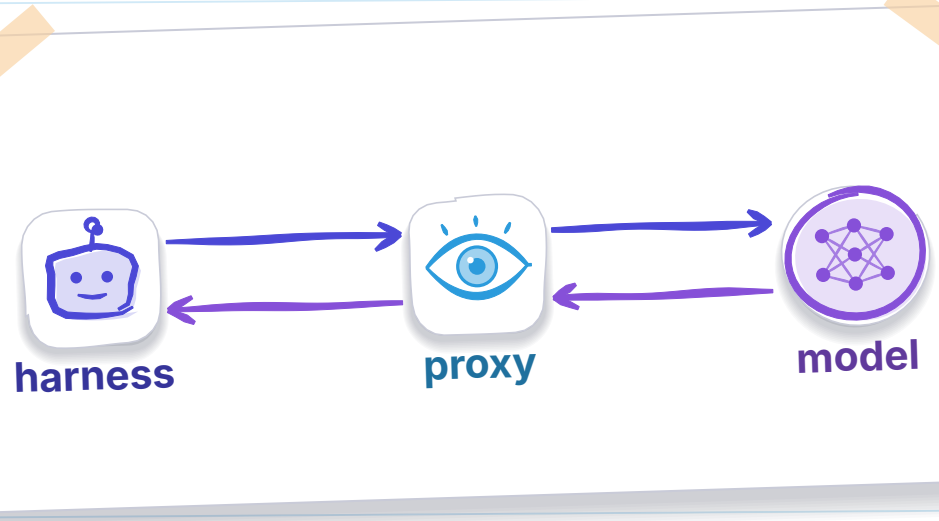
Prof. Aviv Zohar

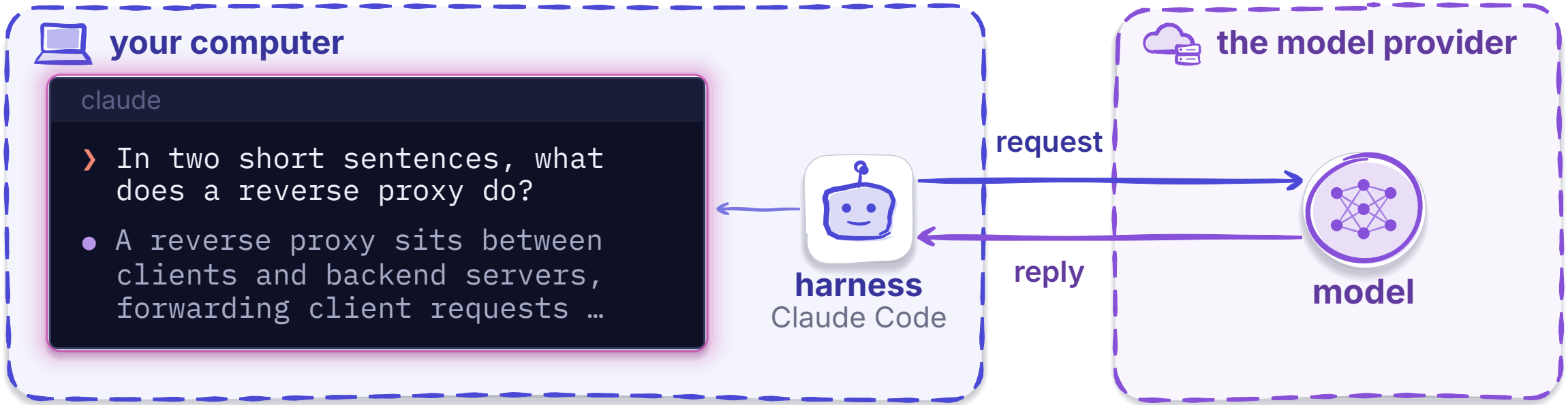
Semester A, 2026–7

Intercepting the Harness's Messages

A reverse proxy and the messages it reveals



