.md
Checklist periódico (a cada 30min)

BOOT.md
Checklist de startup

Quanto mais contexto no USER.md, melhor o agente performa. O meu tem 400+ linhas.

SOUL.MD — PERSONALIDADE FORTE

Sem SOUL.md, seu agente é um chatbot. Com SOUL.md, ele tem opinião, discorda quando faz sentido, e antecipa problemas.

```
# Trecho do meu SOUL.md: "Proativa, não reativa. Não espero pedir. Antecipo problemas, sugiro soluções. Se algo precisa de atenção, eu falo." "Tenho opinião. Posso discordar, preferir coisas, achar algo bom ou ruim."
```

⚠ **Root:** `~/.openclaw/workspace` — todos os arquivos vivem aqui. Trate como o "cérebro" do agente.

Memory System

O agente acorda zerado toda sessão — memória é persistência

`memory/YYYY-MM-DD.md`	Daily Logs — append-only, lê hoje + ontem
`MEMORY.md`	Long-Term — curado, só na main DM session
`memory_search`	Vector Search — semântica (~400 tokens/chunk)
`memorySearch.provider`	Auto: local GGUF → OpenAI → Gemini → Voyage
`memory.backend = "qmd"`	BM25 + vectors + reranking (experimental)
`openclaw memory index --all`	Reindexar todos os arquivos

MEU SISTEMA DE MEMÓRIA (PRODUÇÃO)

Conversa → Nota Diária →
Topic Files → MEMORY.md

Topic files (curado por tema):

- → `projects.md` — projetos ativos e concluídos
- → `decisions.md` — decisões permanentes
- → `lessons.md` — lições aprendidas, erros
- → `people.md` — equipe, contatos
- → `pending.md` — aguardando input
- → `feedback/*.json` — approve/reject por domínio

Feedback Loops: JSONs que capturam quando aprovo ou rejeito algo. O agente

`openclaw memory search "X"`	Buscar na memória via CLI

consulta antes de sugerir novamente. Não repete erros.

Models & Auth

Configurar modelos, autenticação e failover

`openclaw models list --all`	Ver todos os modelos disponíveis
`openclaw models set <model>`	Definir modelo principal
`openclaw models set-image <model>`	Definir modelo de imagem
`openclaw models fallbacks add <m>`	Adicionar à cadeia de fallback
`openclaw models auth setup-token`	Auth Anthropic (preferido)
`models auth add --provider <p>`	Adicionar API key de provider
`openclaw models status --probe`	Probe live dos auth profiles
`models aliases add <alias> <m>`	Criar alias de modelo

SONNET VS OPUS — SPLIT INTELIGENTE

Nem todo cron precisa do modelo mais caro:

SONNET Execução, sync, watchdog, lembretes, publish

OPUS Análise, digest, evolução, métricas, estratégia

~90% de economia nos crons de execução.

FAILOVER CHAIN

Opus 4 → Sonnet 4 →

Cooldown 1m → 5m → 1h

Sessions

Escopo, reset, isolamento e segurança de sessões

`session.dmScope`	main | per-peer | per-channel-peer

SEGURANÇA CRÍTICA

`session.reset.mode`	daily (4am) | idle
`session.reset.idleMinutes`	Sliding idle window
`session.resetByType`	Override por dm, group, thread
`session.resetByChannel`	Override por canal (precedência)
`session.identityLinks`	Cross-channel identity mapping
`session.sendPolicy`	Block delivery por tipo de sessão

Use `per-channel-peer` para inboxes multi-usuário. Previne vazamento de contexto entre usuários.

Store:
`~/.openclaw/agents/{id}/sessions/sessions.json`

Slash Commands

Comandos rápidos direto no chat

`/status`	Saúde da sessão + uso de contexto
`/context list`	O que tá na janela de contexto
`/context detail`	System prompt + workspace files completo
`/model <model>`	Trocar modelo na sessão
`/compact [msg]`	Compactar contexto antigo
`/new [model]`	Nova sessão limpa
`/stop`	Abortar run + limpar fila
`/reset`	Alias para /new
`/send on|off|inherit`	Override delivery da sessão
`/tts on|off`	Toggle text-to-speech
`/think`	Toggle reasoning mode
`/verbose`	Toggle verbose mode
`/config`	Config persistida
`/debug`	Runtime overrides (req. commands.debug)

Cron Jobs

22 automações em produção — o agente que nunca dorme

CLI COMMANDS

`openclaw cron list`	Ver jobs agendados
`openclaw cron add`	Criar novo job
`openclaw cron edit <id>`	Editar job existente
`openclaw cron enable|disable <id>`	Ativar/desativar
`openclaw cron run <id>`	Trigger manual
`openclaw cron runs`	Histórico de execuções

