



DOJO DE AGENTES

Agentes de IA & Fluxos







github.com/SabinoWS/dojoAgentes



Antigravity • Agents • Rules • Skills •
Workflows • SDD • MCP • Canonical Cycle



Agenda

1. **Introdução** aos Agentes de IA
 - O que são e como funcionam?
2. **Setup do Agente** 
 - Rules, Skills e Workflows
3. **Metodologias** 
 - Memória, SDD e MCP
4. **Canonical Cycle** 
5. **Debate e Próximos Passos** 



1. Agentes de IA

Documentação Oficial: antigravity.google/docs/skills

O que são?

Sistemas de software autônomos capazes de perceber seu ambiente, raciocinar sobre ele e tomar ações para atingir objetivos específicos. Diferente de um "chatbot" passivo, um agente tem **autonomia** e **capacidade de execução**.



Para que servem?

Para automatizar tarefas complexas que exigem tomada de decisão, uso de ferramentas e múltiplos passos, atuando como um "parceiro de trabalho".

“ Exemplo Simples:

Um agente que monitora seu repositório git, detecta PRs abertos, roda os testes e, se falhar, comenta sugerindo a correção. O chat do antigravity pode ser usado para interagir com o agente.

”



Configurando agent



2. Por que configurar o Agente?

1. Eficiência de Contexto

Em vez de repetir *"eu uso Arquitetura Hexagonal e testes em Jest"* todo dia, isso vira uma **Rule/Skill/Workflow** permanente.

2. Memória de Longo Prazo

O agente "lembrará" como operar seu sistema e ferramentas via **Skills**, sem você precisar colar documentação técnica a cada prompt.

3. Comandos Naturais

Transforme prompts gigantes e detalhados em pedidos simples do dia a dia.

Antes (Verboso)

"Por favor, faça o deploy, lembrando de rodar testes, verificar a tag, confirmar no chat, um commit no padrão tal e depois push e deploy no Jenkins no job X ... +++"

Depois (Natural)

"Faça o deploy."

A Estrutura `.agent`

A "bancada" do seu Agente é organizada em pastas específicas para garantir modularidade:

```
.agent/  
├── rules/      # 🧠 Contexto e Diretrizes (O que saber?)  
├── skills/     # 🛠️ Ferramentas (Como fazer?)  
└── workflows/ # 📋 Processos (O que fazer passo-a-passo?)
```

Se não está aqui, o agente **não sabe**. (a menos que pedido explicitamente)



3. Rules (Regras)

O que são?

Diretrizes explícitas e imutáveis que governam o comportamento do agente. São o "contexto sistêmico" que garante segurança, conformidade e estilo.

Para que servem?

Para evitar que o agente tome ações indesejadas (ex: apagar banco de produção) e para forçar padrões de qualidade (ex: sempre escrever testes).



“ **Exemplo Simples:**

Regra: "Nunca altere arquivos de configuração (.env, docker-compose) sem pedir confirmação explícita ao usuário."

”

4. Skills (Habilidades)

O que são?

Pacotes de conhecimento operacional. Uma Skill ensina ao agente **como** usar uma ferramenta específica, uma API ou executar um procedimento técnico complexo.

Para que servem?

Para expandir o "canivete suíço" do agente. Em vez de apenas gerar texto, ele se especializa em algo e também pode aprender a interagir com o mundo real (Jira, AWS, Kubernetes).



“ **Exemplo Simples:**

Skill: `git-ops`

Ensina o agente a fazer `git checkout -b`, `git add`, `git commit` seguindo o padrão Conventional Commits do time. ”

5. Workflows (Fluxos)

O que são?

Sequências orquestradas de tarefas. Um Workflow é um "roteiro" passo-a-passo que o agente segue para completar um objetivo maior, garantindo consistência no processo.

Para que servem?

Para padronizar processos repetitivos e propensos a erro humano, como deploys, onboardings ou migrações de banco de dados.



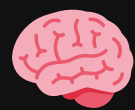
“ **Exemplo Simples:**

Workflow: `deploy-production`

1. Acessar job no Jenkins.
2. Identificar a tag de release (user input).
3. Preencher parâmetros de build.
4. Submeter deploy e notificar Discord.