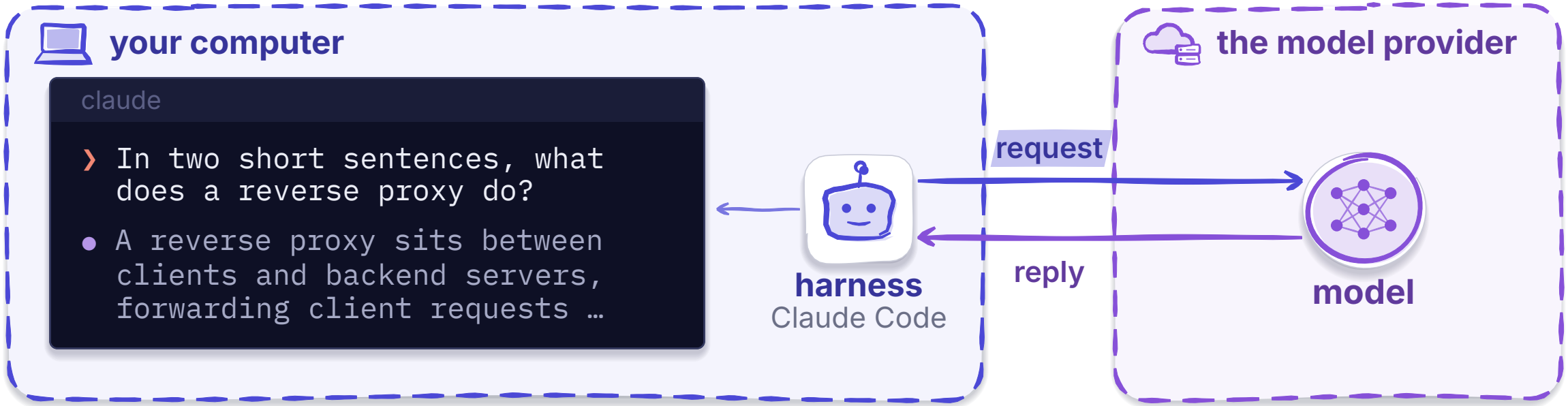
The terminal shows the answer, not the request



What you see: the answers, and what the tools did.

what went into the request?

