Trang 3.3: Workflows - Quy trình làm việc tự động thông qua các lệnh gạch chéo (/).

1. Workflow là gì?

Workflow là các quy trình được định nghĩa sẵn theo từng bước để hoàn thành một tác vụ lập trình phổ biến. Mỗi Workflow đảm bảo tính **nhất quán** và **đúng chuẩn** cho dù bạn yêu cầu bao nhiêu lần.

Workflow giải quyết vấn đề gì?

Không có Workflow, kết quả AI trả về cho cùng một yêu cầu có thể khác nhau mỗi lần. Với Workflow, AI luôn đi theo đúng quy trình đã được định nghĩa — đảm bảo không bỏ sót bước nào.

---

2. Cách kích hoạt và sử dụng

Cú pháp cơ bản

```
/tên-workflow [mô tả yêu cầu]
```

Ví dụ thực tế

```
bash
/brainstorm hệ thống thanh toán cho SaaS app
/plan trang e-commerce với giỏ hàng và checkout
```

```
/create ứng dụng todo có dark mode và đồng bộ real-time
/debug form submit không hoạt động, không báo lỗi gì
/test src/services/auth.ts
/ui-ux-pro-max landing page cho startup fintech, tông màu xanh navy
```

---

3. Tính năng Turbo (`// turbo`)

Một số bước trong Workflow được đánh dấu `// turbo`. Khi gặp dấu này:

- AI có thể **tự động thực thi** lệnh terminal an toàn mà không cần dừng lại hỏi bạn.
- Nếu toàn bộ Workflow có nhãn `// turbo-all`: mọi lệnh terminal trong Workflow đều tự chạy.

> **Lợi ích:** Tăng tốc độ đáng kể cho các tác vụ lặp đi lặp lại như chạy test, lint check, hay preview server.

---

4. Danh sách 11 Workflows và hướng dẫn chi tiết

`/brainstorm` — Lên ý tưởng có cấu trúc

Mục đích: Khám phá ít nhất 3 phương án tiếp cận khác nhau cho một vấn đề trước khi viết code. AI sẽ phân tích ưu/nhược điểm và mức độ nỗ lực của từng phương án, sau đó đưa ra khuyến nghị.

Khi nào dùng:

- Bắt đầu một tính năng mới và chưa chắc về hướng triển khai.
- Cần so sánh các công nghệ hoặc kiến trúc khác nhau.
- Muốn đảm bảo đã cân nhắc đủ các lựa chọn trước khi commit.

Ví dụ:

```
/brainstorm chiến lược quản lý state cho ứng dụng có 50+ màn hình
/brainstorm lựa chọn database: PostgreSQL vs MongoDB cho hệ thống IoT
```

Lưu ý quan trọng: Chỉ đưa ra ý tưởng, **chưa viết code** ở bước này. Sau khi chọn phương án xong mới dùng `/create` hoặc `/enhance` để triển khai.

`/plan` — Lập kế hoạch dự án

Mục đích: Kích hoạt Agent `project-planner` để tạo ra một file kế hoạch chi tiết (`PLAN-{tên}.md`). **Không viết code** trong quá trình này.

Quy trình:

1. Bạn mô tả dự án hoặc tính năng.
2. AI đặt câu hỏi làm rõ (Socratic Questions) về yêu cầu, ràng buộc, tech stack.
3. Bạn trả lời các câu hỏi.
4. AI tạo file `docs/PLAN-tên-dự-án.md` với danh sách tác vụ và phân công Agent.

Ví dụ:

```
/plan hệ thống quản lý nhân sự với phân quyền RBAC
/plan tích hợp cổng thanh toán VNPay vào ứng dụng Next.js
```

`/create` — Tạo ứng dụng/tính năng mới

Mục đích: Trình thuật sĩ tạo ứng dụng hoàn chỉnh. Từ mô tả yêu cầu đến code chạy được.

Quy trình:

1. Bạn mô tả ứng dụng cần tạo.
2. AI phân tích và đề xuất tech stack.
3. Bạn xác nhận (Y) để tiến hành.
4. Các Agent chuyên gia triển khai từng phần (DB → Backend → Frontend → Tests).

Ví dụ:

```
/create blog cá nhân với Next.js, MDX và hệ thống comments
/create REST API cho ứng dụng quản lý thư viện sách
```

`/enhance` — Cải tiến code hiện có

Mục đích: Thêm tính năng mới vào codebase hiện tại một cách "phẫu thuật" — không làm hỏng code đang chạy.

Khác `/create` ở điểm nào?

- `/create` → Xây từ đầu (greenfield).
- `/enhance` → Sửa chữa và bổ sung vào hệ thống đang chạy (brownfield).

Quy trình:

1. AI phân tích codebase hiện có (đọc các file liên quan).
2. Lập kế hoạch thay đổi tối thiểu cần thiết.
3. Triển khai tính năng mới cùng với bài test tương ứng.
4. Chạy kiểm tra để đảm bảo không có regression.

Ví dụ:

```
/enhance thêm tính năng dark mode toggle vào ứng dụng
/enhance tích hợp real-time notifications dùng WebSocket
```

`/debug` — Sửa lỗi có hệ thống

Mục đích: Kích hoạt Agent `debugger` để điều tra lỗi theo giao thức 4 giai đoạn nghiêm ngặt.

