Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

5 правил Claude Code в проде: что не пускаем без присмотра

TDD, память проекта, регрессия, ревью и границы — всё закреплено в settings.json и hooks

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Claude Code ускоряет типовые задачи клиентов в 2–3 раза, но без жёсткого контура за две недели превращает репозиторий в кашу: дубли функций, «зелёные» тесты, которые ничего не проверяют, секреты в контексте сессии. Мы закрепляем пять наших правил не памяткой, а конфигурацией: permissions-deny, hooks и OS-песочница делают нарушение физически невозможным.

Почему это важно бизнесу

- Разбор «каши» после месяца бесконтрольного vibe coding стоит дороже разработки: до 3 дней инженера уходит на чистку дублей
- Утечка боевых паролей или ПДн через контекст ассистента — риск по 152-ФЗ и репутационный удар по клиенту
- Регрессия в проде (парсер прайсов, интеграция 1С) ломает ежедневные процессы бухгалтерии и закупок
- Контур из settings.json и hooks ставится за 1 день и дальше работает без ручного надзора за каждой командой

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.1.x

актуальная ветка Claude Code, на которой мы стандартизируем контуры: sandbox, hooks, auto-режи…

Claude Code CLI reference, code.claude.com

deny

класс правил permissions: Read(./.env), Read(./secrets/**) — секреты закрыты даже от чтения

Claude Code Settings

exit 2

код PreToolUse-хука, блокирующий правку tests/ и миграций; stderr уходит модели как причина

Claude Code Hooks reference

0

доменов в сетевом allowlist песочницы по умолчанию — каждый новый хост только через одобрение

Claude Code Sandboxing

--bare

режим headless-CI: без чужих hooks, plugins, MCP и auto-memory — одинаковый результат на любой…

Claude Code CLI reference

12

golden-файлов в регрессии типового парсера: по 3 на поставщика, эталонный JSON для каждого

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Контур репозитория: settings.json, CLAUDE.md и запреты

Что настраиваем

Каждый клиентский репозиторий + рабочие станции инженеров ITfresh с managed-политиками

Как мы это делаем

1. Кладём .claude/settings.json в git: permissions.deny — Read(./.env), Read(./secrets/**), Edit(tests/**), Bash(curl *)
2. На станции инженеров — /etc/claude-code/managed-settings.json: эти правила нельзя переопределить ни user-, ни project-скоупом
3. CLAUDE.md в корне: архитектура, «НИКОГДА не менять tests/», история инцидентов с решениями; рядом docs/architecture.md и docs/decisions_log.md
4. cleanupPeriodDays: 20 — сессии с контекстом клиента не живут на диске дольше необходимого (дефолт 30 дней)

РЕЗУЛЬТАТ

Пять правил перестают зависеть от дисциплины конкретного инженера: запреты применяются раньше любого действия модели, а новая сессия стартует с полным контекстом проекта вместо изобретения архитектуры заново — встраивание в существующий код занимает минуты.

КЛЮЧЕВОЙ НЮАНС

Precedence важен: settings.local.json перекрывает проектный файл, поэтому всё критичное для безопасности мы поднимаем в managed-скоуп — его не переопределяет никто и ничто.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Хуки-контролёры: PreToolUse и автотесты после каждой правки

Что настраиваем

Все репозитории с прод-кодом: парсеры прайсов, интеграции 1С, telegram-боты

Как мы это делаем

1. PreToolUse с matcher Edit|Write: скрипт сверяет путь с tests/ и migrations/, при попадании — exit 2, причина из stderr возвращается модели
2. PostToolUse на Edit|Write: ruff --fix + mypy --strict + pytest -x — типовой брак разворачивается до коммита
3. Stop-хук гоняет регрессию по golden-набору: эталонные xlsx против tests/fixtures/regression/golden/*.json
4. Хуки лежат в ${CLAUDE_PROJECT_DIR}/.claude/hooks/; allowedHttpHookUrls: [] — сторонние HTTP-хуки заблокированы

РЕЗУЛЬТАТ

Формальный брак (типизация, стиль, упавшие юниты) снимается автоматикой до коммита; человек на ревью тратит 1–2 часа в неделю на смысл, а не на косметику, и ни один PR не уходит клиенту с упавшей golden-регрессией.

КЛЮЧЕВОЙ НЮАНС

Блокировать умеют только события «до действия»: PreToolUse останавливает вызов, а PostToolUse уже ничего не отменит. Поэтому запреты ставим до инструмента, проверки качества — после.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Песочница и headless-CI: автономность без bypass-режима

Что настраиваем

Linux-станции инженеров (Ubuntu 24.04) и GitHub Actions клиентских репозиториев

Как мы это делаем

1. sandbox.enabled: true; на Ubuntu 24.04 — AppArmor-профиль для bwrap (userns), пакеты bubblewrap и socat
2. network.allowedDomains: только pypi.org, files.pythonhosted.org, github.com; strictAllowlist: true в user/managed-скоупе — хост вне списка получает deny без промпта
3. sandbox.credentials: deny на ~/.ssh и ~/.aws/credentials, envVars deny для токенов — реальные секреты командам не видны
4. allowUnsandboxedCommands: false — escape hatch dangerouslyDisableSandbox игнорируется полностью
5. CI: claude --bare -p c --allowedTools "Read,Edit,Bash(git diff *)" и --output-format json; ключ — из секретов CI

РЕЗУЛЬТАТ

Автономные прогоны без --dangerously-skip-permissions: границу держит ОС (Seatbelt на macOS, bubblewrap на Linux), а не осторожность модели. Даже при скомпрометированной команде из песочницы не читается ~/.ssh и нет выхода за allowlist доменов.