The terminal shows the answer, not the request

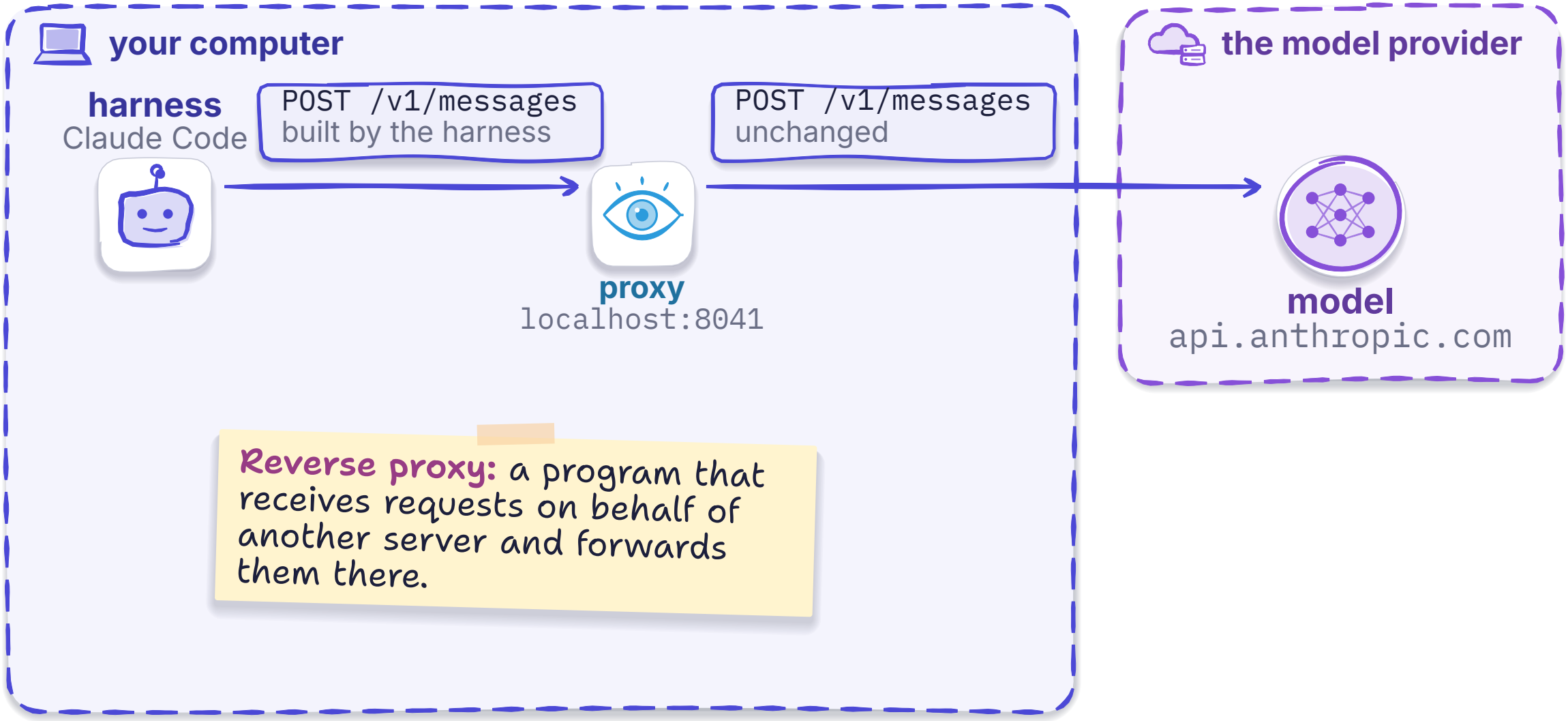


We want to read the request itself, not guess it from the answer.

