

당신이 잠든 사이, 쉬지 않고 일하는 완벽한 동료

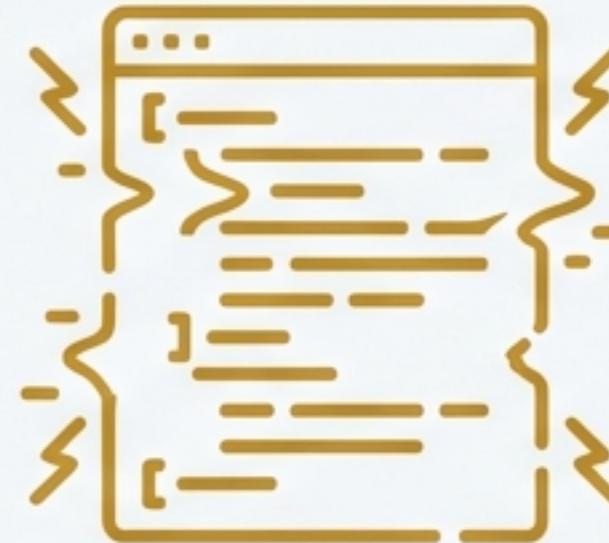
상상해 보십시오. 복잡한 소프트웨어
엔지니어링 작업을 믿고 맡길 수 있는,
지치지 않는 자율적인 협업자를.



Night Shift는 단순한 도구가 아닙니다. 신뢰할 수 있는
자율성을 현실로 만드는 새로운 접근 방식입니다.

하지만 현재의 AI 에이전트는 신뢰하기 어렵습니다

기존의 자율 에이전트들은 종종 감독이 필요하며, 중요한 순간에 불완전한 결과물을 내놓습니다.



환각 (Hallucinations)

지시하지 않은 작업을 수행하거나,
존재하지 않는 파일에 대해
보고합니다.



불완전한 작업 (Incomplete Work)

“완료”라고 보고하지만, 실제로는
코드가 깨져 있거나 테스트가
실패합니다.



지속적인 감독 필요 (Constant Supervision)

신뢰할 수 없기 때문에, 개발자는
AI의 모든 단계를 감시하며 귀중한
시간을 낭비합니다.



› nightshift run deployment --env production
Starting autonomous workflow...

Executing autonomous

Night Shift: The Autonomous Overlord

Executing task: build_docker_image

---> Processing docker directory
-> a2498e8df: build_docker-tmagfjnrhstegrant-manage
-> a28e09031: build_docker-image@7be602871695869

Executing task: build_docker_image

Success: Image built.

Verifying deployment...

STATUS: All tests passed.

Verifying deployment...

STATUS: All tests deployment...

Success: Running task: build_runmster_instruction...

Executing ta
Night Shift는 복잡한 소프트웨어 엔지니어링 작업을 안정적으로 실행하도록

STATUS: Deployment common

Success: Image buil

설계된 커맨드 라인 네이티브 자율 에이전트 오케스트레이터입니다.

Verifying deployment...

STATUS: All tests passed.

› nightshif
프로젝트 작업을 위임하고, 스스로 작업을 완료하며, 자신의 결과물을 검증하여,
STATUS: All tests passed.

