Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Claude Code Agent View: контроль над ИИ-агентами

Разворачиваем парк фоновых агентов: agent view, режим /goal, права и надзор без утечек

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Клиенты внедряют ИИ-агентов, но инженер видит один терминал: ждёт вывод и не контролирует остальные сессии. Мы строим контур управления парком агентов Claude Code — agent view, фоновые сессии, режим /goal — с разграничением прав и аудитом. Без него автономный агент с правами на прод снесёт данные быстрее, чем человек прочитает вывод.

Почему это важно бизнесу

- Один инженер вместо очереди задач ведёт 4-5 параллельных агентов: разбор логов, документов, миграций закрывается за часы, а не дни
- Неконтролируемая автономия — риск: агент с правами на прод и командой «делай до результата» ломает данные без шанса на откат
- Каждая параллельная сессия расходует квоту тарифа пропорционально — без диспетчеризации бюджет на ИИ сгорает впустую
- Персональные данные и коммерческая тайна в облачном сервисе — прямой риск по 152-ФЗ: нужен фильтр того, что попадает в контекст

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.1.212+

наш минимум Claude Code: agent view (доступен с 2.1.140), /fork и /resume-пикер — с 2.1.212

по докам code.claude.com

4 000

лимит условия /goal в символах; всегда зашиваем клаузу «or stop after 20 turns»

по докам /goal

Haiku

модель-оценщик условия /goal после каждого хода (ANTHROPIC_DEFAULT_HAIKU_MODEL)

по докам model-config

15 с

предельная частота обновления строк-статусов в agent view; agent_needs_input дублируем в Teleg…

доки agent view + наш стандарт

5

потолок фоновых агентов на инженера: выше падает качество надзора и горит квота

наш стандарт

2 с

окно повторного Ctrl+X в agent view: первое нажатие останавливает сессию, второе — удаляет её

по докам agent view

РЕАЛИЗАЦИЯ 1 ИЗ 3

Диспетчерская: agent view и фоновый парк

Что настраиваем

Рабочее место инженера сопровождения: терминал + VDS, Claude Code 2.1.212+, репозитории клиентов

Как мы это делаем

1. Открываем пульт командой claude agents; фоновые задачи ставим claude --bg --name <метка> «промпт», группы статусов: Needs input / Working / Completed
2. Изоляция по умолчанию: каждая фоновая сессия правит код в своём git-worktree в .claude/worktrees/ — конфликтов между агентами нет
3. Надзор без переключения: Space открывает peek-панель с вопросом агента, отвечаем прямо из списка; фильтры s:working, s:blocked, a:<агент>
4. Уведомления: Notification-hook с типами agent_needs_input / agent_completed заводим на Telegram-бота дежурного
5. Аварийный стоп: Ctrl+X (повтор за 2 с — удаление) или claude stop <id>; claude logs <id> — разбор, claude respawn <id> — перезапуск

РЕЗУЛЬТАТ

Один инженер ведёт 4-5 параллельных задач: пока один агент разбирает логи, второй мигрирует конфиг, третий чистит документацию. Время человека уходит на приёмку результата, а не на ожидание, и ни одна сессия не висит незамеченной — очередь Needs input всегда наверху списка.

КЛЮЧЕВОЙ НЮАНС

claude rm убирает сессию из списка (транскрипт цел), worktree позже снимает свип по cleanupPeriodDays; незакоммиченную работу свип не трогает, но в ветку она сама не попадёт — принятый результат commit/push ср…

РЕАЛИЗАЦИЯ 2 ИЗ 3

Непрерывный режим /goal с ограничителями

Что настраиваем

Долгие рутинные задачи: миграции конфигураций, разбор бэклога, чистка документации — на выделенной сессии

Как мы это делаем

1. Формулируем условие с измеримым финалом и проверкой: «все тесты в test/auth проходят, npm test выходит с кодом 0, git status чистый»
2. В каждое условие зашиваем стоп-клаузу «or stop after 20 turns» — потолок автономии, который оценщик судит по транскрипту
3. Права не расширяем: /goal не меняет permissions; безнадзорные ходы — пара «/goal + auto mode» и точечные allow-правила, не bypassPermissions
4. Headless-прогоны: claude -p "/goal <условие>" --output-format stream-json --verbose — иначе процесс молчит до самого финала
5. Статус смотрим командой /goal без аргументов: ходы, время, токен-спенд, последний вердикт оценщика; аварийно — /goal clear

РЕЗУЛЬТАТ

Многочасовая рутина уходит в автономный цикл: основной агент работает, а отдельная быстрая модель после каждого хода судит условие и отдаёт причину «не готово» как директиву на следующий ход. Расход на оценку копеечный, а потолок в 20 ходов гарантирует, что цикл не убежит.

КЛЮЧЕВОЙ НЮАНС

Оценщик не запускает команды и не читает файлы — судит только по транскрипту. Поэтому условие обязано требовать доказательство в выводе: код выхода теста, счётчик файлов, пустую очередь.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Контур безопасности: права, hooks, managed settings

Что настраиваем

Все машины с Claude Code у нас и у клиентов: политика прав плюс запреты, которые нельзя снять локально

Как мы это делаем

1. В settings.json задаём permissions: точечные allow на тесты и линтеры, deny на деструктивные Bash-команды и чтение каталогов с кредами и .env
2. Опасные операции держим в режиме ask: деплой, миграции БД, git push --force — подтверждение человека обязательно
3. PreToolUse-hook валидирует команды до запуска: вернул deny — вызов блокируется ещё до permission-диалога
4. Парковые запреты кладём в managed-settings.json (/etc/claude-code/): локальный settings их не перекрывает; allowManagedHooksOnly против чужих хуков
5. ПД и коммерческую тайну в контекст не подаём: агентам достаются обезличенные выгрузки и тестовые контуры, не прод-базы

РЕЗУЛЬТАТ

Автономность растёт, а зона поражения — нет: агент не может выполнить деструктивную команду или прочитать секреты, что бы ни стояло в промпте или условии /goal. Сценарий «агент снёс прод» исключается конфигурацией, а не дисциплиной команды.

