ScopeTreasure Courses

LARAVEL COURSES · LESSON LV.14 · CODE SCOPE SERIES

API Authentication — Sanctum Tokens

A key for the API door: Sanctum personal access tokens, the login call that issues them, Bearer headers, 401 vs 200, and logout that truly revokes.

Introduction

LV.13's API is wide open: anyone who finds `/api/students` can read — and worse, delete. Web login (LV.7) cannot help here: phone apps do not carry browser sessions and cookies. APIs use a different key: a **token** — a long secret string the app obtains at login and attaches to every request, like a hotel key card presented at each door.

Laravel's key-maker is **Sanctum**, deliberately the lightweight choice: it lives inside your app, stores hashed tokens in one table, and asks nothing of outside services. (Its heavier cousin Passport implements full OAuth for third-party ecosystems — power most apps never need.) This lesson issues a token at login, watches the same request fail `401` without it and pass `200` with it, and revokes the key at logout.

(Roman Urdu: Session mehman-nawazi hai, token chaabi hai — API ke darwazay par har baar chaabi dikhani parti hai.)

Key points

- **The token flow is three beats:** (1) the app POSTs email + password to a login endpoint; (2) the server checks them and returns a fresh token; (3) every later request carries `Authorization: Bearer <token>` — and Sanctum matches it to a user before your controller ever wakes.
- **Sanctum setup is short on modern Laravel:** `php artisan install:api` installs Sanctum and the api routes; the User model adds one trait — `use HasApiTokens;` — which grows the token powers (`createToken`, `tokens`). That is nearly the whole ceremony.
- **The token is shown exactly once:** `createToken('phone')` returns an object whose `plainTextToken` (like `1|kX93...fT`) is the only readable copy ever. The DATABASE stores only its hash — the LV.7 password lesson, repeated for keys. Lose the plain text and the cure is a fresh token, never a recovery.
- **`auth:sanctum` is LV.7's lock in API clothes:** routes inside the middleware group reject guests — but an API guest gets no login page; it gets one cold JSON line: `{"message": "Unauthenticated."}` with status **401**.

  (Roman Urdu: Web par bina login ke aap login page par bheje jate hain; API par darwaza bas aik jumla kehta hai — "pehchan nahi mili".)
- **`/api/user` is the identity mirror:** the standard protected route returns the token's owner as JSON — how an app asks "who am I holding a key for?" after reopening.
- **Logout = revoke, not forget:** deleting the app's local copy while the token lives server-side leaves a working key floating. Real logout calls `$user->tokens()->delete()` (or deletes the current token) — the card is cancelled, not just dropped.

- **One token per device is good manners:** name tokens ('phone', 'tablet') so a lost phone's key can be revoked alone, leaving the rest alive.

Code examples

Example 1 — The login endpoint that issues the key

```
// User model — one trait grows the powers
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens;
}
```

```
// routes/api.php — the login door is public; it IS the key office
Route::post('/login', [AuthApiController::class, 'login']);
```

```
// AuthApiController
public function login(Request $request)
{
    $request->validate([
        'email'    => 'required|email',
        'password' => 'required',
    ]);

    $user = User::where('email', $request->email)->first();

    if (! $user || ! Hash::check($request->password, $user->password)) {
        return response()->json(['message' => 'Invalid credentials'], 401);
    }

    $token = $user->createToken('phone')->plainTextToken;

    return response()->json([
        'token' => $token,
        'user'  => $user->only('id', 'name', 'email'),
    ]);
}
```

What it does: The password is checked against the LV.7 hash with `Hash::check` — never compared as text. Success mints a named token and returns its one-time plain text with the user's public face. Failure is a flat 401 — the API does not say WHICH half was wrong, and that silence is deliberate.

What you will see: POST the teacher's email + password from Postman → `200` and a body holding `"token": "1|kX93hF...8fT"` plus the user. Copy that token — the next example spends it.

