

# **OAUTH: AUTHENTICATION PIGGYBACK**

---

*Because we all trust Google, right?*





# OAUTH

*Standard protocol for auth piggybacking*

**Your Application**

**User**

**3rd Party Authority**

**“consumer”**

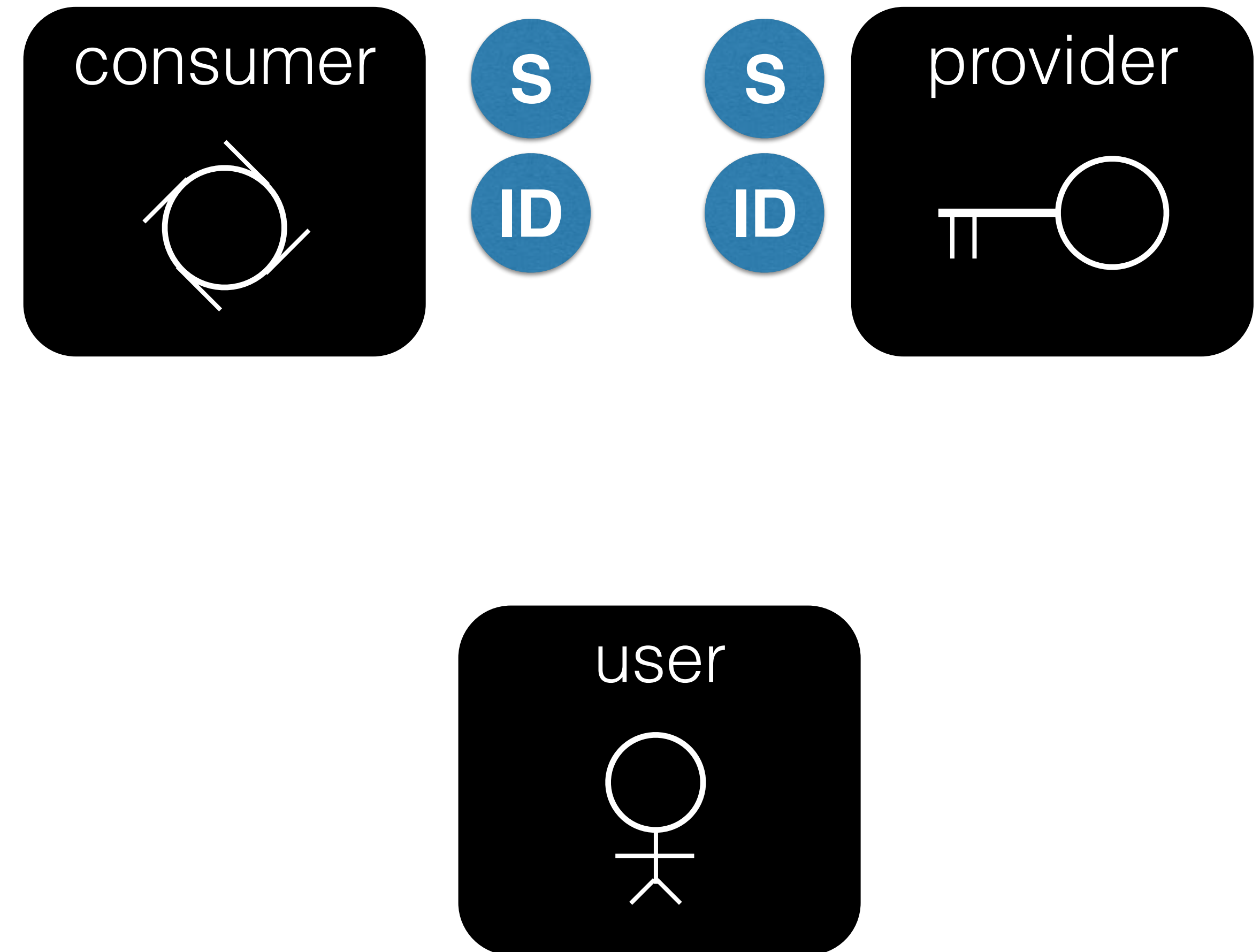
**“user”**

**“provider”**



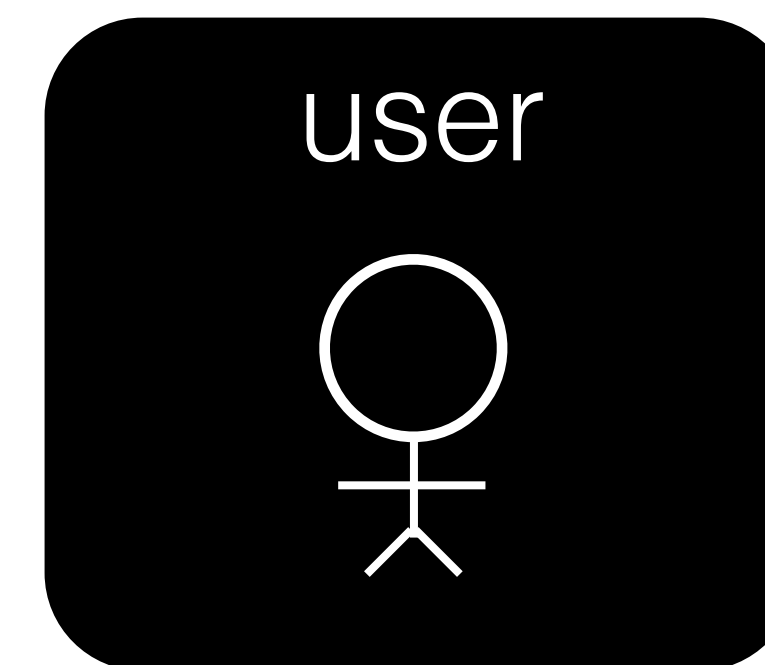
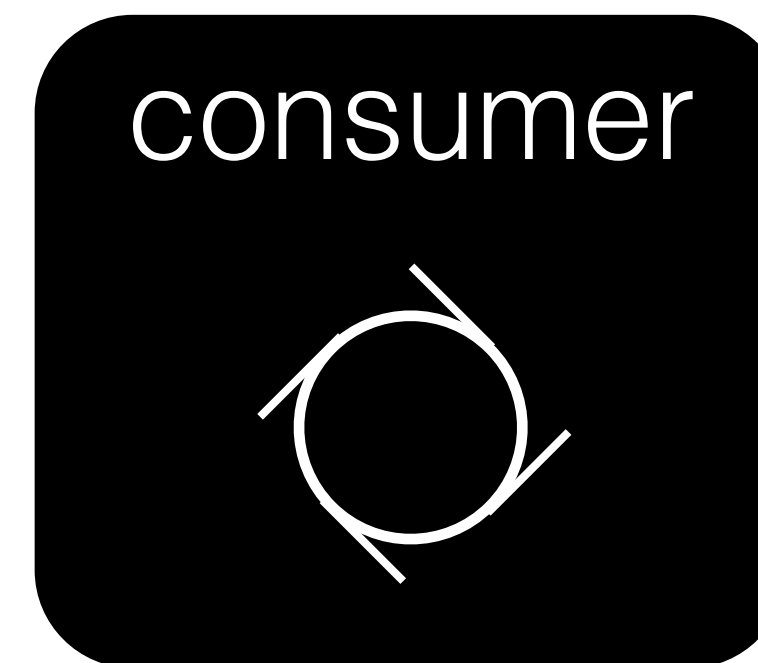
# OAUTH - PREP

- **Consumer app has a registered dev account with the provider.**
- **Provider gives consumer a public client ID and private client Secret.**
- **Both services hold on to these credentials so the consumer can prove who it is to the provider.**





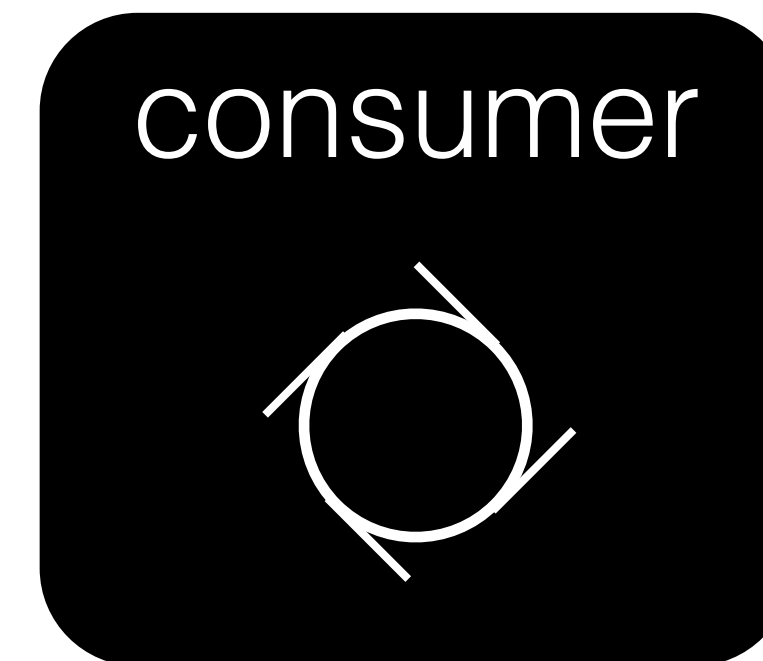
# OAUTH - PETITION



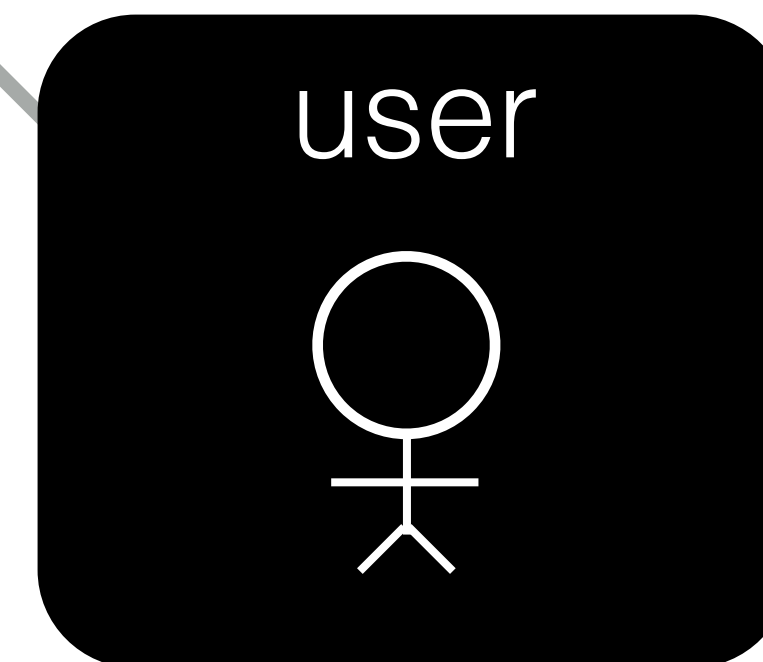


# OAUTH - PETITION

1. Request: load login



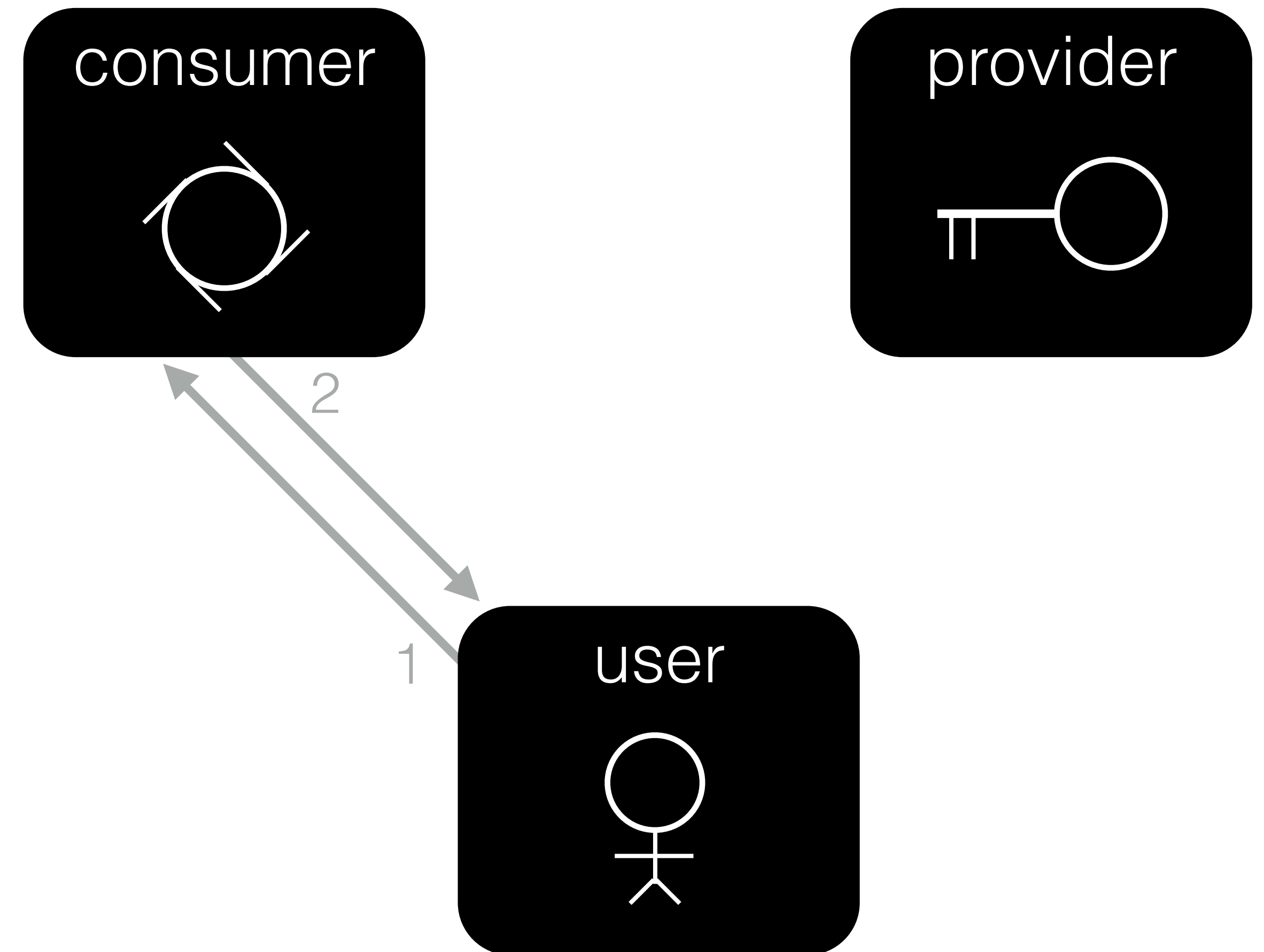
1





# OAUTH - PETITION

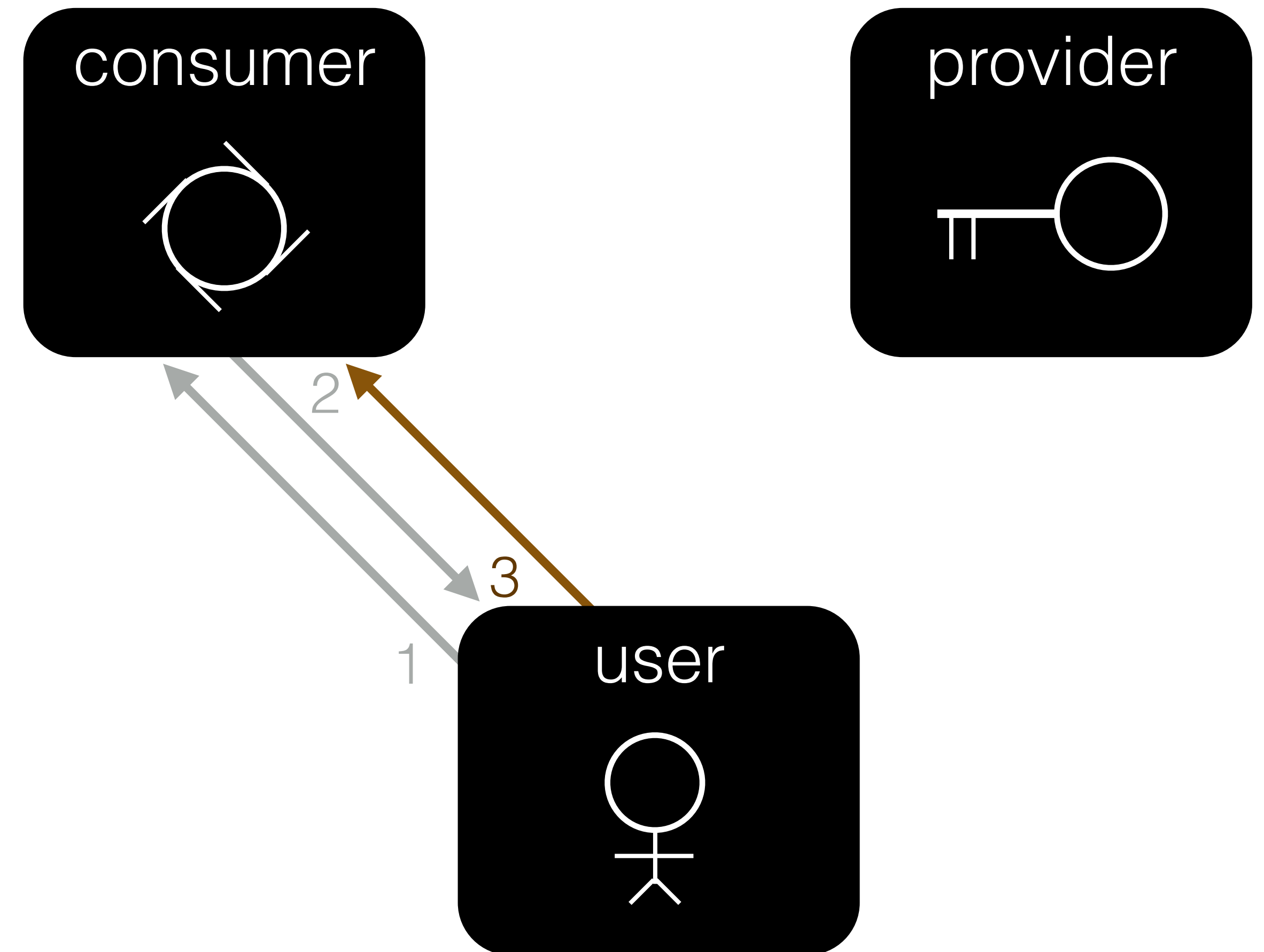
- 1.Request: load login
- 2.Response: rendered login





# OAUTH - PETITION

1. Request: load login
2. Response: rendered login
3. Request: login through provider

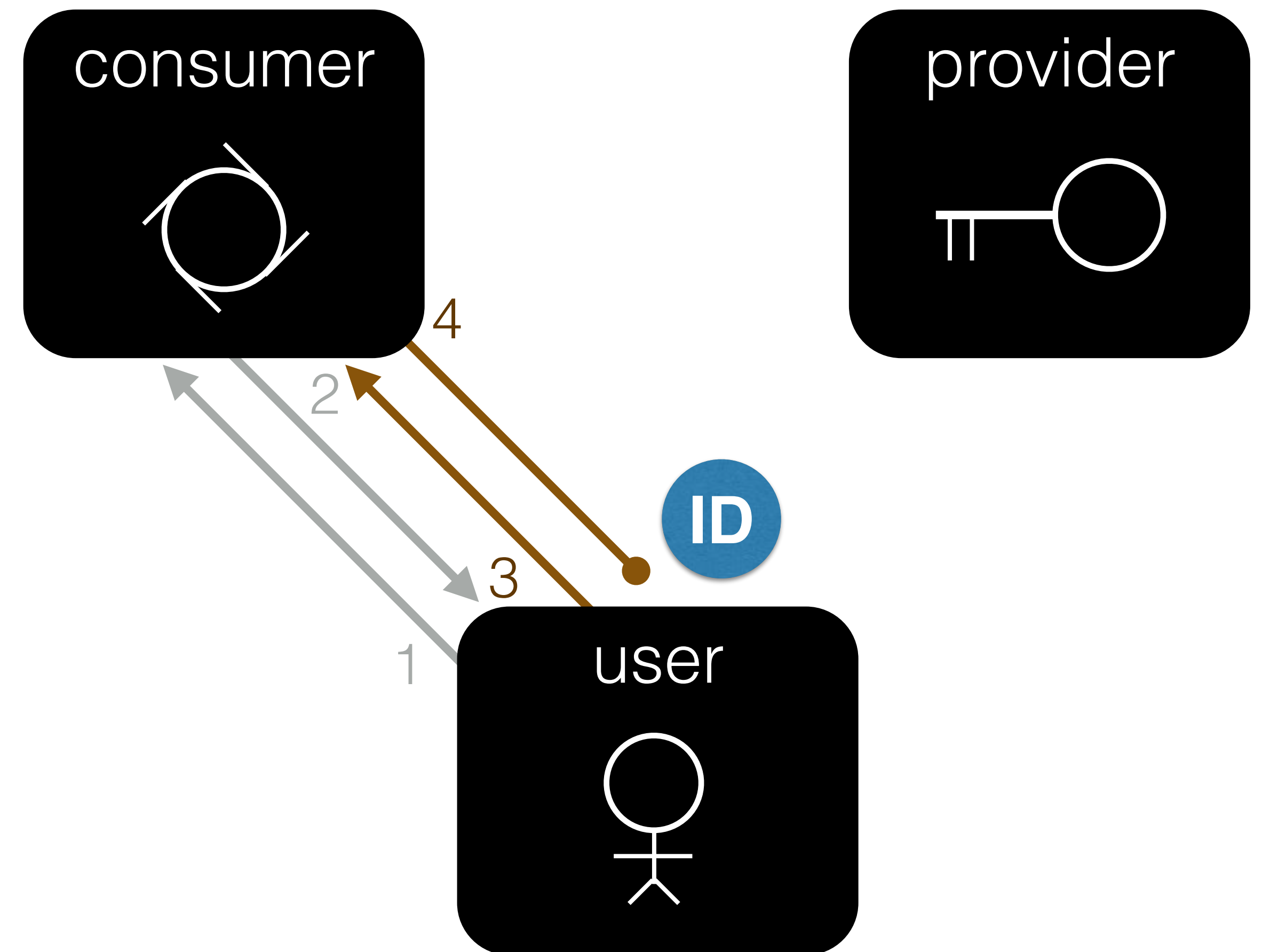






# OAUTH - PETITION

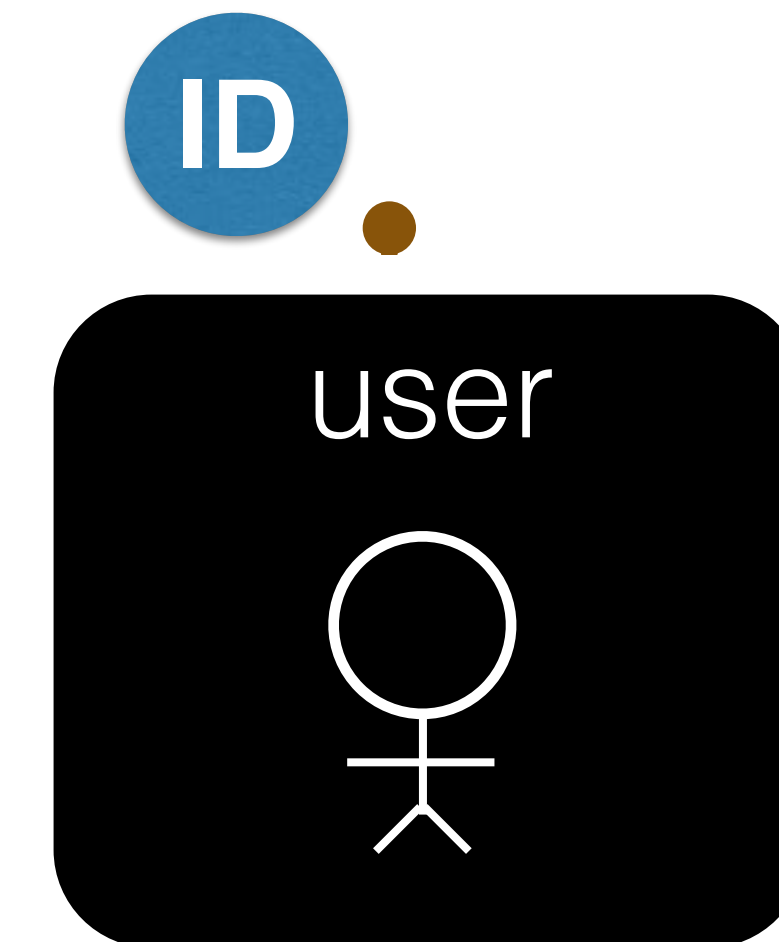
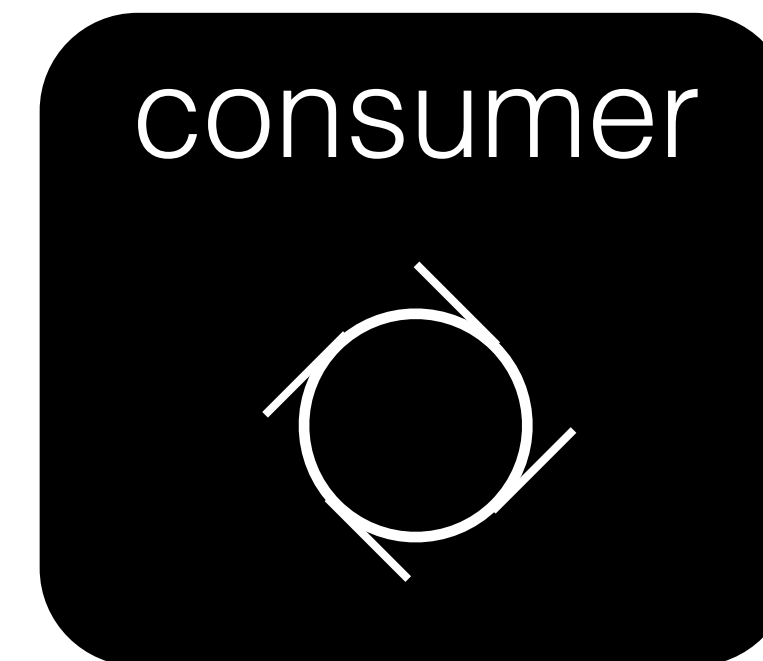
1. Request: load login
2. Response: rendered login
3. Request: login through provider
4. Response: redirect to provider