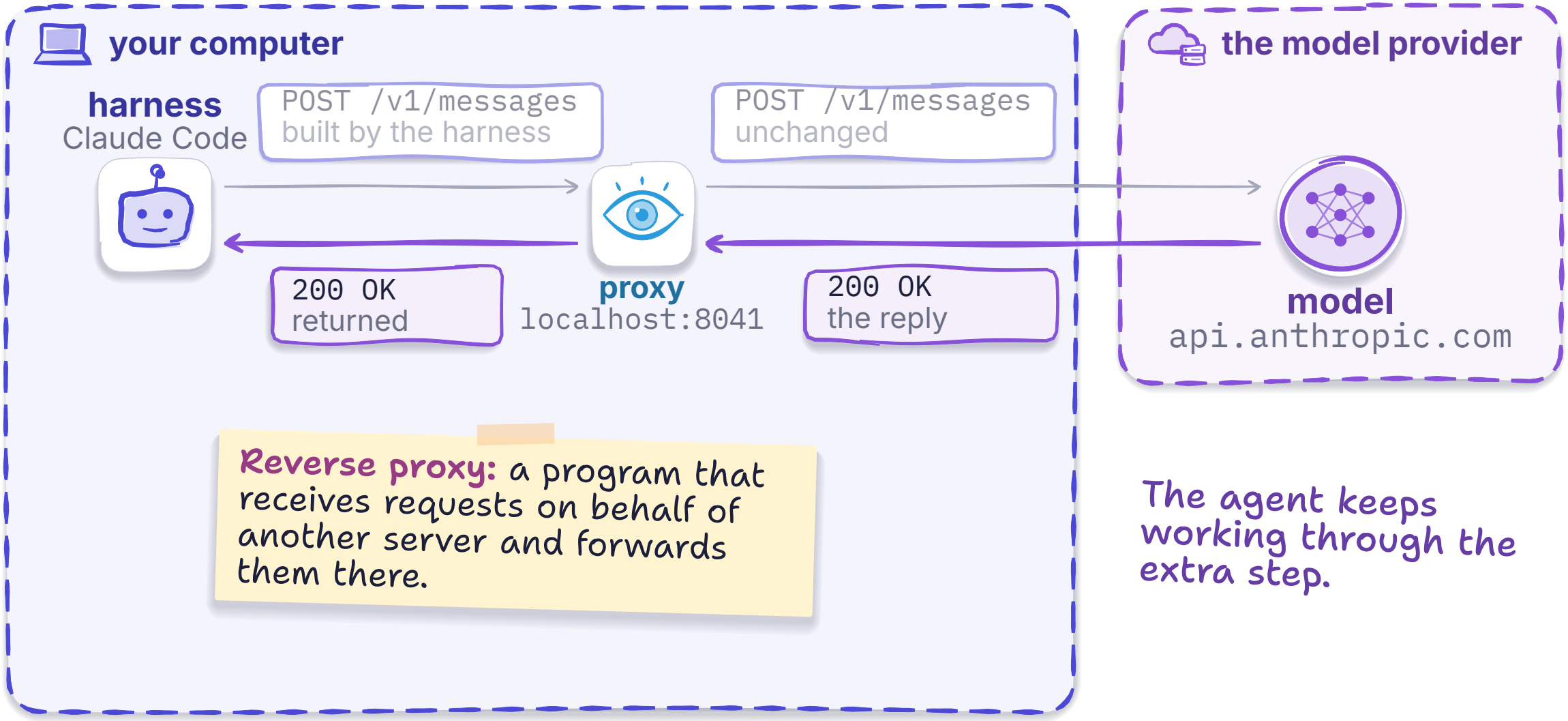
POST `/v1/messages` — **the request**

the conversation	instructions	tool descriptions
your messages and the replies	written by the harness	what the model may ask for

