Как сохранить контекст AI-проекта и не зависеть от одного аккаунта

Полное руководство по созданию локальной системы AI-суверенитета: проект продолжит жить, даже если история чата или аккаунт станут недоступны.

Часть 1. Создайте локальную папку проекта

Шаг 1. Определите назначение папки

Выберите подходящий тип организации:

- Отдельная папка для конкретного продукта или проекта
- Общая рабочая папка для задач, не относящихся к одному проекту

Для каждого большого проекта лучше использовать отдельную папку с собственной документацией и историей.

Шаг 2. Откройте папку в AI-агенте

- **Откройте агент**

  Запустите Codex или Cowork с подключённым AI-агентом

- **Создайте проект**

  Перейдите в раздел проектов и выберите создание нового

- **Укажите папку**

  Выберите уже созданную локальную папку

⚠ Не добавляйте постоянные инструкции только в поле проекта или в чат. Они должны храниться внутри файлов папки.

Часть 2. Система локального хранения контекста

Вставьте в AI-агента основной промпт ниже.

Ты работаешь в локальной папке проекта через Codex, Cowork или IDE-агента. Создай с нуля минимальную, но ПОЛНУЮ систему локального AI-суверенитета: проект должен продолжить жить, даже если история чата или AI-аккаунт станут недоступны.

=== ПРИНЦИПЫ ===

- Модель может быть в облаке. Память проекта (решения, логи, файлы, код, handoff) должна лежать локально на диске и в Git.
- Источник истины — файлы в папке, а не история чата. Если что-то важно — запиши в файл проекта.
- Если правило должно работать всегда, оно живёт в файле инструкций (AGENTS.md / CLAUDE.md), а не в одном длинном промте.
- Не сохраняй секреты (API-ключи, токены, seed phrases, .env) в инструкциях, логах или промтах.
- Перед публикацией, отправкой, деплоем и удалением — всегда спрашивай.
- Не удаляй файлы без явного подтверждения пользователя.
- Не экономь токены. Если не влазит в одно сообщение — раздели на несколько.
- Если из-за ограничений ты что-то не можешь прочитать/создать — скажи об этом прямо и предложи решение (разбивка, zip, csv и т. п.).

=== ШАГ 1. СТРУКТУРА ПАПКИ === Создай следующее дерево (имя корня можно заменить под проект):

ai-sovereign-workspace/ AGENTS.md CLAUDE.md README.md docs/ PROJECT_CONTEXT.md DECISIONS.md CHANGELOG.md RECOVERY.md logs/ sessions/ YYYY-MM-DD-initial-session.md actions.md prompts/ start-session.md work-session.md end-session.md recovery.md export-handoff.md audit-logging.md src/ outputs/ .gitignore

Команды инициализации: mkdir ai-sovereign-workspace cd ai-sovereign-workspace git init mkdir -p docs logs/sessions prompts src outputs

=== ШАГ 2. СОЗДАЙ ФАЙЛЫ С ТОЧНО ТАКИМ СОДЕРЖИМЫМ (заменяя плейсхолдеры <...>) ===

----- FILE: AGENTS.md -----

AGENTS.md - Project Instructions

This file is the source of truth for AI coding agents working in this project. Read it before making changes. If a user request conflicts with this file, stop and ask.

Project

Name: Goal: Audience: Current phase: <idea / prototype / production / maintenance>

Source of Truth

- Project context: docs/PROJECT_CONTEXT.md
- Decisions: docs/DECISIONS.md
- Change history: docs/CHANGELOG.md
- Session logs: logs/sessions/
- Recovery instructions: docs/RECOVERY.md

Do not rely on chat history as the only memory of the project. If something matters, write it to a project file.

Working Rules

- Before work: read this file, docs/PROJECT_CONTEXT.md, latest session log, and docs/DECISIONS.md.
- During work: keep changes scoped to the current task.
- After work: update session log, changelog, and decisions if relevant.
- Never delete files without explicit user approval.
- Never expose secrets, tokens, API keys, private credentials, or .env values.
- Ask before publishing, sending, deploying, deleting, or sharing anything externally.

Logging Rules
Every completed task must leave a local trace.

Update logs/sessions/YYYY-MM-DD-.md with:

- user request;
- files inspected;
- actions performed;
- commands run;
- files created or changed;
- decisions made;
- open questions;
- next steps.

Update docs/DECISIONS.md when:

- architecture changes;
- tool choice changes;
- file structure changes;
- a tradeoff is accepted;
- a risky shortcut is chosen.

Update docs/CHANGELOG.md when:

- user-visible behavior changes;
- files are created;
- configuration changes;
- prompts or workflows change.

File Zones

- src/ - source code.
- docs/ - project memory and documentation.
- logs/ - session and action logs.
- prompts/ - reusable prompts.
- outputs/ - generated deliverables.