[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

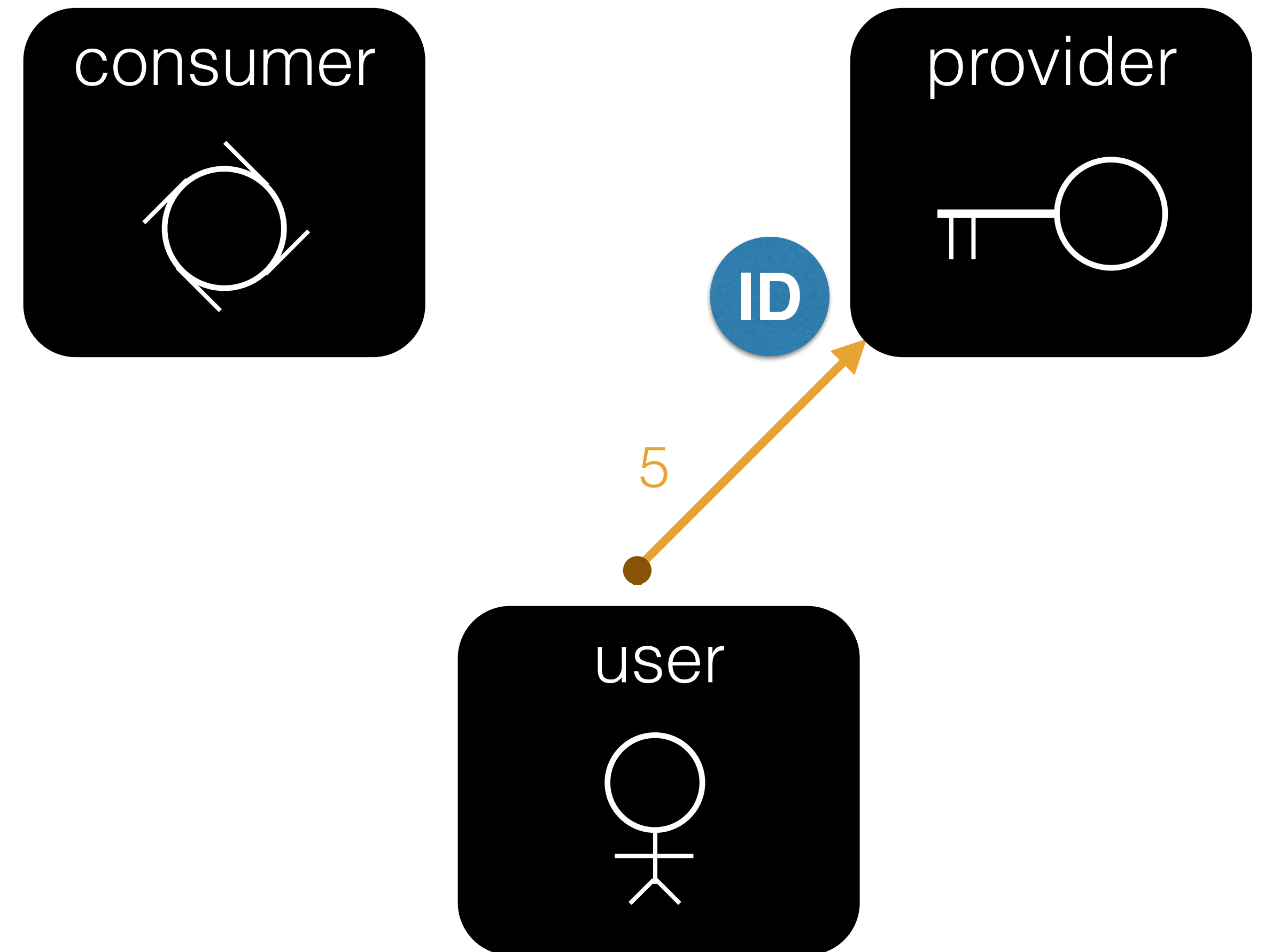


[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

5.Request: load login

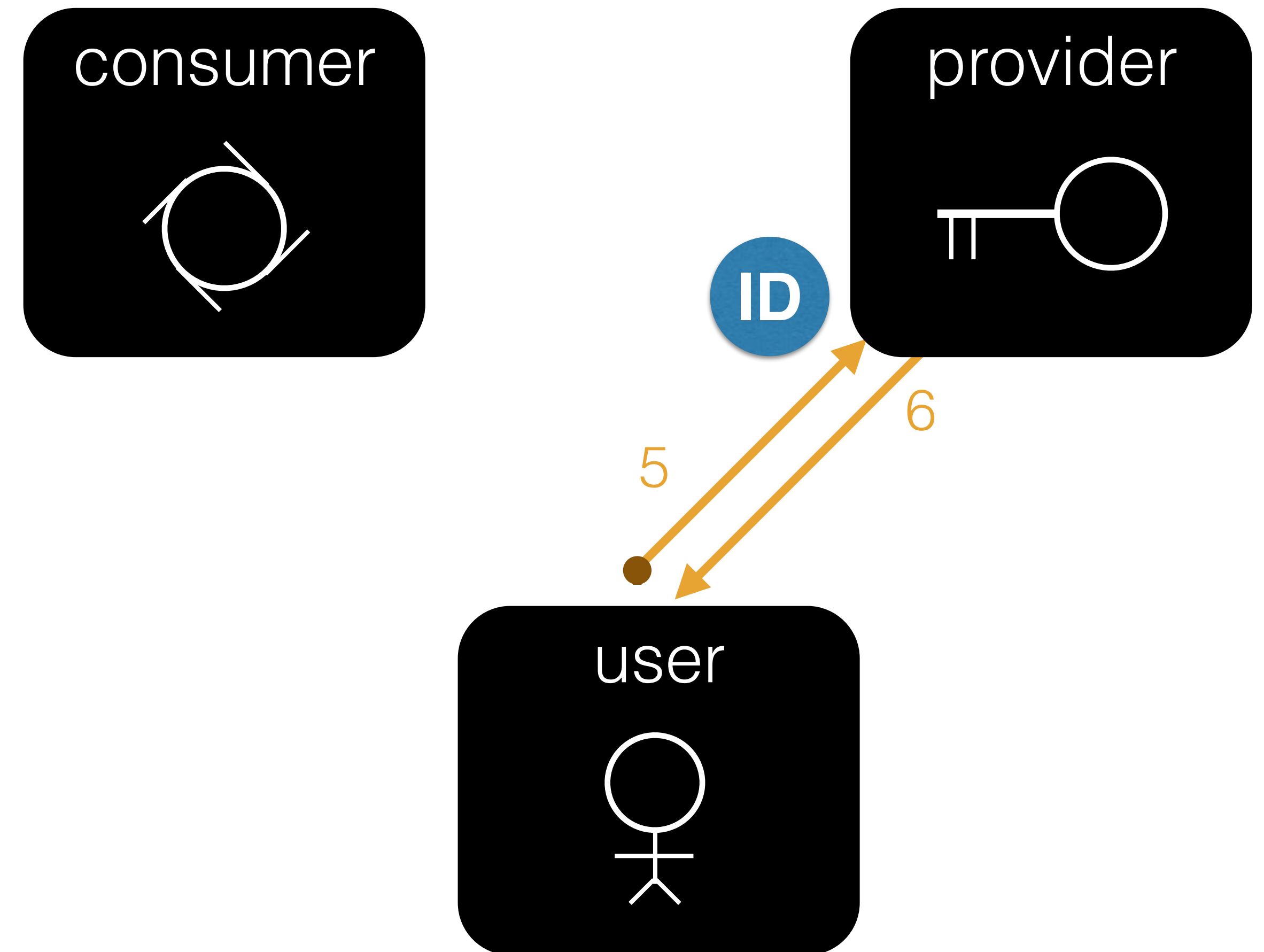


[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

- 5.Request: load login
- 6.Response: rendered login

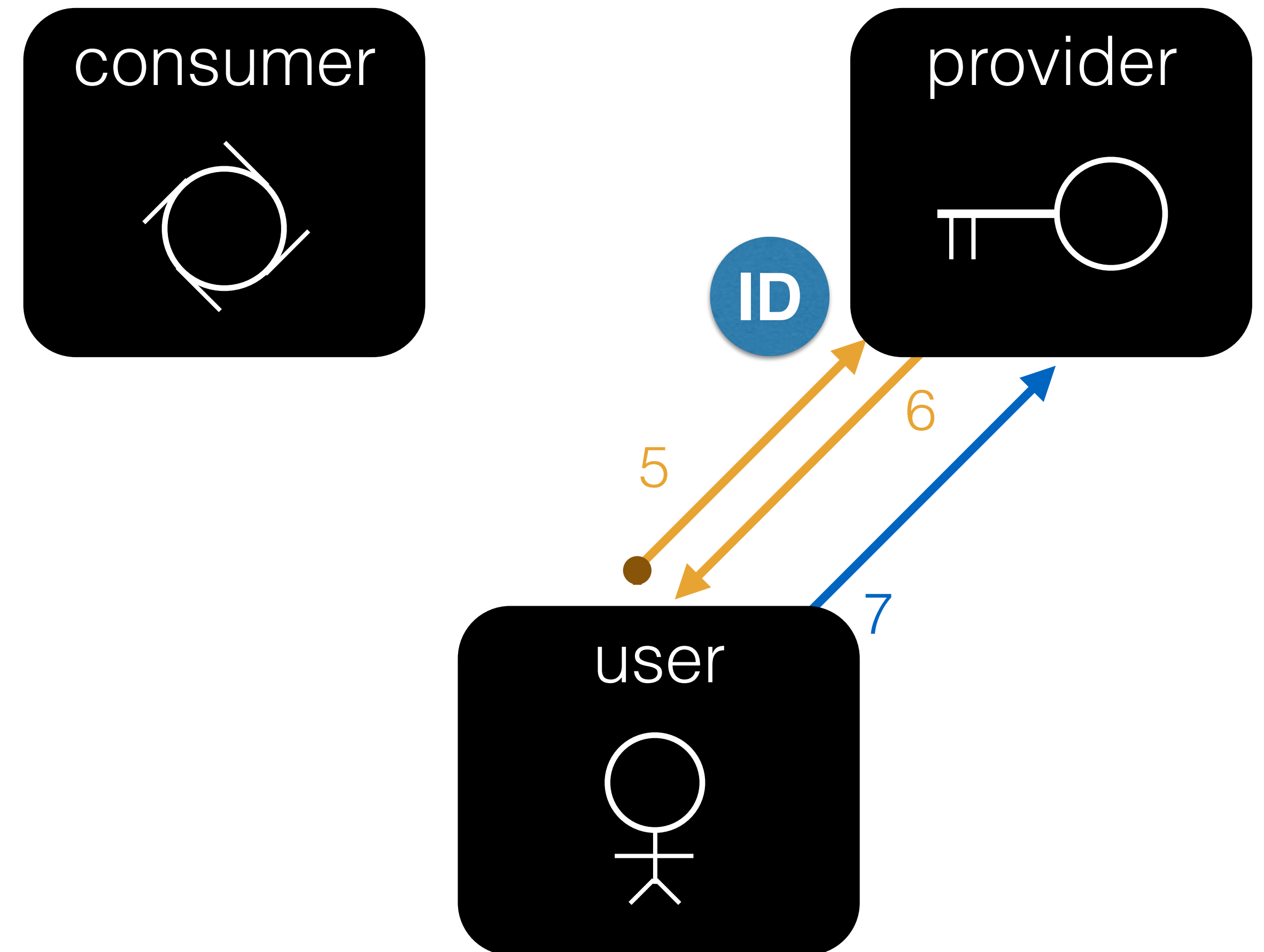


[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider

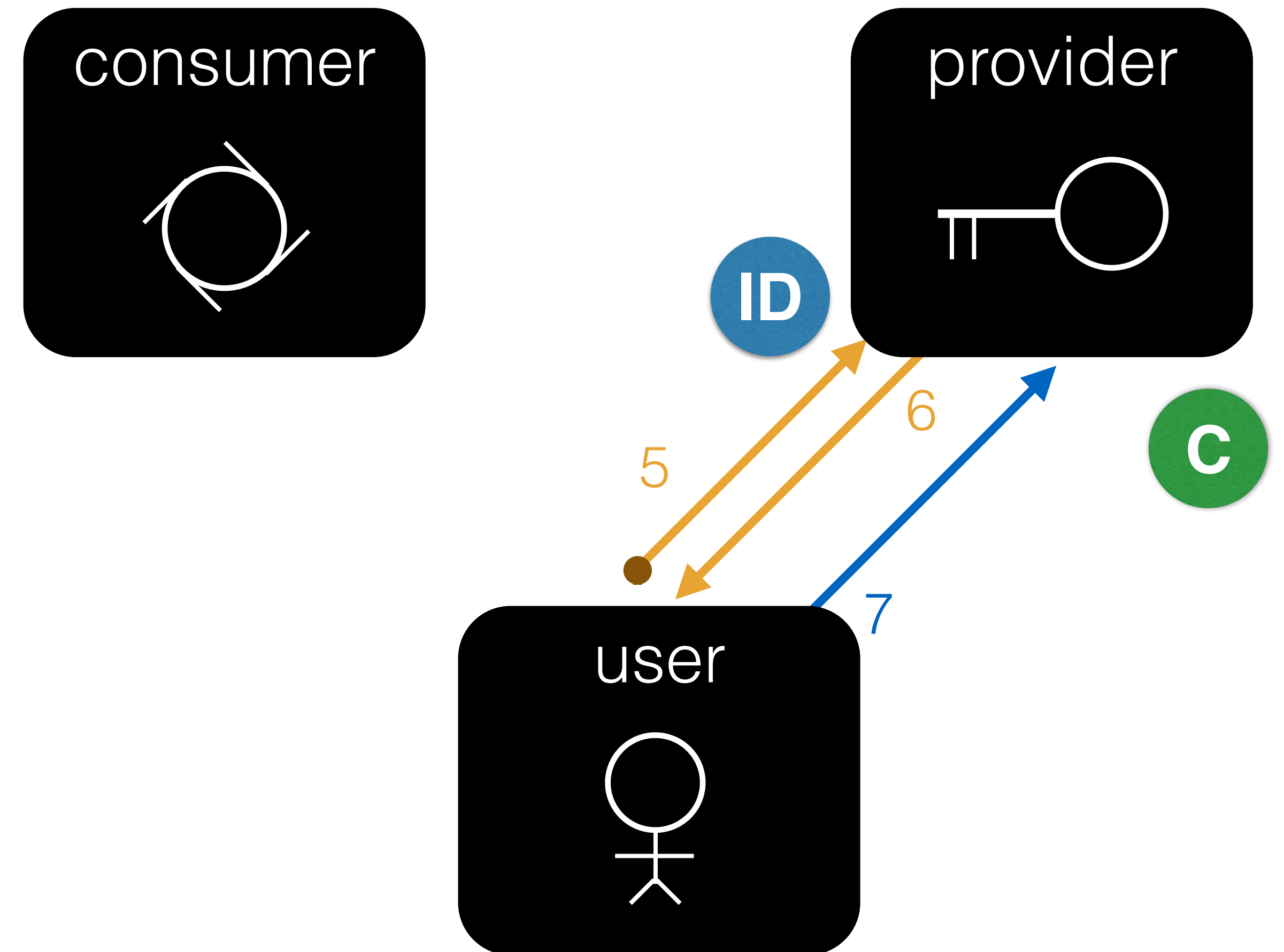


[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider



<http://consumer.com/confirm?authcode=789>

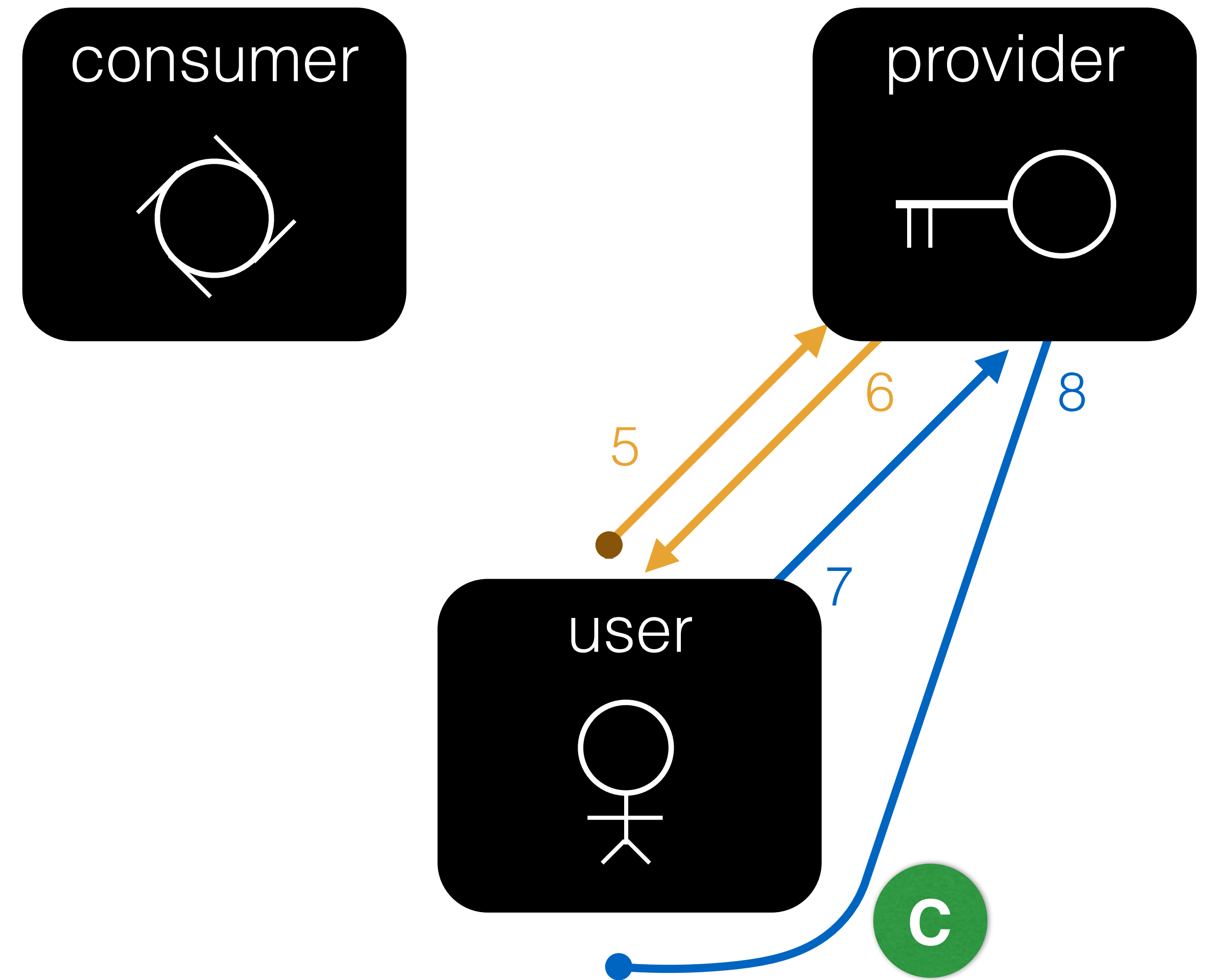
[http://provider.com/oauth/authorize?client\\_id=123&callback\\_url=consumer.com/confirm&scope=read](http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read)



