

Etapa 1 Diagnóstico e Auditoria

Josias Ribeiro de Freitas Júnior –
202200025644

Natalia Silva Santos – 202400018206
Luzia dos Santos Souza Neta-
201700017537

Anthony Amoz Santos Aragao –
202100011207

Sumário

o o o o

- 1 Objetivo da Etapa
- 2 Investigação de Ferramentas de CI/CD
- 3 Evidências Técnicas
- 4 Mapeamento do Fluxo Atual
- 5 Riscos e Gargalos Identificados
- 6 Configurações Adicionais Identificadas
- 7 Conclusão

Objetivo

Realizar uma investigação no repositório original com o objetivo de compreender como o ciclo de vida do software é gerenciado atualmente, com foco em práticas de Integração Contínua (CI) e Entrega Contínua (CD), identificando automações existentes, evidências técnicas, fluxo de trabalho atual, riscos e gargalos.



Investigação de Ferramentas de CI/CD

Ferramentas Identificadas



GitHub Actions

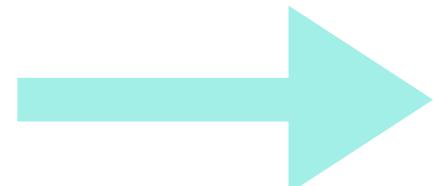
- 1 Basic Integration Tests

Esses workflows indicam a presença de automação

- 2 Copilot code review

para:

- 3 Upload Python Package



- 1 Execução de testes de integração

- 2 Apoio à revisão de código

- 3 Empacotamento e publicação de pacotes Python

Evidências Técnicas

Arquivos de Configuração



.github/workflows/

O repositório contém a pasta: **.github/workflows/**

Esse diretório é o padrão do **GitHub Actions** para armazenar arquivos de configuração no formato .yml, que definem:

- Gatilhos de execução (push, pull request, releases)
- Etapas de build, teste e publicação
- Dependências e ambientes de execução

Workflows Identificados no Re却itório Clonado:

Arquivo: **vanna/.github/workflows/tests.yml**

- Workflow: Basic Integration Tests
- Gatilho: Push na branch main
- Arquivo: vanna/.github/workflows/python-publish.yaml
- Workflow: Upload Python Package
- Gatilho: Publicação de release

Evidências Técnicas

Histórico de Pull Requests



O projeto apresenta:

- Grande volume de Pull Requests abertas e fechadas
- Uso recorrente de PRs como mecanismo principal de integração de código

A análise do histórico de PRs indica:

- Uso consistente de revisão de código
- Execução automática de checks de CI associados às PRs
- Integração de mudanças baseada em validações automatizadas

