Security Concepts :-

Authentication

→ who you are ? , we can validate user here.

Authorization

what the authenticated user allowed to do ?

Authentication

Ex: Authentication factors

(1) knowledge factor
→ something you know

Ex: * * * ——— (9)

(security questions & answers in any site or cycle symbols image symbols)

(2) possession factor
something we have

Ex: authenticate using physical devices
① By call (by microsoft login in laptops)
② By authentication apps
③ By device login (Ex:- gmail login)
④ using device detectors.

(3) Inherence Factor
something we are
→ using fingerprint face id eye

Multifactor Authentication (MFA)

here will use multiple authentication factor to establish MFA.

Ex:- knowledge factor + possession factor + inherence factor.

Ex: we have MFA for gmail login. (2 factor authentication)

SSO (single-sign on) :-

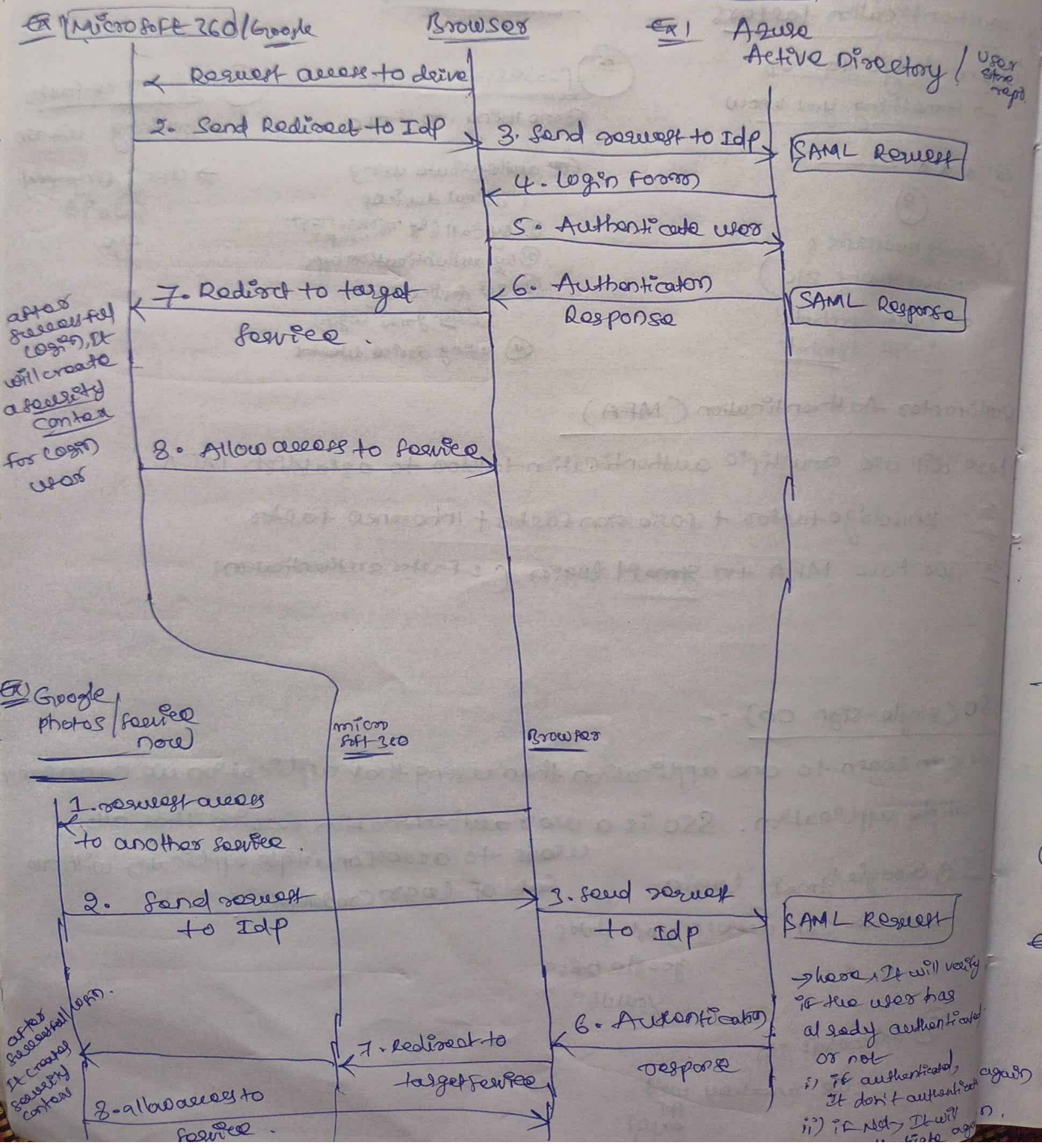
we can login to one application then using that application we can access multiple application. SSO is a user authentication service that allows users to access multiple application with one set of login credentials.