# OAUTH - USER AUTHENTICATION

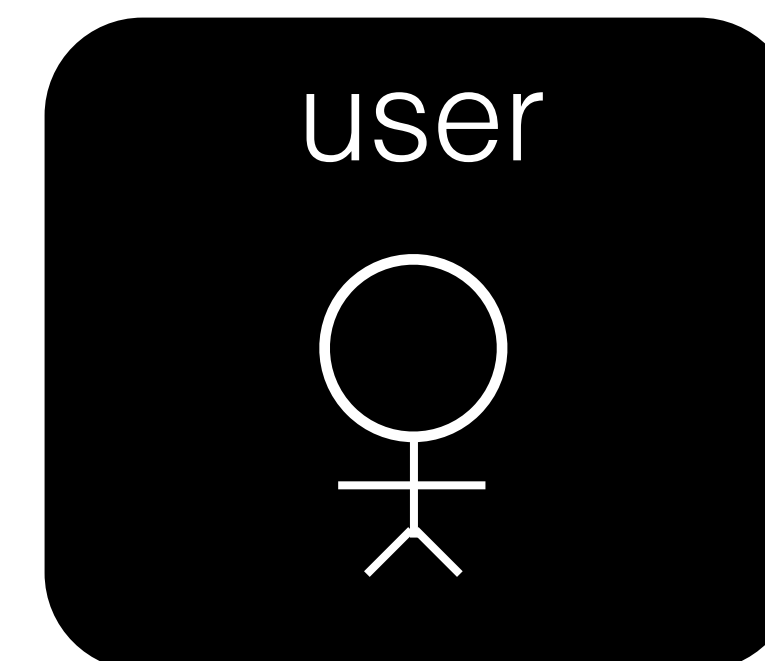
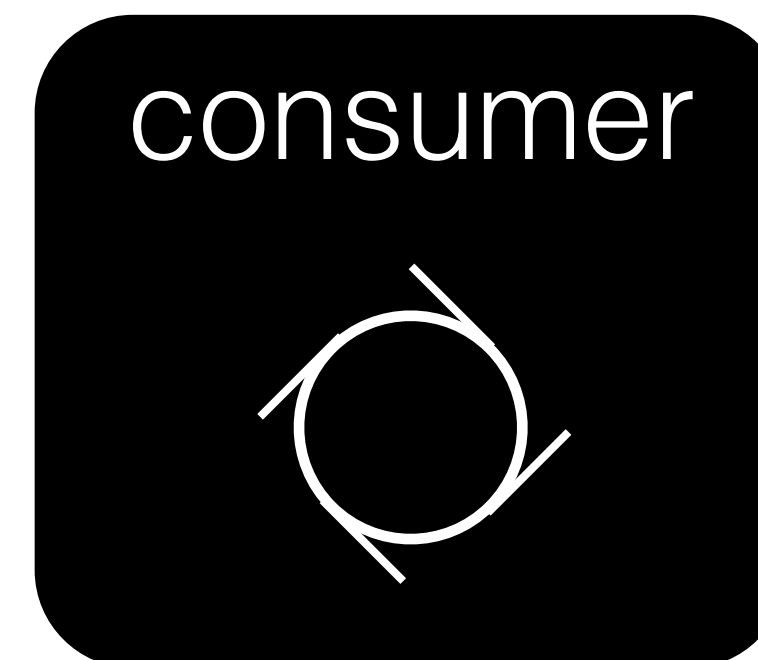
- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider
- 8. Response: redirect callback URL

<http://consumer.com/confirm?authcode=789>

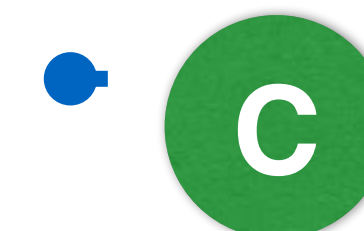




# OAUTH - APP AUTHENTICATION



<http://consumer.com/confirm?authcode=789>



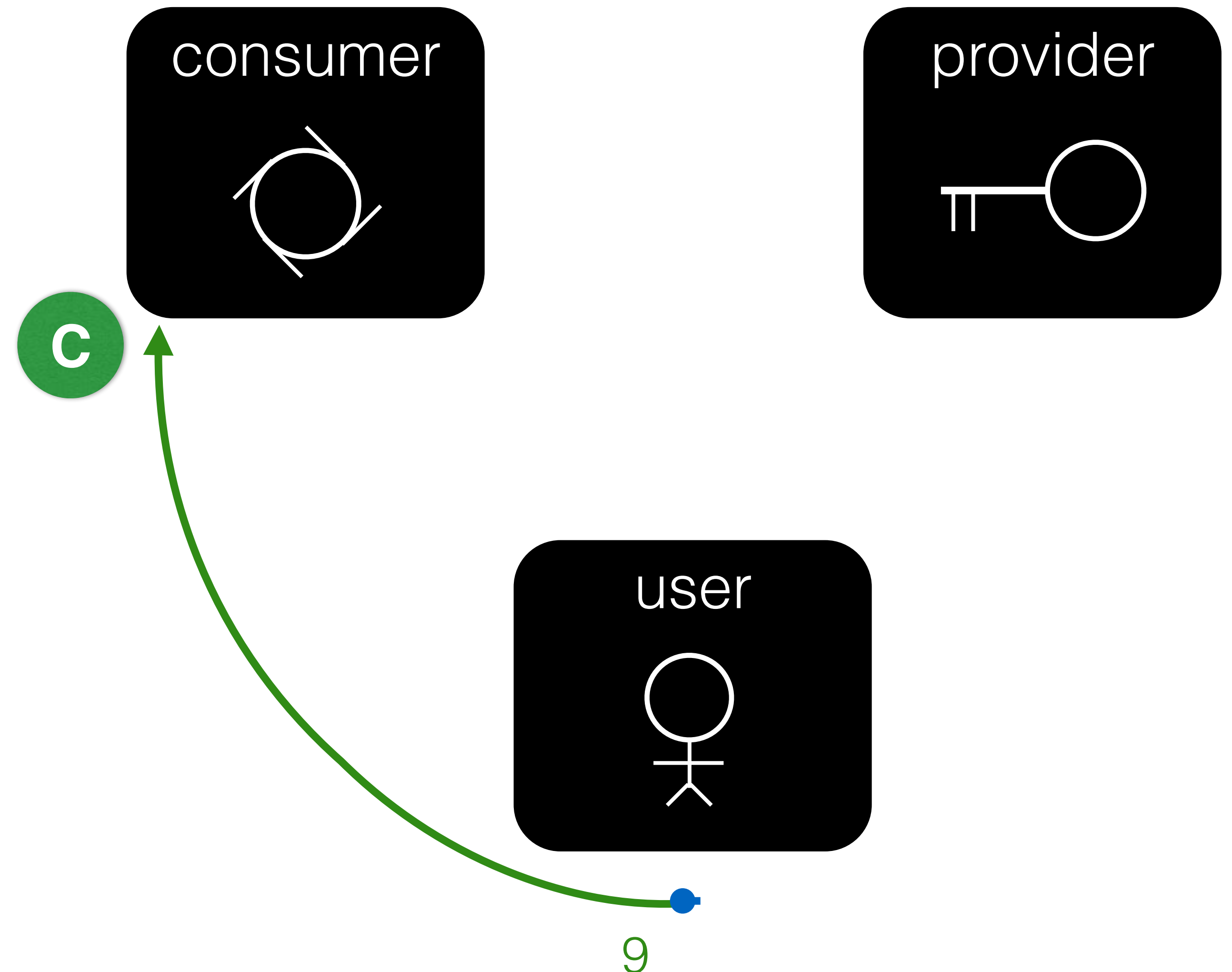




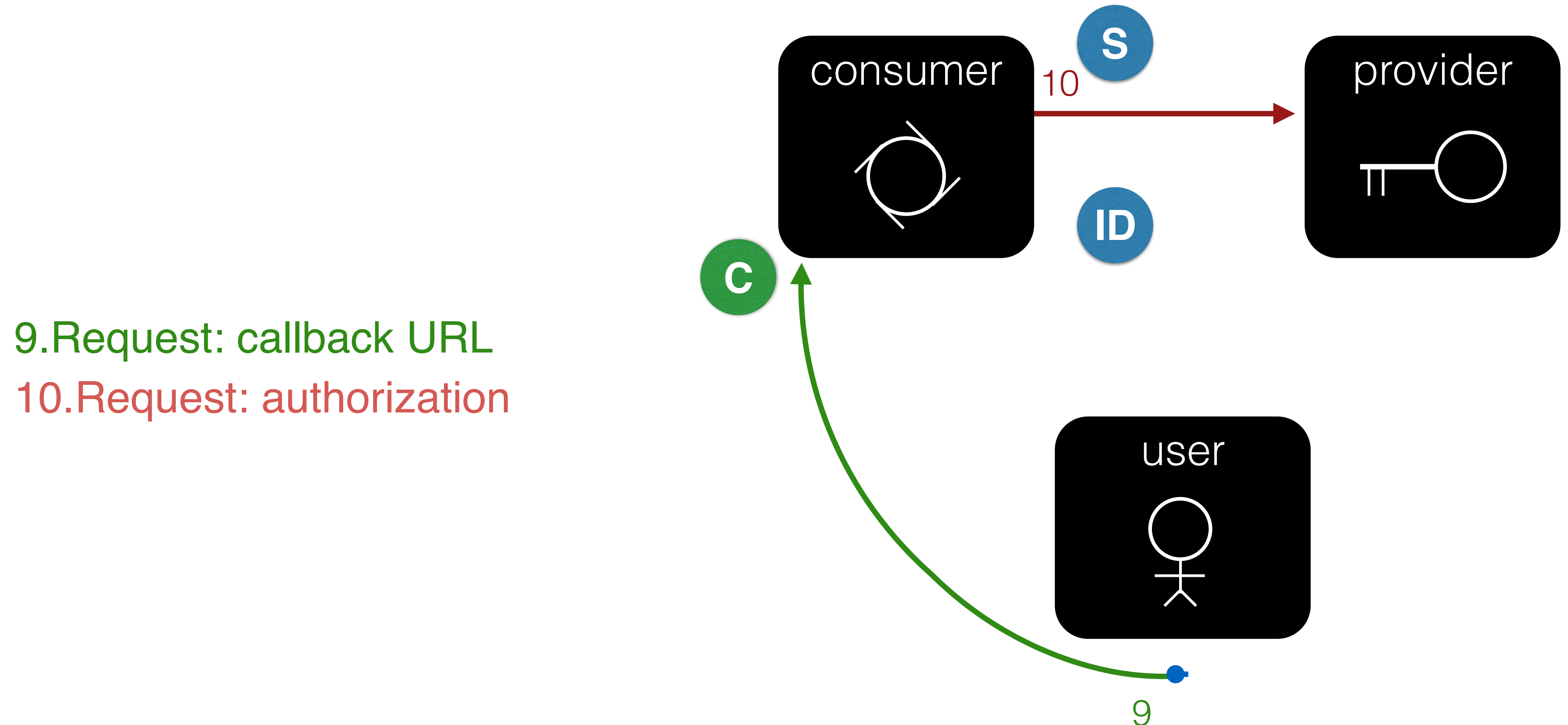
# OAUTH - APP AUTHENTICATION

9.Request: callback URL

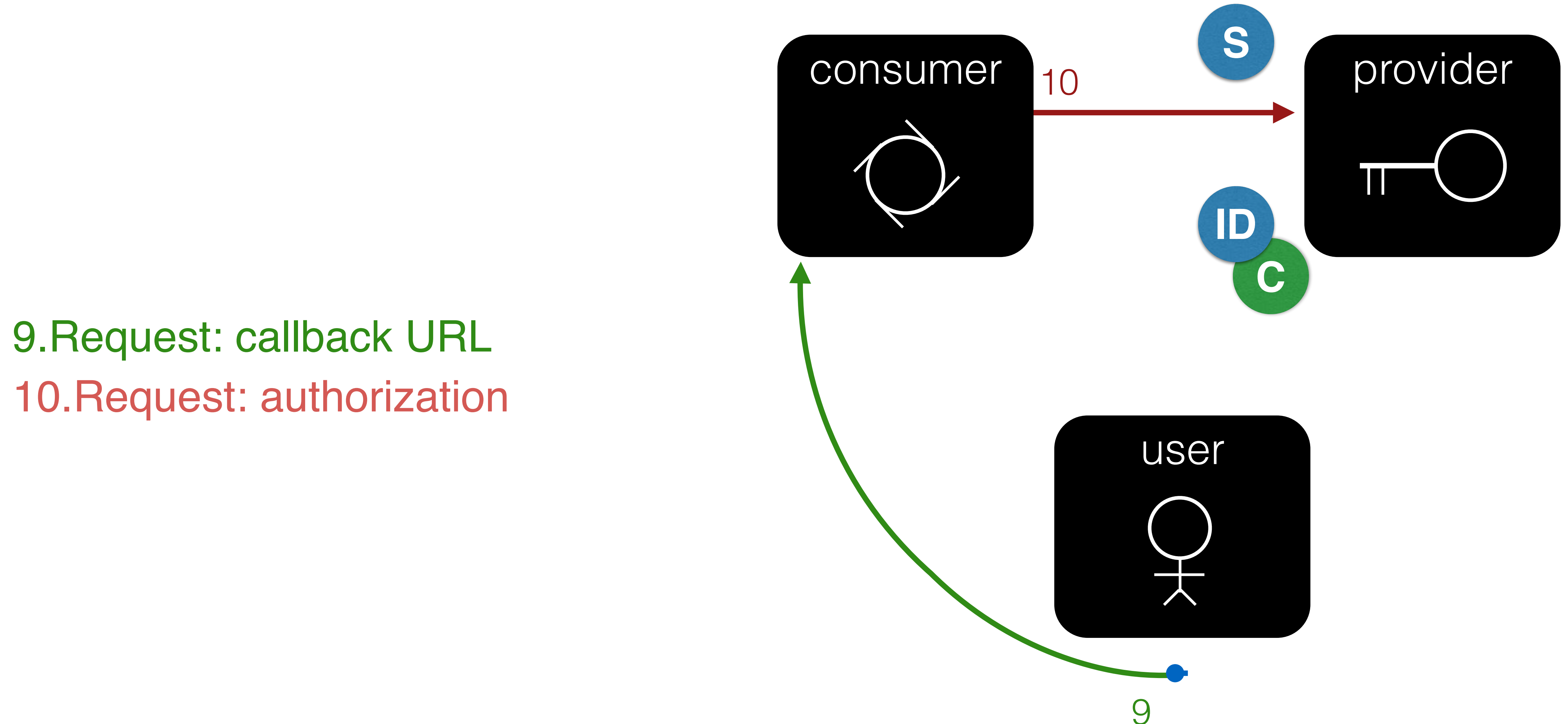
<http://consumer.com/confirm?authcode=789>