КЛЮЧЕВОЙ НЮАНС

Прокси песочницы принимает решение по hostname и по умолчанию не инспектирует TLS: широкий allowlist вроде *.github.com — потенциальный канал утечки, список держим минимальным.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Ассистент правит собственные тесты**
  Круговая порука: код проходит тесты, которые сам и сочинил. Deny на Edit(tests/**) плюс PreToolUse-хук с exit 2 — правка невозможна физически

- ✕ **--dangerously-skip-permissions в проде**
  Флаг снимает даже проверки защищённых путей. Заменяем связкой sandbox auto-allow + deny-правила: автономность та же, границу держит ОС

- ✕ **Секреты в контексте сессии**
  Пароли и дампы оседают в транскриптах сессий. Deny на .env и secrets/**, sandbox.credentials для ~/.ssh, CLAUDE_CODE_SUBPROCESS_ENV_SCRUB для subproc…

- ✕ **settings.local.json сильнее проектного**
  Локальный файл инженера тихо перекрывает командные правила репозитория. Критичные запреты выносим в managed-settings — их не переопределить

- ✕ **Зелёные юнит-тесты ≠ рабочий pipeline**
  Модель охотно проходит юниты, ломая интеграцию. Ловим golden-регрессией: эталонный JSON на реальных файлах клиента, расхождение видно в diff

- ✕ **Дубли функций в каждой сессии**
  Без CLAUDE.md сессия пишет 11-ю parse_price_* вместо существующей. Раздел «Архитектура» в памяти проекта + ревью полного diff закрывают дубли

- ✕ **Широкий сетевой allowlist**
  Прокси песочницы не инспектирует TLS — возможен фронтинг через разрешённый хост. Минимум доменов и strictAllowlist вместо интерактивных промптов

- ✕ **Задача без явных границ «нельзя»**
  «Улучши парсер» кончается переименованием модулей и новыми зависимостями. Шаблон задачи с блоком «что НЕЛЬЗЯ» и ожидаемым составом коммита

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- CLAUDE.md в корне каждого репо: архитектура, жёсткие правила, история инцидентов
- permissions.deny на .env, secrets/**, tests/** в .claude/settings.json
- Ревью diff каждой строки человеком до отправки клиенту

НОРМАЛЬНО

- PreToolUse-хуки на Edit|Write: запрет tests/ и миграций через exit 2
- pre-commit: ruff + mypy --strict + pytest -x до любого коммита
- Golden-регрессия в CI: эталонные входы + зафиксированный JSON-результат
- Шаблон постановки задачи с явным блоком «что НЕЛЬЗЯ делать»

ХОРОШО

- Песочница: sandbox.enabled + узкий network.allowedDomains + credentials deny
- managed-settings.json на станциях инженеров: deny не переопределяется
- Headless CI: claude --bare -p, --allowedTools точечно, отчёт в JSON
- OTEL-телеметрия: CLAUDE_CODE_ENABLE_TELEMETRY=1 → свой OTLP-коллектор

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Есть ли CLAUDE.md с архитектурой и историей инцидентов в каждом клиентском репозитории?
- [ ] Закрыты ли .env и secrets/** deny-правилами, а не устной договорённостью?
- [ ] Может ли ассистент физически изменить файлы tests/ — или это блокирует PreToolUse-хук?
- [ ] Прогоняется ли golden-регрессия на каждом PR до показа результата клиенту?
- [ ] Читает ли человек diff каждой строки перед мержем в прод?
- [ ] Идут ли автономные прогоны в песочнице, а не под --dangerously-skip-permissions?
- [ ] Ограничен ли сетевой доступ Bash-команд явным списком доменов?
- [ ] Вынесены ли критичные запреты в managed-settings, недоступные для переопределения?
- [ ] Есть ли шаблон задачи с блоком «что НЕЛЬЗЯ» и ожидаемым составом коммита?
- [ ] Есть ли decisions_log: понятно ли через год, зачем в коде «странный if»?

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 Claude Code под ключ: settings.json, hooks, песочница, managed-политики на станциях
- Пишем CLAUDE.md, architecture.md и decisions_log под ваш проект и держим их актуальными
- Строим CI с golden-регрессией и headless-прогонами: GitHub Actions, отчёты в JSON
- Разработка типовых задач (парсеры, отчёты, боты, интеграции 1С) на абонентке с ревью каждой строки
- Аудит кода после «vibe coding»: чистка дублей, честные тесты, регрессионный набор

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 Claude Code Settings: файлы, precedence, permissions (code.claude.com — 2026)

02 Claude Code Hooks reference: события, exit-коды, matchers (code.claude.com — 2026)

03 Claude Code Sandboxing: Seatbelt/bubblewrap, сетевой прокси (code.claude.com — 2026)

04 Claude Code CLI reference: --print, --bare, --allowedTools (code.claude.com — 2026)

05 Claude Code Memory: CLAUDE.md и иерархия памяти (code.claude.com — 2026)

06 Шаблон ITfresh: CLAUDE.md + decisions_log + задача с «нельзя» (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.