Architecture micro-service

Point cours - le JWT

Philippe Roussille

3iL INGENIEURS
</les sciences informatiques>

1 JWT par l'exemple

Un **JWT (JSON Web Token)** est une sorte de **badge numérique signé** qu'un utilisateur obtient après s'être connecté. Il contient des informations comme son pseudo, ses rôles, sa date d'expiration… et il permet de **prouver son identité** auprès des autres services.

1.1 Pourquoi utiliser JWT ?

- L'utilisateur s'authentifie **une seule fois** (`/login`)
- Le token est **signé** par le `user-service`, donc infalsifiable
- Les autres services peuvent le **vérifier localement**, sans avoir à redemander

1.2 Exemple de contenu d'un JWT

```
{
  "user": "roger",
  "roles": ["admin"],
  "exp": 1718544300
}
```

1.3 Dans la requête HTTP

Le client ***doit*** envoyer le JWT dans l'en-tête de ***chaque*** requête :

```
Authorization: Bearer <le_token>
```

2 Exemple complet avec Flask et PyJWT

2.1 Créez un fichier Python

```
import jwt
import datetime
from flask import Flask, request, jsonify
```

```python
app = Flask(__name__)
SECRET_KEY = "une_clé_secrète_partagée"

@app.route("/login", methods=["POST"])
def login():
    data = request.get_json()
    pseudo = data.get("pseudo")
    if not pseudo:
        return jsonify({"error": "pseudo manquant"}), 400

    payload = {
        "user": pseudo,
        "roles": ["user"],
        "exp": datetime.datetime.utcnow() +
        ↪  datetime.timedelta(hours=2)
    }
    token = jwt.encode(payload, SECRET_KEY, algorithm="HS256")
    return jsonify({"token": token})

@app.route("/protected")
def protected():
    auth = request.headers.get("Authorization", "")
    if not auth.startswith("Bearer "):
        return jsonify({"error": "Token manquant"}), 401

    token = auth[7:]
    try:
        decoded = jwt.decode(token, SECRET_KEY,
↪   algorithms=["HS256"])
        return jsonify({"status": "ok", "user": decoded["user"]})
    except jwt.ExpiredSignatureError:
        return jsonify({"error": "Token expiré"}), 401
    except jwt.InvalidTokenError:
        return jsonify({"error": "Token invalide"}), 401

if __name__ == "__main__":
    app.run(port=5000)
```

2.2 Installez les dépendances

Dans un terminal :

```
pip install flask pyjwt
```

2.3 Lancez le serveur Flask

```
python votre_fichier_flask.py
```

2.4 Testez l'authentification avec `curl`

2.4.1 Obtenir un token JWT

```
curl -X POST http://localhost:5000/login \
     -H "Content-Type: application/json" \
     -d '{"pseudo": "roger"}'
```

Résultat :

```
{ "token": "eyJOeXAiOiJKV1QiLCJhbGciOi..." }
```

2.4.2 Accéder à une route protégée

```
curl http://localhost:5000/protected \
     -H "Authorization: Bearer VOTRE_TOKEN_ICI"
```

Résultat :

```
{ "status": "ok", "user": "roger" }
```

3 Que retenir

— Le JWT **remplace une session**
— Il est **vérifiable localement** dans un micro-service
— Il expire : on peut contrôler la durée de validité
— On ne stocke **jamais** le token côté serveur : c'est le client qui le garde