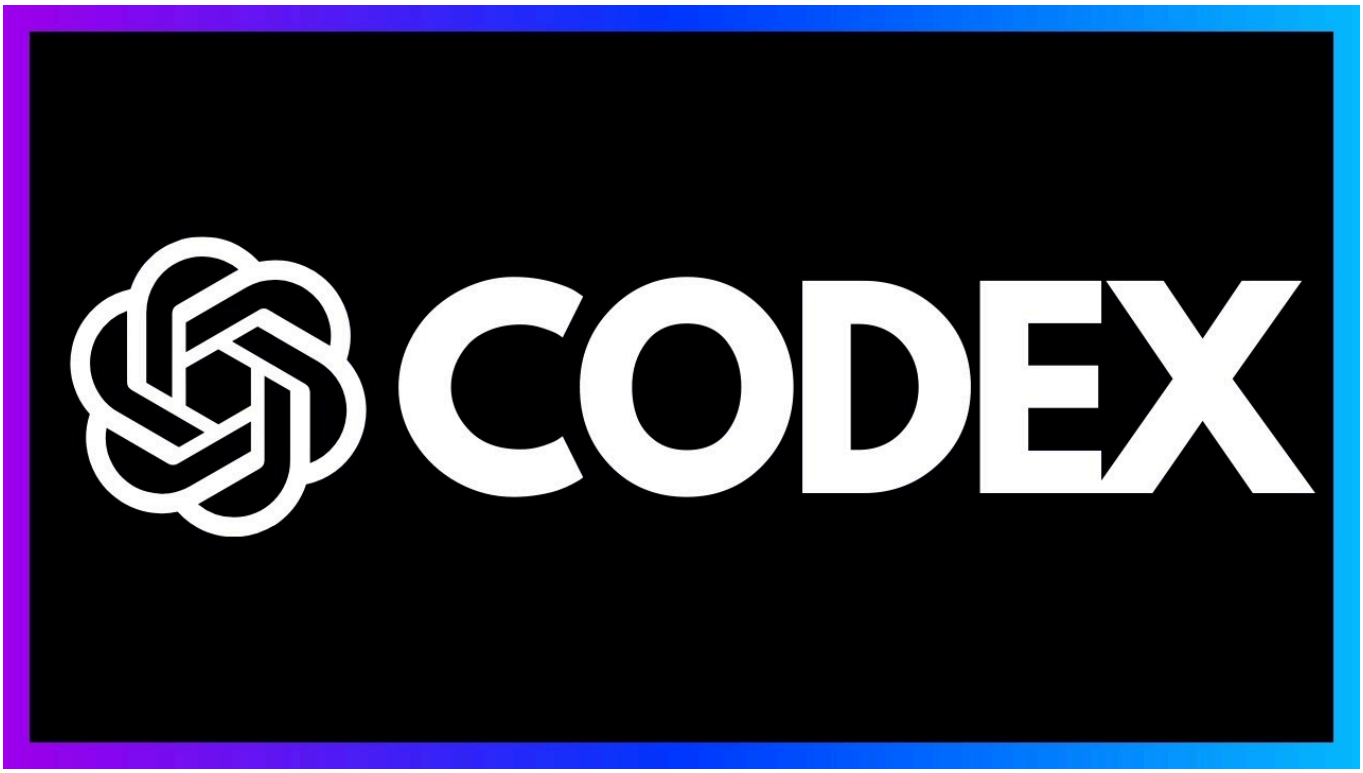
Как_взломать_стоимость_Codex

Как заставить Codex работать на другой модели и не сливать деньги на дорогие лимиты



По умолчанию Codex использует стандартную модель и стандартного провайдера. Но мы можем указать другого провайдера, например **OpenRouter**, и через него запускать более дешёвую модель, например **DeepSeek**.

Схема такая:

```
Codex
  ↓
OpenRouter
  ↓
DeepSeek / Qwen / Kimi / Mistral / другая модель
```

Что понадобится

Нужно:

```
1. Установленный Codex
2. Аккаунт OpenRouter --> https://openrouter.ai/workspaces/default/keys
3. API-ключ OpenRouter --> https://openrouter.ai/workspaces/default/keys
4. Найти и изменить файл config.toml (находится тут ~/.codex/config.toml)
```

Codex хранит пользовательские настройки здесь:

```
~/.codex/config.toml
```

Это основной пользовательский конфиг Codex. Именно в нём указывается модель, провайдер и параметры подключения.