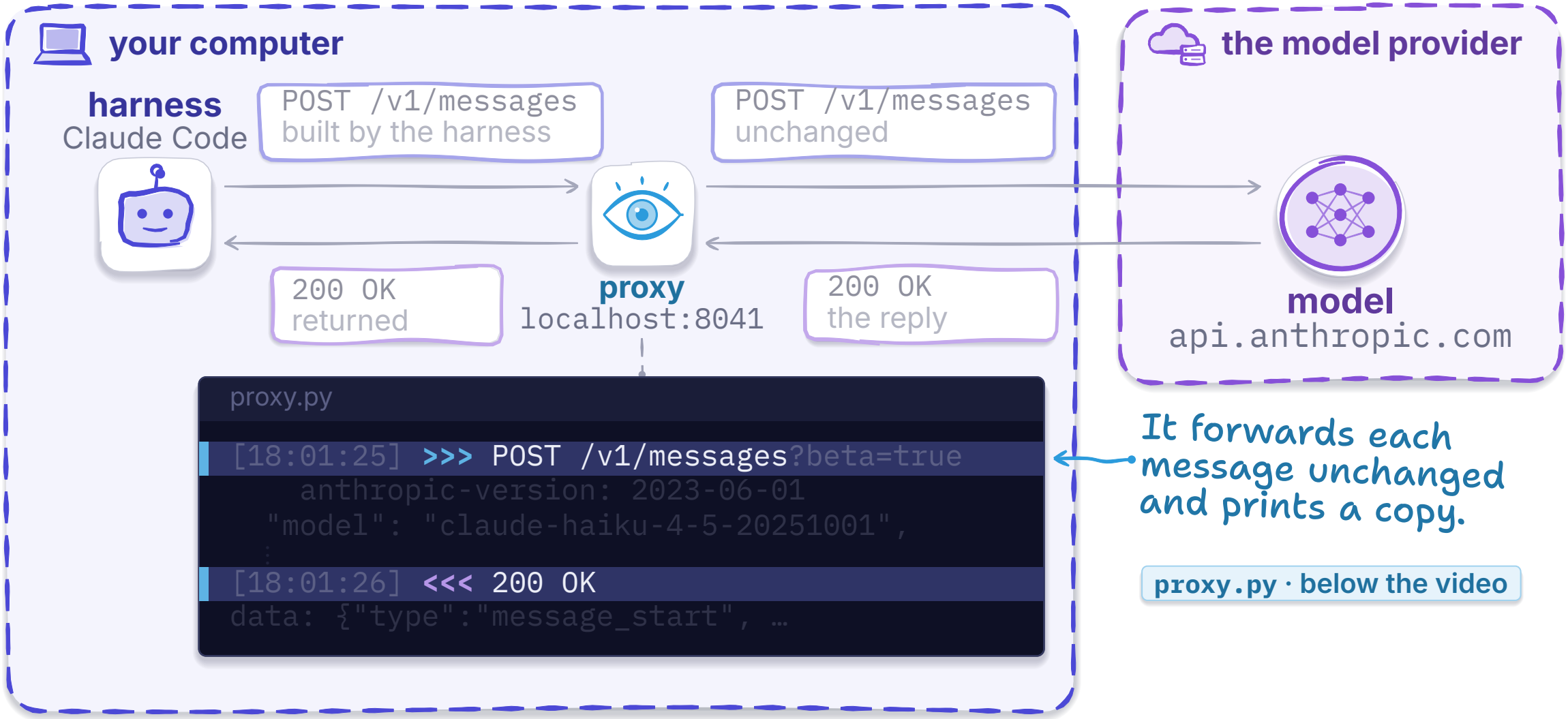
A reverse proxy sits in the middle



A reverse proxy sits in the middle



A reverse proxy sits in the middle



Run the proxy, then point Claude Code at it

terminal 1

shell

```
~ $ python3 proxy.py --port 8041
proxy listening on http://localhost:8041  ->  https://api.anthropic.com
run the agent with:  ANTHROPIC_BASE_URL=http://localhost:8041 claude
```

Python 3, no extra packages. The proxy now listens on your own computer, on port 8041.

proxy startup lines captured 2026-09-16

Run the proxy, then point Claude Code at it

terminal 1

```shell
~ $ python3 proxy.py --port 8041
proxy listening on http://localhost:8041  ->  https://api.anthropic.com
run the agent with:  ANTHROPIC_BASE_URL=http://localhost:8041 claude
```

terminal 2

```shell
~ $ ANTHROPIC_BASE_URL=http://localhost:8041 claude
```

This variable sends Claude Code's model requests to the proxy.

proxy startup lines captured 2026-09-16

Run the proxy, then point Claude Code at it

terminal 1

shell

```
~ $ python3 proxy.py --port 8041
proxy listening on http://localhost:8041  ->  https://api.anthropic.com
run the agent with:  ANTHROPIC_BASE_URL=http://localhost:8041 claude
  ⋮
[18:01:25] >>> POST /v1/messages?beta=true
    Content-Length: 93282
  ⋮
          "text": "In two short sentences, what does a reverse proxy do?",   ⓐ
```

terminal 2

shell

```
~ $ ANTHROPIC_BASE_URL=http://localhost:8041 claude
❯ In two short sentences, what does a reverse proxy do?   ⓐ
```

ⓐ Our question, and the request that carries it, printed by the proxy.

One short question, a 93 KB request

POST /v1/messages?beta=true — 93,282 bytes

```
{
  "model": "claude-haiku-4-5-20251001",
  "messages": [
    { "role": "user",
      "content": [ ▸ 3 <system-reminder> blocks ,
        { "type": "text",
          "text": "In two short sentences,
                   what does a reverse proxy do?" }
    ] } ],
  "system": [ ▸ 3 text blocks ],
  "tools": [ ▸ 29 tools · 76 KB ],
  ▸ 5 more fields: max_tokens, thinking, stream, ...
}
```

Our question: 53 characters.

Everything the harness sent: **93,282 bytes**. Most of it lies outside this excerpt.