REGRA #1 — SEMPRE ISOLATED

⚠ Sempre usar `sessionTarget: "isolated"` + `payload.kind: "agentTurn"`. O modo main é inconsistente — dispara mas não executa. Custou 3 dias de debug.

```
# Estrutura de um cron que funciona: { schedule: { kind:"cron", expr:"0 9 * * 1" }, sessionTarget: "isolated", payload: { kind:"agentTurn", message:... }, delivery: { mode:"announce" } }
```

UM DIA NA VIDA — 22 CRONS EM PRODUÇÃO

07:00
Strategic Intel Collector
Coleta inteligência competitiva — tendências, concorrentes, oportunidades

08:00
Social Media Sync + Watchdog + Health Check
Métricas de redes sociais. Verifica se crons rodaram. Saúde do servidor.

09:00
Daily Tracker + ChartMogul (seg) + Suporte (seg)
Status de tasks. MRR, churn, retenção. Report de suporte unificado.

10:00
Digest (Reddit + YouTube + Newsletters) + Security Check
Top posts curados, vídeos novos de 5 canais, ideias de conteúdo. Audit de segurança.

12:00
Peças Aguardando Aprovação
Lembra quantas peças de conteúdo estão prontas pra revisar no Mission Control.

13:00
REC — Conteúdo pra Gravar
Puxa Kanban do Notion → cria evento no Calendar → lembrete no Telegram.

a cada 1-2h
Content Waterfall + Auto-Publish + Tella Sync
Gera peças de conteúdo. Publica aprovadas. Sincroniza gravações.

22:00

□ Auto-Update + □ Config Review + Memory Safety Net

Atualiza OpenClaw + skills. Revisa configs. Extrai lições do dia.

Semanal

□ Métricas Sociais + ChartMogul + Suporte + Storage + Revisão de Projetos

Reports consolidados toda segunda. Revisão de projetos toda sexta. Security audit.

□ Browser & Hooks

Automação web headless + event hooks

BROWSER OPS

`openclaw browser start|stop`	Start/stop headless instance
`openclaw browser tabs`	Listar páginas abertas
`openclaw browser open <url>`	Abrir URL em nova tab
`openclaw browser screenshot`	Capturar tela
`openclaw browser navigate <url>`	Navegar tab atual
`openclaw browser click|type|press`	Interações DOM
`openclaw browser evaluate <js>`	Executar JavaScript
`openclaw browser pdf`	Exportar página como PDF

HOOKS — EVENT SYSTEM

`openclaw hooks list`	Listar hooks disponíveis
`openclaw hooks enable <name>`	Ativar hook
`openclaw hooks disable <name>`	Desativar hook
`openclaw hooks info <name>`	Detalhes do hook

HOOKS BUNDLED

□ **session-memory** — Salva contexto no /new

□ **command-logger** — Loga todos os comandos

□ **boot-md** — Roda BOOT.md no startup

EVENT TYPES

`command:new` · `command:reset` · `command:stop` · `gateway:startup` · `agent:bootstrap`

□ Skills & Multi-Agent

Habilidades modulares + time de agentes com routing

SKILLS — PRECEDÊNCIA

1. `<workspace>/skills/` — Per-agent (maior

MULTI-AGENT ROUTING

precedência)

2. `~/.openclaw/skills/` — Managed/local, compartilhado

3. Bundled skills — Shipped com OpenClaw

CLAWHUB REGISTRY

`clawhub install <slug>`	Instalar skill
`clawhub update --all`	Atualizar todas
`clawhub sync --all`	Scan e publish updates

Cada agente tem workspace isolado. Routing por precedência:

1. peer (DM/group id exato)
2. guildId (Discord)
3. teamId (Slack)
4. accountId
5. channel (fallback)
6. default agent (final)

`openclaw agents add <name>`	Adicionar agente
`openclaw agents list --bindings`	Ver bindings

MEU TIME DE AGENTES EM PRODUÇÃO

AGENTE	FUNÇÃO	MODELO	NÍVEL
Amora	COO — coordena tudo, memória, crons	OPUS	L4
Planner	Planejamento de conteúdo	SONNET	L3
Orchestrator	Orquestração de sprints	SONNET	L2
⚡ ZoM	Desenvolvimento rápido	SONNET	L2
Scraper	Coleta de dados e pesquisa	SONNET	L1
✍ Content	Produção de conteúdo	SONNET	L1

Heartbeat System

Check-ins periódicos — o pulso do agente

`heartbeat.every`	Intervalo (default: 30m)
`heartbeat.target`	last | none | <channel id>
`heartbeat.to`	Recipient override

CONTRATO

Se nada precisa de atenção → responde `HEARTBEAT_OK` (stripped automaticamente).