1 Открываем config.toml

В терминале вводим:

```
#Это команда открывает в терминале файл по пути ~/.codex/config.toml.
sudo nano ~/.codex/config.toml
```

Если используешь VS Code, можно открыть так:

```
code ~/.codex/config.toml
```

Теперь в начало файла нужно добавить настройки модели и провайдера.

2 Вставляем строки

Вставляем эти строки в самом вверху:

```
model = "deepseek/deepseek-v4-flash"
model_provider = "openrouter"
model_reasoning_effort = "xhigh"

[model_providers.openrouter]
name = "openrouter"
base_url = "https://openrouter.ai/api/v1"
env_key = "OPENROUTER_API_KEY"
wire_api = "responses"
```

Если там есть строки с прошлыми значениями, у примеру:

```
model = "gpt-5.5"
model_reasoning_effort = "xhigh"
```

То просто замени их на наши строки. (УДАЛИ/ЗАМЕНИ).

Если не понял че надо 😂 - удаляй содержимое всего файла ~/.codex/config.toml и вставь это:

```
model = "deepseek/deepseek-v4-flash"
model_provider = "openrouter"
model_reasoning_effort = "xhigh"

[model_providers.openrouter]
name = "openrouter"
base_url = "https://openrouter.ai/api/v1"
env_key = "OPENROUTER_API_KEY"
wire_api = "responses"
```

после сохрани и выйди.

Объясняю что значит каждая строка (просто ознакомься)

```
model = "deepseek/deepseek-v4-flash"
```

Это модель, которую будет использовать Codex.
В данном случае мы говорим: используй DeepSeek-модель через OpenRouter.

```
model_provider = "openrouter"
```

Это имя провайдера.
Codex смотрит на эту строку и ищет ниже блок:

```
[model_providers.openrouter]
```

То есть название должно совпадать:

```
model_provider = "openrouter"
        ↓
[model_providers.openrouter]
```

```
name = "openrouter"
```

Это человекочитаемое имя провайдера. Можно оставить так.

```
base_url = "https://openrouter.ai/api/v1"
```

Это адрес API OpenRouter.
Codex будет отправлять запросы модели через этот URL.

```
env_key = "OPENROUTER_API_KEY"
```

Это не сам ключ.
Это название переменной окружения, из которой Codex возьмёт ключ.

То есть в `config.toml` мы не пишем секретный ключ напрямую. Мы пишем только имя переменной:

```
env_key = "OPENROUTER_API_KEY"
```

А настоящий ключ передаём отдельно через терминал.

```
wire_api = "responses"
```

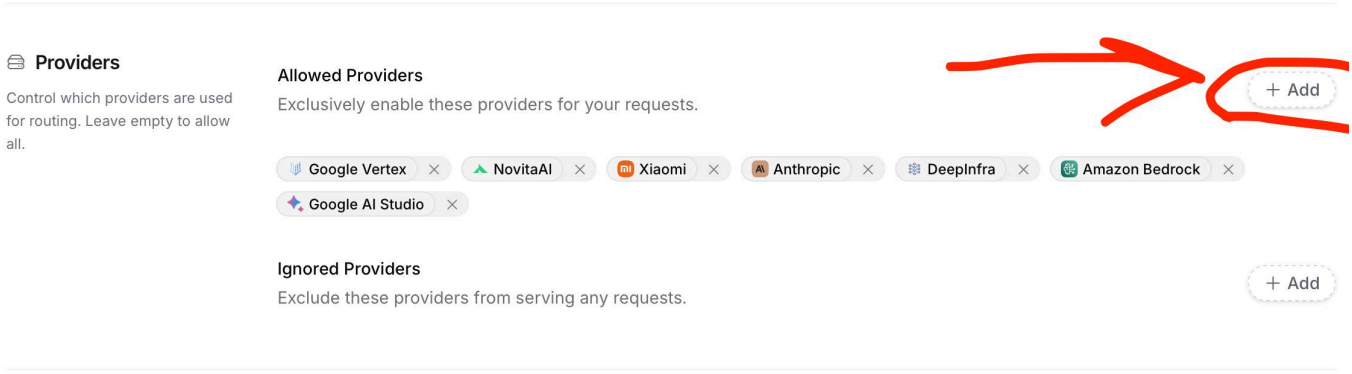
Это формат API, через который Codex общается с провайдером. Для custom model provider в Codex используется Responses API.