Ex: ① Google/gmail login.
→ we can access google photos
google drive
youtube

② microsoft 365
→ we can access word
PPT
Excel
Aloud

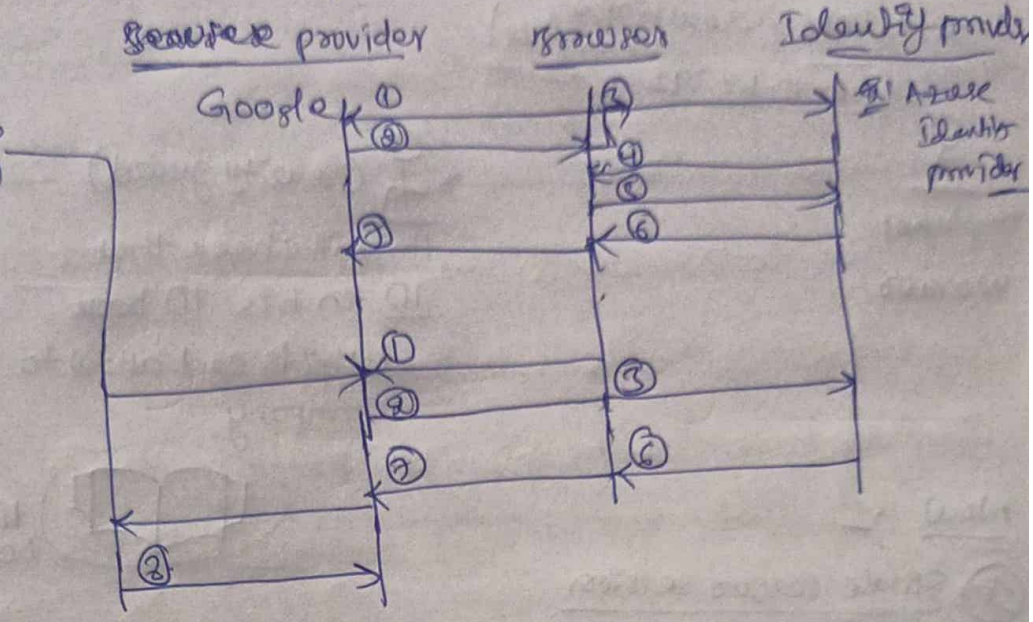
SAML (Security assertion markup language) :-

The format is like .xml and consist of information reg, service, Destination, provider names, ID, time, versions, format etc --)