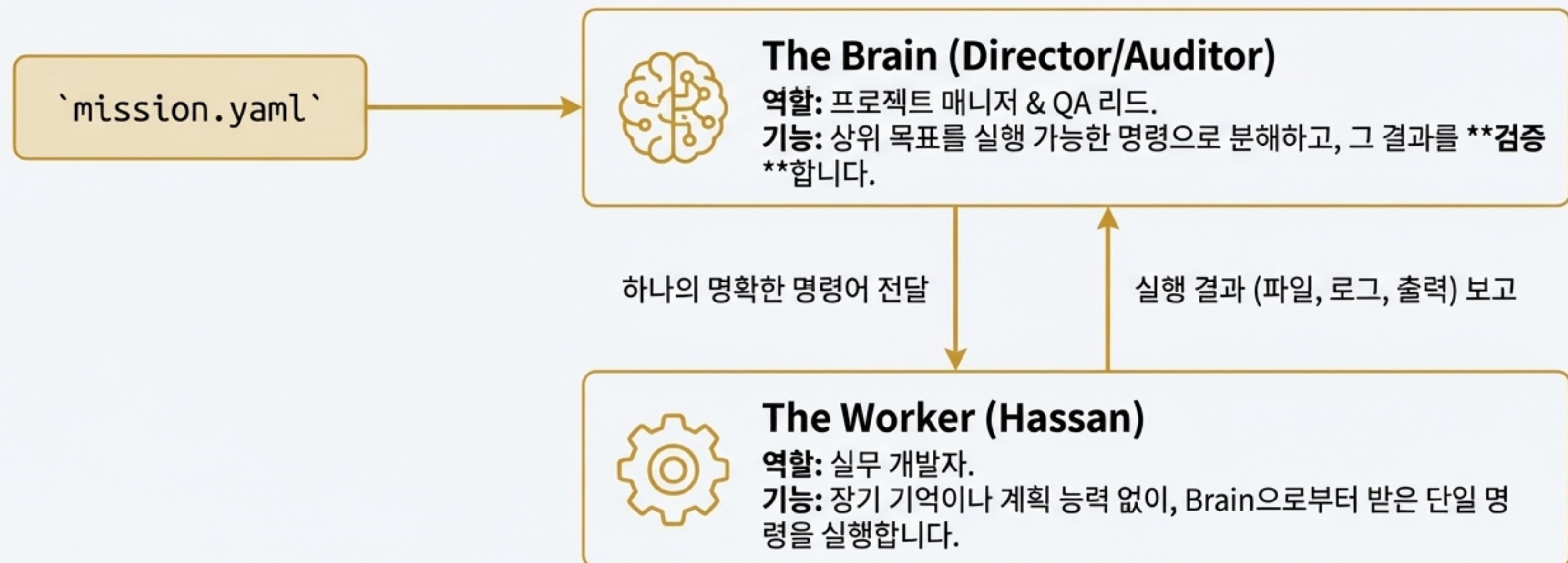
› █

당신이 자신감을 갖고 위임할 수 있게 합니다.