”

Conceitos e Metodologias



6. Memória por Arquivo (vs Chat)

A Filosofia "Manus":

O chat é efêmero, volátil e não confiável para guardar decisões.

- **Chat**: Apenas para pedir, ajustar e comandar.
- **Arquivo (.md)**: A única fonte de verdade e persistência.

Se você não escreveu no arquivo, para a IA, **não existe**.

Mantenha a documentação viva e o contexto salvo.

7. Spec Driven Development (SDD)

O problema do Chat:

Conversar "de boca" com a IA gera loops de erro, esquecimento de contexto e alucinações ("telefone sem fio").

A solução SDD:

1. Escreva o que você quer num arquivo Markdown (Spec).
2. A IA lê a Spec.
3. A IA implementa baseado na Spec (não na sua memória).



Exemplo Prático de SDD

1. O Humano escreve a Spec (`feature-login.md`):

```
# Funcionalidade de Login
- O usuário deve logar com email e senha.
- Se errar 3 vezes, bloquear por 15min.
- Retornar erro 401 se credenciais inválidas.
```

2. O Agente lê e Implementa:

O agente gera o código **exatamente** como definido nos business rules do Markdown, sem inventar moda ou esquecer detalhes.



8. MCP: Model Context Protocol

Como conectar a IA ao mundo real?

O **MCP** é um padrão aberto que permite ao Agente atuar fora da IDE.

- Ler tickets no Jira 
- Consultar documentação no Confluence 
- Acessar Banco de Dados 
- Pesquisar na Web 

É a ponte segura entre o "cérebro" da IA e os dados da sua empresa.

Exemplo de Configuração (Rovo)

Para conectar o Atlassian Rovo (Jira/Confluence), basta adicionar ao `mcp_config.json`:

```
{
  "mcpServers": {
    "rovo": {
      "command": "/home/sabino/.local/bin/mcp-npx-node24",
      "args": [
        "-y",
        "mcp-remote",
        "https://mcp.atlassian.com/v1/sse"
      ]
    }
  }
}
```

O Canonical Cycle

Framework de Consistência e Verdade




 github.com/SabinoWS/canonicalCycle



Filosofia: O Contrato IA-Humano

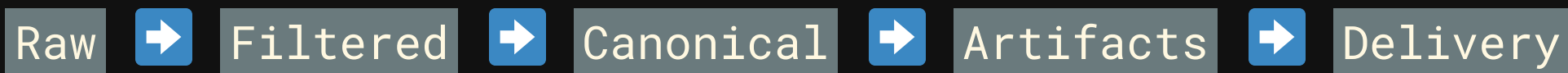
1. **IA nunca decide a verdade:** Apenas propõe interpretações.
2. **Humano nunca reinterpreta raw material sozinho:** Usa o fluxo estruturado.
3. **Canonical Material é o Ponto de Controle:** Onde a verdade é firmada.






Os 3 Pilares

-  **Bancada (Workspace Agent)**: Contexto de código e produto.
-  **Regras (Canonical Agent)**: Guardião do processo.
-  **Externo (MCPs)**: Dados do Jira, Web, Docs.

O Fluxo do Ciclo




A Jornada da Informação:



-  **Raw**: Dados brutos (reuniões, anotações). *Sem verdade.*
-  **Filtered**: IA estrutura e organiza. *Proposta.*
-  **Canonical**: Humano aprova. **VERDADE.**
-  **Artifacts**: Tickets, Docs, Código. *Descartáveis.*
-  **Delivery**: O mundo real (Deploy, PR, Publish).

Roles Sequenciais

O ciclo conecta especialistas em carrossel. O artefato de um vira o **Raw** do outro.

1.  **Analista**: Entende o negócio → *Requisitos*
2.  **Designer**: Cria a experiência → *Protótipos*
3.  **Arquiteto**: Define a estrutura → *ADRs / Diagramas*




4.  **Engenheiro**: Planeja a mudança → *Tasks Técnicas*

5.  **Desenvolvedor**: Codifica → *Commits / PRs*

“ **Atalho**: Roles como Designer e Arquiteto são opcionais e puláveis. ”

Conversa & Experimentos

Espaço Aberto

-  Mostrar outros experimentos
-  Dúvidas e Discussões
-  Próximos Passos