🌙 Night Shift: The Autonomous Overlord

An Autonomous Agent Orchestrator designed to reliably execute complex software engineering tasks.

Night Shift acts as a tireless collaborator that works on your projects, finishes tasks, and verifies its own work, allowing you to delegate with confidence.

Our Core Philosophy: Autonomy Through Verification

True autonomy isn’t just about executing commands; it’s about achieving a verifiable goal.

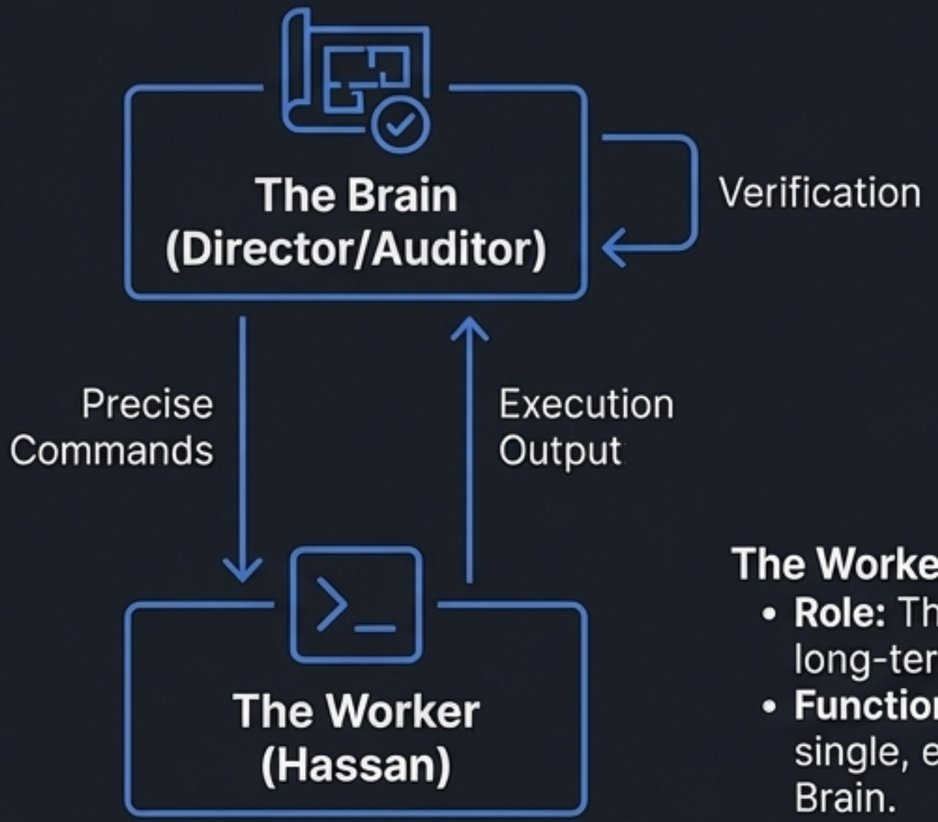
We call this the **“Evidence-Based Done”** protocol.

Night Shift requires the agent to find physical proof of completion by inspecting file content, logs, or command outputs. This approach is designed to significantly reduce agent hallucinations and incomplete work.

A Simple and Robust Two-Agent System

The Brain (Director/Auditor)

- **Role:** The project manager and quality assurance lead.
- **Function:** Breaks down high-level goals into precise commands and, crucially, **verifies*** the outcome. Its core identity is that of a ***strict auditor***.



The Worker (Hassan)

- **Role:** The hands-on developer with no long-term memory.
- **Function:** Receives and executes single, explicit commands from the Brain.

This separation of duties—strategic planning and verification vs. simple execution—creates a reliable, debuggable, and effective workflow.

The Brain’s Mandate: ‘Evidence-Based Done’ in Action

The Brain (Director) no longer blindly trusts the Worker's reports. Completion is only granted when physical evidence is visible in the session history.

1 Assign

```
[BRAIN]: "Implement the
feature with unit tests"
```

2 Execute

```
[WORKER]: "Task complete.
All tests pass."
```

3 Verify

```
[BRAIN]: read_file tests/test_feature.py
[OUTPUT]:
import pytest
from new_feature import core_function

def test_core_function_positive():
    assert core_function(2) == 4
```

This strict, evidence-based protocol makes the Brain an objective and skeptical auditor, ensuring the mission is truly complete.

Advantages of the v5.6 Evidence-Based Architecture

Brain as Auditor & Architect

The Brain's fixed identity as a high-level Auditor ensures it remains objective and skeptical, enforcing mission constraints.

Lean Orchestration

The separate Critic module has been integrated directly into the Brain's verification logic, reducing token overhead and agent-to-agent friction.