Report Format

At the end of each task, report:

1. What was done.
2. Which files changed.
3. Which checks passed.
4. What still needs attention.
5. What the next agent should do.

----- FILE: CLAUDE.md -----

CLAUDE.md - Claude Project Instructions

Follow AGENTS.md as the canonical project instruction file.

Additional Claude-specific rules:

- Treat this project as a local working system, not a one-off chat.
- Write durable project memory to files in docs/ and logs/.
- Do not assume chat history will be available in the future.
- Before ending a session, produce a handoff in logs/sessions/.
- If you cannot write files, return the exact Markdown that the user should save.

Start every session by reading:

1. AGENTS.md
2. docs/PROJECT_CONTEXT.md
3. latest file in logs/sessions/
4. docs/DECISIONS.md

End every session by updating:

1. a session log
2. docs/CHANGELOG.md
3. docs/DECISIONS.md, if decisions changed

----- FILE: README.md -----

Local AI-sovereign workspace. The model can be in the cloud, but the project memory lives here.

Start here:

1. Read AGENTS.md (canonical instructions for agents).
2. Read docs/PROJECT_CONTEXT.md (what this project is).
3. Read docs/RECOVERY.md to restore context with a new agent/account.

Daily flow: prompts/start-session.md -> work -> prompts/end-session.md -> git commit.

----- FILE: docs/PROJECT_CONTEXT.md -----

Project Context

What This Project Is

<One paragraph. What are we building?>

Why It Exists

<What problem does it solve?>

Current State

- Phase:
- Working version:
- Main branch:
- Last important change:

Users / Audience

<Who uses this?>

Constraints

- Budget:
- Deadline:
- Stack:
- Legal/security:
- Things we will not do:

Important Paths

- Source:
- Outputs:
- Logs:
- Prompts:
- Docs:

Definition of Done

A task is done only when:

- files are updated;
- checks are run or skipped with reason;
- session log is updated;
- next step is clear.

----- FILE: docs/DECISIONS.md -----

Decisions

Use this file for durable project decisions.

Template

YYYY-MM-DD -

Status: proposed | accepted | rejected | replaced

Context: <What forced the decision?>

Decision: <What did we decide?>

Why: <Why this option?>

Consequences: <What becomes easier/harder?>

Rollback: <How to undo this if needed?>

----- FILE: docs/CHANGELOG.md -----

Changelog

YYYY-MM-DD

Added

- ...

Changed

- ...

Fixed

- ...

Notes

- ...

----- FILE: docs/RECOVERY.md -----

Recovery Guide

Use this file when opening the project with a new AI account, new agent, or after losing chat history.

Recovery Order

1. Read AGENTS.md.
2. Read docs/PROJECT_CONTEXT.md.
3. Read docs/DECISIONS.md.
4. Read the latest 3 files in logs/sessions/.
5. Run git status.
6. Summarize current state before changing anything.

What To Reconstruct

- What the project is.
- What has been done.
- Which decisions matter.
- Which files are source of truth.
- What is risky.
- What the next step should be.

Recovery Prompt

Paste this into a new agent:

Read AGENTS.md, docs/PROJECT_CONTEXT.md, docs/DECISIONS.md, and the latest session logs in logs/sessions/. Then reconstruct:

1. current project state;
2. last completed work;
3. open risks;
4. next recommended step. Do not edit files until you finish the reconstruction.

----- FILE: logs/actions.md -----

Action Log

YYYY-MM-DD

Task

<What was requested?>

Files inspected

- ...

Commands run

- ...

Files changed

- ...

Checks

- ...

Notes

- ...

----- FILE: logs/sessions/YYYY-MM-DD-initial-session.md -----

Session Log - YYYY-MM-DD

Request

Context Loaded

- AGENTS.md
- docs/PROJECT_CONTEXT.md
- ...

Work Done

- ...

Decisions

- ...

Files Changed

- ...

Commands Run

- ...

Checks

- ...

Open Questions

- ...

Next Step

----- FILE: .gitignore ----- .env .env.* secrets/ *.key *.pem node_modules/ pycache/ .DS_Store tmp/

----- FILE: prompts/start-session.md ----- Начни новую рабочую сессию в этом проекте.

Сначала прочитай:

1. AGENTS.md
2. CLAUDE.md, если есть
3. docs/PROJECT_CONTEXT.md
4. docs/DECISIONS.md
5. последний session log из logs/sessions/

Затем ответь кратко:

- что это за проект;
- в каком он состоянии;
- какой последний выполненный шаг;
- какие правила особенно важны;
- какой первый безопасный следующий шаг.

Пока ничего не меняй в файлах.