# OAUTH - APP AUTHENTICATION

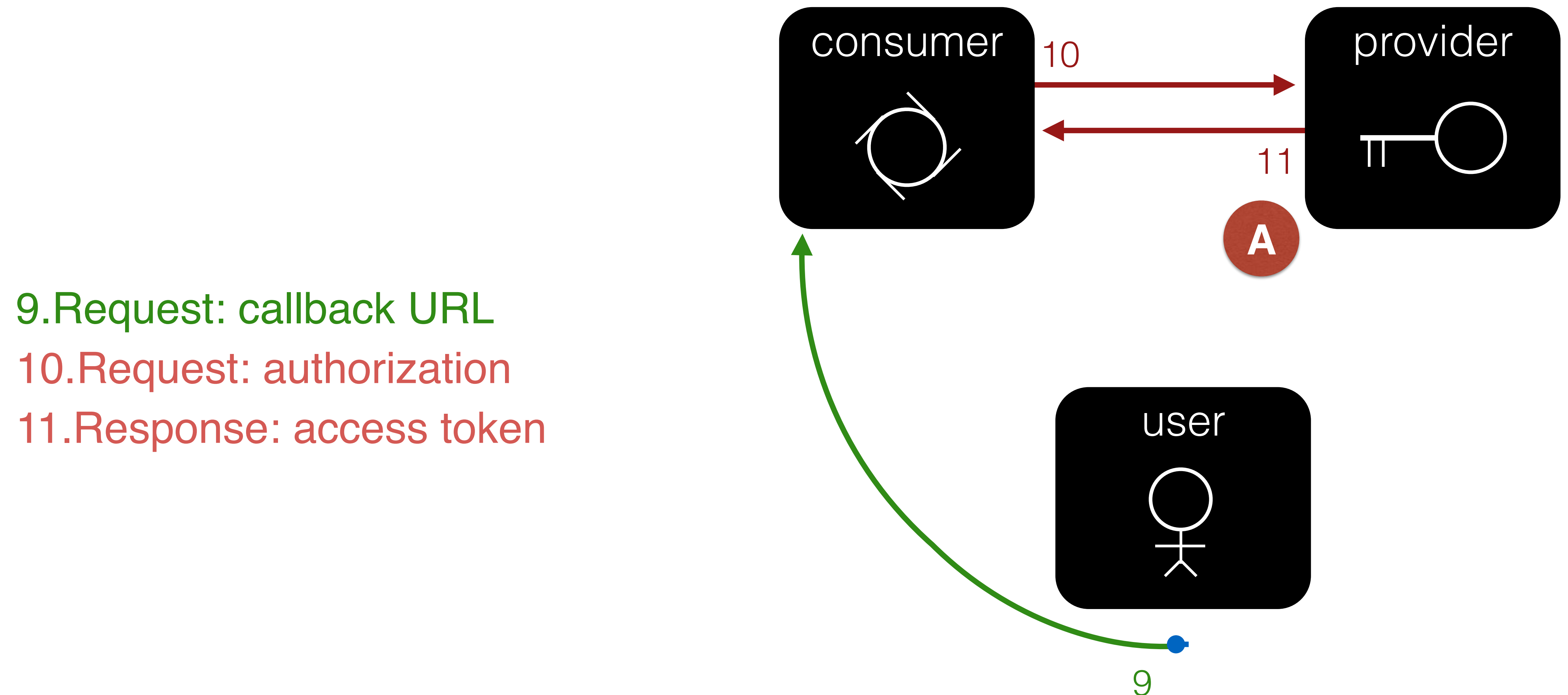


# OAUTH - APP AUTHENTICATION



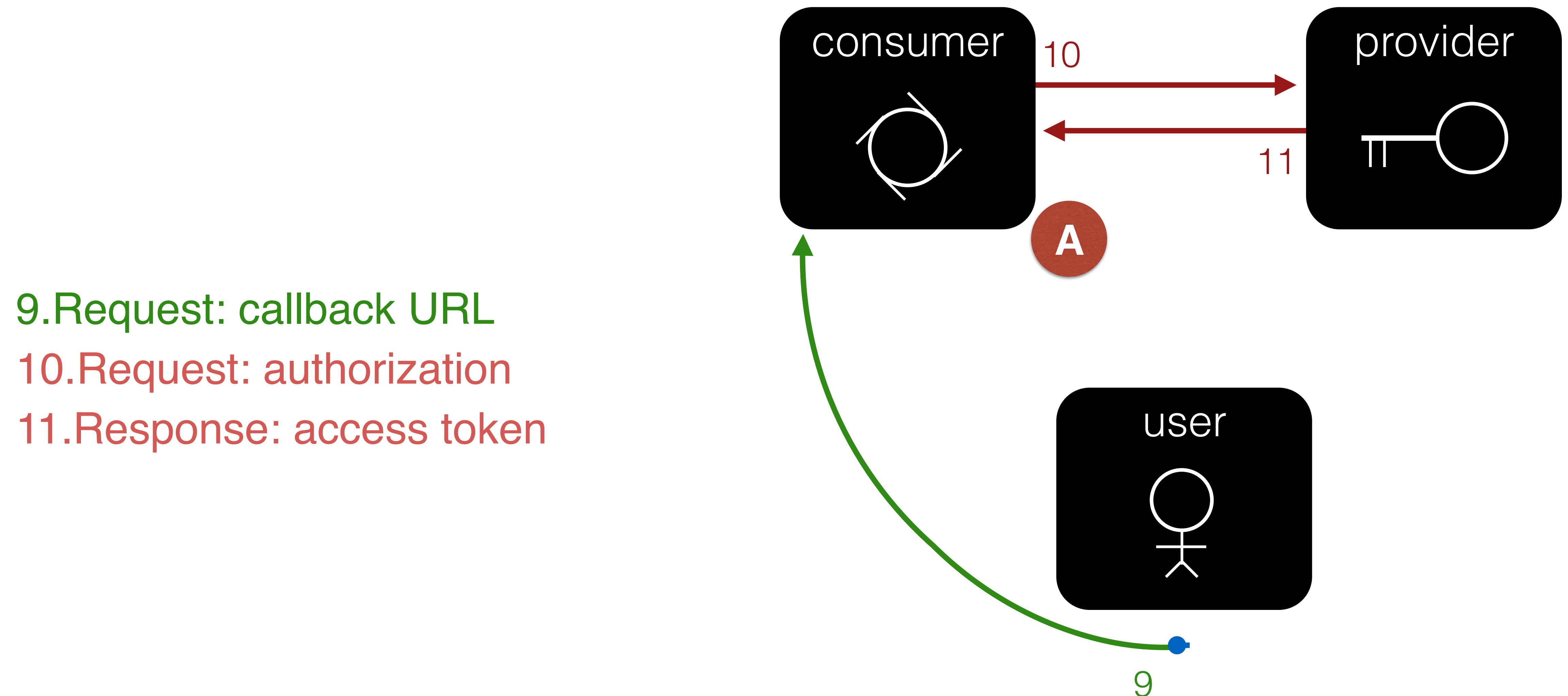


# OAUTH - APP AUTHENTICATION



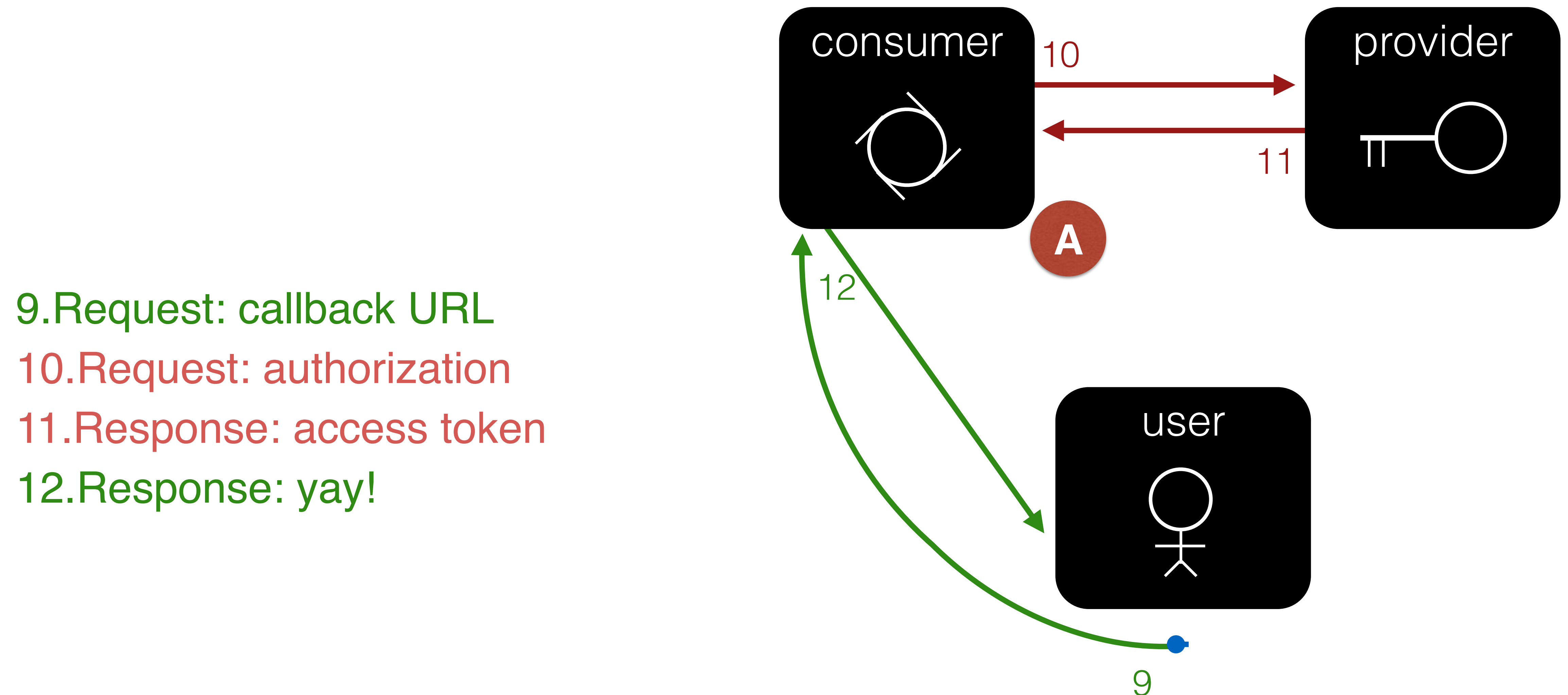


# OAUTH - APP AUTHENTICATION





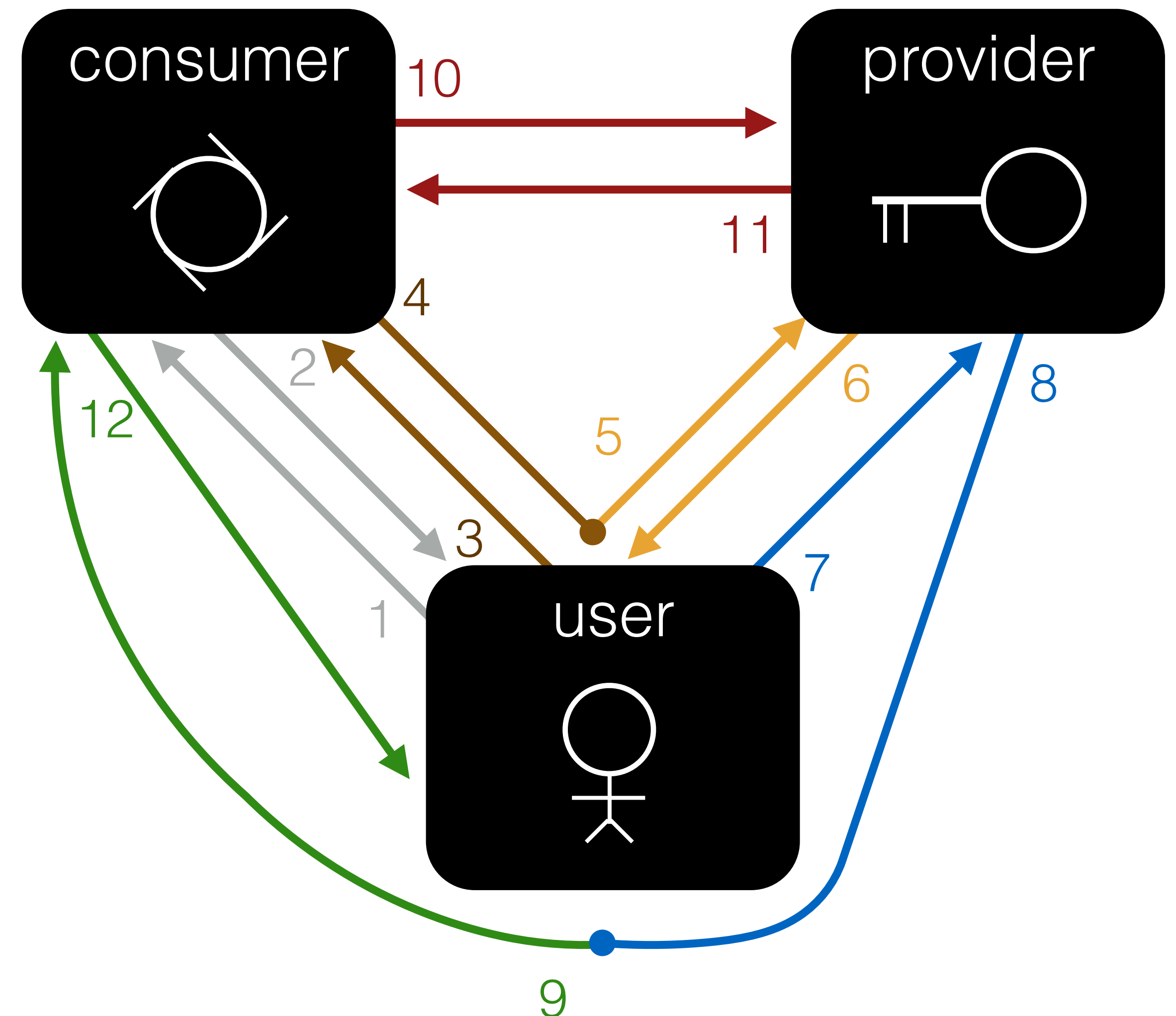
# OAUTH - APP AUTHENTICATION





# OAUTH - ALL OF IT

1. Request (user to app): load login page
2. Response (app to user): rendered login page
3. Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]
4. Response (app to user): redirect to provider login, passing along (to provider) an app id, a success "callback URL", and a permissions "contract" [app transfers this petition to provider]
5. Request (implicit, user to provider): load login page
6. Response (provider to user): rendered login page
7. Request (user to provider): login to provider [the user signs the contract]
8. Response (provider to user): on success, redirect to callback URL, passing along a new temporary code [the provider approves the user's signature]
9. Request (implicit, user to app): initiate callback
10. Request (app to provider): request for authorization given temporary code and app secret key [the app signs the contract]
11. Response (provider to app): on success, passes back an access token [the provider approves the app's signature and puts the contract into effect]
12. Response (app to user): we're good to go!



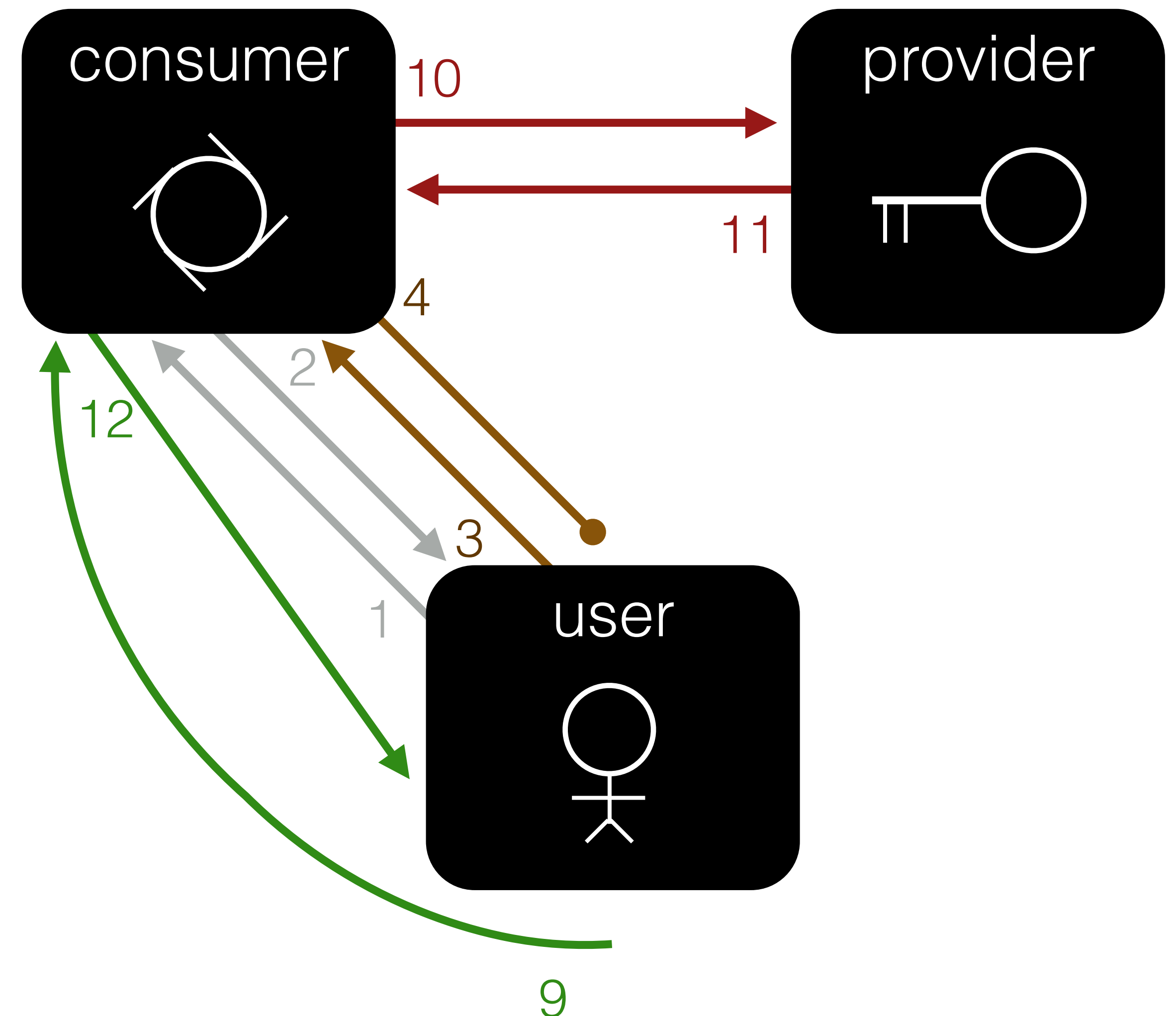




# OAUTH - CONSUMER ROLE

1. Request (user to app): load login page
2. Response (app to user): rendered login page
3. Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]
4. Response (app to user): redirect to provider login, passing along (to provider) an app id, a success "callback URL", and a permissions "contract" [app transfers this petition to provider]

9. Request (implicit, user to app): initiate callback
10. Request (app to provider): request for authorization given temporary code and app secret key [the app signs the contract]
11. Response (provider to app): on success, passes back an access token [the provider approves the app's signature and puts the contract into effect]
12. Response (app to user): we're good to go!





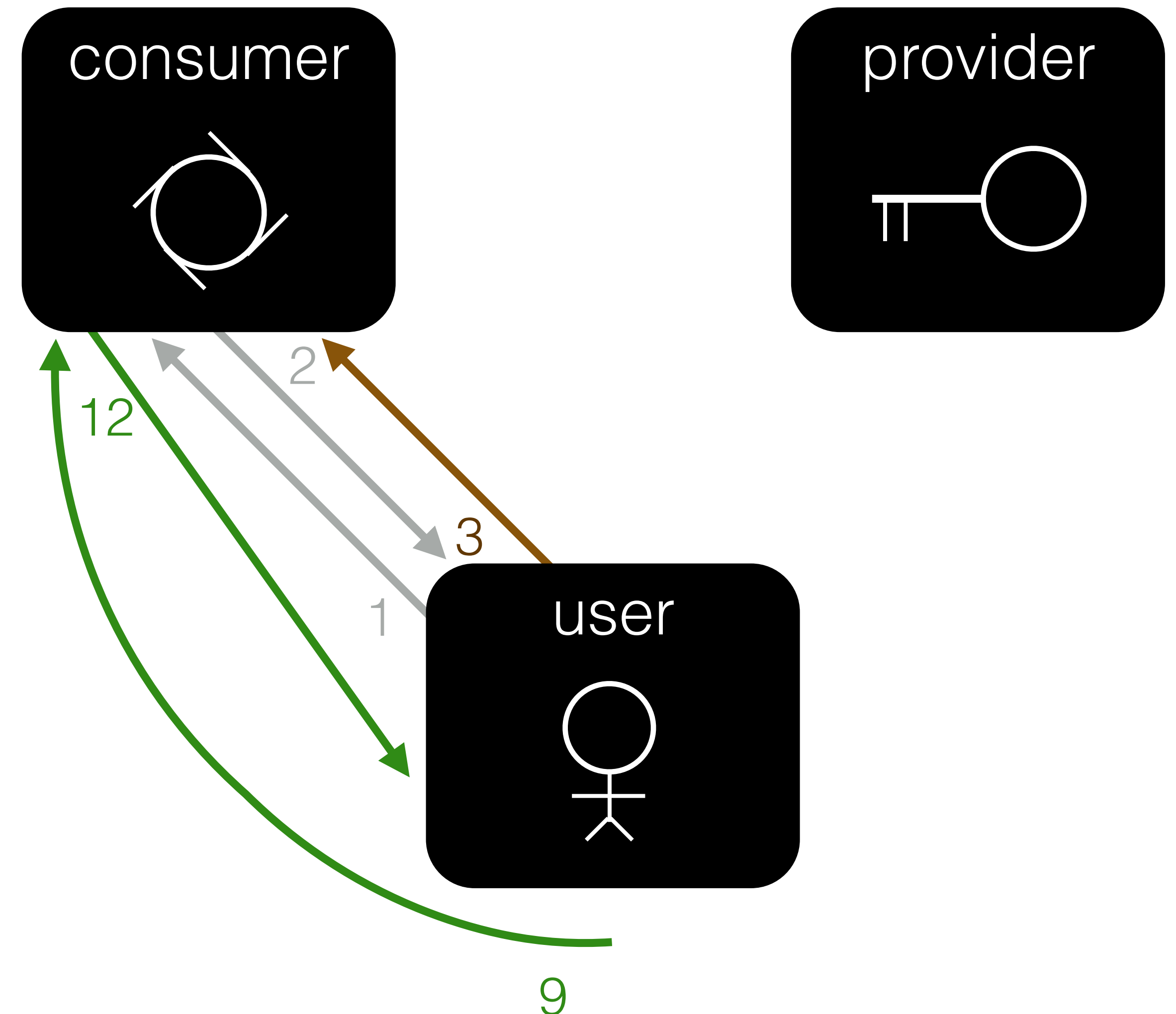


# OAUTH - CONSUMER ROLE WITH PASSPORT

1. Request (user to app): load login page
2. Response (app to user): rendered login page
3. Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]

9. Request (implicit, user to app): initiate callback

12. Response (app to user): we're good to go!



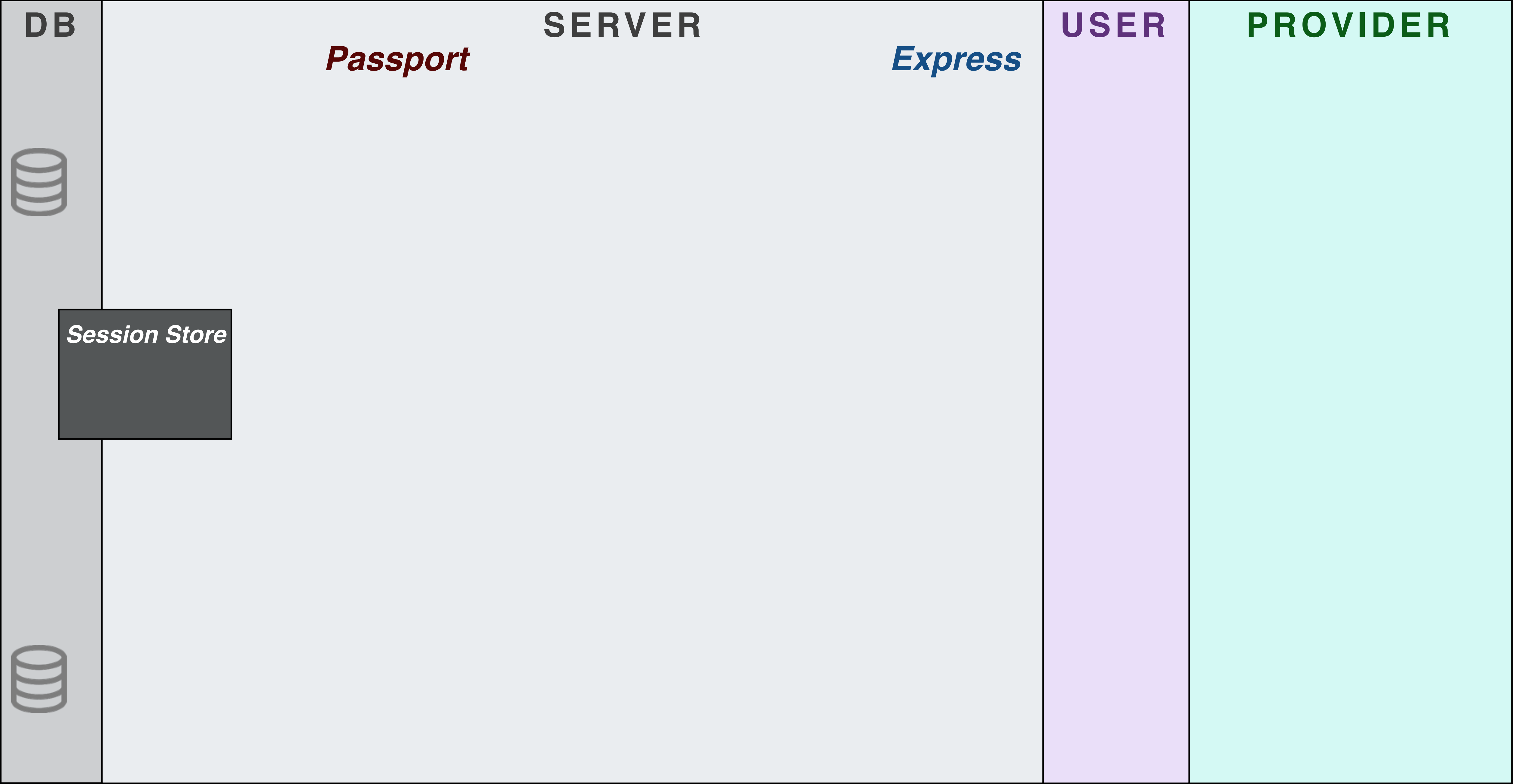


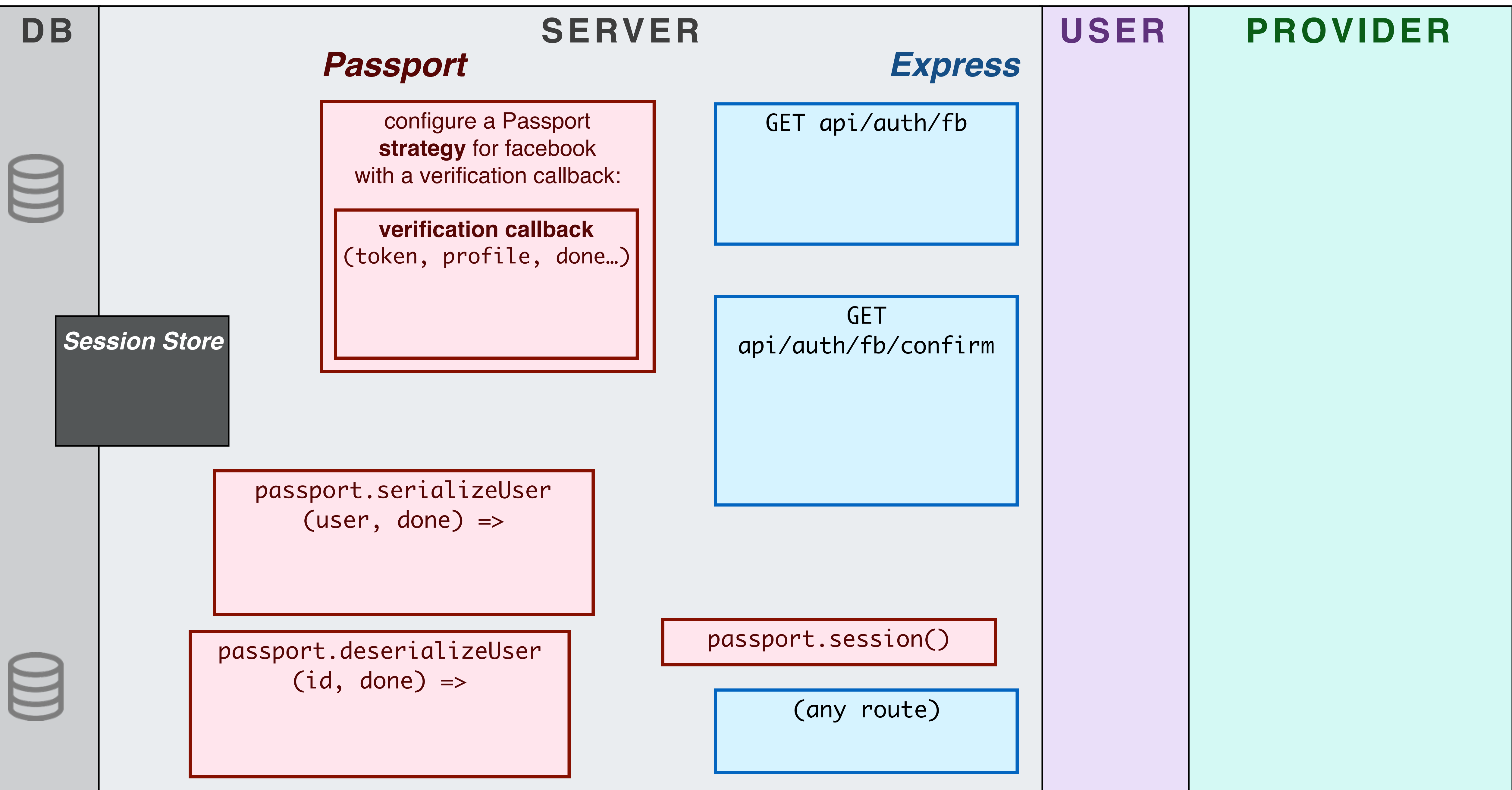
# TL;DR

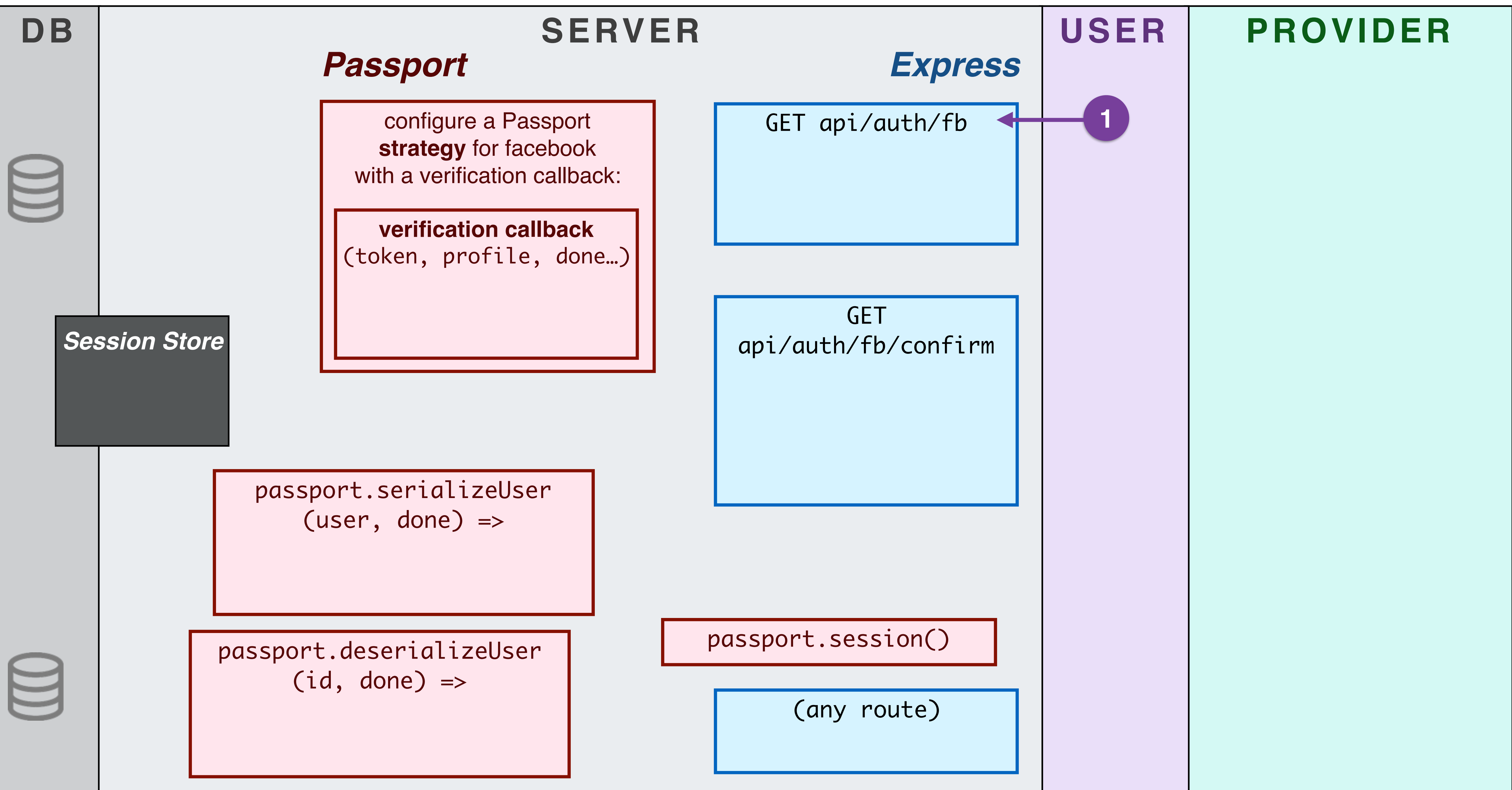
- **Assuming you set everything up right:**
- **Client can log in by requesting GET /auth/google or similar**
- **In routes you will now have req.user to check user info**

