Oauth2 & OIDC

Questo è un sottotitolo esplicativo

Autenticazione

L'autenticazione è il processo di verifica dell'identità di un utente.

Risponde alla domanda: **CHI**?

Protocollo a supporto **OpenID Connect**

Autorizzazione

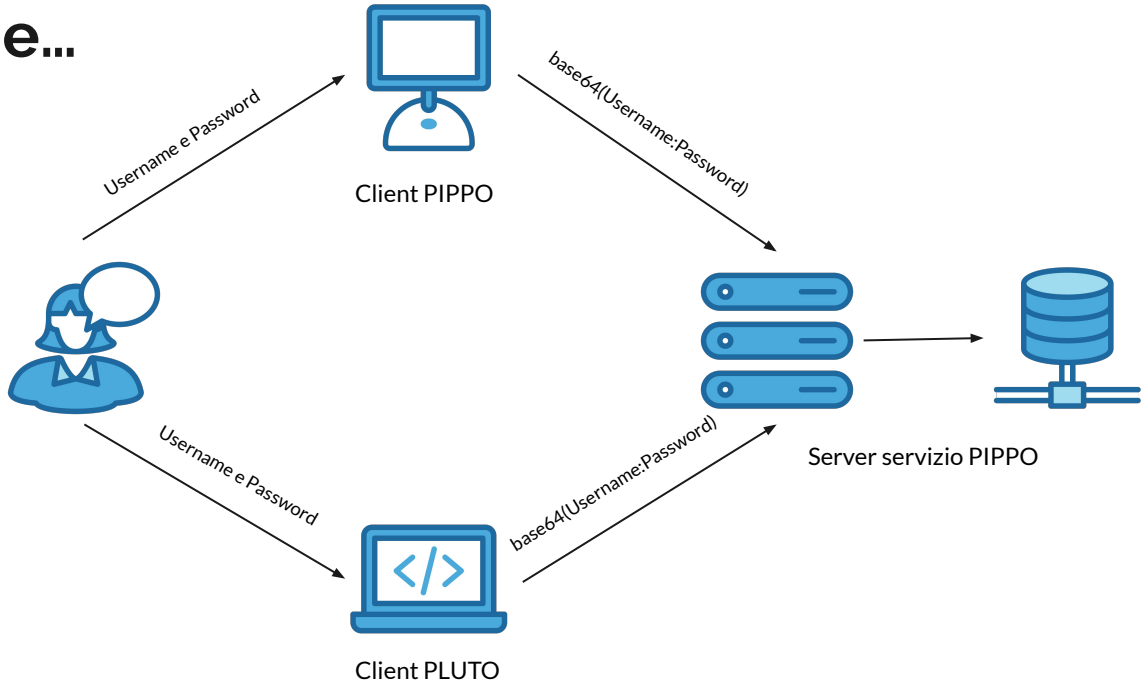
L'autorizzazione è il processo di concessione di un permesso a un utente di accedere a una risorsa.

Risponde alla domanda: **COSA** *può fare*?

Protocollo a supporto **Oauth2**

Once upon a time...

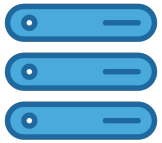
Oauth nasce con lo scopo di permettere l'accesso alle risorse in maniera condivisa senza la necessità di condividere anche le credenziali



Chi partecipa al flusso Oauth2?

- **Resource owner** or **End user** è un entita in grado di concedere l'accesso a delle risorse protette.

- **Client** un'applicazione che richiede risorse protette per conto di un Resource Owner con la sua autorizzazione



- **Resource server** è Il server che ospita le risorse ad accesso limitato deve essere in grado renderle disponibili previa presentazione di un Acces Token valido

- **Authorization server** Il server che autentica il resource owner ed emette l'access token una volta ottenute le autorizzazioni

Cosa è l'access token?

Le specifiche di Oauth2 (**RFC 6749**) non definisco un implementazione dell'access token.

per questo motivo ne esistono diverse adatte a diversi usi. **Nessuno dei due è più sicuro.**

Esistono 2 principali categorie:

- Opachi
- Non opachi

- I token **opachi** non contengono nessuna informazione. Le informazioni relative al token opaco sono storate nel Authorization server. es:(ajikjfds453sdfgdsg)
- I token **non opachi** contengono le informazioni relative a quel token stesso (es: jwt) quindi possono essere letti da tutti e per questo non dovrebbero contenere informazioni sensibili.