Example 2 — The same door, without and with the key: 401 vs 200

```
// routes/api.php — everything else moves behind the guard
Route::middleware('auth:sanctum')->group(function () {
    Route::apiResource('students', StudentApiController::class);

    Route::get('/user', function (Request $request) {
        return $request->user();
    });
});
```

What it does: One group line locks the whole API behind Sanctum. Now repeat LV.13's innocent GET twice in Postman — first bare, then with the header `Authorization: Bearer 1|kX93hF...8fT` pasted in.

What you will see: without the header, `401` and the cold single line:

```
{ "message": "Unauthenticated." }
```

With the header, `200` and the full students JSON — and GET `/api/user` returns the teacher's own JSON. Same address, same app: the key made the difference.

Example 3 — Logout that actually revokes

```
Route::middleware('auth:sanctum')->group(function () {
    Route::post('/logout', [AuthApiController::class, 'logout']);
});
```

```
public function logout(Request $request)
{
    $request->user()->tokens()->delete();

    return response()->json(['message' => 'Logged out, tokens revoked']);
}
```

What it does: The endpoint lives INSIDE the guard (you must hold a key to surrender it). `tokens()->delete()` wipes the user's minted tokens from the database — every device signed out at once. A finer aim deletes only the current token, sparing the tablet.

What you will see: after logout, repeat Example 2's proud GET with the same Bearer header — the once-mighty token now earns the same cold `401 Unauthenticated.` A copied key, a stolen phone, an old screenshot of a token: all dead. That is what logout means on an API.

Quick recap

- APIs authenticate with tokens, not sessions: login returns a token once; every request presents it as a Bearer header.
- Sanctum = `install:api` + `HasApiTokens` + named `createToken()`; the DB stores only token hashes.
- `auth:sanctum` guards the group; guests receive 401 JSON, never a page.
- Real logout revokes server-side (`tokens()->delete()`), not just forgets client-side.

Common mistakes

1. **Comparing passwords as plain text in the API login** — `where('password', $request->password)` can never match a hash and tempts someone to store passwords raw. Fix: fetch by email, then `Hash::check` — the LV.7 discipline, unbroken.
2. **Logout that only clears the app** — the token still works for anyone holding a copy. Fix: revoke on the server; the client-side delete is just tidiness.
3. **One eternal token for everything** — a single unnamed key shared by all devices means one leak revokes the world. Fix: named tokens per device (`createToken('phone')`), revoked individually.
4. **Hiding routes instead of guarding them** — removing the students list from the app's menus while `/api/students` stays public. Fix: the `auth:sanctum` group is the lock; menus are decoration (the LV.7 door-not-signboard law, API edition).

Practice

1. Run the full key ceremony in Postman: login (save the token), 401 without the header, 200 with it, `/api/user`, logout, and the final 401 with the old token. Screenshot the two 401s — same line, different stories.
2. Issue two tokens for the teacher ('phone' and 'tablet'), then revoke only 'phone' by deleting that one token row through a small endpoint or Tinker. Prove 'tablet' still opens the door.

Practice quiz

1. What are the three beats of the Sanctum token flow?

- Login, logout, login again
- The app POSTs email plus password to a login endpoint, the server checks them and returns a fresh token, and every later request carries Authorization: Bearer <token>
- The server emails a token, the user types it in, the app stores it in a cookie
- Tokens are created by the browser

Answer: The app POSTs email plus password to a login endpoint, the server checks them and returns a fresh token, and every later request carries Authorization: Bearer <token>

2. How many times is a Sanctum token's plain text shown, and what does the database store?

- As many times as you like, and the database stores the plain text
- Exactly once, from plainTextToken; the database stores only its hash, so losing the plain text means a fresh token, never a recovery
- Never, tokens are invisible
- Once a day, and the database stores the hash of the password

Answer: Exactly once, from plainTextToken; the database stores only its hash, so losing the plain text means a fresh token, never a recovery

3. What does real logout mean with Sanctum tokens?

- Deleting the app's local copy, which is enough
- Revoking server-side with $user->tokens()->delete() or deleting the current token; deleting only the local copy leaves a working key floating
- Changing the user's password
- Logging out is impossible with tokens

Answer: Revoking server-side with $user->tokens()->delete() or deleting the current token; deleting only the local copy leaves a working key floating

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/laravel-api-authentication-sanctum · More free tools and notes at https://scopetreasure.com