성공의 비결: 두뇌와 일꾼의 분리

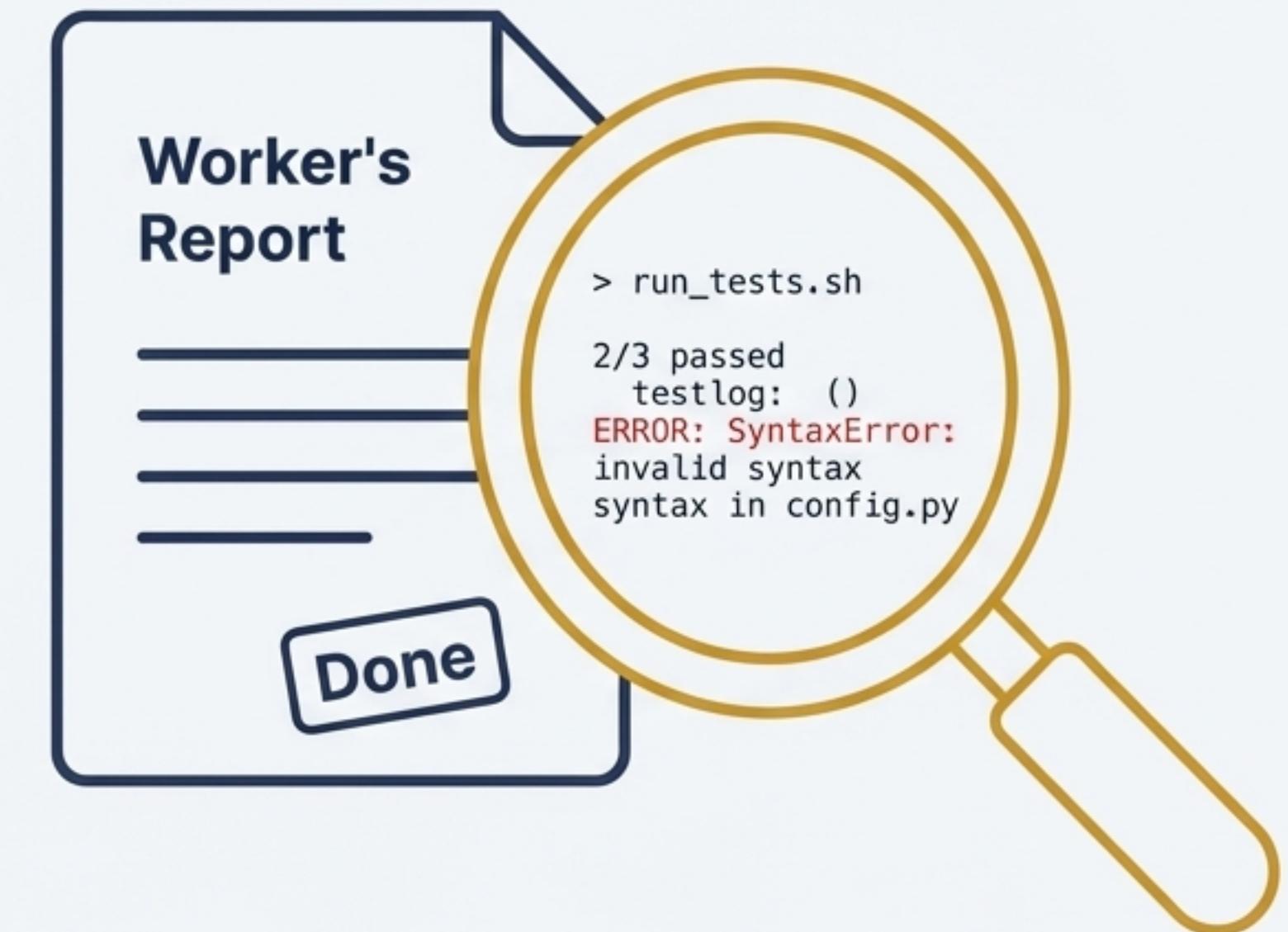
Night Shift의 아키텍처는 두 개의 명확히 구분된 에이전트 역할의 협력을 기반으로 합니다. 전략적 계획과 단순 실행을 분리하여 안정성과 디버깅 용이성을 확보합니다.



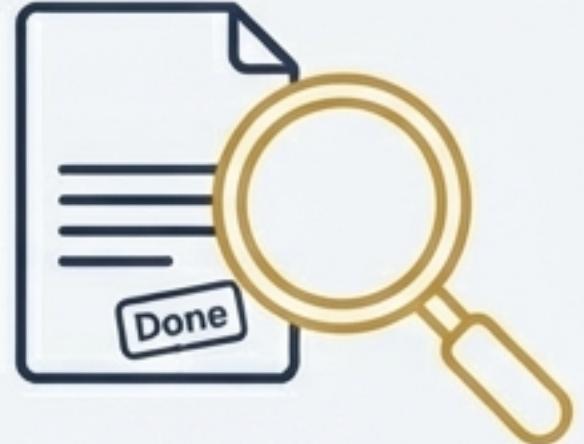
핵심 철학: 증거 기반 완료 (Evidence-Based Done)

진정한 자율성은 단순히 명령을 실행하는 것이 아니라, 목표를 달성하는 것입니다.

- Night Shift의 **Brain**은 Worker의 보고를 맹목적으로 신뢰하지 않습니다.
- 작업 완료는 오직 **물리적 증거**—파일 내용, 테스트 로그, 명령어 출력—가 세션 히스토리에 명확히 나타날 때만 승인됩니다.
- 이 프로토콜은 에이전트의 환각과 불완전한 작업을 극적으로 줄여줍니다.



