

# Setup Codex with GPT-5

Requirements

[Codex Setup Guide](#)

  Installation

  Authentication

  Configuration

    Codex settings

  Environment Variables

    MacOS

    Windows

  Token Allocation Strategy

    Request

    Response

Model Compatibility

  Testing Methodology

    GPT-5

  Other Models WIP

## Requirements

- ⓘ As everything in the Group, CLI connectivity is dependant on having the right proxy and SSL settings.

Sounds easy, it isn't... To solve proxy issues, you should run Alpaca:

 [Setting up Alpaca on MacOS](#) or  [Setting up Alpaca on Windows](#)

 [CLI - Why you should not set CNTLM nor Prisma](#)

To solve SSL issues, you should build a PEM cert store and point your CLI to it using various environment variables...

 [Group Internal Root CAs Certificate Store](#)

Or if you're `on MacOS`, you can head there and let my script do it for you: [Proxy and SSL Certificate issues](#)

- ⓘ Request `AR - GenAI Studio - Prod - User` access in [identity.cba](#) to access it at: <https://studio.genai.cba> and get an API Key

- ⓘ The below was tested on `Codex 0.40`, `LiteLLM 1.72.2` ([api.studio.genai.cba](#)),  
`MacOS 15.5 (24F74)`, `zsh 5.9` & `bash 3.2.57(1)`

# Codex Setup Guide

## Installation

Follow the official documentation for the most up-to-date installation instructions: [GitHub - openai/codex: Lightweight coding agent that runs in your terminal](#)

Current installation command: `npm install -g @openai/codex` or `brew install codex`

## Authentication

Follow the official documentation for the most up-to-date authentication instructions:

1. Update/Save the `sk-` key in the Keychain:

```
security delete-generic-password -s "LITELLM_API_KEY";
security add-generic-password -s "LITELLM_API_KEY" -a
"$(whoami)" -w "KEY"
```

where `KEY` is your `sk-` key from Open-WebUI (<https://studio.genai.cba>)

2. Add the following line to your shell configuration file (`~/.zshrc`, `~/.bashrc`, etc.)

```
1 export LITELLM_MASTER_KEY=$(security find-generic-password -s "LITELLM_API_KEY" -w)
2 export OPENAI_API_KEY="$LITELLM_MASTER_KEY"
```

3. Restart your terminal or run `source ~/.zshrc` (or equivalent for your shell)

## Configuration

### Codex settings

Create or update your `~/.codex/config.toml` file with the following configuration:

*I built it manually based on: [GitHub - openai/codex/docs/config.md at main · openai/codex](#)*

```
1 # Recall that in TOML, root keys must be listed before tables.
2 model = "gpt-5_v2025-08-07_EASTUS2"
3 model_provider = "litellm"
4
5 approval_policy = "on-failure"
6 sandbox_mode = "danger-full-access" # "workspace-write", or "read-only"
7 tools.web_search = true
```

```

8
9 # Setting `profile` is equivalent to specifying `--profile o3` on the command
10 # line, though the `--profile` flag can still be used to override this value.
11 profile = "gpt5"
12
13 [model_providers.litellm]
14 name = "Chat Completions"
15 base_url = "https://api.studio.genai.cba/v1"
16 env_key = "OPENAI_API_KEY"
17 wire_api = "chat"
18
19
20 [profiles.gpt5]
21 model = "gpt-5_v2025-08-07_EASTUS2"
22 model_provider = "litellm"
23 approval_policy = "on-failure"
24 request_max_retries = 4           # retry failed HTTP requests
25 stream_max_retries = 10          # retry dropped SSE streams
26 stream_idle_timeout_ms = 300000   # 5m idle timeout
27 model_reasoning_effort = "minimal" # minimal, low, medium (default), or high
28 model_reasoning_summary = "none"   # none, auto, concise or detailed
29 model_verbosity = "low"           # low, medium or high
30 sandbox_mode = "danger-full-access" # "workspace-write", or "read-only"
31 tools.web_search = true

```

## Environment Variables

### MacOS

In our Shell config (e.g. `~/.zshrc` ), create the following entries:

```

1 export OPENAI_API_KEY="$LITELLM_MASTER_KEY"           # Already performed in Authentication
2 export OPENAI_API_BASE="https://api.studio.genai.cba"
3 export OPENAI_BASE_URL="https://api.studio.genai.cba"

```

⚠ Alas it seems `export OPENAI_MODEL="gpt-5_v2025-08-07_EASTUS2"` is not recognised.