Cognitive Architecture

Agents learn from past mistakes using **ReflexionMemory** and apply rigorous integrity checks before declaring a mission success.

MCP Support

Seamlessly integrate external tools and memory via the **Model Context Protocol** (e.g., Serena, Sequential Thinking).

Your Project's Central Command: The `mission.yaml` Dashboard

Manage your entire project via a single YAML file that evolves in real-time.

```yaml
project:
  project_root: "."
mission:
  title: "Develop a new feature and document it"
  persona: "general"
  constraints:
    - "Use Python 3.11+"
    - "All code must be formatted with black."
  tasks:
    - id: task_1
      title: "Design the core module"
      status: done # done
    - id: task_2
      title: "Implement the feature with unit tests"
      status: in_progress # in_progress
    - id: task_3
      title: "Write user documentation for the new feature"
      status: todo # todo
```

- Night Shift automatically adds `id` and `status` to tasks.
- Statuses update from `todo` → `in_progress` → `done` directly in the file.
- The mission is persistent: stop and resume exactly where you left off.

A Feature Set Built on a Solid Foundation

Optimized for Local LLMs
Enhanced support for models like DeepSeek and Llama via **Smart Tools** (`view`, `list`, `edit`) and advanced prompt engineering to combat recency bias.

Parallel Execution
Run independent tasks simultaneously by setting `parallel: true` in your mission.

Safety Net
Automatic Git checkpoints are created, with an option for auto-rollback on task failure.

Message Efficiency
Suppress redundant persona text in long sessions to save tokens and reduce costs.

Model-Agnostic
Leverage any local or remote LLM that has a CLI wrapper (Ollama, Claude, Gemini, etc.).

Quick Start, Step 1: Initialize Your Project

For any new or existing project, create a `mission.yaml` file in the project's root directory. This file is your main dashboard.

```yaml
# mission.yaml
project:
  project_root: "." # Defines the root directory for this mission
mission:
   title: "Develop a new feature and document it"
   persona: "general" # Default persona for all tasks
   constraints:
     - "Use Python 3.11+"
     - "All code must be formatted with black."
   tasks:
     - title: "Design the core module"
       persona: "system-architect" # Optional: override persona
     - title: "Implement the feature with unit tests"
     - title: "Write user documentation for the new feature"
       persona: "technical-writer"
```

Quick Start, Steps 2 & 3: Run Night Shift and Observe

Step 2: Run Night Shift

```
# Ensure you are in the directory containing mission.yaml
nightshift mission.yaml
```

Step 3: Observe and Collaborate

- **Strict Audit**: The Brain will command `view` or `read_file` to verify the Worker's output before finishing.
- **Auto-Injection**: Night Shift automatically adds `id: task_n` and `status: todo` to your tasks.
- **State Tracking**: Statuses update from `todo` → `in_progress` → `done` directly in your YAML file.
- **Persistence**: You can stop and resume the mission exactly where you left off.

Effortless Installation for macOS & Linux

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

What The Script Does:

- Clones the repository to `~/.night_shift_app`.
- Sets up a dedicated Python virtual environment.
- Installs all dependencies.
- ****Auto-installs MCP servers**** (Serena, etc.) if `uv` and `npm` are detected.
- Adds the `nightshift` command to your `PATH`.

🗣️ ****Pro-Tip****: To update your installation, simply run the same command again.

Fine-Tuning Your Agent via `settings.yaml`

Configure Night Shift's core behavior in the `settings.yaml` file located in your project root.

🧠 Intelligence (Brain & Body)

- `active_driver`: Choose your LLM engine (`claude`, `gemini`, `deepseek`, etc.).
- `output_format`: Set to `json` for more reliable autonomous operation.

🛡️ Safety & Verification

- `auto_rollback_on_failure`: Roll back Git changes if a task fails.
- `require_approval_for_destructive`: Gate commands like `rm -rf`.
- `qa.run_tests`: Automatically run your test suite after tasks.

🔌 Model Context Protocol (MCP)

- `mcp_enabled`: Globally enable or disable MCP connections.
- `mcp_servers`: Define commands to launch MCP servers.

Explore the Documentation and Source

Dive deeper into the system's design and capabilities.

- `docs/architecture.md`: Detailed breakdown of the system structure and data flow (v5.6 Auditor Model).
- `docs/features.md`: A comprehensive overview of all capabilities and available personas.
- `docs/quality_gates.md`: An in-depth explanation of how Night Shift ensures 'true completion'.

github.com/appff/NightShift

Acknowledgements: Personas are derived from the SuperClaude framework, thanking the community for their contributions.