CORS (Cross-Origin Resource Sharing) is a **browser security mechanism** — `curl` itself is **not restricted by CORS** because it's not a browser.

However, you can **simulate** a CORS preflight or actual cross-origin request by adding the same HTTP headers a browser would send.

1️⃣ Simulate a Simple CORS Request

```
curl -i \
  -H "Origin: https://example.com" \
  https://api.yourserver.com/data
```

Explanation:

- `-i` → Show response headers.
- `-H "Origin: ..."` → Mimics a browser sending the origin.
- Server should respond with `Access-Control-Allow-Origin` if CORS is enabled.

2️⃣ Simulate a CORS Preflight (OPTIONS request)

```
curl -i -X OPTIONS \
  -H "Origin: https://example.com" \
  -H "Access-Control-Request-Method: GET" \
  -H "Access-Control-Request-Headers: Content-Type" \
  https://api.yourserver.com/data
```

Explanation:

- `-X OPTIONS` → Sends an HTTP OPTIONS request (preflight).
- `Access-Control-Request-Method` → Method you intend to use (GET, POST, etc.).
- `Access-Control-Request-Headers` → Any custom headers you plan to send.

3️⃣ Simulate a CORS Request with Custom Headers

```
curl -i \
  -H "Origin: https://example.com" \
  -H "Content-Type: application/json" \
  -d '{"name":"John"}' \
  https://api.yourserver.com/data
```

Explanation:

- Sends JSON data with a custom `Content-Type`.
- Will trigger a preflight in browsers, but `curl` sends it directly.

✅ **Tip:**
If you want to **debug CORS** on a server, check:

- `Access-Control-Allow-Origin`
- `Access-Control-Allow-Methods`
- `Access-Control-Allow-Headers`

If you want, I can give you a **single curl command that tests both preflight and actual CORS request in one go** so you can fully debug your API.
Do you want me to prepare that?