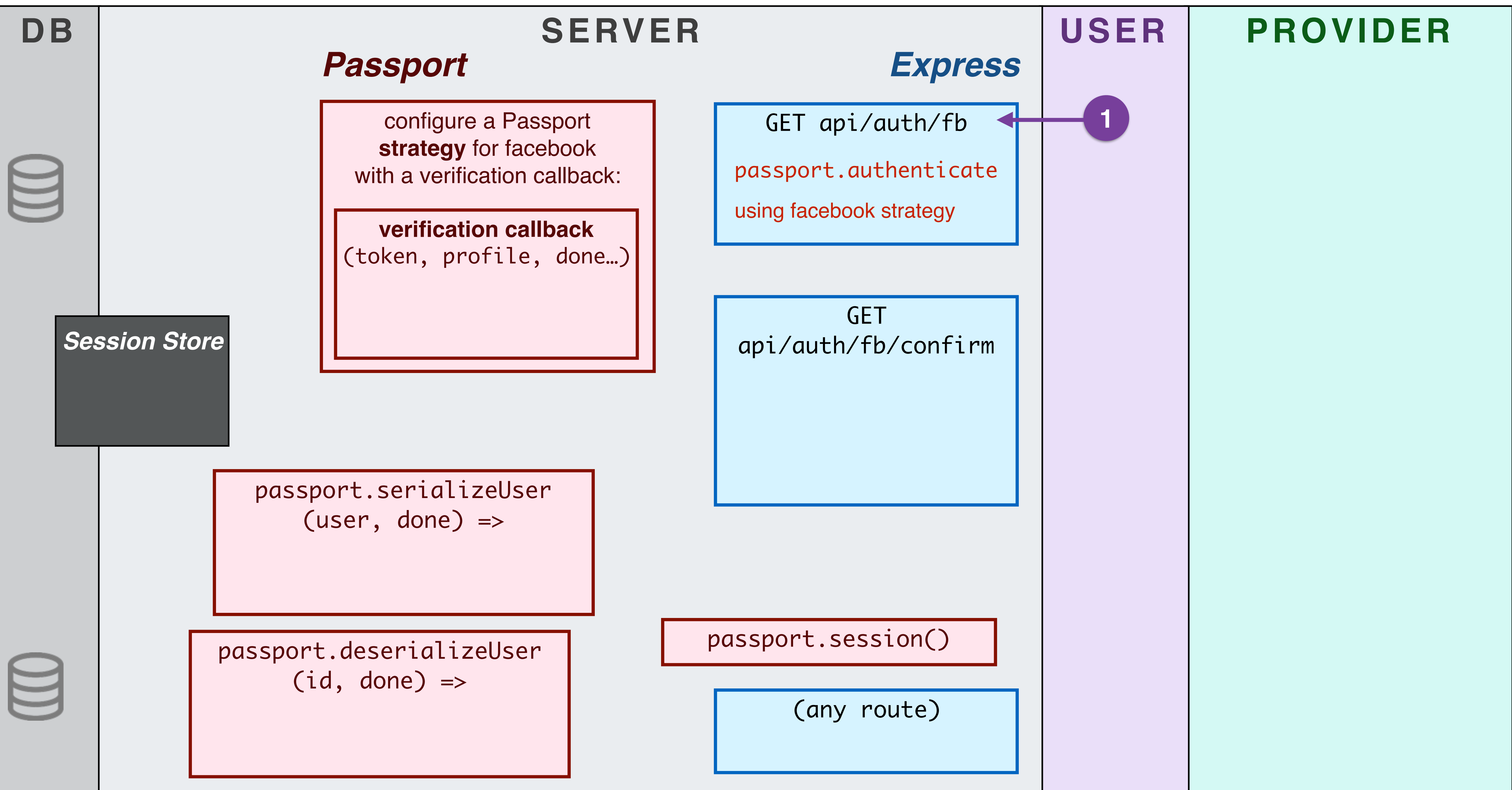
# PASSPORT INGREDIENTS

- **attach passport.session() middleware**
  - Only after express-session
- **Define how to minimally store & look up user using session**
  - passport.serializeUser / passport.deserializeUser
- **Must configure a Strategy**
  - Strategy needs a *verification callback* you write — longest part
- **passport.authenticate (in two different routes)**
  - Uses strategy
  - Slightly different call in each route