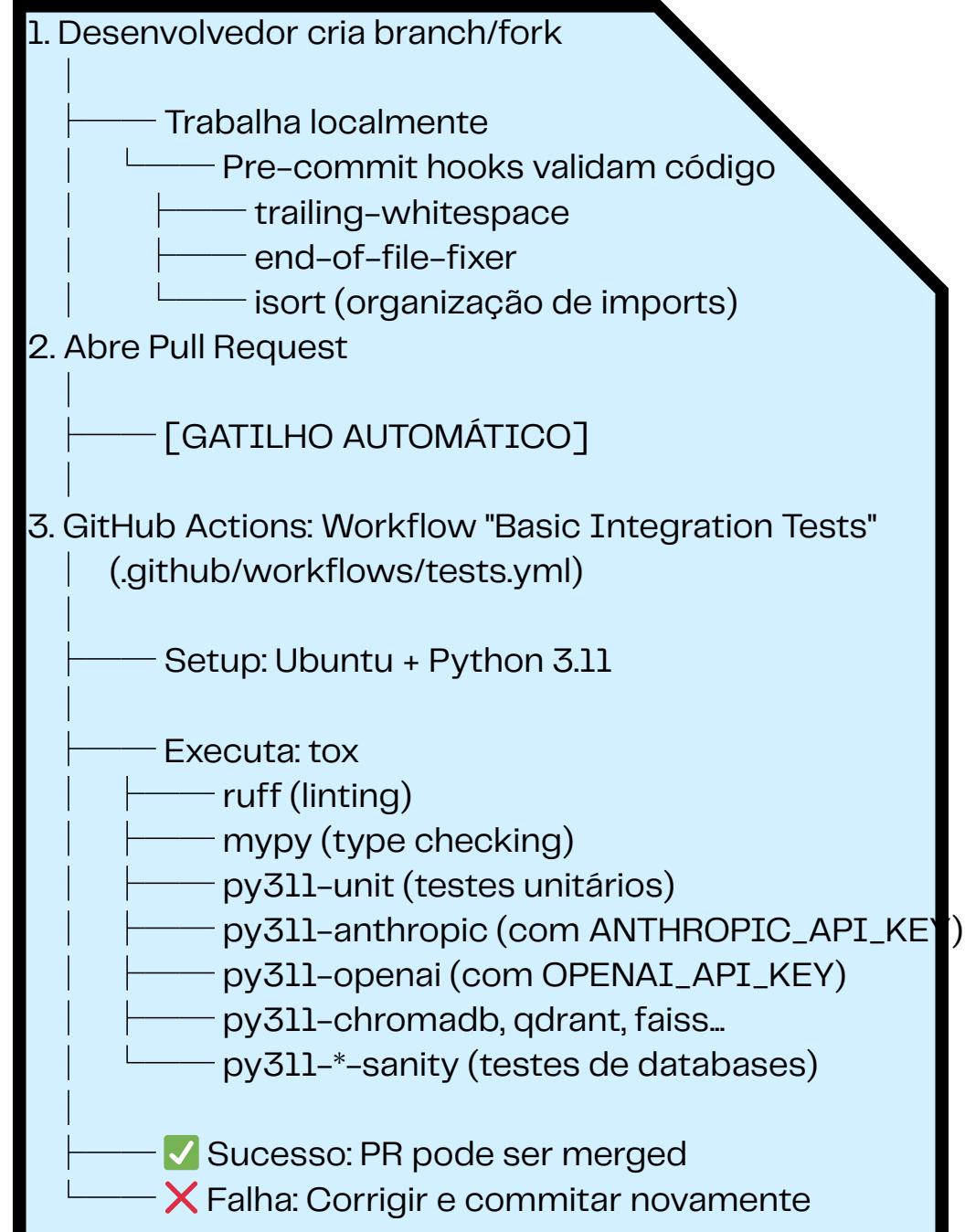
Mapeamento do Fluxo Atual



Fluxo de Desenvolvimento Identificado

Pontos de automação:

- 1** Validação local (pre-commit)
- 2** Testes em PR
- 3** Testes em push para main
- 4** Build e publicação após release



Mapeamento do Fluxo Atual



Integração Contínua (CI)

- Presente e funcional
- Automatiza testes e validações
- Reduz riscos de integração de código defeituoso

Workflow: Basic Integration Tests

Gatilho configurado:

```
on:  
  push:  
    branches:  
      - main
```

Ambiente de execução:

```
jobs:  
  build:  
    runs-on: ubuntu-latest  
    steps:  
      - uses: actions/checkout@v4  
      - name: Set up Python 3.11  
        uses: actions/setup-python@v5  
        with:  
          python-version: "3.11"
```

Execução dos testes:

```
- name: Run tests  
env:  
  PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION: python  
  VANNA_API_KEY: ${{ secrets.VANNA_API_KEY }}  
  OPENAI_API_KEY: ${{ secrets.OPENAI_API_KEY }}  
  MISTRAL_API_KEY: ${{ secrets.MISTRAL_API_KEY }}  
  ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }}  
  GEMINI_API_KEY: ${{ secrets.GEMINI_API_KEY }}  
  SNOWFLAKE_ACCOUNT: ${{ secrets.SNOWFLAKE_ACCOUNT }}  
  SNOWFLAKE_USERNAME: ${{ secrets.SNOWFLAKE_USERNAME }}  
  SNOWFLAKE_PASSWORD: ${{ secrets.SNOWFLAKE_PASSWORD }}  
run: tox
```

Mapeamento do Fluxo Atual



Integração Contínua (CI)

Análise Técnica:

- O workflow usa tox para orquestrar os testes
- Requer múltiplas chaves de API de serviços externos (OpenAI, Mistral, Anthropic, Gemini)
- Depende de credenciais do Snowflake
- Gargalo identificado: A execução dos testes depende de serviços externos e APIs de terceiros, o que pode causar instabilidade

35 workflow runs	Event	Status	Branch	Actor
Copilot code review Copilot code review #1: by Copilot AI	refs/pull/1065/head	Jan 4, 4:26 AM GMT-3		4m 6s
v2.0.1 Upload Python Package #77: Release v2.0.1 published by zainhoda		Nov 20, 2025, 3:35 PM GMT-3	main	37s
Merge pull request #1021 from vanna-ai/fix/text-... Basic Integration Tests #531: Commit f7adb62 pushed by zainhoda	main	Nov 20, 2025, 3:23 PM GMT-3		10m 53s
v2.0.0 Upload Python Package #76: Release v2.0.0 published by zainhoda		Nov 19, 2025, 3:21 PM GMT-3	main	35s
Update package.json for publishing Basic Integration Tests #530: Commit c0d2cf4 pushed by zainhoda	main	Nov 19, 2025, 3:03 PM GMT-3		9m 13s
Fix frontend send button issue Basic Integration Tests #529: Commit f99bdf9 pushed by zainhoda	main	Nov 19, 2025, 1:05 PM GMT-3		10m 8s

Mapeamento do Fluxo Atual



Entrega Contínua (CD)

O projeto implementa CD parcial através de publicação automatizada no PyPI (Python Package Index).