3 Добавляем API-ключ в терминал (чтобы codex подхватил)

После сохранения `config.toml` нужно открыть терминал для запуска CODEX (как обычно) и сразу ввести эту строчку кода:

```
export OPENROUTER_API_KEY="sk-or-ТВОЙ КЛЮЧИ API КОТОРЫЙ ВЫПУСТИШЬ ИЗ
https://openrouter.ai/workspaces/default/keys"
```

Добавлю - на сайте https://openrouter.ai/workspaces/default/keys обязательно разреши провайдера (как на картинке снизу)



После этого запускаем Codex:

```
codex
```

Если запускаешь Codex внутри проекта, сначала перейди в папку проекта:

```
cd /путь/к/твоему/проекту
codex
```

Например:

```
cd ~/projects/my-app
codex
```

Теперь Codex будет работать в этой папке, но запросы к модели будут идти через OpenRouter.

Что произошло после настройки

До настройки схема была такая:

```
Codex
  ↓
дефолтный провайдер
  ↓
дефолтная модель
```

После настройки:

```
Codex
  ↓
OpenRouter
  ↓
deepseek/deepseek-v4-flash
```

То есть Codex остался тем же агентом.
Изменился только провайдер и модель.

Почему это дешевле

Стоимость работы Codex зависит от модели, которая обрабатывает твои запросы.

У разных моделей разная цена за input tokens и output tokens. Дефолтные сильные модели обычно стоят дороже, потому что они мощнее, тяжелее и рассчитаны на более сложные задачи.

Но не для каждой задачи нужна дорогая модель.

Например:

```
исправить README
написать простой тест
```

```
поправить компонент
найти очевидный баг
объяснить структуру проекта
сделать небольшую правку
```

Для таких задач часто достаточно более дешёвой модели.

OpenRouter позволяет выбрать конкретную модель и платить по её цене. Например, DeepSeek Flash обычно стоит сильно дешевле премиальных моделей, поэтому одна и та же работа агента может обходиться заметно дешевле. У OpenRouter в каталоге DeepSeek V4 Flash указан как бюджетная модель с низкой стоимостью за 1M input/output tokens. (OpenRouter)

Идея такая:

```
дорогая модель — для сложной архитектуры и тяжёлых задач
дешёвая модель — для рутины, правок, тестов и простых багов
```

Так можно не тратить дорогие токены там, где они не нужны.

Как сделать отдельный профиль DeepSeek (ДЛЯ ПРОФИ)

Если не хочется менять основной конфиг каждый раз, можно сделать отдельный профиль.

Например:

```
обычный запуск:
codex

запуск через DeepSeek-профиль:
codex --profile deepseek
```

Codex поддерживает profile-файлы формата:

```
~/.codex/profile-name.config.toml
```

При запуске с `--profile deepseek` Codex сначала читает базовый `~/.codex/config.toml`, а потом накладывает сверху `~/.codex/deepseek.config.toml`. (OpenAI Разработчики)

Базовый config.toml

В основном файле:

```
nano ~/.codex/config.toml
```

оставляем провайдера:

```
[model_providers.openrouter]
name = "openrouter"
base_url = "https://openrouter.ai/api/v1"
env_key = "OPENROUTER_API_KEY"
wire_api = "responses"
```

Создаём профиль

Открываем файл профиля:

```
nano ~/.codex/deepseek.config.toml
```

Вставляем:

```
model = "deepseek/deepseek-v4-flash"
model_provider = "openrouter"
model_reasoning_effort = "high"
```

Теперь запуск:

```
codex --profile deepseek
```

Или коротко:

```
codex -p deepseek
```

Смысл профиля простой:
не нужно каждый раз руками менять глобальный `config.toml`. Можно иметь несколько режимов:

```
codex
codex --profile deepseek
codex --profile strong
codex --profile cheap
```

И переключаться между ними под задачу.

Итоговая схема

```
1. Открываем ~/.codex/config.toml
2. Добавляем OpenRouter как model provider
3. Указываем модель DeepSeek
4. В новом терминале экспортируем OPENROUTER_API_KEY
```

```
5. Запускаем codex
6. При необходимости создаём отдельный профиль deepseek
```

Главная мысль:

```
Codex — это агент.
OpenRouter — это провайдер.
DeepSeek — это модель.
config.toml — это место, где мы связываем всё вместе.
```

Так можно оставить удобство Codex, но управлять стоимостью через выбор более дешёвой модели.