Also there appear to be a glitch where codex will read

`GOOGLE_GEMINI_BASE_URL="http://localhost:9201"` (if declared), so you might have to unset this...

### Windows

Same as above using Windows System Properties / Env Var:

See → [🔒 Group Internal Root CAs Certificate Store | Method 1: Using System Properties \(Recommended\)](#)

## Token Allocation Strategy

This step was inconclusive since our team did not input figures for max output / token.

- ✓ The inconclusive tests...

#### Request

```
1 curl -X 'GET' \
2   'https://api.studio.genai.cba/api/v1/v1/model/info?litellm_model_id=3f705214-f147-406a-
3   9b80-5de2d76662a1' \
4   -H 'accept: application/json' \
5   -H 'x-litellm-api-key: sk-1234567890'
```

#### Response

```
1 {
2   "model_name": "guardrails-gpt-5_v2025-08-07_EASTUS2",
3   "litellm_params": {
4     "api_base": "http://localhost:9099/v1",
5     "use_in_pass_through": false,
6     "use_litellm_proxy": false,
7     "merge_reasoning_content_in_choices": false,
8     "model": "openai/guardrails.gpt-5_v2025-08-07_EASTUS2"
9   },
10  "model_info": {
11    "id": "0f74ca6b1ff0bc3b9629c0690897815a5f12213182d476598119b264dfcdcdce",
12    "db_model": false,
13    "key": "openai/guardrails.gpt-5_v2025-08-07_EASTUS2",
14    "max_tokens": null,
15    "max_input_tokens": null,
16    "max_output_tokens": null,
17    "input_cost_per_token": 0,
18    "cache_creation_input_token_cost": null,
19    "cache_read_input_token_cost": null,
20    "input_cost_per_character": null,
21    "input_cost_per_token_above_128k_tokens": null,
22    "input_cost_per_token_above_200k_tokens": null,
23    "input_cost_per_query": null,
24    "input_cost_per_second": null,
25    "input_cost_per_audio_token": null,
26    "input_cost_per_token_batches": null,
27    "output_cost_per_token_batches": null,
28    "output_cost_per_token": 0,
29    "output_cost_per_audio_token": null,
30    "output_cost_per_character": null,
31    "output_cost_per_reasoning_token": null,
32    "output_cost_per_token_above_128k_tokens": null,
33    "output_cost_per_character_above_128k_tokens": null,
34    "output_cost_per_token_above_200k_tokens": null,
35    "output_cost_per_second": null,
36    "output_cost_per_image": null,
37    "output_vector_size": null,
38    "litellm_provider": "openai",
39    "mode": null,
40    "supports_system_messages": null,
41    "supports_response_schema": null,
42    "supports_vision": null,
43    "supports_function_calling": null,
44    "supports_tool_choice": null,
45    "supports_assistant_prefill": null,
46    "supports_prompt_caching": null,
47    "supports_audio_input": null,
```

```

48     "supports_audio_output": null,
49     "supports_pdf_input": null,
50     "supports_embedding_image_input": null,
51     "supports_native_streaming": null,
52     "supports_web_search": null,
53     "supports_url_context": null,
54     "supports_reasoning": null,
55     "supports_computer_use": null,
56     "search_context_cost_per_query": null,
57     "tpm": null,
58     "rpm": null,
59     "supported_openai_params": [
60         "frequency_penalty",
61         "logit_bias",
62         "logprobs",
63         "top_logprobs",
64         "max_tokens",
65         "max_completion_tokens",
66         "modalities",
67         "prediction",
68         "n",
69         "presence_penalty",
70         "seed",
71         "stop",
72         "stream",
73         "stream_options",
74         "temperature",
75         "top_p",
76         "tools",
77         "tool_choice",
78         "function_call",
79         "functions",
80         "max_retries",
81         "extra_headers",
82         "parallel_tool_calls",
83         "audio",
84         "web_search_options",
85         "response_format"
86     ]
87 }