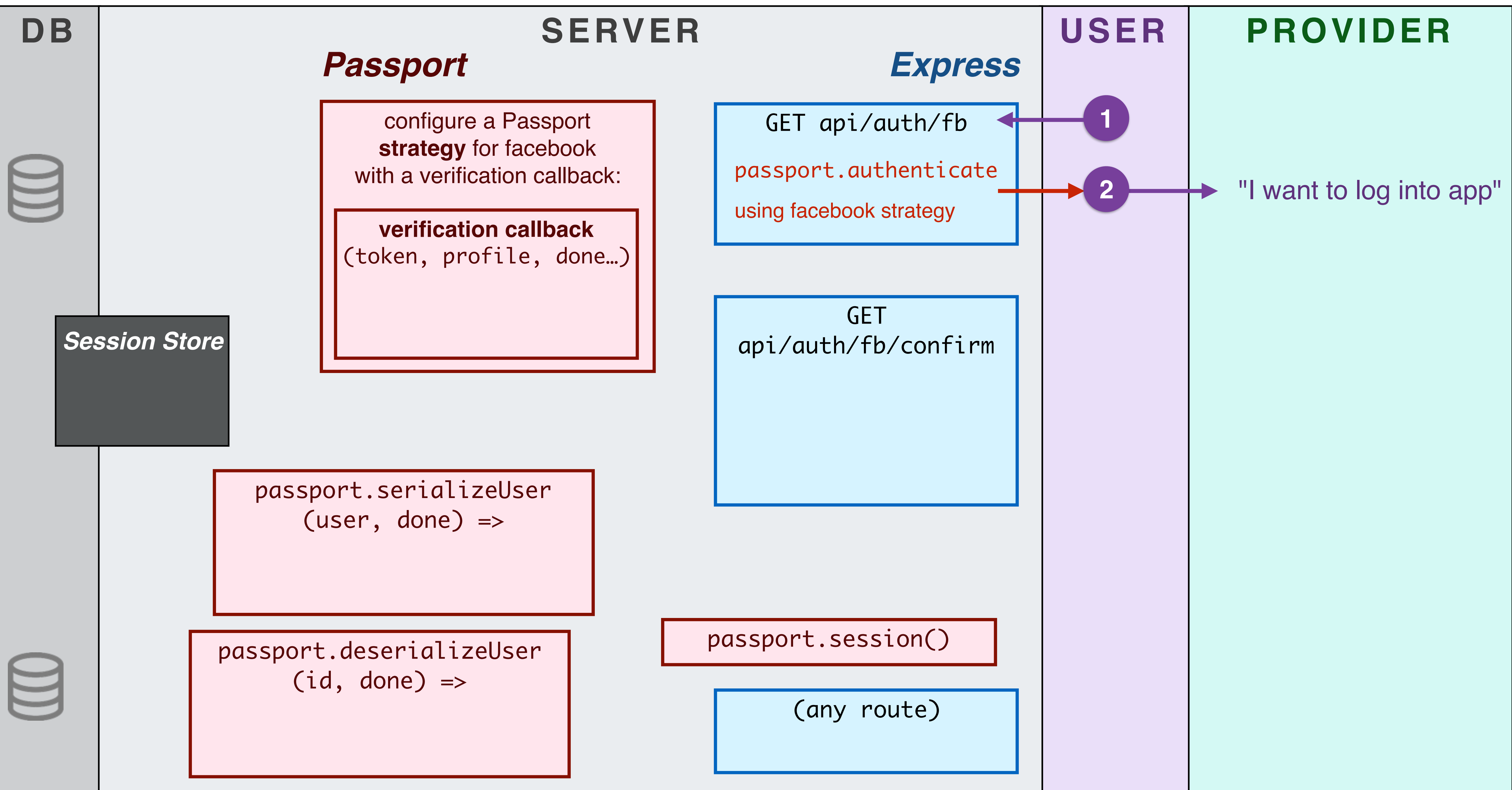


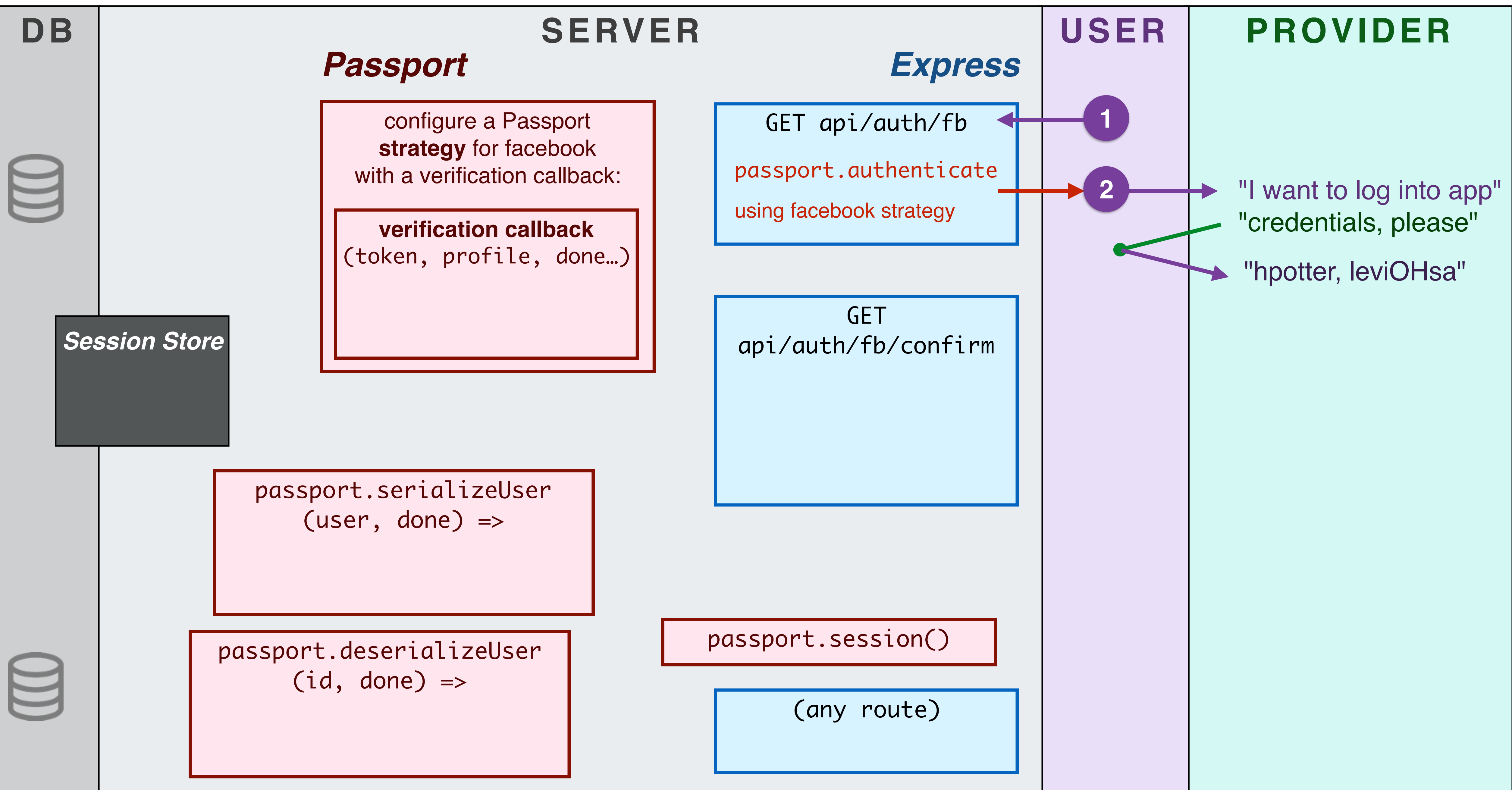


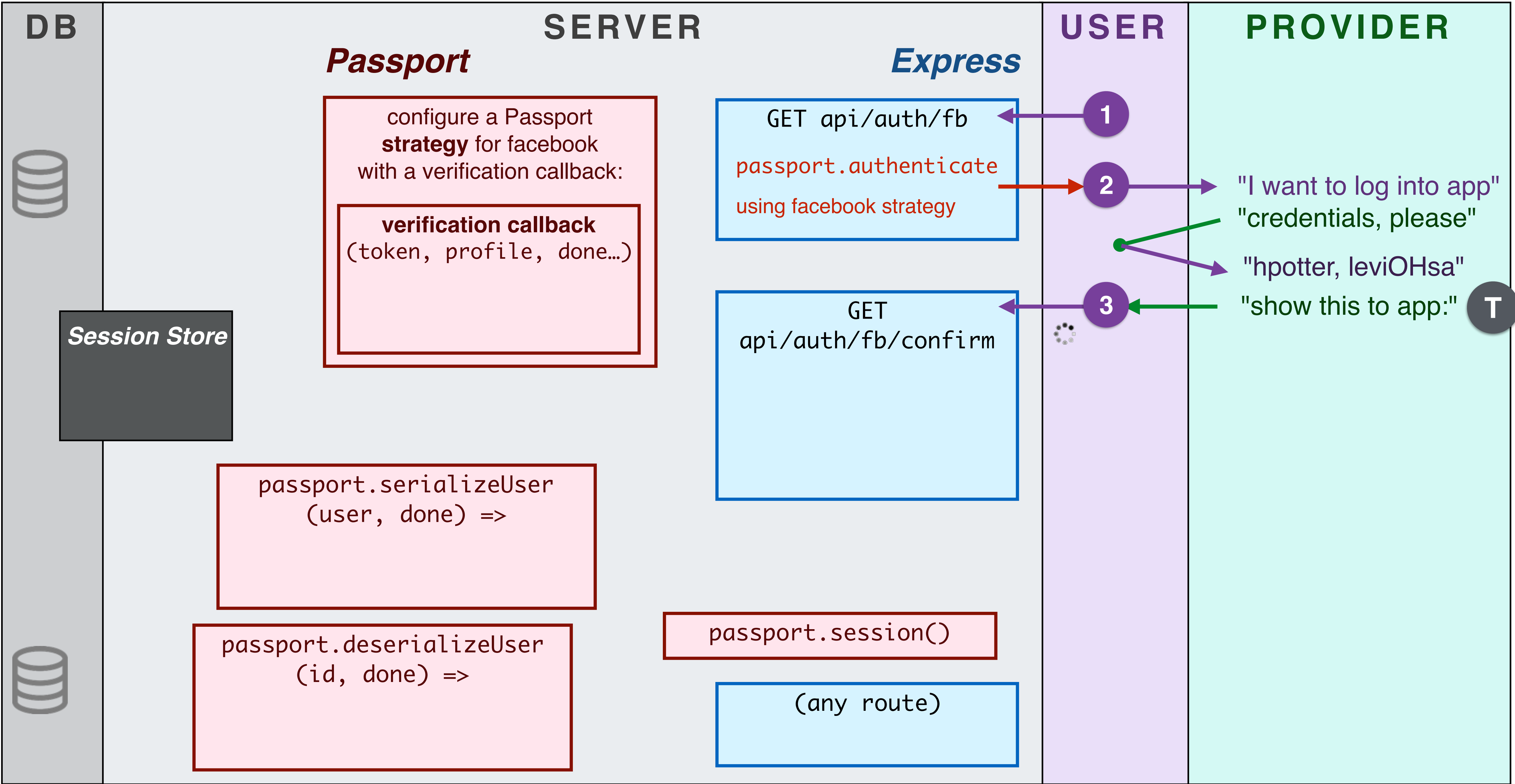


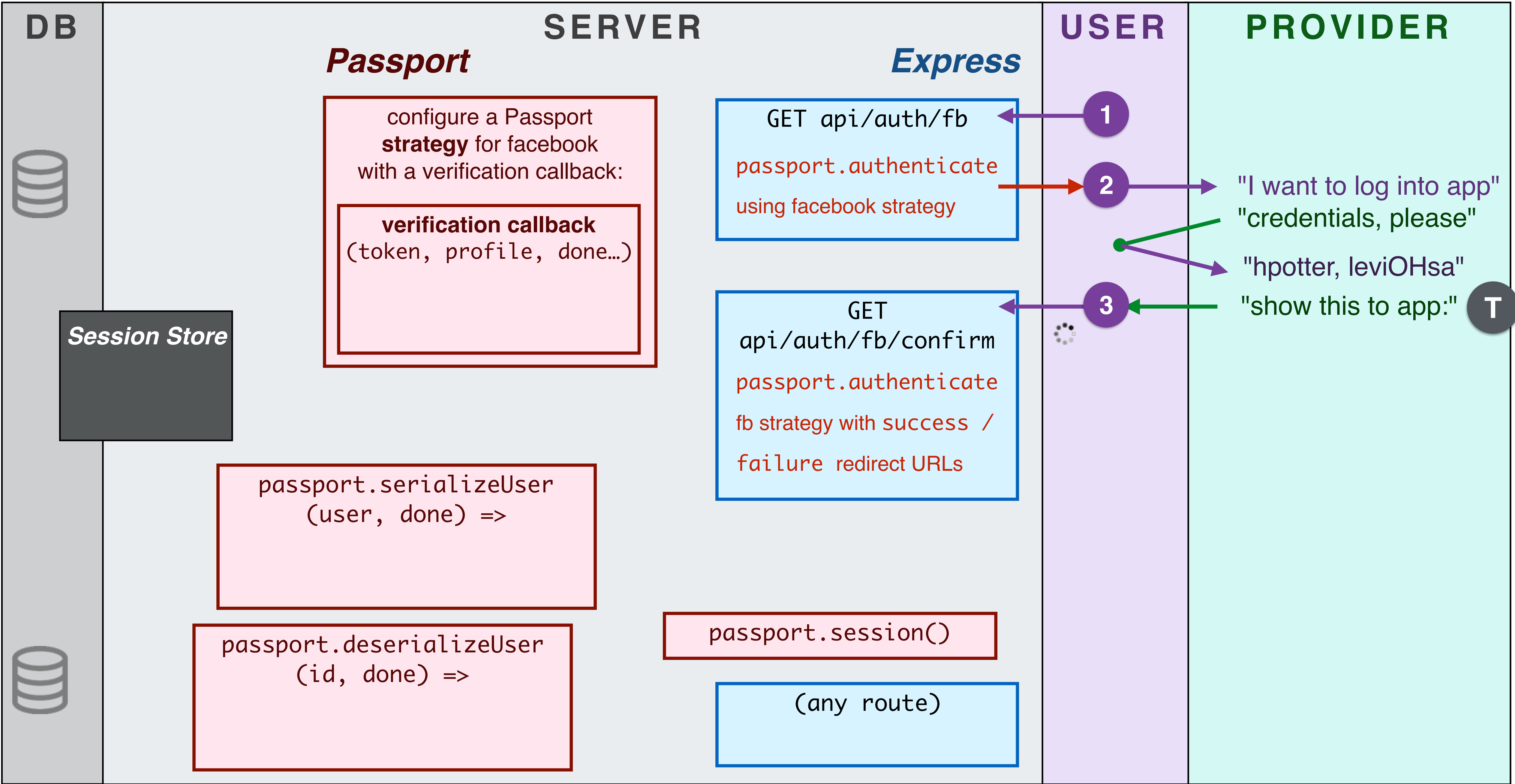


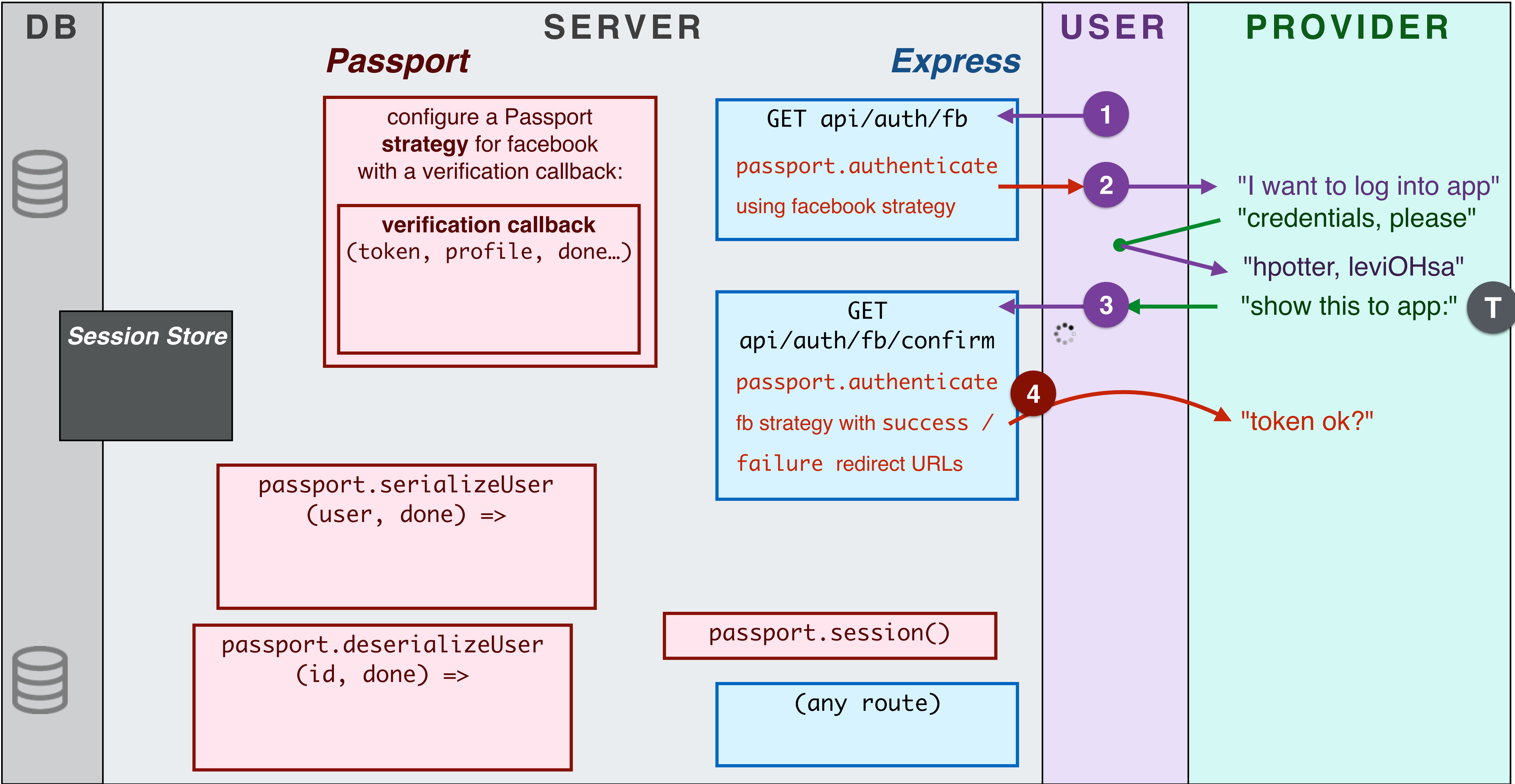


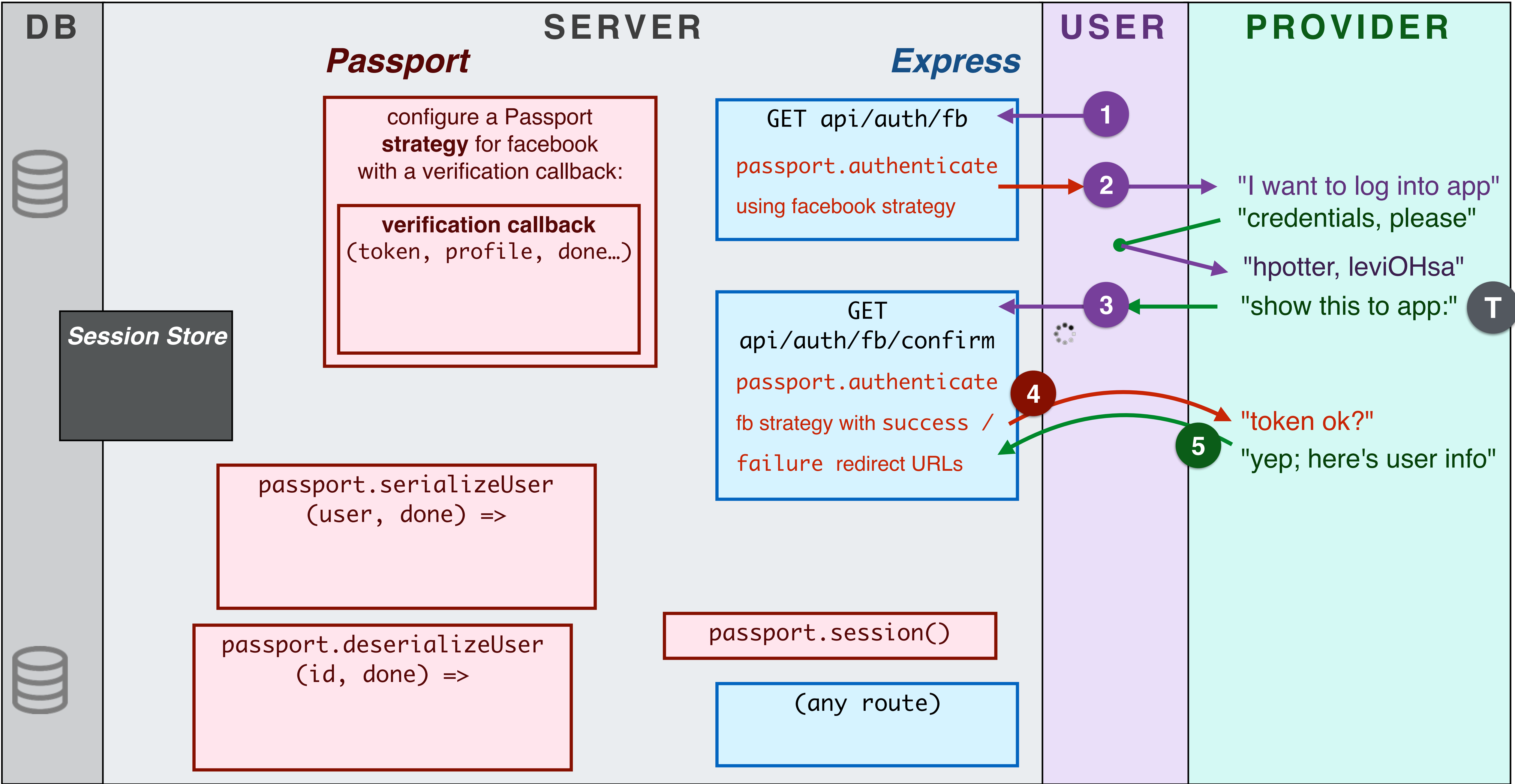




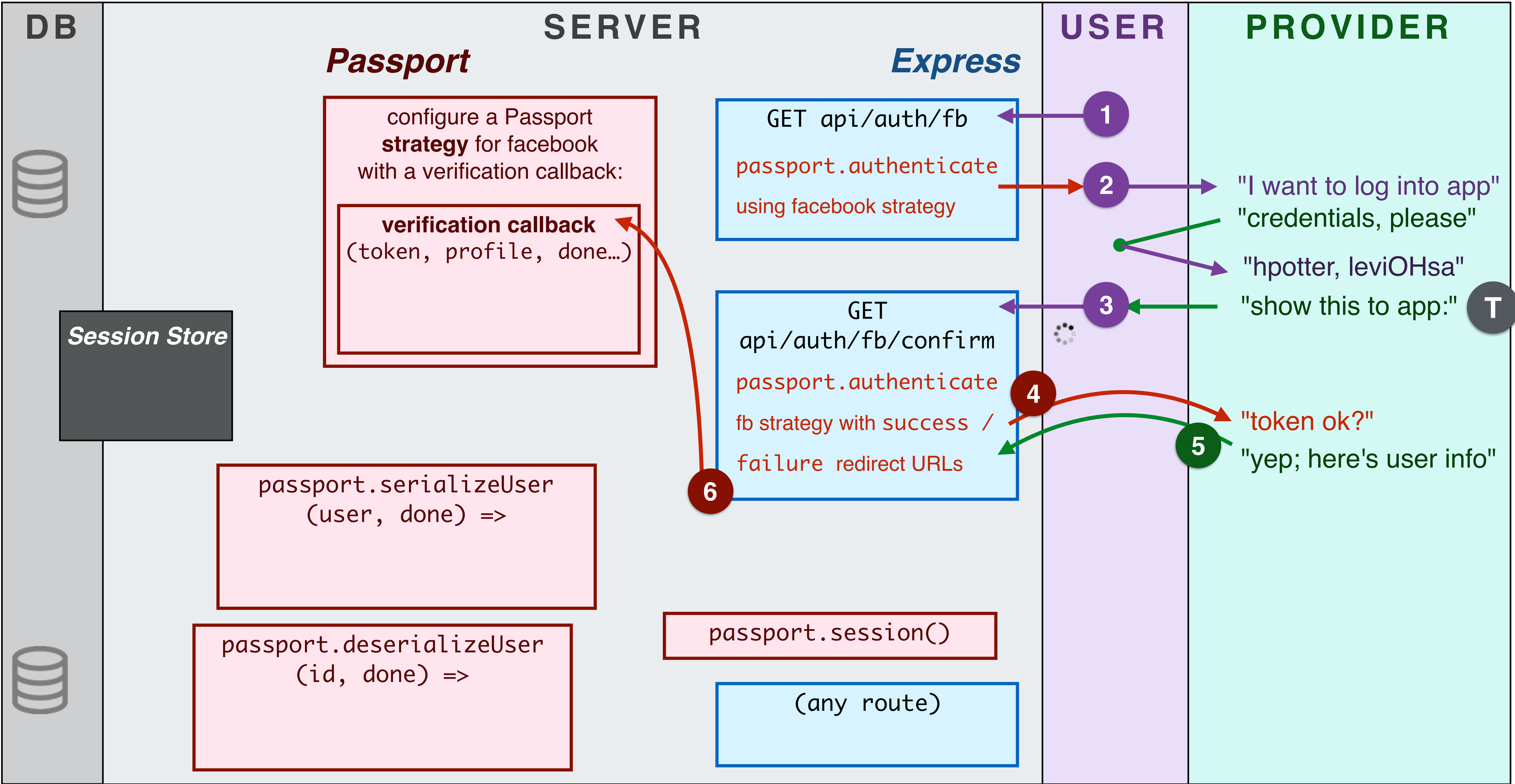


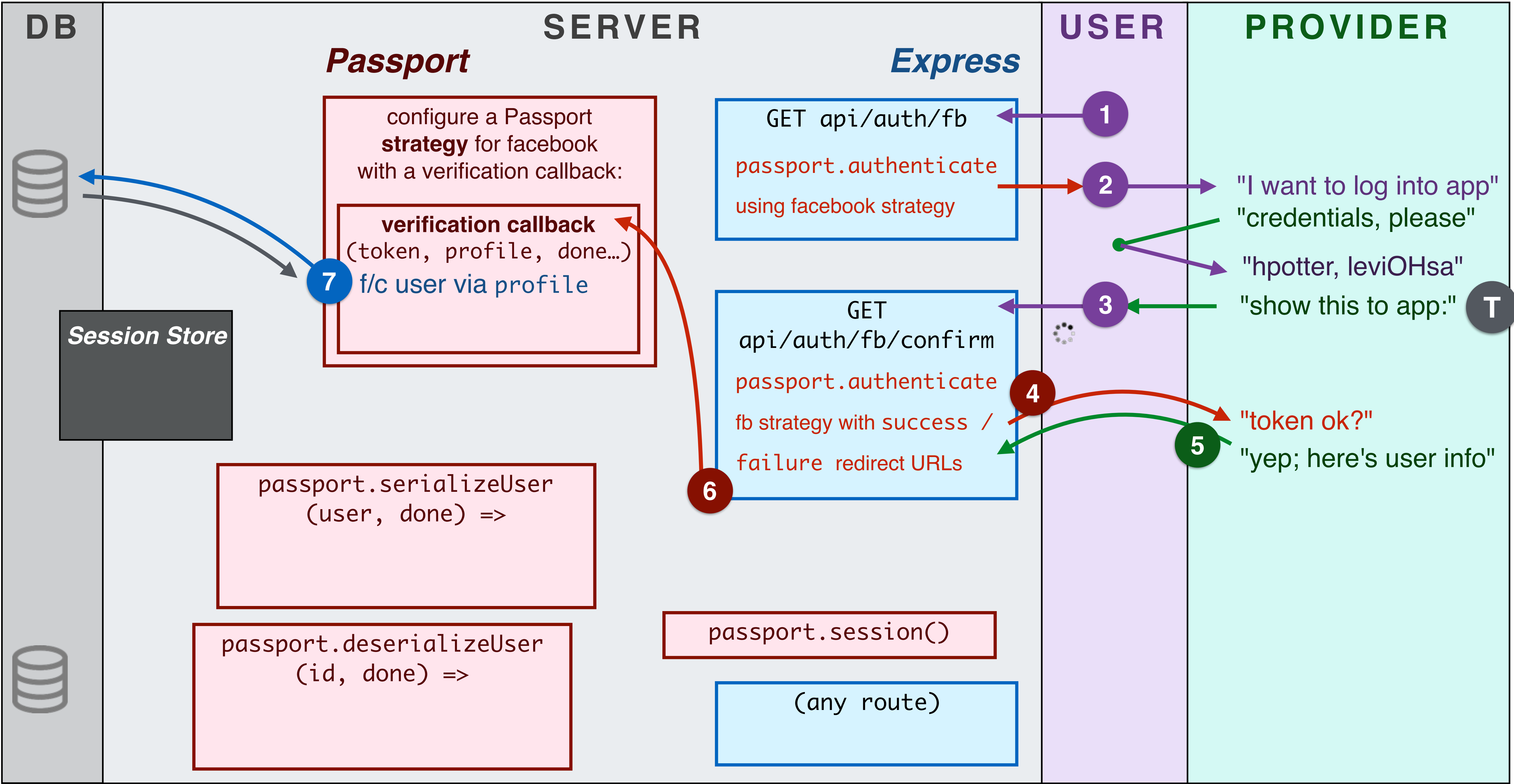




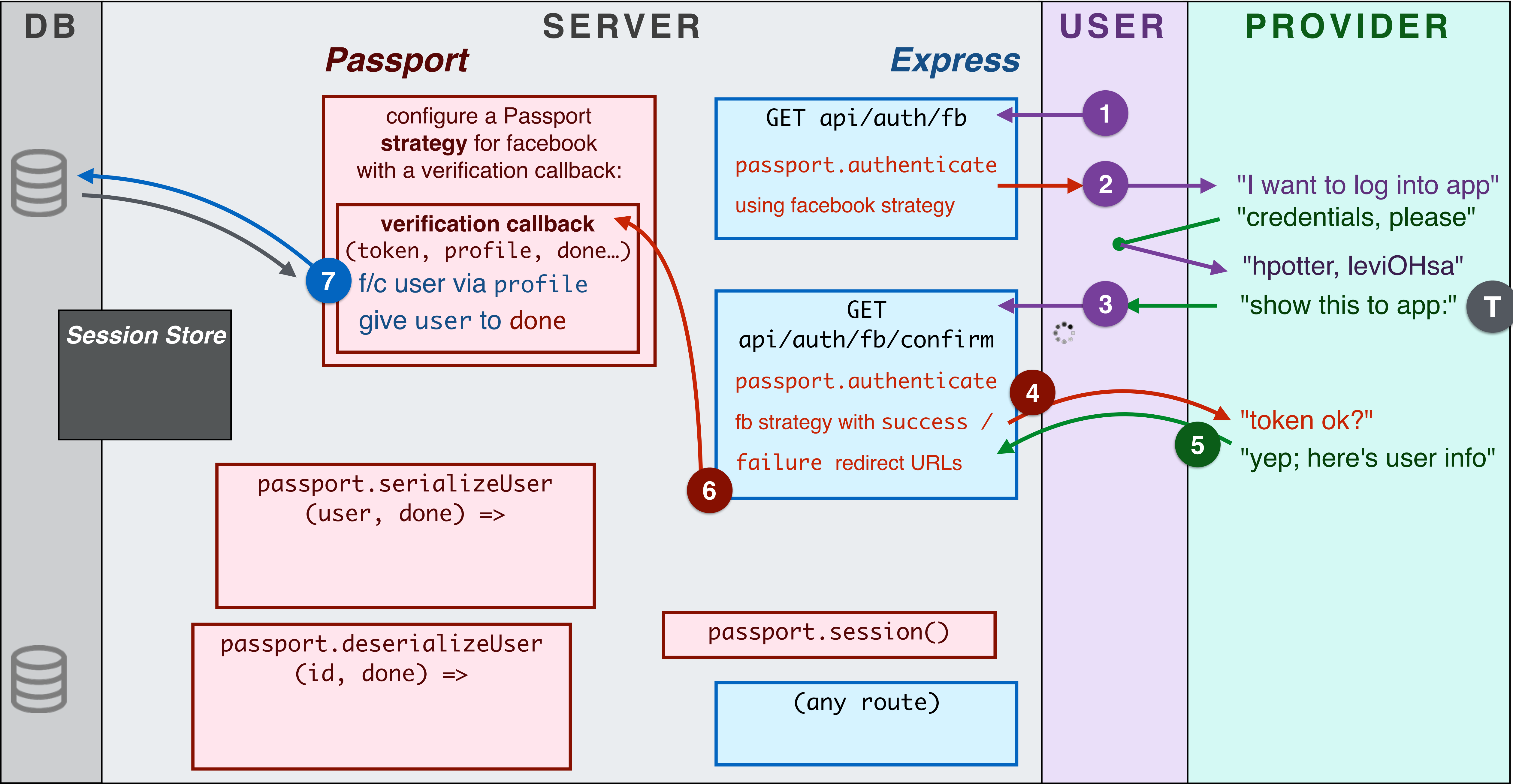


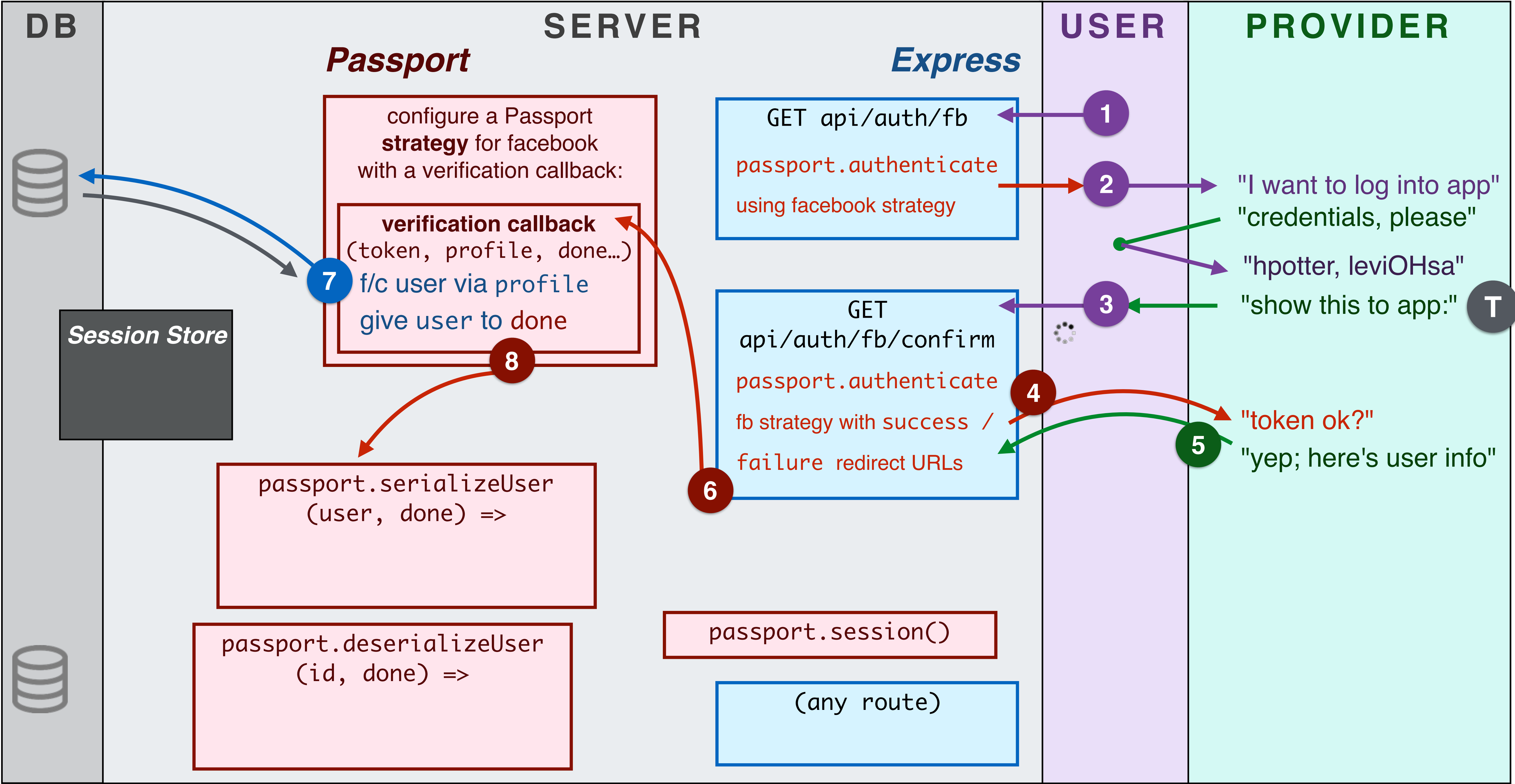