4 giai đoạn:

1. **Discovery (Khám phá):** Thu thập log, báo cáo lỗi, các bước tái hiện.
2. **Hypothesis (Giả thuyết):** Đưa ra danh sách nguyên nhân tiềm năng, sắp xếp theo khả năng xảy ra.
3. **Verification (Xác minh):** Kiểm tra từng giả thuyết bằng bằng chứng cụ thể (không đoán mò).
4. **Resolution (Giải quyết):** Áp dụng bản sửa lỗi và thêm regression test.

Ví dụ:

```
/debug API trả về 403 dù đã đăng nhập đúng
/debug animation bị giật trên mobile Safari
```

`/test` — Tạo và chạy kiểm thử

Mục đích: Tự động hóa toàn bộ quy trình kiểm thử.

Các chức năng:

```
bash
/test # Chạy toàn bộ test suite hiện có
/test src/auth.ts # Tạo test cases cho file cụ thể (ở đây là auth.ts)
/test coverage # Báo cáo độ bao phủ (coverage report)
```

Kết quả trả về: File test mới (`tests/auth.test.ts`), số lượng test cases, tỉ lệ coverage.

`/ui-ux-pro-max` — Thiết kế UI nâng cao

Mục đích: Tạo landing page hoặc giao diện chất lượng cao theo phong cách cụ thể trong thời gian ngắn.

Tích hợp với NextLevelBuilder: Bạn có thể chọn phong cách từ thư viện tại nextlevelbuilder.io và dán vào lệnh.

Kết quả trả về:

- HTML semantic chuẩn SEO.
- Components có thể tái sử dụng.
- Micro-animations và responsive design.

Ví dụ:

```
/ui-ux-pro-max tạo landing page SaaS với glassmorphism cards, pricing 3 tới, testimonials
/ui-ux-pro-max portfolio cá nhân cho designer, phong cách minimalist với dark mode
```

`/deploy` — Triển khai ứng dụng

Mục đích: Quy trình 5 giai đoạn để triển khai an toàn lên môi trường production.

5 giai đoạn: Pre-flight checks → Build → Test → Deploy → Verify.

Ví dụ:

```
/deploy lên Vercel
/deploy lên VPS với Nginx và PM2
```

`/preview` — Quản lý server xem trước

Mục đích: Khởi động, dừng hoặc kiểm tra trạng thái server phát triển cục bộ.

```
bash
/preview start # Khởi động server (tự nhận diện Next.js, Vite...)
/preview stop # Dừng server
/preview # Kiểm tra trạng thái hiện tại
```

`/status` — Xem tổng quan dự án

Mục đích: Hiển thị bảng điều khiển (dashboard) tổng thể: thông tin dự án, tech stack, tiến độ các Agent, trạng thái server.

```
bash
/status
```

`/orchestrate` — Điều phối đa Agent

Mục đích: Kích hoạt `orchestrator` để phối hợp nhiều Agent chuyên gia cùng giải quyết một vấn đề lớn, phức tạp đòi hỏi nhiều góc nhìn chuyên môn.

Khi nào dùng: Tác vụ yêu cầu đồng thời Frontend + Backend + Security + QA.

Ví dụ:

```
/orchestrate build toàn bộ hệ thống authentication (DB + API + UI + Tests + Security review)
```

---

5. Tạo Workflow tùy chỉnh của riêng bạn

Bất kỳ ai cũng có thể thêm Workflow riêng phù hợp với quy trình nội bộ của đội:

Bước 1: Tạo file trong `.agent/workflows/`:

```
.agent/workflows/code-review.md
```

Bước 2: Viết nội dung theo format:

```
markdown
---
description: Tiến hành code review theo tiêu chuẩn nội bộ của đội
---



# Code Review Workflow



1. Đọc toàn bộ diff thay đổi
// turbo
2. Kiểm tra security issues
3. Kiểm tra performance implications
4. Kiểm tra code style theo team conventions
5. Tạo báo cáo review với mức ưu tiên (critical/major/minor)
```

Bước 3: Gọi bằng lệnh:

```
/code-review
```

---

6. Tóm tắt nhanh — Chọn Workflow nào?

Tình huống	Workflow
Chưa biết nên làm theo hướng nào	`/brainstorm`
Cần kế hoạch trước khi code	`/plan`
Xây dự án/tính năng mới từ đầu	`/create`
Thêm tính năng vào dự án hiện có	`/enhance`
Cần giao diện đẹp nhanh	`/ui-ux-pro-max`
Ứng dụng bị lỗi không rõ nguyên nhân	`/debug`
Muốn kiểm thử code	`/test`
Sẵn sàng đưa lên production	`/deploy`
Muốn xem app đang chạy	`/preview start`
Muốn biết dự án đang ở đâu	`/status`
Tác vụ cần nhiều chuyên gia cùng lúc	`/orchestrate`

Phiên bản #2
Được tạo 2026-03-04 06:10:45 UTC bởi Nam Đặng
Được cập nhật 2026-03-04 07:22:33 UTC bởi Nam Đặng