One short question, a 93 KB request

POST /v1/messages?beta=true — **93,282 bytes**

```
{
  "model": "claude-haiku-4-5-20251001",
  "messages": [
    { "role": "user",
      "content": [ ▸ 3 <system-reminder> blocks,
        { "type": "text",
          "text": "In two short sentences,
                   what does a reverse proxy do?" }
    ] } ],
  "system": [ ▸ 3 text blocks ],
  "tools": [ ▸ 29 tools · 76 KB ],
  ▸ 5 more fields: max_tokens, thinking, stream, ...
}
```

the conversation

the system instructions

the tool descriptions

One short question, a 93 KB request

POST /v1/messages?beta=true — **93,282 bytes**

```
{
  "model": "claude-haiku-4-5-20251001",
  "messages": [
    { "role": "user",
      "content": [ ▸ 3 <system-reminder> blocks ,
        { "type": "text",
          "text": "In two short sentences,
                   what does a reverse proxy do?" }
    ] } ],
  "system": [ ▸ 3 text blocks ],
  "tools": [ ▸ 29 tools · 76 KB ],
  ▸ 5 more fields: max_tokens, thinking, stream, ...
}
```

We typed one line. The harness added these instructions.

Part of the system prompt in that request

"system" block 3 of 3, excerpt

```
ⓐ You are an interactive agent that helps users
  with software engineering tasks.

  # Harness
   - Text you output outside of tool use is
ⓑ    displayed to the user as Github-flavored
     markdown in a terminal.

ⓒ Write code that reads like the surrounding code:
  match its comment density, naming, and idiom.
```

ⓐ what the agent is for

ⓑ where its text appears

ⓒ match the code around it

Part of the system prompt in that request

"system" block 3 of 3, excerpt

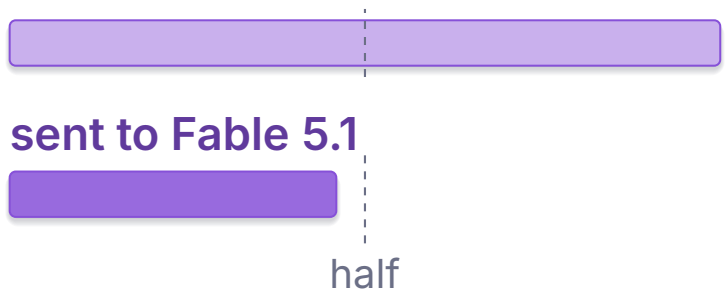
```
You are an interactive agent that helps users
with software engineering tasks.

# Harness
 - Text you output outside of tool use is
   displayed to the user as Github-flavored
   markdown in a terminal.

Write code that reads like the surrounding code:
match its comment density, naming, and idiom.
```

We did not type this. The harness sent it alongside our message.

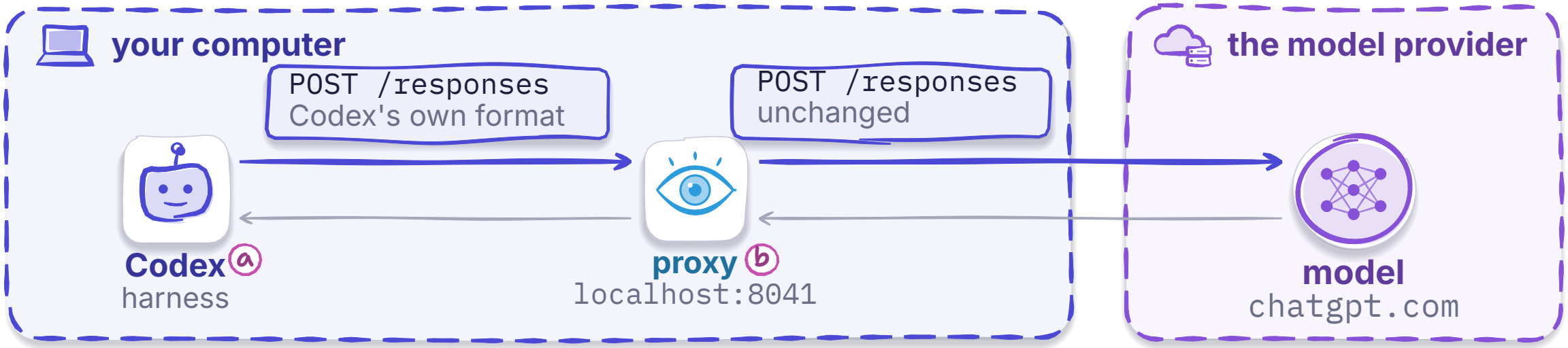
a later capture: the same harness



`system-prompts.html` · below the video

excerpt: 2026-09-06 capture · bars to scale: word counts in system-prompts.html (Claude Code 2.1.278)