```

Regardless, the Codex CLI doesn't expose granular controls for separate input/output caps, so we'll need to manage headroom ourselves against [GPT-5's ~400k-token context window and ~128k max output](#). I'll start with `model_reasoning_effort="minimal"` to reduce internal token burn and latency, then report on “low, medium, high” options that trade cost and latency for quality, especially on long-context runs (where GPT-5 high stays strong at 128k–256k tokens).

```

Run `npm install -g @openai/codex@latest` to update.

> _OpenAI Codex (v0.36.0)
model: guardrails-gpt-5_v2025-08-07_EASTUS2 /model to change|
directory: ~/Documents/_RBS_Card_Agent_Validation

To get started, describe a task or try one of these commands:
/init - create an AGENTS.md file with instructions for Codex
/status - show current session configuration
/approvals - choose what Codex can do without approval
/model - choose what model and reasoning effort to use

I give me a summary of the claudie_notes folder
> I'll list the contents of claudie_notes and skim each file to summarize.

• Explored
└ List ls -la
⚠stream error: exceeded retry limit, last status: 500 Internal Server Error; retrying 1/5 in 193ms.
⚠stream error: exceeded retry limit, last status: 500 Internal Server Error; retrying 2/5 in 402ms.
⚠stream error: exceeded retry limit, last status: 500 Internal Server Error; retrying 3/5 in 743ms.
⚠stream error: exceeded retry limit, last status: 500 Internal Server Error; retrying 4/5 in 1.493s.

Working (23s • Esc to interrupt)
#improve documentation in @filename
at send ^J newline ^T transcript ^C quit

```

HTTP 500 screenshot  
courtesy of [@Dom Nasrabad](#)

```

> _OpenAI Codex (v0.36.0)
model: gpt-5_v2025-08-07_EASTUS2 /model to change|
directory: ~/codex

To get started, describe a task or try one of these commands:
/init - create an AGENTS.md file with instructions for Codex
/status - show current session configuration
/approvals - choose what Codex can do without approval
/model - choose what model and reasoning effort to use

Hello
> Hi! What can I help you with today?
Tell me a poem
* stream error: exceeded retry limit, last status: 429 Too Many Requests; retrying 1/5 in 216ms.
# Conversation interrupted - tell the model what to do differently
Hello
> Hello! Here's a short poem:
A small hush hums behind the screen,
pixels breathing like distant stars.
Coffee cools, the cursor blinks-
a lighthouse in a quiet harbor.
We stitch our hopes in simple lines,
loops of thought, a gentle thread.
Outside, dawn rehearses light-
inside, a spark becomes a star.
Token usage: total=6,807 inputs=6,395 (+ 6,144 cached) output=412 (reasoning 328)
To continue this session, run codex resume 01997e25-77b3-7082-9e55-342e440bedbe.
codex

```

HTTP 429

## Model Compatibility

### Testing Methodology

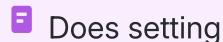
GPT-5

I ran `/init` in a busy project directory containing lots of code and pdf/png images.

The model performed well and did not crash using the above settings.



Using  
`model_reasoning_effort="minimal"`  
 (along with low verbosity and no reasoning summary) reduces hidden “thinking” tokens and extra text, which lowers tokens-per-request and compute time. That can reduce the likelihood of 429s caused by tokens-per-minute quotas and some 500s due to timeouts or overloaded requests. It won’t prevent 429s driven by requests-per-minute limits, nor 500s from provider-side faults. This is require ongoing testing...



Does setting  
`model_reasoning_effort="minimal"` on GPT-5 negate its advantages and make it behave like GPT-3/3o?

No, this doesn’t change the underlying model or its capabilities. It only reduces the internal reasoning/compute budget, making responses faster and cheaper but potentially less robust on complex, multi-step tasks. Use minimal for simple tasks; increase effort for harder problems.

```
> codex
↑ Update available! 0.40.0 -> 0.41.0.
Run npm install -g @openai/codex@latest to update.

>_ OpenAI Codex (v0.40.0)
model: gpt-5_v2025-08-07_EASTUS2 minimal /model to change|
directory: ~/Github/video-to-gif

To get started, describe a task or try one of these commands:

/init - create an AGENTS.md file with instructions for Codex
/status - show current session configuration
/approvals - choose what Codex can do without approval
/model - choose what model and reasoning effort to use

Find and fix a bug in @filename
send ^J newline ^T transcript ^C quit

0:00 / 0:44 1x [ ] ↓
```

codex `/init` and `/status` (sped up)

#### Other Models WIP

Each model **will be** tested using the following procedure:

1. Running `codex --model <model> "What's your model?"` to test if the model responds correctly
2. Recording the response or error message

🎉 Helpful? Drop me a thanks on [Achievers](#)! And if you've got knowledge to share, don't hold back - we all grow when we learn from each other 💡