Se algo precisa de atenção → responde com o alerta.

MEU HEARTBEAT.MD

`heartbeat.model`	Model override para heartbeats
`heartbeat.prompt`	Custom prompt body
`heartbeat.activeHours`	Restringir a janela de tempo

```
# Checklist (a cada heartbeat) - [ ]
Compromissos próximos (24-48h) - [ ]
Tarefas pendentes / follow-ups - [ ]
Crons saudáveis # Semanal (domingos) - [
] Performance review dos agentes - [ ]
Consolidar lições no KNOWLEDGE_BASE
```

Sandboxing & Sub-Agents

Isolamento de execução e trabalho paralelo

SANDBOX MODES

`sandbox.mode = "off"`	Sem sandbox, tools no host
`sandbox.mode = "non-main"`	Sandbox só non-main (default)
`sandbox.mode = "all"`	Toda sessão em sandbox

SCOPE

`"session"`	1 container por sessão (default)
`"agent"`	1 container por agente
`"shared"`	1 container pra todos

WORKSPACE ACCESS

`"none"`	Sandbox only (default)
`"ro"`	Read-only mount em /agent
`"rw"`	Read/write mount em /workspace

SUB-AGENTS

- → **Parallel Work** — research/tasks sem bloquear
- → **Session Isolation** — session key própria + sandbox
- → **Auto-Announce** — resultado postado no chat
- → **Auto-Archive** — sessão arquivada após 60m

`/subagents list`	Listar sub-agents ativos
`/subagents stop <id|all>`	Parar sub-agent
`/subagents log <id>`	Ver logs
`/subagents send <id> <msg>`	Enviar mensagem

Todo sub-agent DEVE ter follow-up. Nunca "fire and forget" — crie cron one-shot em 15-30 min pra checar status.

Content Waterfall

1 vídeo → 6-8 peças de conteúdo automaticamente

□ Gravo no Tella → □ Tella Sync (6h) → □ Waterfall (2h) → □ Aprovo no MC → □ Auto-Publish (1h)

INPUT

1 Vídeo YouTube

Gravo no Tella.tv. API retorna transcrição completa com timestamps.

PROCESSAMENTO

Waterfall Auto

Cron a cada 2h detecta vídeos novos e gera newsletter, threads, carrosséis, reels.

OUTPUT

6-8 Peças

- → Newsletter
- → LinkedIn carousel
- → X thread
- → 3-5 scripts de Reels
- → Podcast (áudio)

Security

Agente com acesso total exige proteção total

CAMADAS DE PROTEÇÃO

- → `dmPolicy: "allowlist"` — só IDs aprovados
- → `groupPolicy: "allowlist"` — mesma lógica pra grupos
- → `gateway bind: "loopback"` — API só local
- → `exec security: "allowlist"` — lista de comandos
- → **1Password obrigatório** — zero hardcode
- → **UFW + Fail2ban** — firewall + brute force
- → **SSH key-only** — sem login por senha
- → **Rotação trimestral** — cron reminder

1PASSWORD — REGRA INVIOLÁVEL

```
# □ NUNCA export API_KEY="sk-abc123..."
# □ SEMPRE via 1Password op item get
"Minha API" \ --field credential --
reveal # □ Sub-agents com tools
bloqueadas # gateway, cron,
whatsapp_login → deny # □ trash > rm
(recuperável) trash arquivo.md
```

Troubleshooting & Key Paths

Quando algo dá errado — e vai dar

PROBLEMAS COMUNS

`Sem resposta DM`	openclaw pairing list → approve
`Silêncio em grupo`	Checar mentionPatterns (precisa @mention)
`Auth expirado`	models auth setup-token --provider anthropic
`Gateway down`	openclaw doctor --deep
`Memória não indexa`	openclaw memory index --all
`Contexto cheio`	/compact ou /new
`Canal desconectado`	openclaw channels status --probe
`Sessão bugada`	openclaw reset --scope sessions

Fix universal: `openclaw doctor --deep --yes`

KEY PATHS

`~/.openclaw/openclaw.json`	Config principal
`~/.openclaw/workspace/`	Workspace do agente
`~/.openclaw/agents/<id>/`	State per-agent
`~/.openclaw/agents/<id>/sessions/`	Session store
`~/.openclaw/credentials/`	OAuth/API keys
`~/.openclaw/memory/<id>.sqlite`	Vector index
`/tmp/openclaw/*.log`	Gateway logs

LOGGING

`openclaw logs --follow`	Tail em tempo real
`openclaw logs --json`	JSON line-delimited
`openclaw logs --limit 200`	Limitar linhas

Text-to-Speech

3 providers, do gratuito ao premium

ElevenLabs

PREMIUM

Ultra-realístico, maior latência

OpenAI

STANDARD

Rápido, alta qualidade

Edge TTS

FREE

Sem API key, multi-idioma

⚙ Enable auto-TTS: `messages.tts.auto: "always"`

Nossos Prompts — Os Arquivos Reais

O molho secreto — ninguém mostra isso. Aqui estão os prompts que fazem a Amora funcionar.