----- FILE: prompts/work-session.md ----- Выполни задачу: <описание задачи>.

Правила:

- работай внутри текущего проекта;
- не опирайся только на историю чата;
- если принимаешь важное решение, обнови docs/DECISIONS.md;
- если меняешь проект, обнови docs/CHANGELOG.md;
- в конце создай или обнови session log в logs/sessions/;
- не удаляй файлы без явного подтверждения;
- не публикуй, не отправляй и не деплой без отдельного подтверждения.

В финале дай:

1. что сделал;
2. какие файлы изменил;
3. какие проверки прошли;
4. что осталось;
5. первый следующий шаг.

----- FILE: prompts/end-session.md ----- Закрой текущую сессию.

Обнови:

- logs/sessions/YYYY-MM-DD-.md
- logs/actions.md
- docs/CHANGELOG.md
- docs/DECISIONS.md, если были решения
- docs/RECOVERY.md, если изменился способ восстановления проекта

Session log должен содержать:

- исходную задачу;
- что было прочитано;
- что сделано;
- какие команды запускались;
- какие файлы изменены;
- какие решения приняты;
- что не сделано;
- следующий шаг для нового агента.

После обновления файлов дай краткий handoff для человека.

----- FILE: prompts/recovery.md ----- Я открываю этот проект после потери старого AI-чата или аккаунта.

Восстанови контекст только по локальным файлам.

Прочитай:

- AGENTS.md
- CLAUDE.md, если есть
- docs/PROJECT_CONTEXT.md
- docs/DECISIONS.md
- docs/CHANGELOG.md
- docs/RECOVERY.md
- 3 последних файла в logs/sessions/

Затем составь:

1. краткую карту проекта;
2. последнее известное состояние;
3. список важных решений;
4. список изменённых файлов;
5. риски;
6. следующий безопасный шаг.

Ничего не меняй, пока я не подтвержу план.

----- FILE: prompts/export-handoff.md ----- Сожми эту переписку в локальный handoff для проекта.

Формат Markdown:

- цель сессии;
- исходный запрос;
- ключевые решения;
- что было сделано;
- какие артефакты созданы;
- какие промты сработали;
- ошибки и ограничения;
- следующий шаг;
- что нужно записать в docs/DECISIONS.md;
- что нужно записать в docs/CHANGELOG.md;
- что нужно записать в logs/sessions/YYYY-MM-DD-.md.

Не переписывай всю переписку. Сохрани только то, что нужно, чтобы новый агент продолжил работу без старого чата.

----- FILE: prompts/audit-logging.md ----- Проверь, можно ли восстановить этот проект без истории чата.

Прочитай AGENTS.md, docs/PROJECT_CONTEXT.md, docs/DECISIONS.md, docs/CHANGELOG.md и logs/sessions/.

Оцени по шкале 0-10:

- понятна ли цель проекта;
- понятен ли текущий статус;
- видны ли последние решения;
- видны ли последние изменения;
- понятен ли следующий шаг;
- достаточно ли правил безопасности;
- можно ли передать проект новому агенту.

Дай список пробелов и конкретные правки в Markdown-файлы. Не редактируй файлы без подтверждения.

=== ШАГ 3. GIT === После создания файлов сделай первый коммит: git add . git commit -m "init ai sovereign workspace"

После каждой сессии: git status git diff git add . git commit -m "log ai session and update project state"

=== ШАГ 4. (опционально) Codex config === Если нужны кастомные имена fallback-инструкций, создай ~/.codex/config.toml: project_doc_fallback_filenames = ["CLAUDE.md", "PROJECT.md", ".agents.md"] project_doc_max_bytes = 65536

=== RECOVERY DRILL (проверка системы) === После создания структуры предложи пользователю тест:

1. Закрой текущий чат.
2. Открой новый агент в той же папке.
3. Дай ему prompts/recovery.md.
4. Проверь, может ли он ответить: что за проект; что сделано; какие решения; какие файлы важны; что дальше. Если новый агент не восстанавливается за 5 минут — система логирования не работает, её надо усилить.

=== ФИНАЛЬНЫЙ ОТЧЁТ === После создания всех файлов дай краткий отчёт:

1. Что создано (дерево).
2. Зачем нужен каждый ключевой файл.
3. Что делать дальше (открыть в Codex/Cowork/IDE, запустить start-session).
4. Напоминание: модель можно заменить, систему — нет.

=== ВАЖНЫЕ ОГОВОРКИ ===

- Не сохраняй API-ключи, токены, seed phrases и приватные данные в AGENTS.md, CLAUDE.md, логах или промтах.
- Если проект коммерческий, не выкладывай локальный репозиторий в публичный GitHub.
- Логи не обязаны хранить полные диалоги слово в слово. Важнее решения, действия и следующий шаг.
- Для чувствительных проектов добавь шифрование диска и приватный backup.
- Перед публикацией, отправкой, деплоем и удалением агент всегда должен спрашивать.