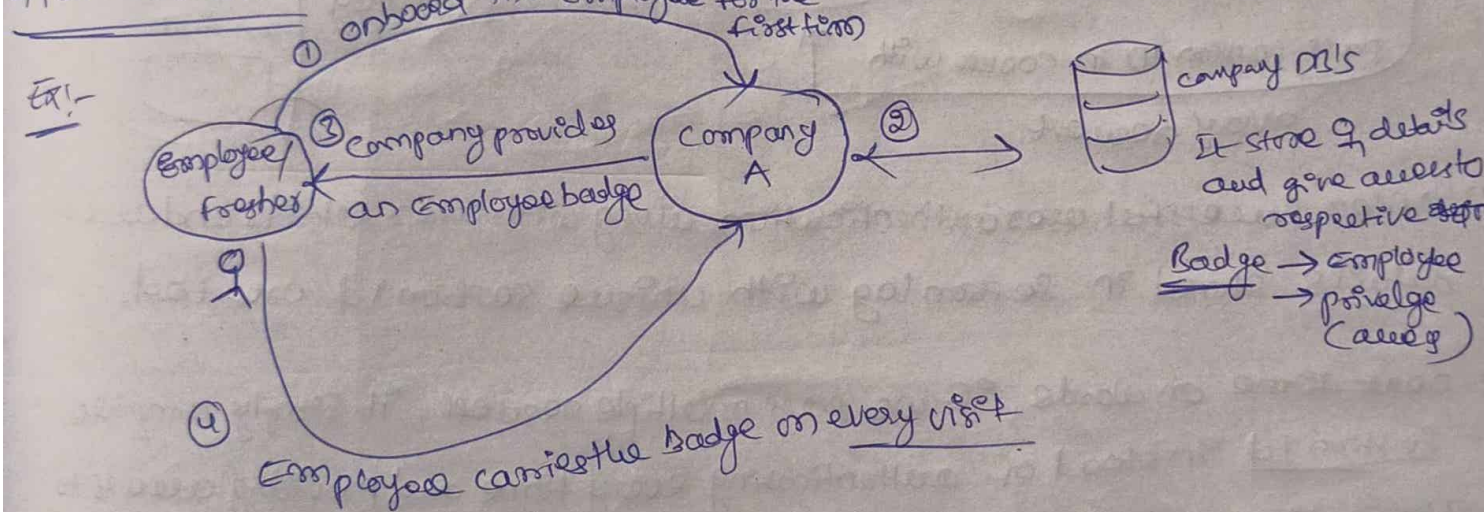


If the user exists in azure directory as a Active mode, It won't authenticate every time if we use respective application.

Ex:- Google photos
Google drive
Google calender
youtube
gmail



Authorization (cookie/session based, Token based, OAuth2.0 & OIDC)



How long is this access Token (Badge) Valid?

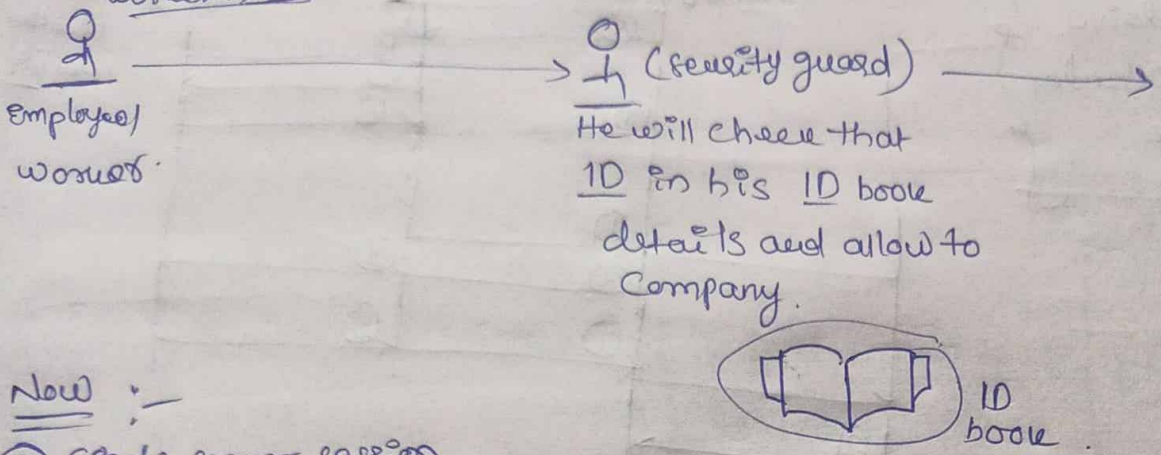
1. until it is invalidated
   - (a) Lost and reported
   - (b) Left the organization
2. until it is Expired.

~~Cookie~~

Cookies/session based Authorization :-

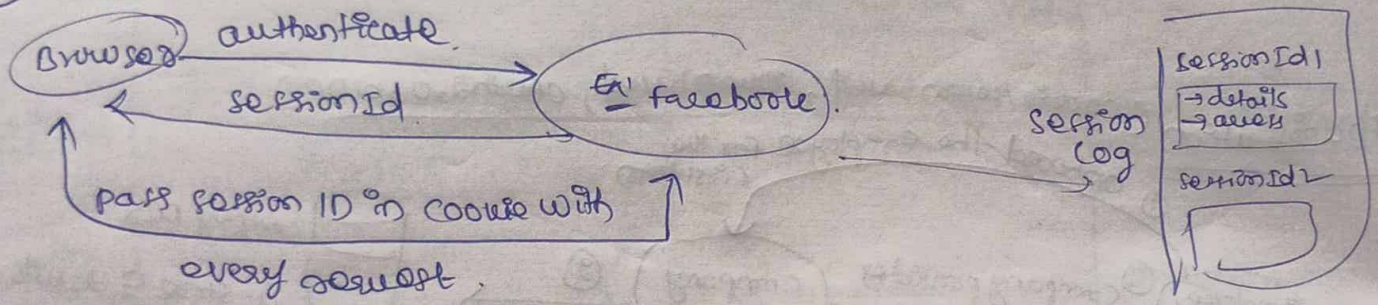
In old days (real time examples)