왜 Night Shift를 선택해야 하는가? (v5.6 기준)



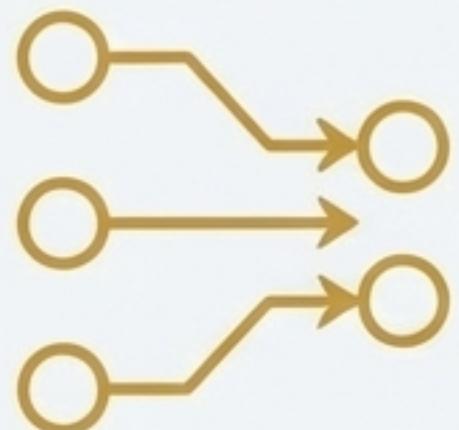
🔍 증거 기반 완료 (Evidence-Based Done)

Worker의 결과물에 대한 물리적 증거를 Brain이 직접 확인하여 완료를 보장합니다.



⚖️ 감사자로서의 Brain (Brain as Auditor & Architect)

Brain의 정체성은 객관적이고 회의적인 '감사자'로 고정되어, Worker의 페르소나와 관계없이 미션 제약 조건을 엄격히 준수합니다.



↖ 간결한 오케스트레이션 (Lean Orchestration)

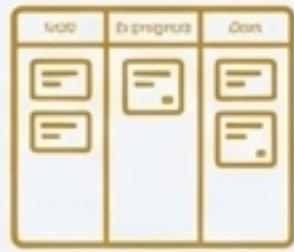
별도의 Critic 모듈을 Brain의 검증 로직에 통합하여 토큰 오버헤드와 에이전트 간 마찰을 줄였습니다.



🧠 인지 아키텍처 (Cognitive Architecture)

에이전트는 과거의 실수로부터 배우고(ReflexionMemory), 미션 성공 선언 전 엄격한 구조적 무결성 검사를 적용합니다.

개발 워크플로우를 위한 강력한 기능들



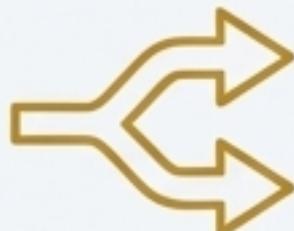
▣ 프로젝트가 곧 대시보드 (Project-as-a-Dashboard)

`mission.yaml` 파일을 통해 전체 프로젝트를 관리합니다. 태스크 상태(`todo`, `in_progress`, `done`)가 실시간으로 업데이트됩니다.



🛠 로컬 LLM 최적화 (Optimized for Local LLMs)

`view`, `list`, `edit` 같은 스마트 도구와 향상된 프롬프트 엔지니어링으로 DeepSeek, Qwen 등 로컬 모델 지원을 강화했습니다.



⚡ 병렬 실행 (Parallel Execution)

독립적인 작업들을 `parallel: true`로 설정하여 동시에 실행할 수 있습니다.



🔌 MCP 지원 (MCP Support)

Model Context Protocol을 통해 Serena, Sequential Thinking 등 외부 도구 및 메모리를 완벽하게 통합합니다.



◀ 안전망 (Safety Net)

실패 시 자동으로 Git 체크포인트를 생성하고, 이전 상태로 롤백하는 기능을 제공합니다.

지금 바로 시작하기: 간편한 설치

macOS 및 Linux 사용자는 한 줄의 명령어로 Night Shift를 설치할 수 있습니다.

```
# bash  
curl -fsSL https://raw.githubusercontent.com/appff/NightShift/main/install.sh | bash
```

설치 스크립트가 하는 일

- ✓ `~/.night_shift_app`에 리포지토리 클론
- ✓ 전용 Python 가상 환경 설정 및 의존성 설치
- ✓ `uv`와 `npm`이 감지되면 MCP 서버 (**Serena** 등) 자동 설치
- ✓ `nightshift` 명령어를 `PATH`에 추가

업데이트도 동일한 명령어를 다시 실행하면 됩니다.