Часть 3. От структуры к Git и GitHub

1 Проверьте структуру папки. Убедитесь, что все файлы находятся непосредственно в открытой директории, а не во вложенной папке.

2 Заполните плейсхолдеры: название проекта, цель, аудитория, этап, описание, ограничения, текущее состояние и следующий шаг.

3 Проверьте логирование сессий. Отправьте агенту реальную задачу и убедитесь, что он обновил все нужные файлы.

4 Настройте Git и приватный репозиторий GitHub через GitHub CLI. Агент выполнит всё автоматически.

Запрос для настройки Git и GitHub

Пожалуйста, самостоятельно с помощью GitHub CLI проверь, настроен ли Git в нашем проекте, и настрой его.

Создай приватный репозиторий на GitHub, который будет сохранять мои проекты после каждой рабочей сессии.

Если я не авторизован в GitHub, открой мне окно авторизации или создания аккаунта, чтобы я мог войти или зарегистрироваться.

Если Git или GitHub CLI не установлены, установи необходимые инструменты либо покажи точные действия, которые мне нужно выполнить вручную.

Не создавай публичный репозиторий.

Не добавляй в GitHub API-ключи, токены, пароли, файлы .env и другие секретные данные. Перед удалением, перезаписью, публикацией или отправкой данных обязательно запроси моё подтверждение.

После настройки проверь:

1. Git инициализирован в текущей папке.
2. Создан первый commit.
3. Создан приватный репозиторий GitHub.
4. Локальная папка подключена к удалённому репозиторию.
5. Файлы отправлены в GitHub.
6. Секретные файлы исключены через .gitignore.
7. После каждой завершённой сессии изменения можно сохранять в Git и отправлять в GitHub.

Часть 4. Рабочий процесс с промптами

Готовые промпты обеспечивают стандартный цикл работы: каждая сессия начинается с восстановления контекста и заканчивается сохранением состояния.

Начало сессии

`prompts/start-session.md`

Агент читает инструкции, изучает контекст, смотрит решения и последний лог. Ничего не меняет до начала задачи.

Рабочая задача

`prompts/work-session.md`

Добавьте конкретную задачу внутрь файла. Агент в рамках проекта обновляет DECISIONS.md и CHANGELOG.md.

Завершение сессии

`prompts/end-session.md`

Агент обновляет лог сессии, журнал действий, историю изменений и инструкцию восстановления.

Сохранение в Git

После каждой сессии — commit и push в приватный репозиторий GitHub.

Каждая рабочая сессия должна оставлять след в локальной папке: обновлённый session log, CHANGELOG.md и при необходимости DECISIONS.md.

Часть 5. Recovery Drill и восстановление версий

Как провести Recovery Drill

01

Завершите сессию

Убедитесь, что агент обновил session log и сохранил изменения в Git

02

Закройте чат

Отправьте изменения в GitHub и закройте текущий чат

03

Новый агент

Откройте новый чат или другого AI-агента

04

Передайте recovery.md

Дайте агенту содержимое `prompts/recovery.md`

Что должен определить новый агент

Проект

Что это и для кого

Состояние

Текущий статус и последняя сессия

Решения

Принятые архитектурные решения

Риски

Открытые вопросы и опасные места

Следующий шаг

Первое безопасное действие

Безопасность и финальный чек-лист

Что нельзя хранить в проекте и GitHub

🔑 Ключи и токены

API-ключи, токены доступа, приватные ключи, seed phrases

🔒 Учётные данные

Пароли, содержимое .env, платёжные данные

👤 Персональные данные

Персональные и конфиденциальные данные клиентов без необходимости

⊗ **Перед публикацией, отправкой, деплоем, удалением или передачей данных агент обязан запросить подтверждение.**

Чек-лист за 30 минут

Отметьте каждый пункт — и проект переживёт любой пропавший аккаунт.

- [ ] Создана папка проекта
- [ ] Инициализирован Git
- [ ] Созданы AGENTS.md и CLAUDE.md
- [ ] Создан docs/PROJECT_CONTEXT.md
- [ ] Секреты не попадают в логи
- [ ] Есть первый commit
- [ ] Созданы DECISIONS.md, CHANGELOG.md и RECOVERY.md
- [ ] Создана папка logs/sessions/
- [ ] Созданы промпты start-session, end-session и recovery
- [ ] Агент умеет обновлять session log
- [ ] Проведён recovery drill в новом чате
- [ ] Handoff пишется в конце каждой сессии

> **Главный принцип:** модель и аккаунт можно заменить, но память проекта должна оставаться в локальных файлах и истории Git.