The same proxy works with Codex



Codex setting ⓐ

```
base_url = "http://localhost:8041"
```

proxy option ⓑ

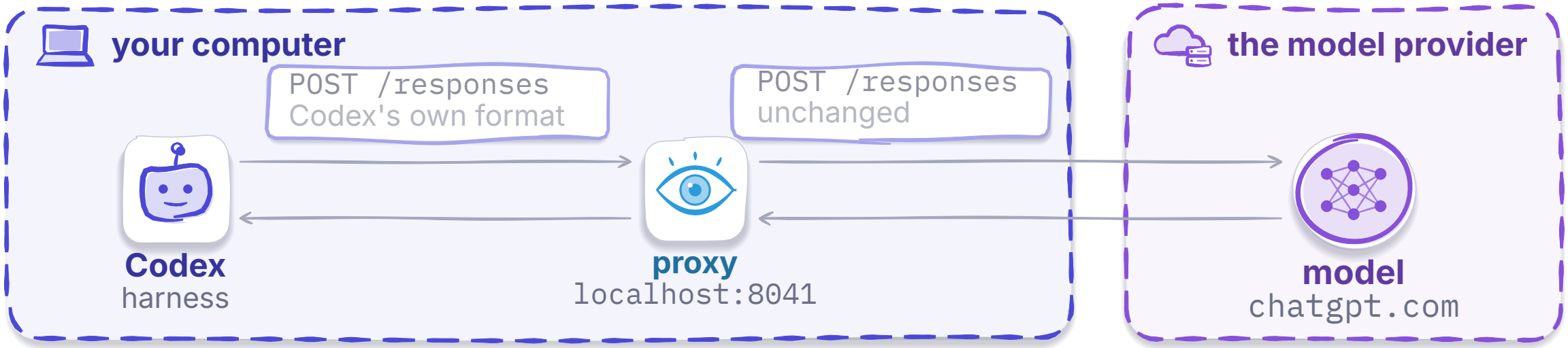
```
--upstream https://chatgpt.com/backend-api/codex
```

ⓐ Codex sends to the proxy.

ⓑ The proxy forwards to Codex's service.

A different request format, the same idea.

The same proxy works with Codex



Codex setting

```
base_url = "http://localhost:8041"
```

proxy option

```
--upstream https://chatgpt.com/backend-api/codex
```

The complete commands for both agents, including Codex with a ChatGPT login.

setup.md · below the video

From printed traffic to a live monitor

the proxy's terminal

proxy.py

```
[18:01:25] >>> POST /v1/messages?beta=true
    anthropic-version: 2023-06-01
    Content-Length: 93282
{
  "model": "claude-haiku-4-5-20251001",
  "messages": [
    { "role": "user", "content": [ …
  ⋮
  "tools": [ { "name": "Agent", …
  ⋮
[18:01:26] <<< 200 OK
event: content_block_delta
data: {"type":"content_block_delta",…
event: content_block_delta
data: {"type":"content_block_delta",…
  ⋮
```

Printed traffic works, but a growing conversation is hard to read here.

