

ai-safety-lab | 端到端工作流程（可打印版）

目标：把仓库中的“适配器→编排→沙盒→测试套件→合规规则→报告”这一套跑通的路径，用一张图 + 一页步骤说明清楚。

① 一图看懂（Mermaid 流程图）

```
graph TD; U[用户/调用者] --> O{Orchestrator  
orchestrator/run.py};  
  
subgraph A [Adapters 适配器层]  
AH[HTTP Agent  
adapters/http_agent.py];  
end  
  
subgraph S [Sandbox 沙盒层]  
MS[Mock Model Server  
sandbox/mock_model_server.py];  
end  
  
O -->|加载配置/规则| C[Compliance 规则库  
compliance/*.yaml];  
O -->|选择&排程| T{Testsuites 测试套件}  
  
T --> T1[Adversarial 对抗  
testsuites/adversarial/*];  
T --> T2[Ethics/Compliance 伦理合规  
testsuites/ethics/compliance_audit.py];  
T --> T3[Consistency/Explain 可解释/一致性  
testsuites/consistency/*\n testsuites/explainability/*];  
  
O -->|目标模型交互| A;  
A -->|真实接口或沙盒| M{Model 接口};  
M -->|可切换| AH;  
AH -->|直连或转向| S;  
MS --> AH;  
  
T1 --> R[Reporting 报告层\njson_reporter.py\nhtml_reporter.py];  
T2 --> R;  
T3 --> R;  
  
C --> R;  
R --> OUT[产出：JSON/HTML 报告\n风险清单/违规项/分数]
```

② 水道图 (角色×阶段)

```
sequenceDiagram
    participant User as 用户/测试发起
    participant Orc as Orchestrator (编排)
    participant Ada as Adapter (HTTP/自定义)
    participant Box as Sandbox (Mock)
    participant TS as Testsuites (对抗/合规/一致性)
    participant Rep as Reporting (报告)

    User->>Orc: 提交配置 (模型端点、Key、测试集、规则)
    Orc->>TS: 解析计划并排程用例
    loop 对每一组用例
        TS->>Ada: 生成攻击/提问/检查请求
        alt 使用沙盒
            Ada->>Box: 调用 Mock Server 获取响应
            Box-->>Ada: 返回仿真响应
        else 直连模型
            Ada->>Ada: 以HTTP/API方式请求真实模型
        end
        Ada-->>TS: 返回模型输出
        TS->>TS: 用规则/度量打分、提取证据
        TS-->>Rep: 上报单次用例结果 (分数/证据/日志)
    end
    Rep-->>Orc: 汇总为总报告 (JSON/HTML)
    Orc-->>User: 提供报告下载/查看
```

③ 端到端步骤 (含关键文件)

- 1) 配置阶段： - 选择目标模型（真实 API 或沙盒 `sandbox/mock_model_server.py`）。 - 选择要跑的测试套件（对抗/伦理合规/一致性/可解释）。 - 选择/加载规则与标准（`compliance/*.yaml`）如 EU AI Act、UN Ethics）。
- 2) 编排与调度（`orchestrator/run.py`）： - 读取配置 → 解析测试计划 → 按套件/用例排程执行顺序与并发。
- 3) 适配器调用（`adapters/http_agent.py` 等）： - 统一请求格式（prompt、参数、温度、top_p...）→ 向真实接口或 Mock 发起请求。
- 4) 沙盒/仿真（可选）： - 用 FastAPI 的 Mock Server 注入可控回包，便于离线调试与复现。
- 5) 执行测试套件： - **Adversarial**: 构造 prompt injection、图攻击等，观察是否误导/越权。 - **Ethics/Compliance**: 对输出跑关键词/模式匹配 + 规则评估，产出违规项与分数（`testsuites/ethics/compliance_audit.py`）。 - **Consistency/Explainability**: 多次采样、一致性对比、可解释特征抓取（部分脚本为骨架待完善）。

- 6) 聚合与报告 (`reporting/json_reporter.py`, `html_reporter.py`) : - 汇总每个用例的输入/输出/证据/分数/规则命中。- 产出 **JSON 报告** (便于系统对接) 与 **HTML 报告** (审阅/归档)。
-

④ 数据流与产物 (What-In / What-Out)

- **输入**: 模型端点信息、API Key、测试套件清单、合规/伦理规则、测试参数 (重试、并发、超时)。
 - **中间产物**: 调用日志、原始响应、证据快照 (关键词命中、截图/片段ID)。
 - **输出**:
 - Per-Case: 是否通过、分数、命中的违规条款、证据链接。
 - Per-Run: 汇总指标 (通过率、平均分)、高风险清单、整改建议、可追溯运行ID。
-

⑤ 一次“典型运行”的画面化示例

- 用户在配置中勾选 **Adversarial + Ethics**, 加载 `eu_ai_act.yaml`。
 - Orchestrator 读取计划 → 生成 120 条用例队列。
 - Adapter 依次把 120 条请求打到 **Mock** (开发期) 或 **真实模型 API** (联调期)。
 - Testsuites 对每条响应:
 - 对抗用例检查是否越权、泄密;
 - 伦理用例用规则库标注“偏见/歧视/暴力”等标签并计分;
 - 保存证据 (原文、命中项、可复现的随机种子)。
 - Reporting 汇总为 **HTML 报告** (含总览雷达图/风险Top-N列表) 与 **JSON 报告** (供 CI/工单系统接入)。
-

⑥ 与常见使用场景的对齐

- **本地/离线开发**: 以 Mock 起步, 先打通编排→适配→测试→报告。
 - **联调与回归**: 切换到真实 API, 固定随机种子, 版本化规则与测试集, 实现可复现对比。
 - **合规模拟审查**: 按地区/机构挑选规则 YAML, 批量生成审查材料与证据清单。
-

⑦ 可扩展点 (结合仓库现状)

- 补齐 **Consistency/Explainability** 的度量与报告维度。
- **模型版本管理**: 同一套用例对多版本模型做 A/B 对比与趋势图。
- **前端控制台**: 可视化配置、运行监控、报告浏览与工单闭环。
- **CI/CD**: 集成到 PR/发布流水线, 设定阻断阈值 (如伦理分数<阈值则不准上线)。