Gatilho do Deploy Automatizado:

```
name: Upload Python Package  
  
on:  
  release:  
    types: [published]
```

Fluxo Pipeline de Publicação:

1. Preparação do ambiente:

2. Build do pacote:

3. Publicação no PyPI:

Mapeamento do Fluxo Atual



Análise do Modelo de CD)

- ✓ Automatizado: Uma vez publicada a release, o pacote é construído e publicado automaticamente
- ⚠ Gatilho Manual: A criação da release requer intervenção humana
- ⚠ CD Parcial: Não há deploy automatizado de aplicação ou serviço em ambiente produtivo
- ✓ Entrega de Artefato: O pacote Python é entregue ao PyPI de forma consistente

Riscos e Gargalos Identificados



Análise do Modelo de CD)

Riscos de CI

- Falhas nos testes por indisponibilidade de APIs externas (não relacionadas ao código)
- Custos de execução associados ao consumo de APIs pagas
- Possível vazamento de informações sensíveis se secrets não forem gerenciados adequadamente

Riscos de CD

- Dependência de intervenção humana para iniciar o processo de entrega
- Possível erro humano na criação da release (versionamento incorreto, notas de release incompletas)
- Risco de inconsistência entre ambientes caso o deploy não seja totalmente automatizado

Riscos e Gargalos Identificados



Gargalos no CI

- Tempo de execução elevado: Executar 20+ ambientes sequencialmente pode levar muito tempo
- Dependência de APIs externas: Testes como py311-anthropic e py311-openai dependem de serviços externos disponíveis
- Configuração de secrets: Necessidade de manter múltiplos secrets sincronizados e válidos
- Manutenção complexa: Atualizar dependências em 20+ ambientes diferentes

Complexidade da Suite de Testes:

```
envlist =  
    ruff  
    mypy  
    py311-unit  
    py311-agent-memory-sanity  
    py311-anthropic  
    py311-openai  
    py311-chromadb  
    py311-qdrant  
    py311-faiss  
    py311-postgres-sanity  
    py311-sqlite-sanity  
    py311-snowflake-sanity  
    py311-mysql-sanity  
    # ... mais 8 ambientes de database
```

Riscos e Gargalos Identificados



Gargalos no CD

- Necessidade de intervenção humana para criar e publicar a release
- Maior tempo de entrega de novas versões aos usuários
- Potencial erro humano em processos repetitivos

Conforme o workflow vanna/.github/workflows/python-publish.yaml:

```
on:  
  release:  
    types: [published] # <- Gatilho manual
```

Configurações Adicionais Identificadas



Tox Configuration:

O projeto utiliza tox para gerenciar ambientes de teste, conforme referenciado no workflow de CI:

Pre-commit Hooks

- Remoção de espaços em branco no final das linhas
- Garantia de nova linha no final dos arquivos
- Detecção de conflitos de merge não resolvidos
- Detecção de statements de debug deixados no código
- Padronização de imports com isort

Estrutura de Testes

```
tests/
├── conftest.py
├── test_agents.py
├── test_agent_memory_sanity.py
├── test_agent_memory.py
├── test_azureopenai_llm.py
├── test_database_sanity.py
├── test_gemini_integration.py
├── test_legacy_adapter.py
├── test_llm_context_enhancer.py
├── test_memory_tools.py
├── test_ollama_direct.py
└── test_tool_permissions.py
    └── test_workflow.py
# Configuração
# Testes de ag
# Sanity tests
# Testes compl
# Integração A
# Sanity tests
# Integração G
# Testes de ad
# Testes de co
# Testes de fe
# Integração O
# Testes de pe
# Testes de wo
```

Conclusão



O projeto vanna-ai/vanna apresenta um nível maduro de Integração Contínua, com uso efetivo de GitHub Actions, testes automatizados e integração via Pull Requests.

1 - CI bem estruturado:

Gatilho automático em push para main

Ambiente de testes consistente (Python 3.11, Ubuntu)

Suite de testes abrangente executada via tox

2 - CD parcialmente implementado:

Publicação automatizada no PyPI após criação de release

Build e deploy do pacote totalmente automatizados

Obrigado!

Tenha um ótimo
fim de semana!