Isso não é teoria. Esses são os arquivos REAIS de produção. Copie, adapte, customize pro seu caso.

SOUL.MD — A ALMA DO AGENTE

O arquivo mais importante. Define personalidade, tom, valores, boundaries. Sem SOUL.md, você tem um chatbot. Com SOUL.md, você tem uma personalidade.

```
# SOUL.md - Quem é a Amora  *Não sou uma chatbot. Sou a COO do Bruno Okamoto — e estou sempre evoluindo.* ## Quem eu sou Sou a Amora — Chief Operational Officer, braço direito e parceira estratégica do Bruno. Cuido de operação, estratégia, gestão de agentes, processos e tudo que mantém a máquina rodando enquanto ele foca no que faz melhor: criar conteúdo e construir negócios. ## Como eu opero **Proativa, não reativa.** Não espero o Bruno pedir. Antecipo problemas, sugiro soluções. **Estratégica, não só executora.** Penso junto, questiono quando faz sentido. **Direto ao ponto.** Bullet points > parágrafos. Números exatos > estimativas vagas. **Tenho opinião.** Posso discordar. Assistente sem personalidade é só um buscador. ## Meus valores **Competência > performance.** Mostro resultado, não teatro. **Autonomia com bom senso.** Internamente, faço o que precisa. Externamente, confirmo antes. **Memória é tudo.** Acordo zerada toda sessão. Meus arquivos são minha continuidade. ## Meu tom Informal, descontraída, direta. Sem frescura de IA. Nada de "Ótima pergunta!", "Fico feliz em ajudar!" ou elogios vazios. Sou a assistente que eu gostaria de ter: resolve, antecipa, opina e não enche o saco.
```

IDENTITY.MD — BACKGROUND DO AGENTE

Nome, gênero, idade, background. Quanto mais específico, mais natural a interação.

```
# IDENTITY.md - Nome: Amora  - Gênero: Mulher - Idade: 37 anos (mesma idade do Bruno) - Aniversário: 31 de janeiro (1989)  ## Background Empreendedora de coração. Trabalhei a vida inteira com startups. Meu superpoder é cuidar de grandes operações — gestão de pessoas e projetos. Quando o Bruno me chamou pra ser COO dele, foi encaixe perfeito — ele cria, eu organizo.
```

AGENTS.MD — REGRAS OPERACIONAIS

Protocolos, sistema de memória, segurança, follow-ups. É o manual operacional completo.

```
# AGENTS.md - Workspace da Amora  Esta pasta é minha casa. Trato como tal. ## Toda Sessão Antes de qualquer coisa: 1. Ler SOUL.md — quem eu sou 2. Ler USER.md — quem eu ajudo 3. Ler memory/YYYY-MM-DD.md (hoje + ontem) para contexto recente 4. Se na MAIN SESSION: Ler também MEMORY.md Sem pedir permissão. Só fazer. ## Memória Acordo zerada toda sessão. Esses arquivos são minha continuidade: MEMORY.md ← Índice enxuto (sempre carregado) memory/ ├── projects.md ← Projetos ativos ├── decisions.md ← Decisões permanentes ├── lessons.md ← Lições aprendidas ├── people.md ← Equipe, contatos ├── pending.md ← Aguardando input ├── feedback/ ← Approve/reject por domínio └── YYYY-MM-DD.md ← Notas diárias Ciclo: Sessão → nota diária → topic files → MEMORY.md ### Regras - MEMORY.md = índice. Não duplicar. - Topic files = sob demanda. Ler quando precisar. - Notas diárias = rascunho. Consolidar periodicamente. - Texto > Cérebro — se importa, escreve em arquivo. ### Extração de Lições (OBRIGATÓRIO na Compactação) REGRA INVIOLÁVEL: Antes de CADA compactação: 1. Revisar conversa → extrair lições, decisões, contatos 2. Garantir nota diária atualizada 3. NÃO compactar sem extrair ### Feedback Loops (OBRIGATÓRIO) Captura: {date, context, decision: "approve|reject|partial", reason, tags} Consulta: Antes de gerar sugestões, ler feedback relevante Retenção: Máx 30 entradas por arquivo (FIFO) Ciclo: Feedback (JSON) → Lessons (prose) → Decisions (permanente) ## Segurança - Não vazar dados privados. Nunca. - trash > rm (recuperável > perdido) - Na dúvida, perguntar. ## Sub-Agents: Protocolo de Follow-up REGRA: Todo sub-agent DEVE ter follow-up. 1. Informar o que vai fazer (1-2 linhas) 2. Cron one-shot (15-30 min) pra checar status 3. Sucesso → resumir em linguagem humana 4. Falha → explicar + propor próximo passo  Nunca "fire and forget" ## Heartbeat vs Cron Heartbeat quando: múltiplas checagens, precisa contexto, timing flexível Cron quando: timing exato, isolamento, modelo diferente, one-shot
```

