curl: why your POST works in bash but not in a script

📅 2026-09-15 📊 beginner ["curl" "bash" "http" "apis"]

The Problem

The same `curl -X POST` command works perfectly when you type it in your terminal, but returns HTTP 400 (or a JSON parse error) when you paste it into a `.sh` script. Or worse: it works locally but fails in CI.

The Mistake

Embedding JSON inside single quotes (`-d '{...}'`). The shell treats the single quotes as literal delimiters, but the moment your JSON contains an apostrophe (a name like `O'Brien` , a comment with `it's` , or a SQL `WHERE name = 'foo'`) the quote-pair breaks and the shell starts parsing the rest of the command as code.

The Fix

Use `--data-raw` with a here-doc, or pipe the body via stdin:

```
curl -X POST https://api.example.com/things \
  -H "Content-Type: application/json" \
  --data-binary @- <<'EOF'
{"name": "O'Brien", "tag": "it's a test"}
EOF
```

The `<<'EOF'` (with quotes around the delimiter) disables variable expansion inside the body, so `$` characters and backticks stay literal.

Why It Works

The shell only does its quote-matching at parse time. Inside `'...'` the shell treats every byte as literal, but the *first* unescaped `'` ends the string. JSON often contains apostrophes, so single-quoting is fragile. Heredocs with a quoted delimiter bypass shell parsing entirely.

Copy-Paste

```
# Read JSON body from a file (cleanest)
curl -X POST https://api.example.com/things \
  -H "Content-Type: application/json" \
  --data-binary @body.json
```

Next Steps

- For interactive testing, use `curl ... | jq` so you can see parsed JSON.
- For repeatable calls, save the full request as a `.http` file and use the REST Client extension in VS Code.
- For production code, use a real HTTP client (Python `requests` , Node `fetch`) instead of shelling out.

Download PDF (1 page)