worker ID 6Z392



Now :-

1) Single server session

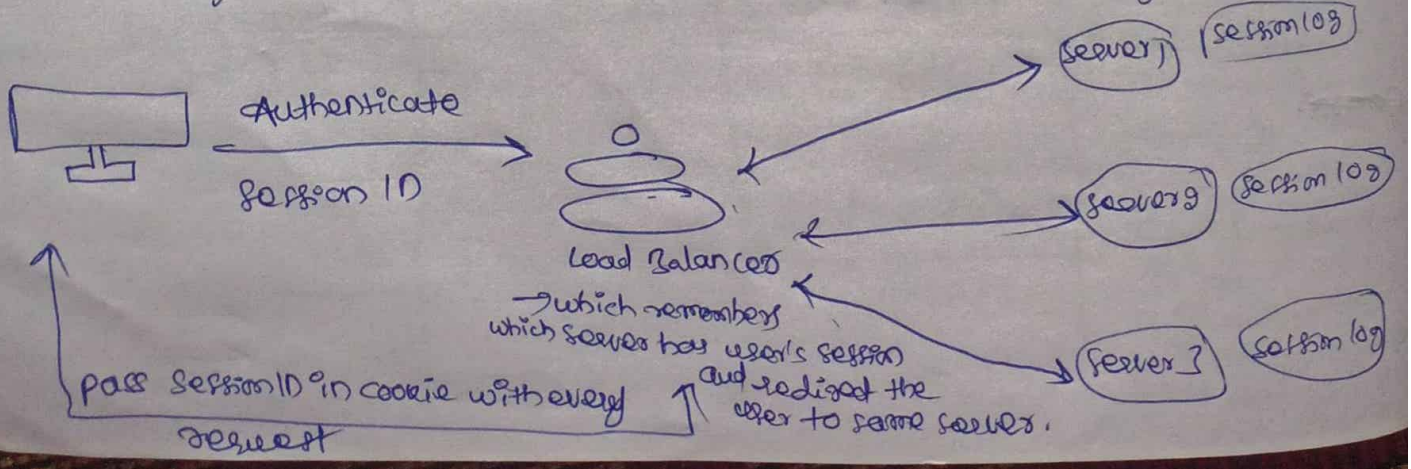


1) After successful user authentication, along with user details and access details stored in session log with unique sessionId created.

2) Next time on words, if we pass multiple request, it simply compare sessionId instead of authenticating every time and providing access to applications.

2) Sticky session

It comes into picture, when the applications installed in various servers, for reducing load on one server, we will use multiple server and it is maintained by load balancer to balance the request weight on servers.