КЛЮЧЕВОЙ НЮАНС

/goal построен на системе hooks: он недоступен при disableAllHooks=true и при allowManagedHooksOnly в managed settings — всегда проверяем порядок применения настроек (settings precedence).

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Автопилот поверх bypassPermissions**
  «Делай до результата» плюс отключённые права = снос данных. Автономию даём только точечными allow-правилами при живых deny-запретах.

- ✕ **Условие /goal без доказательства**
  Оценщик судит по транскрипту и не запускает команд. Условие «сделай красиво» зацикливает; требуем измеримое: «npm test выходит с кодом 0».

- ✕ **Отключённая worktree-изоляция**
  worktree.bgIsolation: "none" (умолчание — "auto") заставляет параллельные сессии писать в один каталог — гонки правок. Отключаем только для одиночног…

- ✕ **Молчащий headless-/goal**
  claude -p с целью не печатает ничего до финала и выглядит зависшим. Всегда добавляем --output-format stream-json --verbose.

- ✕ **Глобальный ANTHROPIC_DEFAULT_HAIKU_MODEL**
  Переменная меняет не только оценщика /goal — на неё переезжают алиас haiku и все фоновые функции (суммаризация и др.). Меняем осознанно, тестируем.

- ✕ **Брошенный worktree после приёмки**
  Результат фоновой сессии живёт в .claude/worktrees/<имя> и сам в основную ветку не попадёт; чистый каталог снимет свип по cleanupPeriodDays. Правило:…

- ✕ **Квота сгорает на параллелизме**
  Каждая фоновая сессия ест лимит тарифа пропорционально. Держим потолок агентов и снимаем «зомби» через claude stop / claude rm.

- ✕ **Секреты в контексте агента**
  Claude Code — облачный сервис: контекст уходит на сторону. Прод-креды и ПД закрываем deny-правилами чтения и обезличенными выгрузками.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Claude Code 2.1.140+ (agent view; /fork и /resume-пикер — с 2.1.212) вместо пачки те…
- permissions.deny на деструктивные команды и креды, опасное — через ask
- Стоп-клауза «or stop after 20 turns» в каждом условии /goal

НОРМАЛЬНО

- Worktree-изоляция фоновых сессий включена, приёмка результата через PR
- Notification-hook agent_needs_input → Telegram дежурного инженера
- PreToolUse-hook с валидацией команд до исполнения
- Потолок 5 фоновых агентов на инженера, разбор очереди Needs input

ХОРОШО

- managed-settings.json: парковые запреты, неснимаемые локально
- allowManagedHooksOnly + контроль settings precedence
- Headless-прогоны /goal по расписанию со stream-json-логом в аудит
- Тестовые контуры и обезличенные данные вместо прод-доступа

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Видите ли вы все запущенные ИИ-сессии на одном экране (claude agents), а не по разным терминалам?
- [ ] Есть ли deny-правила на деструктивные команды и чтение каталогов с секретами?
- [ ] Проходят ли деплой, миграции и платёжные операции через подтверждение человека (ask)?
- [ ] Ограничен ли каждый /goal стоп-клаузой по числу ходов или времени?
- [ ] Изолированы ли параллельные агенты по git-worktree, а не пишут в один каталог?
- [ ] Приходят ли уведомления agent_needs_input дежурному, а не теряются в терминале?
- [ ] Коммитится ли результат до удаления фоновой сессии и её worktree?
- [ ] Закрыт ли доступ агентов к персональным данным и прод-базам (обезличка, тестовый контур)?
- [ ] Зафиксированы ли парковые запреты в managed settings, недоступных для локальной правки?
- [ ] Проверяет ли человек результат каждого автономного прогона перед выкаткой?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Разворачиваем контур агентов Claude Code под ключ: agent view, фоновые сессии, worktree-изоляция, уведомления
- Пишем политику прав: permissions allow/deny/ask, PreToolUse-хуки, managed settings для всего парка машин
- Настраиваем безопасный /goal: шаблоны условий, стоп-клаузы, headless-прогоны с логированием в аудит
- Отделяем данные: тестовые контуры, обезличенные выгрузки, запрет ПД и коммерческой тайны в контексте
- Обучаем инженеров-диспетчеров, строим дежурство по очереди Needs input, сопровождаем по SLA

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Manage multiple agents with agent view (code.claude.com — 2026)

02 Keep Claude working toward a goal (/goal) (code.claude.com — 2026)

03 Run parallel sessions with worktrees (code.claude.com — 2026)

04 Hooks reference: Stop, PreToolUse, Notification (code.claude.com — 2026)

05 Permissions and managed settings (code.claude.com — 2026)

06 Model configuration (small fast model) (code.claude.com — 2026)

07 Шаблон ITfresh: политика прав ИИ-агентов (наш стандарт — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.