Quick Start: 프로젝트 기반 워크플로우

1단계: `mission.yaml` 파일 생성

새 프로젝트 또는 기존 프로젝트의 루트에 `mission.yaml` 파일을 생성하여 당신의 메인 대시보드로 사용하세요.

```
project:
  project_root: "." —— # 미션의 루트 디렉토리 정의
mission:
  title: "Develop a new feature and document it"
  persona: "general" —— # 모든 태스크의 기본 페르소나
constraints:
  - "Use Python 3.11+"
  - "All code must be formatted with black."
tasks:
  - title: "Design the core module"
    persona: "system-architect" # 특정 태스크 페르소나 오버라이드
  - title: "Implement the feature with unit tests"
  - title: "Write user documentation"
    persona: "technical-writer"
```

Quick Start: 실행과 관찰

2단계: Night Shift 실행

`mission.yaml` 파일이 있는 디렉토리에서 간단히 명령어를 실행하세요.
Night Shift가 태스크 ID와 진행 상황을 자동으로 관리합니다.

```
nightshift mission.yaml
```

3단계: 관찰 및 협업

-  **엄격한 감사 (Strict Audit):** Brain은 `view` 또는 `read_file` 명령어로 Worker의 결과물을 직접 확인합니다.
-  **자동 상태 관리 (State Tracking):** Night Shift가 자동으로 `id: task_n`과 `status: todo`를 추가하고, 상태는 `todo` → `in_progress` → `done`으로 YAML 파일에 직접 업데이트됩니다.
-  **지속성 (Persistence):** 언제든 중단하고 정확히 그 지점부터 다시 시작할 수 있습니다.

당신의 필요에 맞게: 상세 설정 (`settings.yaml`)

프로젝트 루트의 `settings.yaml` 파일을 통해 Night Shift의 모든 측면을 제어할 수 있습니다.



🧠 Intelligence (두뇌 & 신체)

```
active_driver: "claude"  
output_format: "json"
```



🛡 Safety & Verification (안전 & 검증)

```
auto_rollback_on_failure: true  
require_approval_for_destructive: true
```



🕒 Efficiency & Context (효율성 & 컨텍스트)

```
message_efficiency: true  
context_reduction: "auto"
```

모든 LLM과 도구를 당신의 편으로

Night Shift는 특정 모델에 종속되지 않으며, MCP를 통해 외부 도구와 원활하게 통합됩니다.

모델 독립성 (Model-Agnostic)

Ollama, Claude, Gemini 등 CLI 래퍼를 가진 모든 로컬 또는 원격 LLM의 강력함을 활용할 수 있습니다.



모델 컨텍스트 프로토콜 (MCP) 지원

MCP 호환 서버에 연결하여 외부 도구와 메모리를 Night Shift의 작업에 통합하세요.

```
mcp_enabled: true  
mcp_servers:  
  serena:  
    command: "uvx"  
    args: ["--from", "git+...", "serena", "start-mcp-server"]  
  sequential_thinking:  
    command: "npx"  
    args: ["-y", "@modelcontextprotocol/server-sequential-thinking"]
```

더 깊이 알아보기

시스템 구조, 기능, 품질 보증 방식에 대한 상세 문서를 제공합니다.

-  • docs/architecture.md: 시스템 구조와 데이터 흐름 (v5.6 Auditor Model)
-  • docs/features.md: 기능 개요 및 사용 가능한 모든 페르소나
-  • docs/quality_gates.md: Night Shift가 '진정한 완료'를 보장하는 방법

커뮤니티 및 기여자



Acknowledgements: 이 프로젝트에 사용된 전문 페르소나는 SuperClaude 프레임워크에서 파생되었습니다. 프롬프트 엔지니어링에 기여한 SuperClaude 커뮤니티에 감사드립니다.



이제, 자신감을 갖고 위임하세요



Night Shift는 단순한 자동화 도구를 넘어,
당신의 **가장 신뢰할 수 있는 자율적인 동료**가 될 것입니다.



신뢰 (Trust)

증거 기반 완료로 결과물을
보장합니다.



자율 (Autonomy)

당신의 감독 없이도 복잡한
작업을 완수합니다.



통제 (Control)

CLI 네이티브 환경과 상세한 설정을
통해 모든 것을 제어할 수 있습니다.