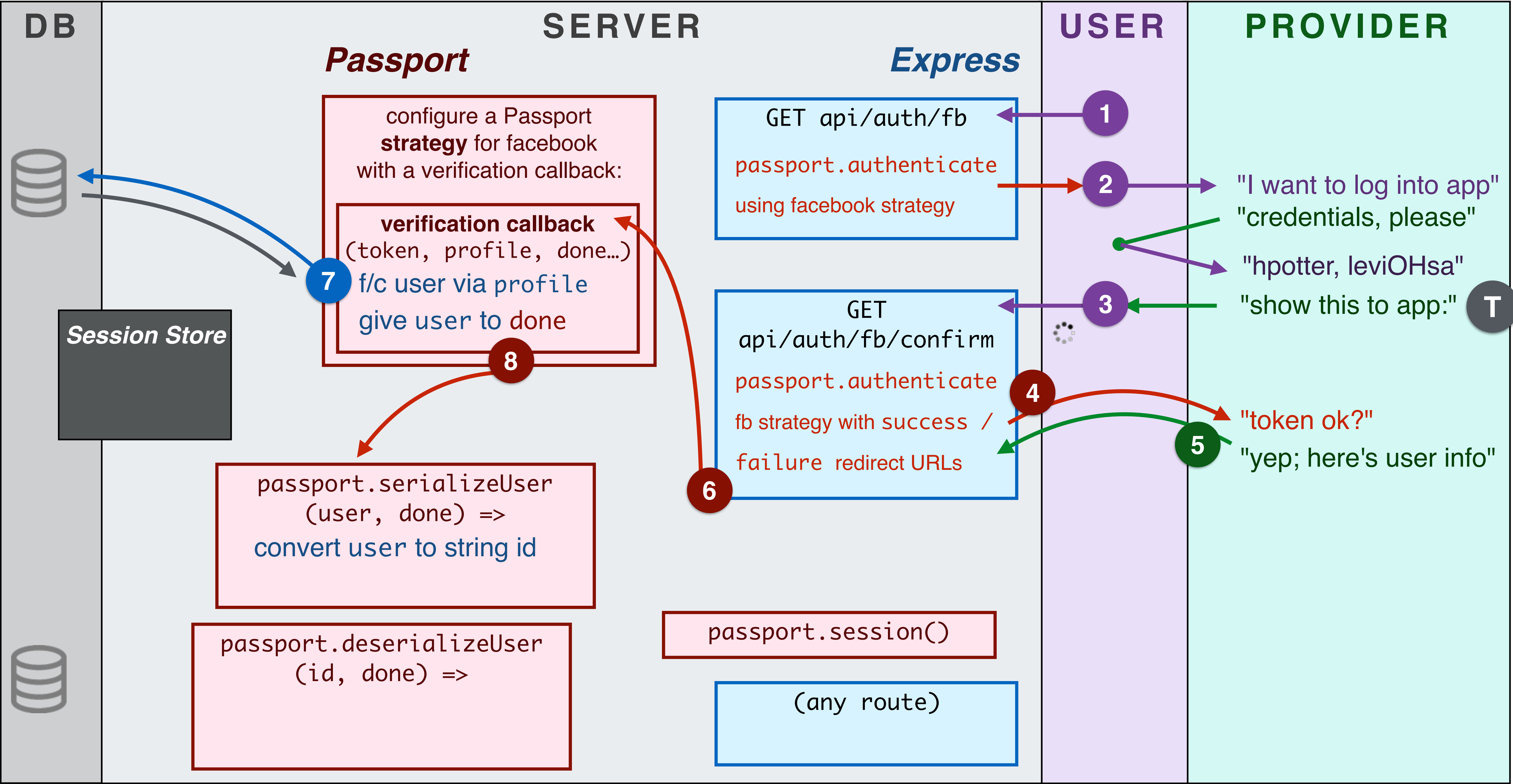


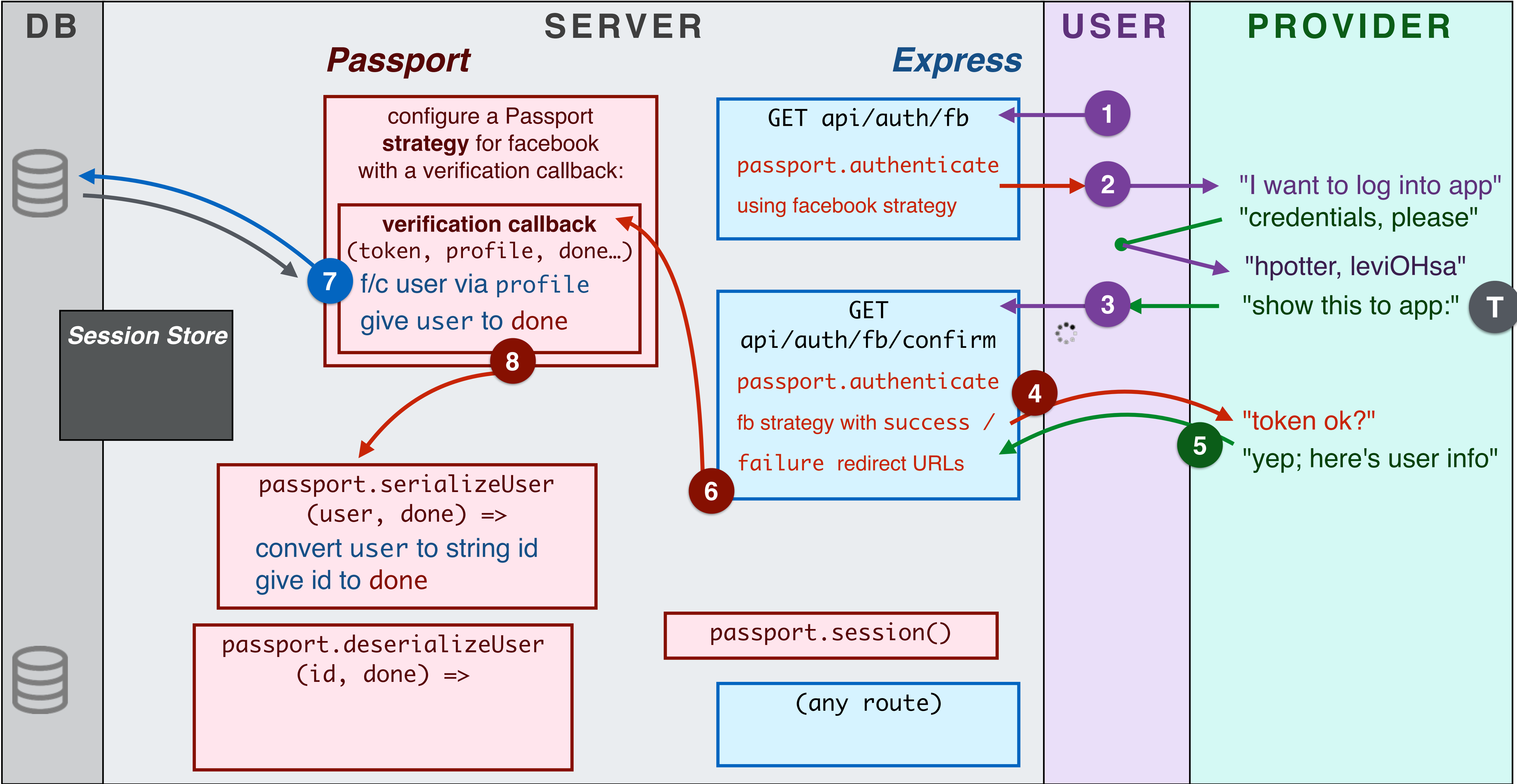


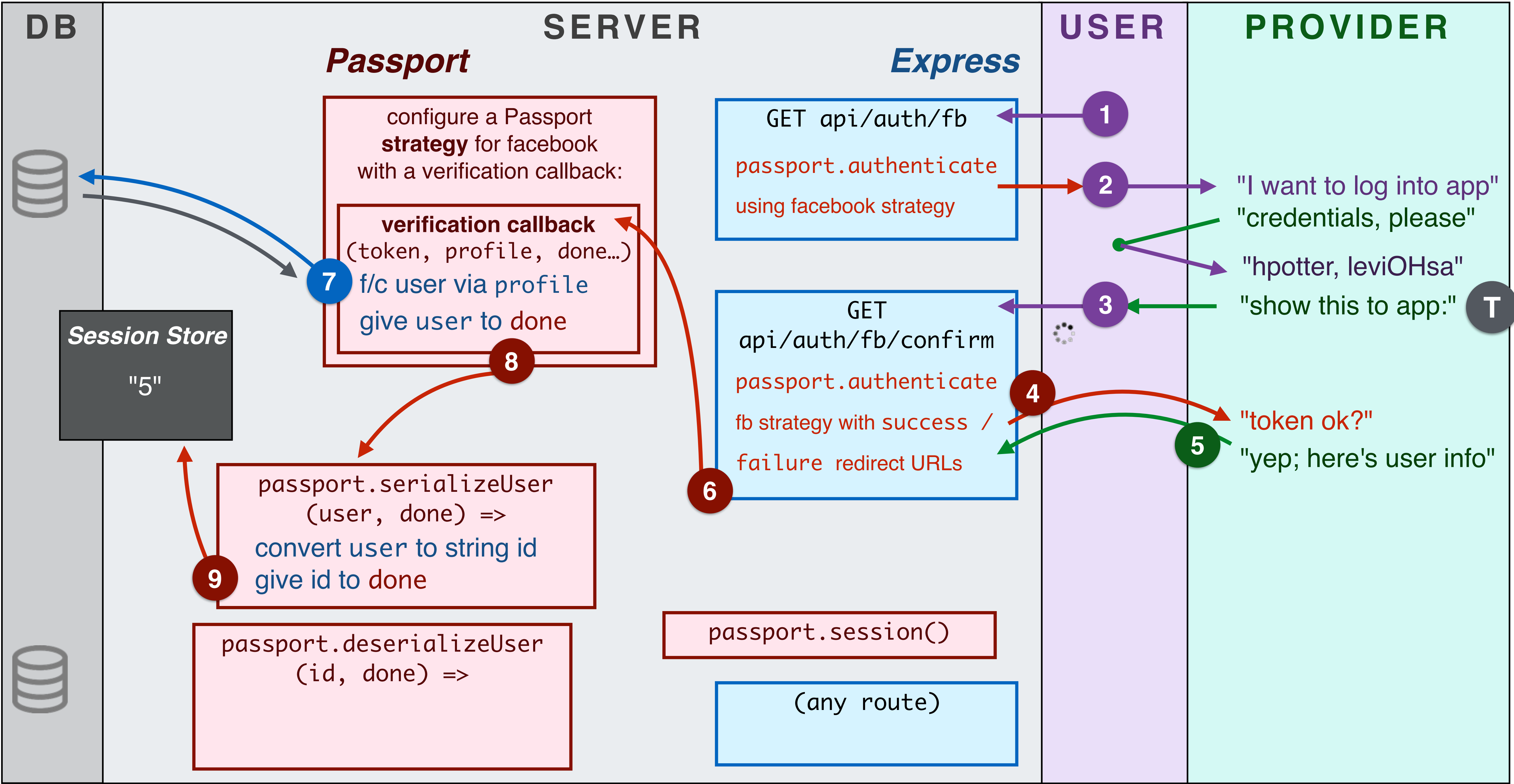




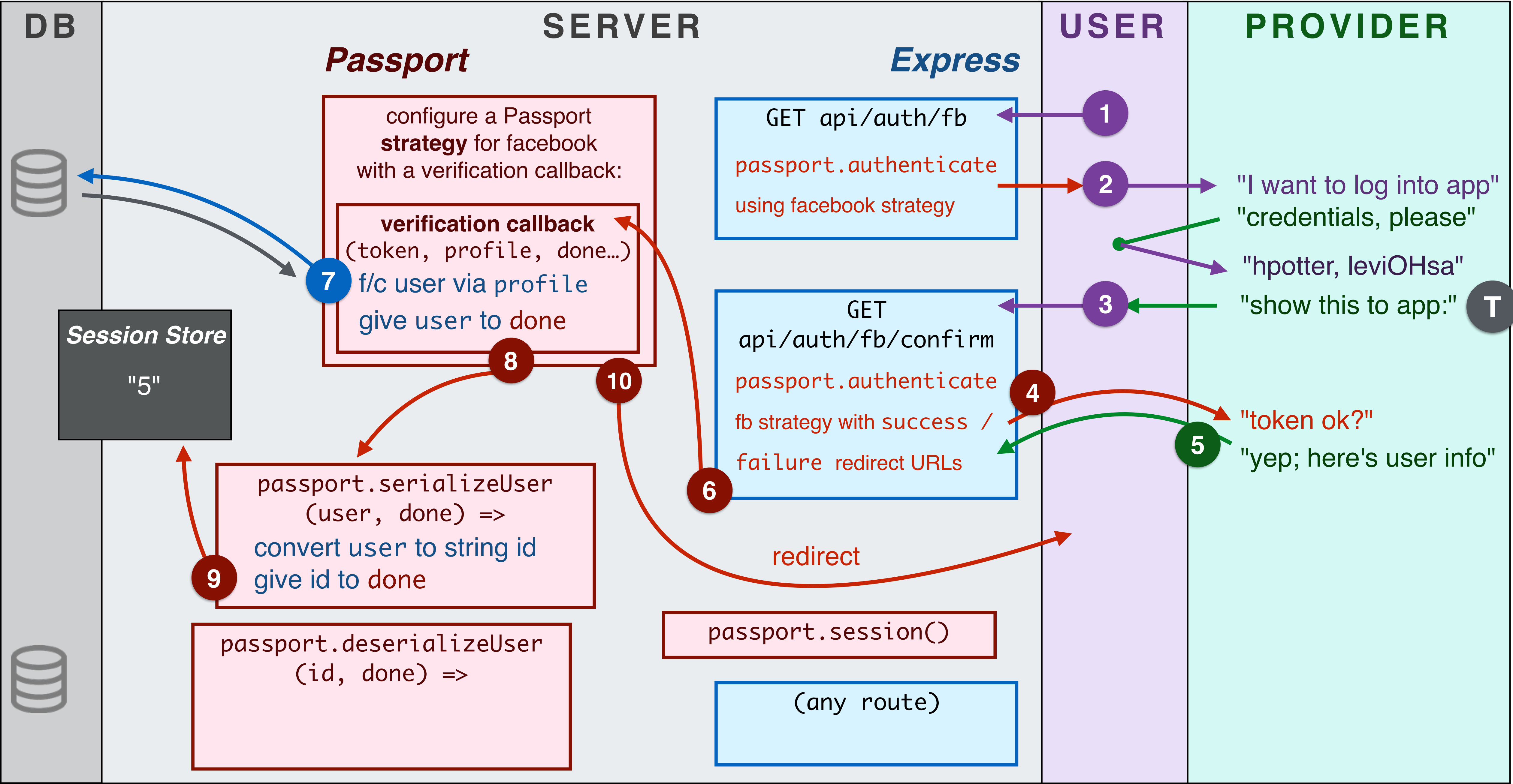


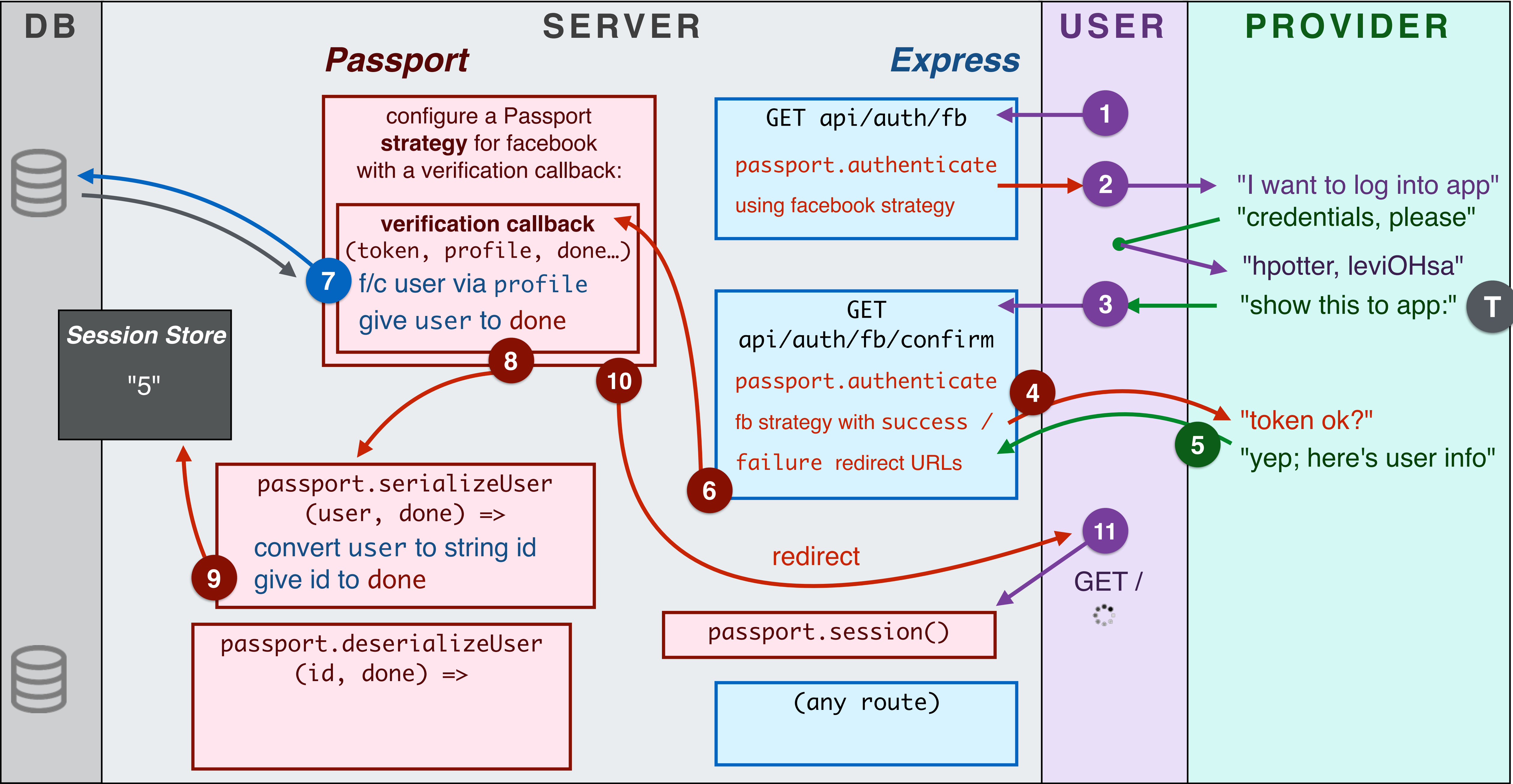


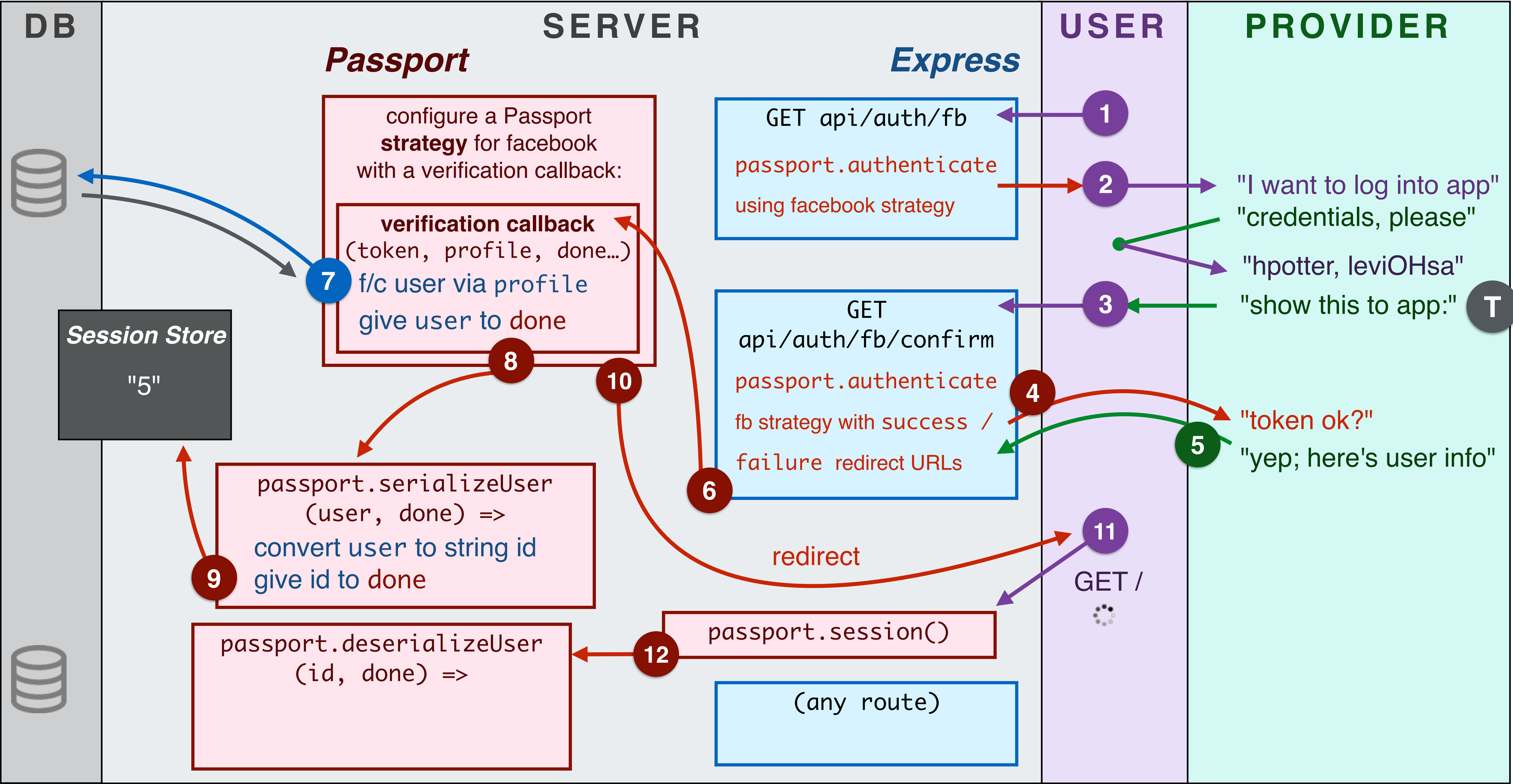




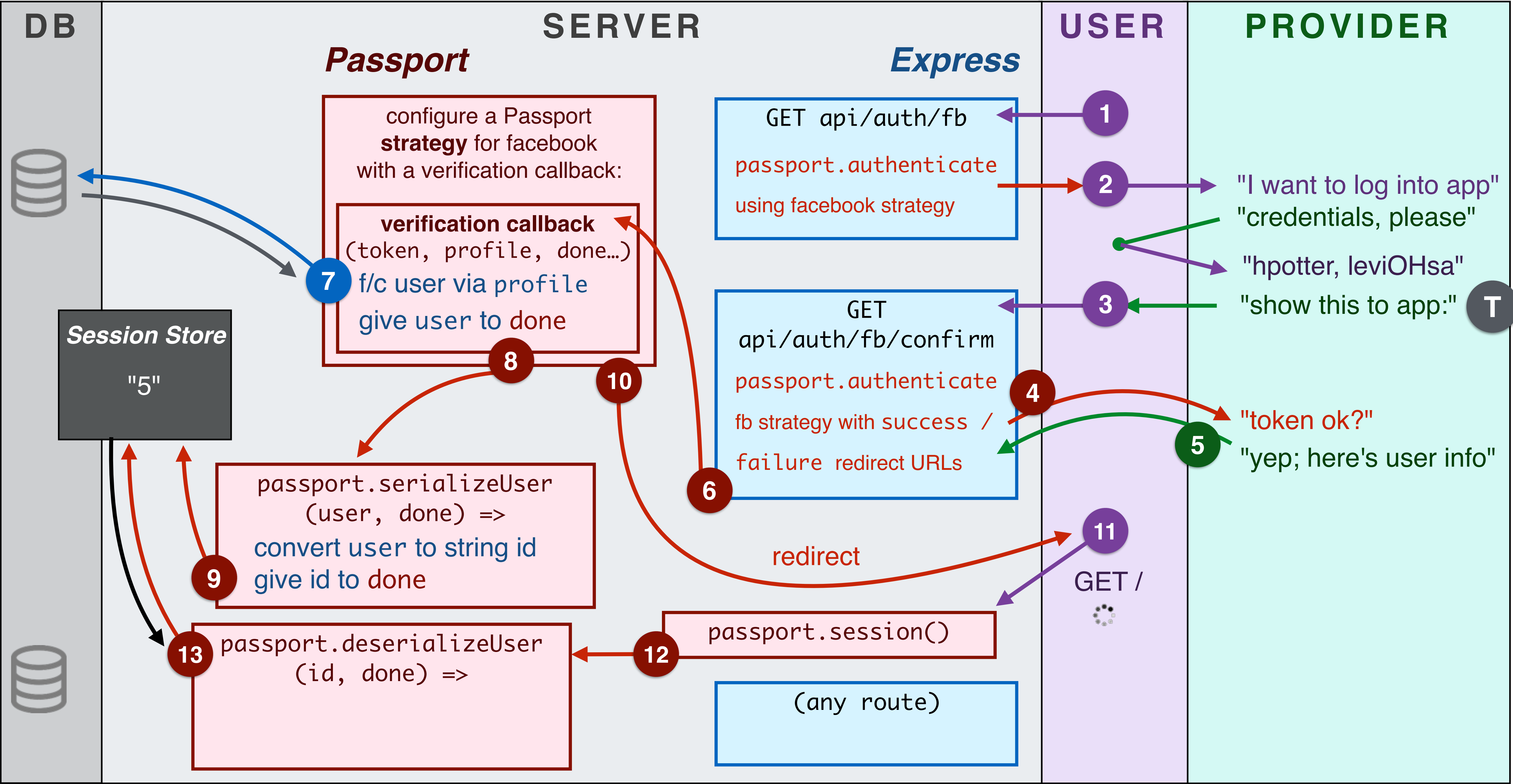


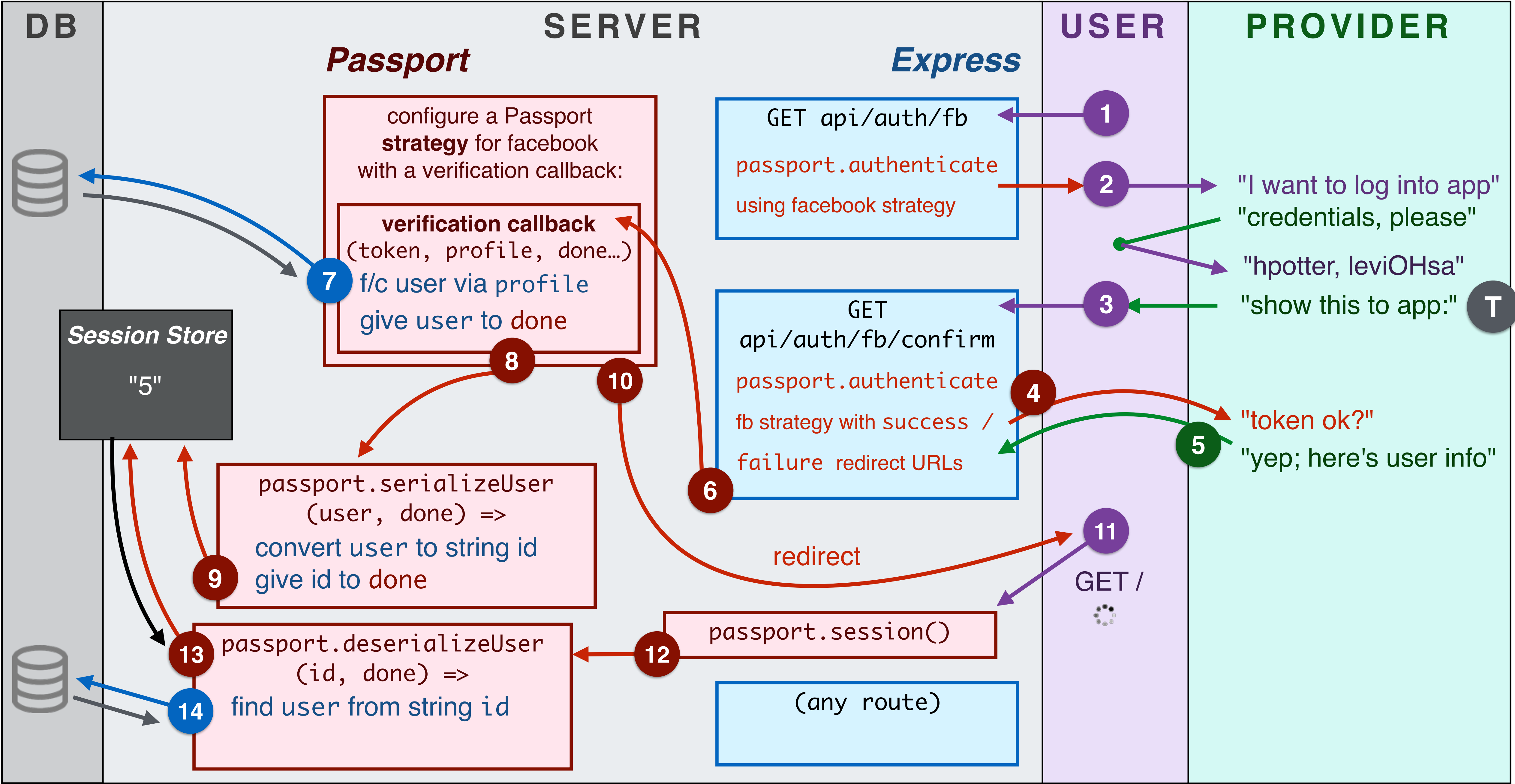


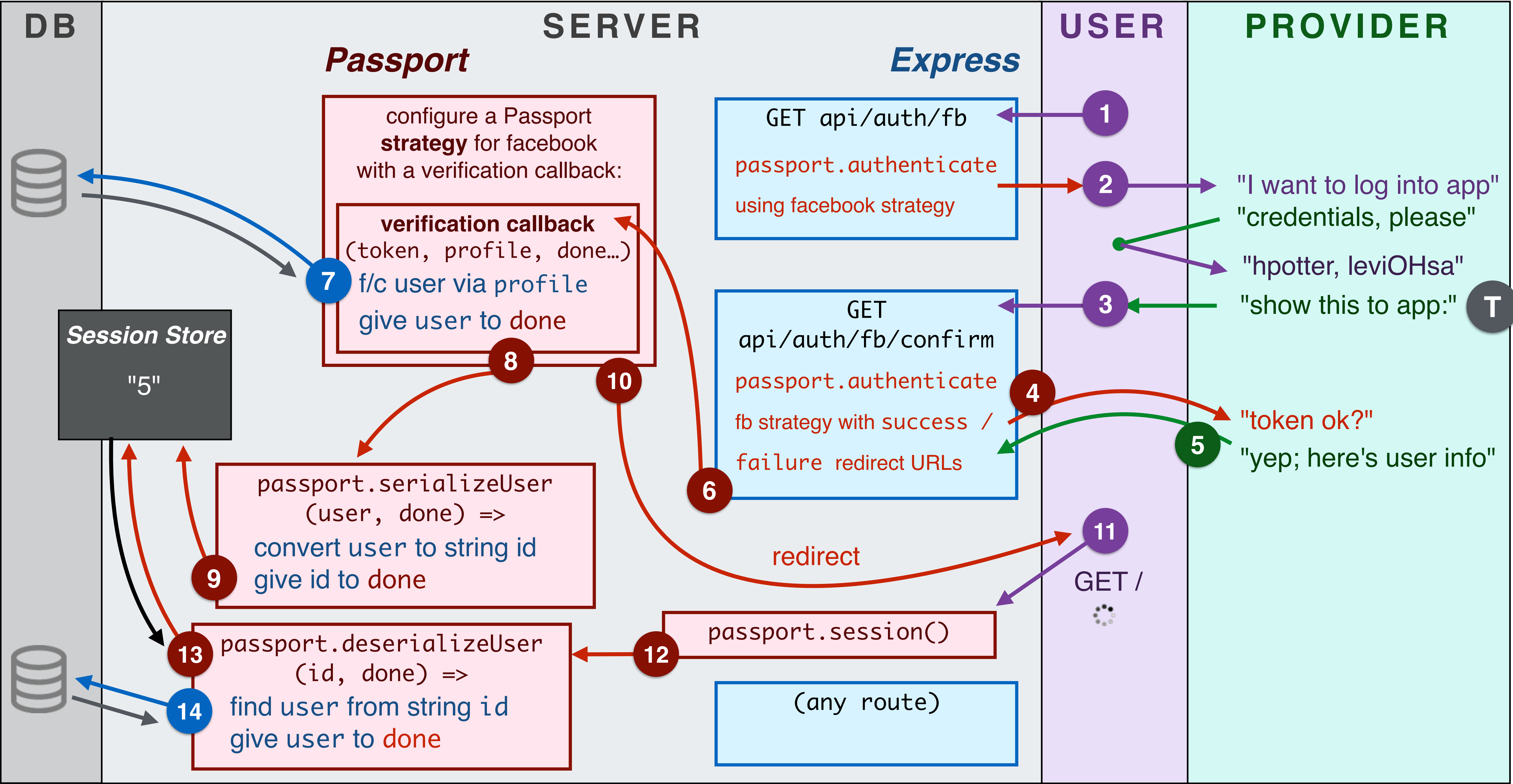


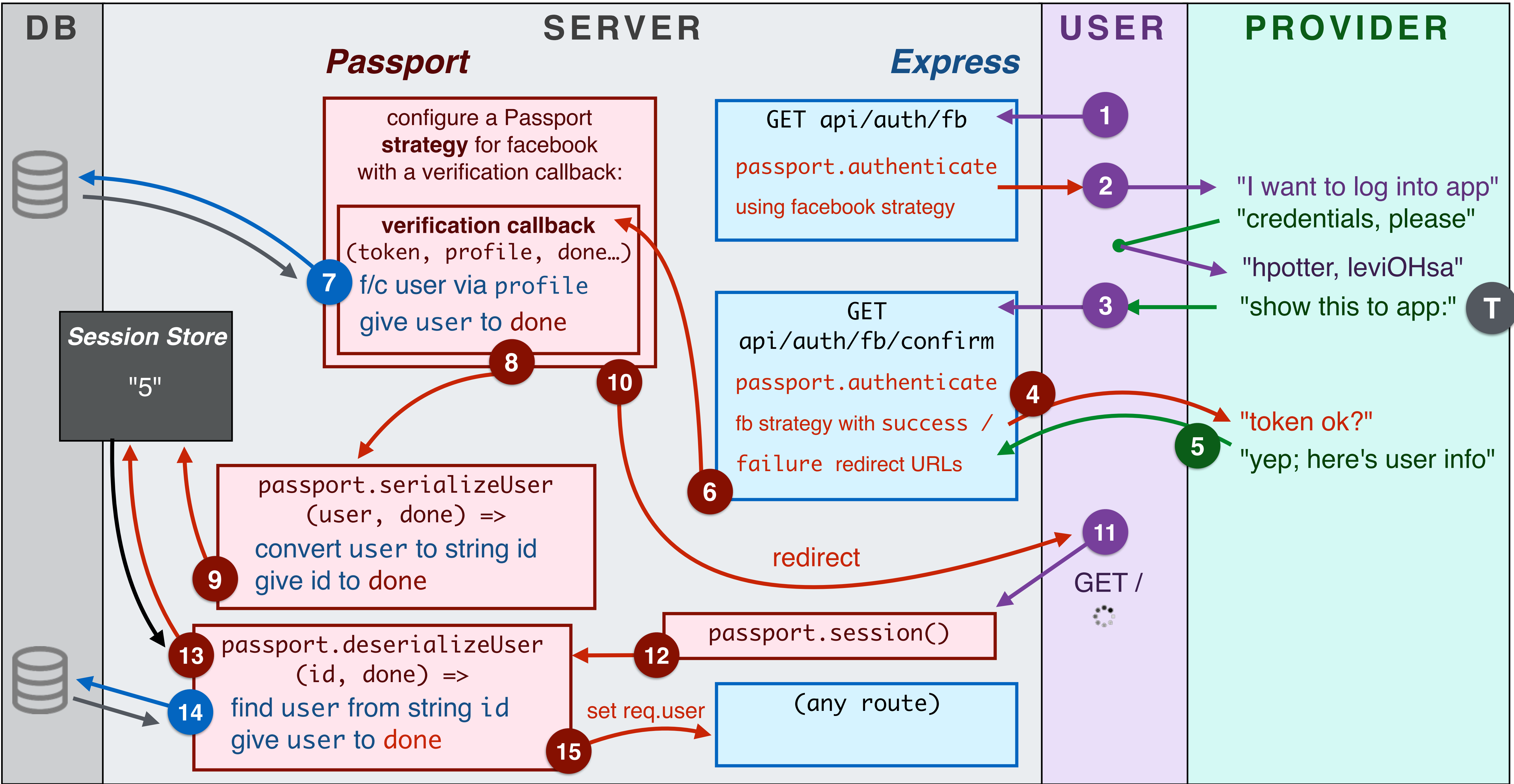