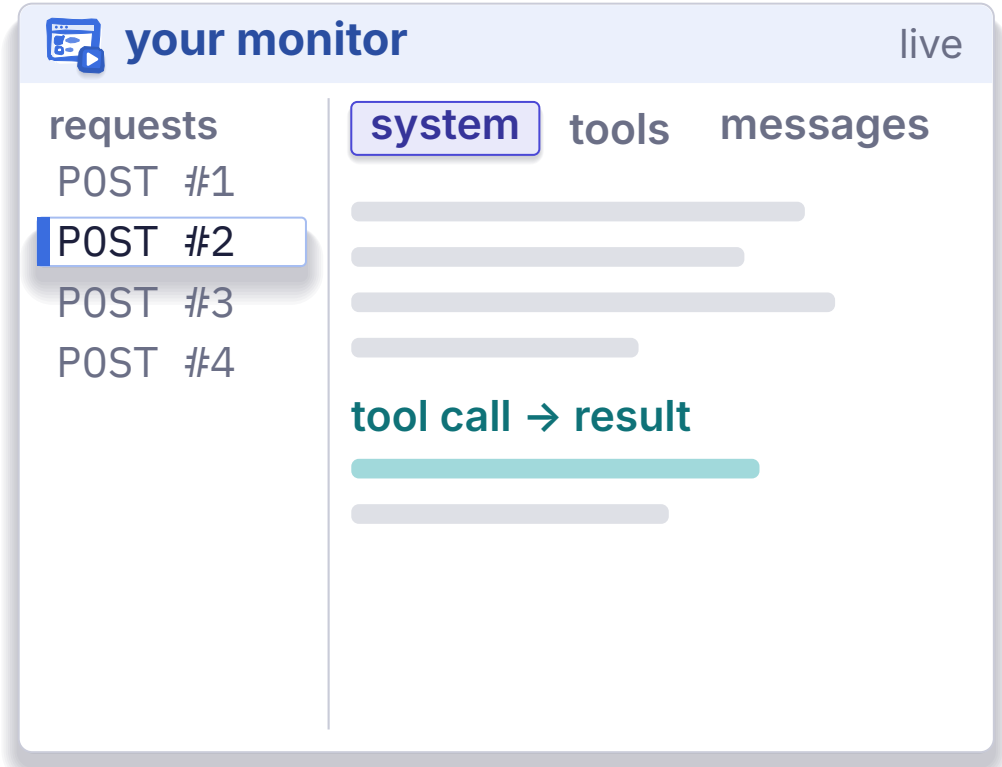
We want to open one request, and go straight to the part we need.

From printed traffic to a live monitor

the proxy's terminal

proxy.py

```
[18:01:25] >>> POST /v1/messages?beta=true
    anthropic-version: 2023-06-01
    Content-Length: 93282
{
  "model": "claude-haiku-4-5-20251001",
  "messages": [
    { "role": "user", "content": [ …
  "tools": [ { "name": "Agent", …
[18:01:26] <<< 200 OK
event: content_block_delta
data: {"type":"content_block_delta",…
event: content_block_delta
data: {"type":"content_block_delta",…
```



The lab: build it on our proxy.

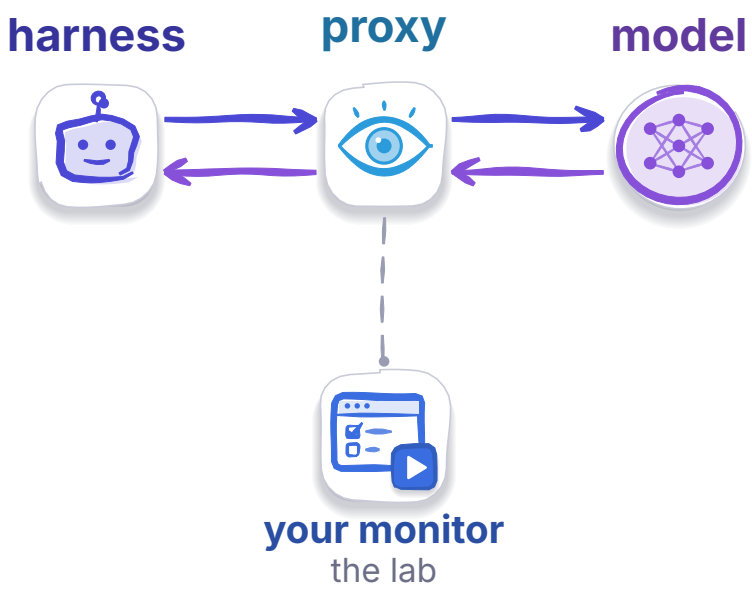
monitor: schematic

A proxy shows what the harness sends

our question
in `"messages"`, a small part of the request

the tool definitions
in `"tools"`: what the model may ask for

the harness's instructions
in `"system"`: text we never typed



Put a proxy between the harness and the model, and read the messages.