Come è fatto un token Jwt

```
//header
{
  "alg": "RS256",
  "typ": "JWT"
}
//payload
{
  "iss": "https://example.auth0.com/",
  "aud": “https://api.example.com/cal/v1/",
  "sub": "usr_123",
  "scope": "read write",
  "iat": 1458785796,
  "exp": 1458872196
}

encodedHeader = base64UrlEncode(header)
encodedPayload = base64UrlEncode(payload)
signature = RSASHA256(
  encodedHeader + . + encodedPayloady, PrivateKey)

token =  encodedHeader + . +
         encodedPayload + . +
         signature
```

Il token è sicuro in quanto la signature viene creata dal authorization server con una **chiave privata.**

Il token **non può essere modificato** senza possedere la chiave privata mantenendolo valido.

i vari client possono richiedere una **chiave pubblica** all’Authorization server per la decodifica del token

110% di protezione in più*

* la percentuale riportata non ha nessun significato

Come si ottiene un token

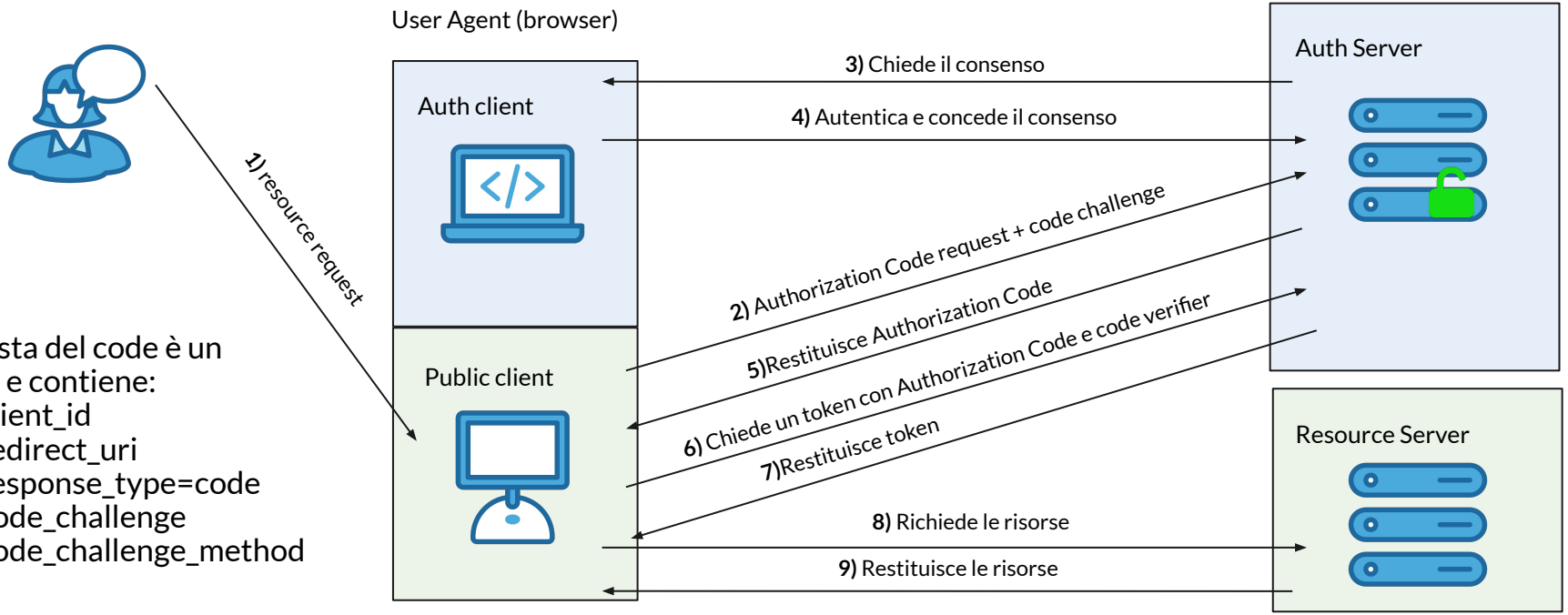
Oauth2 definisce diversi flussi, chiamandoli Authorization Grant, utilizzabili nella richiesta di un token.

