🤖 Claude Code Routines: проактивные агенты для разработки

🚀 Превращение Claude Code из инструмента в проактивного помощника

Ключевые тезисы:

- Claude Code — мощный инструмент, но пока он ждёт, пока пользователь нажмёт Enter.
- Цель — превратить его в **проактивного тиммейта**, который сам реагирует на события и проблемы.
- Для этого создана функция **Routines**, которая автоматизирует запуск сессий Claude Code по расписанию или событиям.
- Routines избавляют разработчика от необходимости строить и поддерживать инфраструктуру для proactive-агентов.

🔥 Проблемы создания проактивных агентов сегодня

Создание proactive-агентов на Claude Code возможно, но связано с трудностями:

1. **Где запускать?** 👉 Запуск на локальной машине ненадёжен (ноутбук можно закрыть). Требуется своя инфраструктура: хостинг, сохранение состояния, аутентификация.
2. **Как запускать?** 👉 Нужно самостоятельно настраивать триггеры (cron, эндпоинты), что создаёт много шаблонного кода.
3. **Контроль и наблюдение:** 👉 Сложно отслеживать, что делает агент в реальном времени, направлять его или возобновить сессию. Нет возможности быть "человеком в цикле" по необходимости.

💡 Решение: Routines в Claude Code

Routines — это автоматизация, где вы задаёте только **промпт, репозитории, коннекторы и триггер**. Claude Code управляет всем остальным.

Три ключевых принципа Routines:

1. **Всегда доступны** ✅: Запускаются на управляемой инфраструктуре Claude Code. Не зависят от вашего ноутбука. Claude Code сам управляет хостингом, состоянием сессии и коннекторами.
2. **Проактивны с настраиваемыми триггерами** ⏰: Запуск по расписанию (time-based) или по событиям (event-based), включая нативные события GitHub и кастомные вебхуки.
3. **Интерактивны и управляемы** 🎮: Каждая routine — это обычная сессия Claude Code "под капотом". Её можно открывать, наблюдать, направлять и возобновлять через веб, CLI или десктоп.

📚 Реальный кейс: Автоматизация создания документации в Anthropic

Проблема: Инженер по документации не успевала за ростом количества PR в Claude Code (увеличение на 200% с начала года).

Решение с помощью Routines:

1. **Рутина по расписанию:**
    - **Триггер:** `once a week` (каждый понедельник в 10:00).
    - **Промпт:** *"Пожалуйста, просмотри все новые изменения, смерженные в main, сравни с репозиторием документации и создай PR для обновления docs, если увидишь изменения."*
    - **Контекст:** Доступ к репозиториям исходного кода Claude Code и документации, коннекторы GitHub и Slack.

2. **Рутина по событию GitHub:**
    - **Триггер:** `on issue open` в репозитории документации.
    - **Промпт:** *"Исследуй issue, на которое среагировала эта сессия. Определи, указывает ли оно на пробел в документации. Если да, открой PR и уведоми меня в Slack."*

🎯 Три ключевых решения при создании любой Routine

1. Триггер (Когда запускать?)

- **По расписанию:** Например, еженедельный обзор изменений.
- **По событию:**
    - Нативные события GitHub (мерж PR, создание issue, релиз).
    - Кастомные события через POST-запрос на вебхук (например, после деплоя).

2. Контекст (Что нужно агенту для успеха?)

- **Репозитории:** Доступ к исходному коду, документации и т.д.
- **Коннекторы и инструменты:**
    - Google Drive — для доступа к маркетинговым материалам и стилю.
    - Slack — для уведомлений.
    - Мониторинг (DataDog, Grafana) — для проверок.
    - **Важно:** Контекст определяет "потолок" возможностей Claude в этой задаче.

3. Управляемость (Как направлять Claude и контролировать качество?)

- **Агент-на-агенте (генератор-критик):** Одна рутина создаёт PR, вторая — автоматически проверяет и комментирует его.
- **Человеческий надзор в реальном времени:** Можно зайти в веб-интерфейс, посмотреть live-сессию, задать вопросы, перенаправить агента.
- **Верификация вывода:** Например, просмотр сгенерированной страницы документации перед мержем.

💼 Примеры применения Routines для разработчиков

🛡️ Верификатор деплоя (Deploy Verifier)

- **Триггер:** Вебхук от CI/CD пайплайна после деплоя.
- **Контекст:** Исходный код сервиса, коннекторы к мониторингу (DataDog), Slack для алертов.
- **Управление:** Claude проводит расследование, даёт рекомендацию "go/no-go" по откату. Разработчик может проверить анализ и либо сам откатить с помощью Claude, либо доверить это агенту.

📋 Исследователь инцидентов (On-call Investigator) или Помощник PM

- **Триггер:** По расписанию (еженедельно).
- **Контекст:** Доступ к GitHub Issues, Slack-каналам с багрепортами.
- **Задача:** Автоматический анализ и приоритизация backlog, создание PR для самых важных issues.

🎯 Ключевые выводы

- **Проактивные агенты лучше реактивных.** Цель — превратить Claude Code из инструмента в **тиммейта**, который сам решает проблемы.
- **Routines избавляют от рутины.** Не нужно строить инфраструктуру — можно сфокусироваться на предметной области и бизнес-логике.
- **Начать легко.** Достаточно одной команды `/schedule` в Claude Code, чтобы создать свою первую routine.