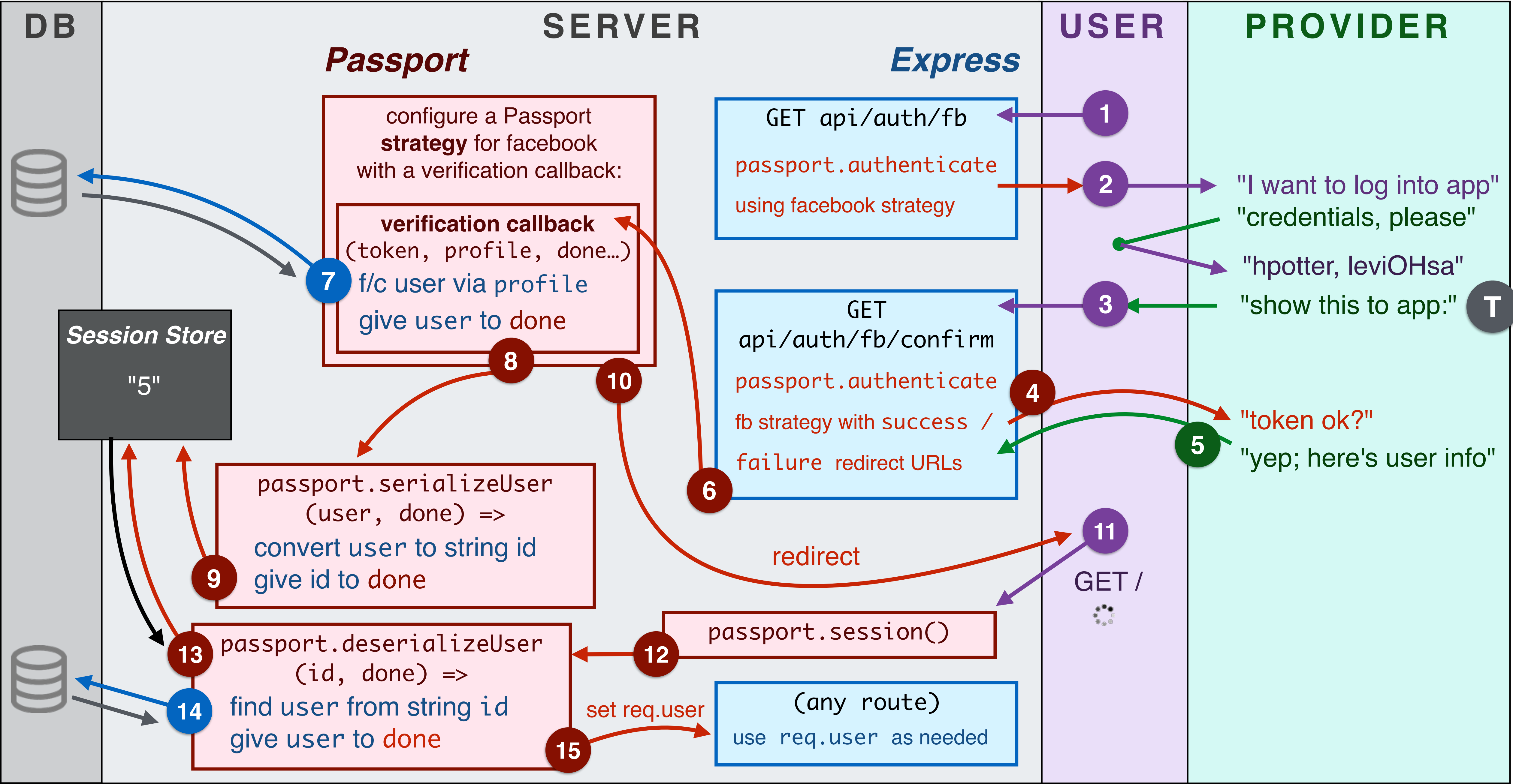


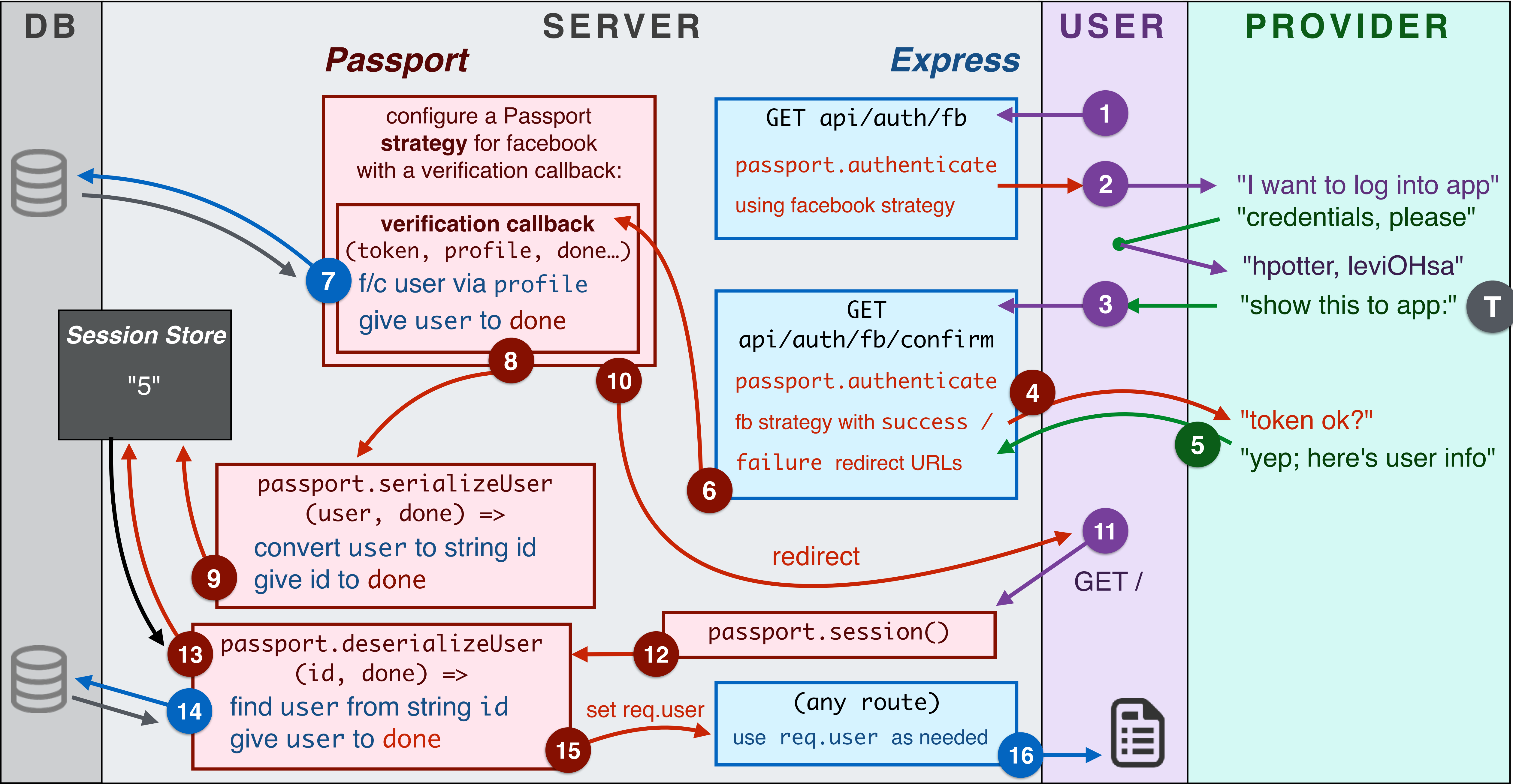


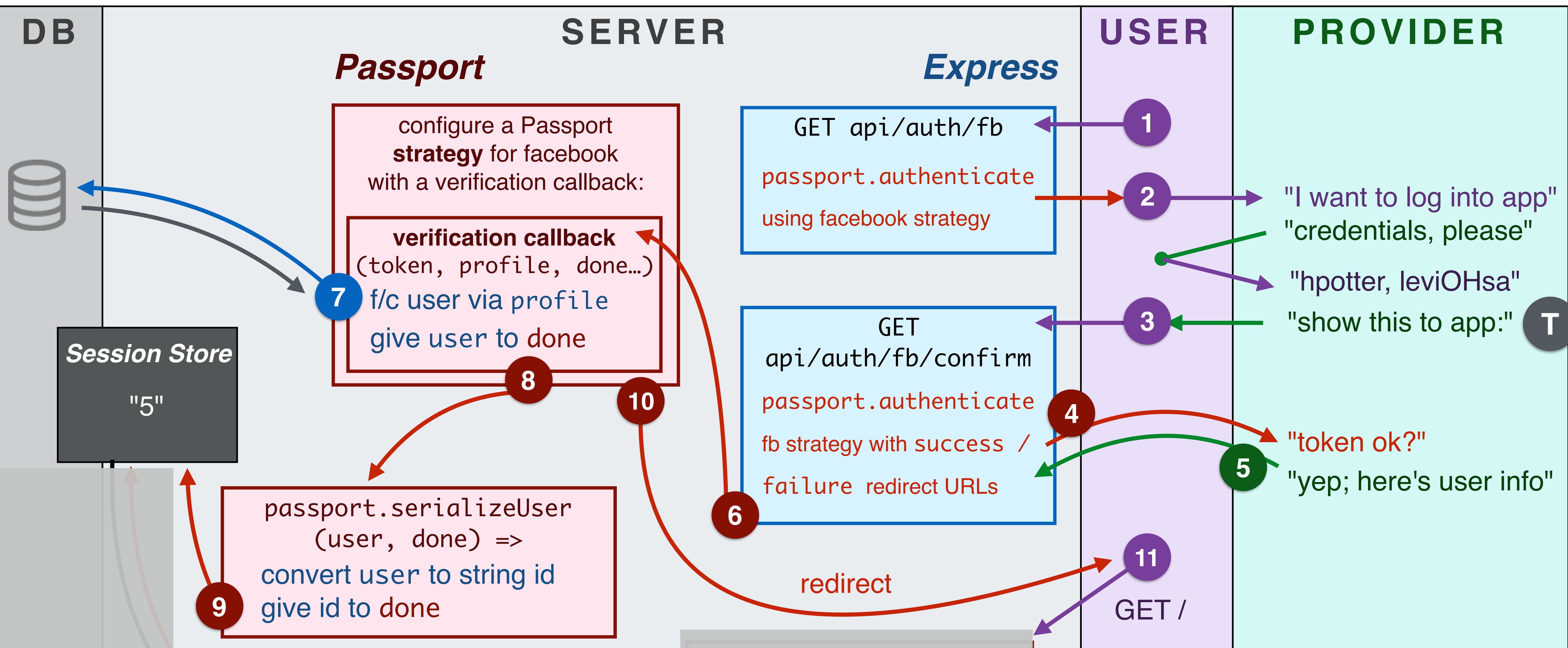












Initial login request: use `passport.authenticate` in routes. Token / profile will be passed into verification callback. Find / create user in DB based on profile info. Passport stores ID in session for easy future lookup, then tells client to redirect to the "success" URL.