Il nome di questi flussi diventato lo standard de facto è **Grant Type.**

Alcuni di questi grant type sono stati deprecati, quelli considerati sicuri e che analizzeremo sono:

- Authorization code with PKCE
- ClientCredentials
- Refresh Token

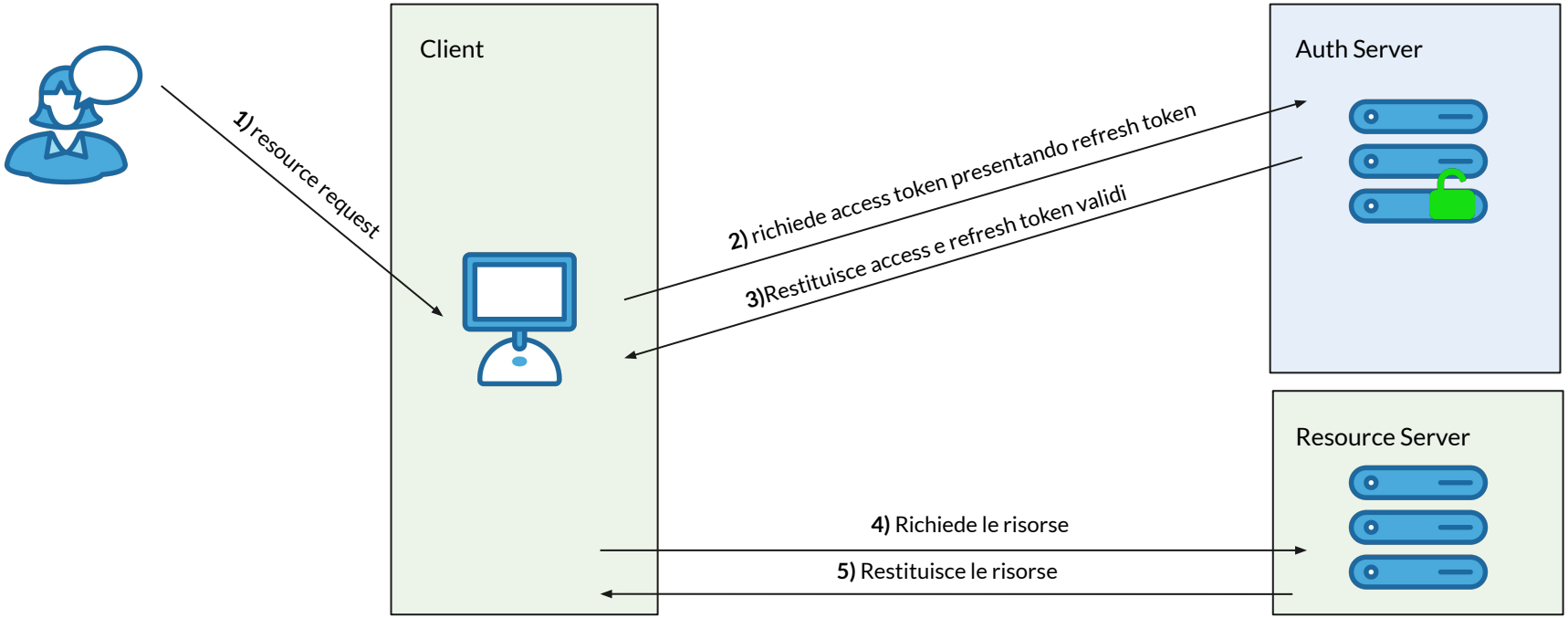
Authorization code with PKCE



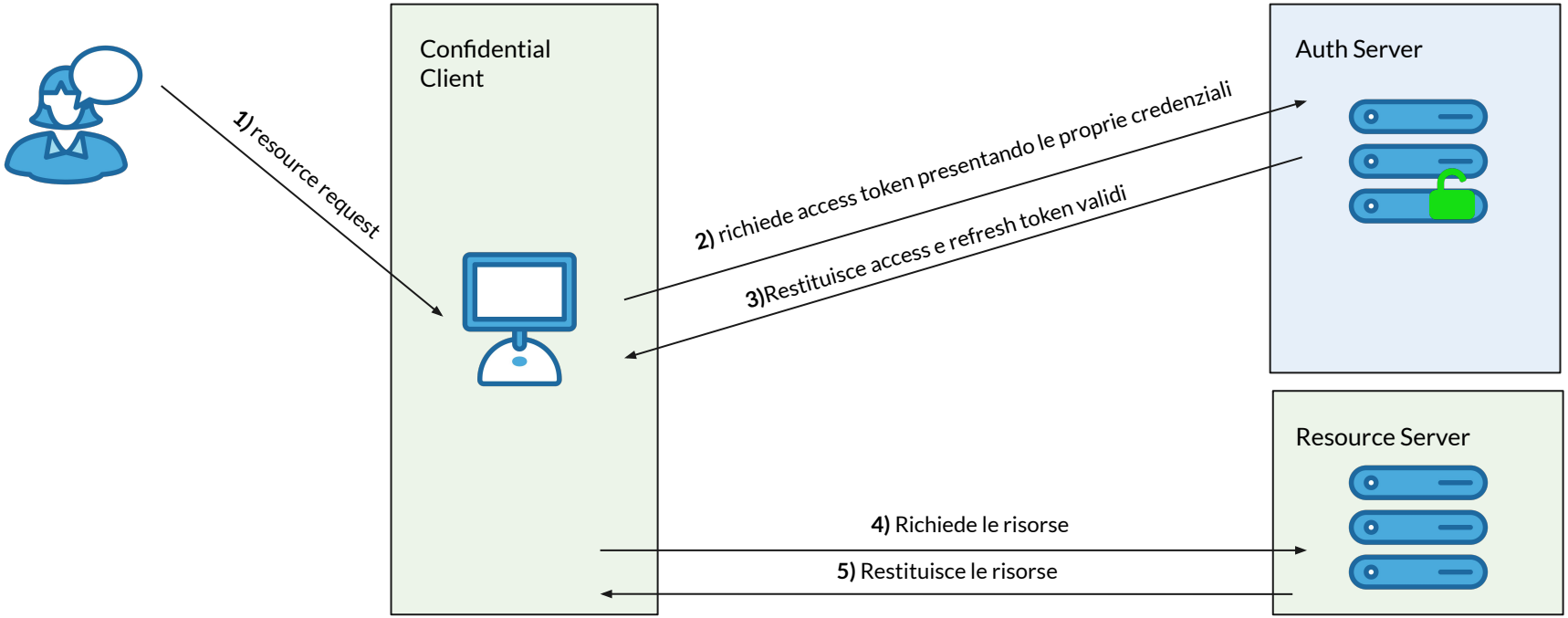
la richiesta del code è un redirect e contiene:

- client_id
- redirect_uri
- response_type=code
- code_challenge
- code_challenge_method

Refresh Token



Client Credentials Token



C’era questo tra i layout ma non so come usarlo

Client Registrations

Dopo mi è venuto in mente come usarlo

Chi può usare questi Grant type

Le specifiche definiscono due tipi di client ed in base alle loro caratteristiche possono o meno usare un determinato Grant Type. I Client Type sono:

- **Public**: Non sono in grado di mantenere in maniera sicura dei dati confidenziali (le credenziali)
  - Applicazioni eseguite in un Web Browser (es: tutte le web app)
  - Client eseguiti sul sistema operativo locale dell'utente (es: il client di un servizio db)
  - Client eseguiti su dispositivi mobili (es: tutte le app)
- **Confidential**: Sono in grado di mantenere in maniera sicura dei dati confidenziali:
  - Applicazioni eseguite lato server

Client Registration

I client per poter richiedere un token all authorization server hanno la necessità di essere registrati presso l'authorization server stesso.

Ci sono tre tipi di Client registration tra le specifiche ma ne esistono diverse non standard implementate dai diversi provider.

- **Web Application** (Confidential Client) mantengo le informazioni confidenziali nel server.
- **User Agent Based Application** (Public client) i nostri front end che girano nel browser molti provider li indicano come **Single Page Application**
- **Native Application** (public client) applicazioni che per loro natura non necessitano di un user agent. es: applicazioni che girano direttamente sul sistema operativo

Open Id Connect

Open ID Connect, **OIDC** per gli amici, e solo un layer aggiuntivo a Oauth2.

Se l'authorization server è configurato per supportare OIDC e se durante la richiesta del token tra gli scope richiesti è presente lo scope **openId** oltre agli altri token verrà fornito un **id_token** contenente informazioni sull'identità dell'utente.

```
//Payload

{
  "iss": "http://my-domain.auth0.com",
  "sub": "auth0|123456",
  "aud": "my_client_id",
  "exp": 1311281970,
  "iat": 1311280970,
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "picture": "http://example.com/janedoe/me.jpg"
}
```

Grazie a tutti!