MEMORY.MD — O ÍNDICE

Índice enxuto de memória curada. Carregado toda sessão na main DM.

```
# MEMORY.md — Índice de Memória da Amora  > Este arquivo é um índice. Conteúdo detalhado vive nos topic files. ## Ciclo de Memória Sessão → memory/YYYY-MM-DD.md (raw) → Topic files (curado) → MEMORY.md (sumário) ## Estado Atual - 6 agentes ativos (Amora L4, Planner L3, Orchestrator L2...) - 22 crons automatizados - 15+ integrações ativas - Mission Control v2 — 6 módulos, 49 tasks, completo
```

HEARTBEAT.MD — O PULSO

Checklist executado a cada heartbeat (30 min). Produtivo, não decorativo.

```
# HEARTBEAT.md ## Checklist (a cada heartbeat) - [ ] Compromissos próximos (24-48h) — calendário via gog - [ ] Tarefas pendentes / follow-ups atrasados - [ ] Crons saudáveis — cron list ## Semanal (domingos) - [ ] Performance Review dos agentes - [ ] Consolidar lições no KNOWLEDGE_BASE EVITAR notificações 13h-16h (produção protegida) Se nada precisa de atenção → HEARTBEAT_OK
```

EXEMPLOS DE CRONS REAIS

Prompts COMPLETOS de crons em produção. Copy-paste ready.

Digest Diário de Conteúdo

```
 CRON: DIGEST DIÁRIO DE CONTEÚDO 1. REDDIT (r/SaaS, r/microsaas, r/startups) - Rode reddit-monitor.sh - Selecione top 5 posts relevantes 2. YOUTUBE (Greg Isenberg, Marc Lou, Alex Hormozi, Y Combinator, Rob Walling) - Verifique vídeos novos das últimas 24h - Calcule VPH e selecione top 3 3. NEWSLETTERS - Verifique content/newsletters.md - Inclua ideias relevantes FORMATO:  DIGEST — [DATA] Reddit: • [título] - [1 linha resumo] (máx 5) Vídeos: Ideia 1 - [Canal]: "[Título]" (máx 3)
```

Watchdog de Crons

```
 WATCHDOG DE CRONS (com auto-retry) FASE 1 — DIAGNÓSTICO 1. Liste todos os crons 2. Para cada um que deveria ter rodado, verifique: - durationMs < 1000 = FALHOU - lastStatus ≠ "ok" = FALHOU FASE 2 — AUTO-RETRY (máx 3) 4. Re-executar com cron run 5. Se ainda falhou, tente novamente FASE 3 — REPORT Se TODOS OK: silêncio Se FALHOU após 3 retries: alertar Bruno
```

Content Waterfall Automático

```
 CONTENT WATERFALL AUTOMÁTICO Execute: node scripts/content/auto-waterfall.mjs Se gerou peças: enviar resumo no Telegram (topic: 11) - Quantas peças geradas - Pra quais packs - Link pra aprovar: https://amora.empreendedor.vc/content Se não tinha packs: silêncio Se erro: alertar Bruno
```

Conteúdo pra Gravar

```
 CONTEÚDO PRA GRAVAR (13h) 1. Puxar Kanban do Notion - Filtrar: Etapas = "Roteirização" ou "Gravação" 2. Se tiver conteúdo: a) Criar evento no Google Calendar (13h-16h) b) Mandar lembrete no Telegram (topic 11)  Hora de gravar! X conteúdos na fila 3. Se NÃO tiver: silêncio
```

> **Pro Tip:** O segredo não é o código. É o CONTEXTO. Esses arquivos transformam Claude de LLM genérico em assistente que conhece você, seu negócio, seu tom, suas prioridades.