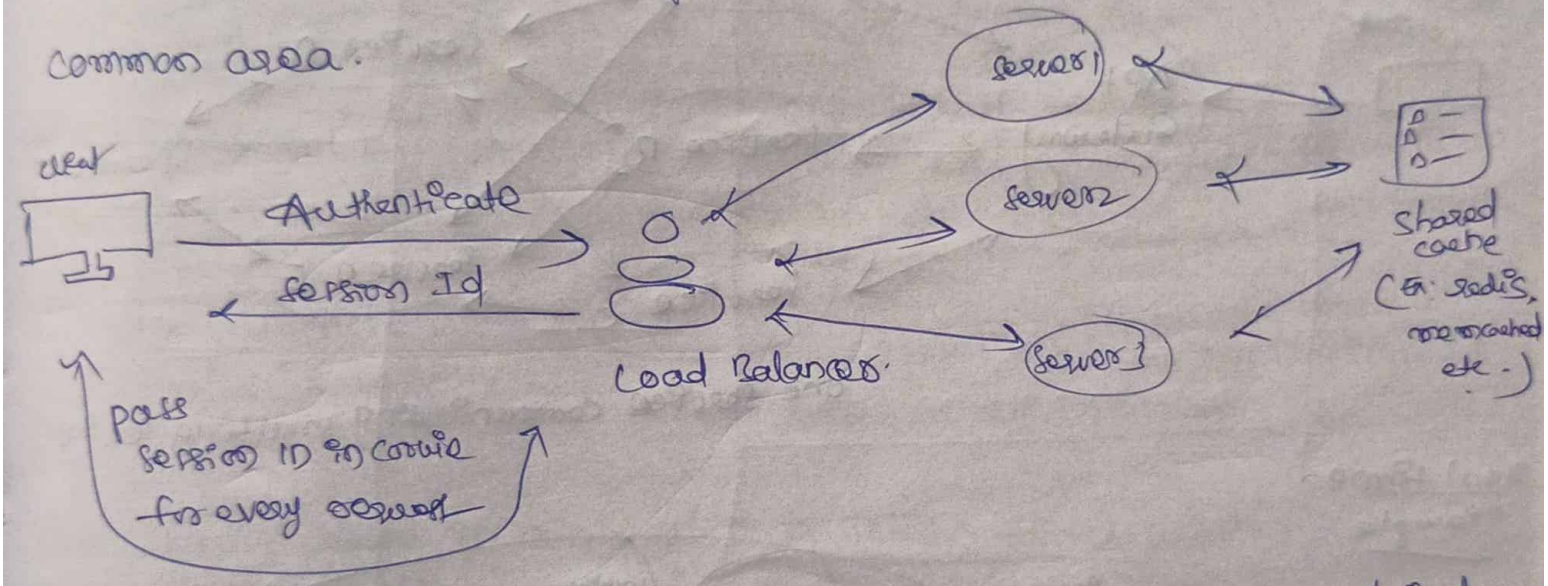


But the drawback of Sticky session is. If it redirect users to respective same server like if initially multiple request goes to one server then that particular requests won't goes to other servers

→ we won't achieve proper load balancing here.
scalability.

(3) Shared session Cache

To overcome drawbacks in sticky session, we will use shared session cache to store various/multiple server's session log file in a common area.



⇒ The drawback of shared session cache is like if one shared cache instance goes to down then it will impact whole application.

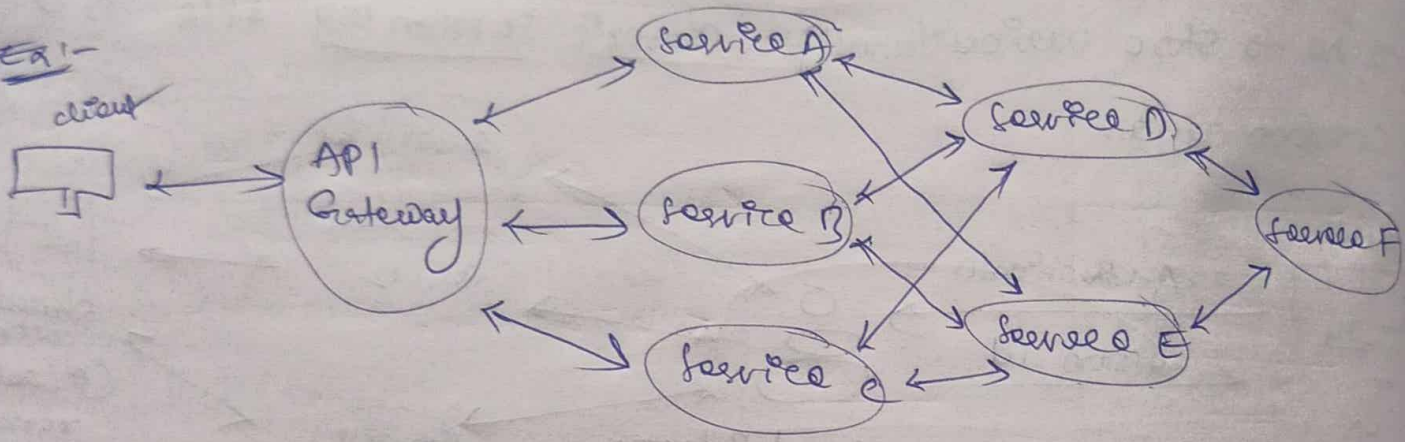
⇒ Based on application requirement will use simple session authorization or sticky session authorization or shared session cache.

These are stateful servers, means, It will store the session or user authorization information data in servers.

Token based authorization

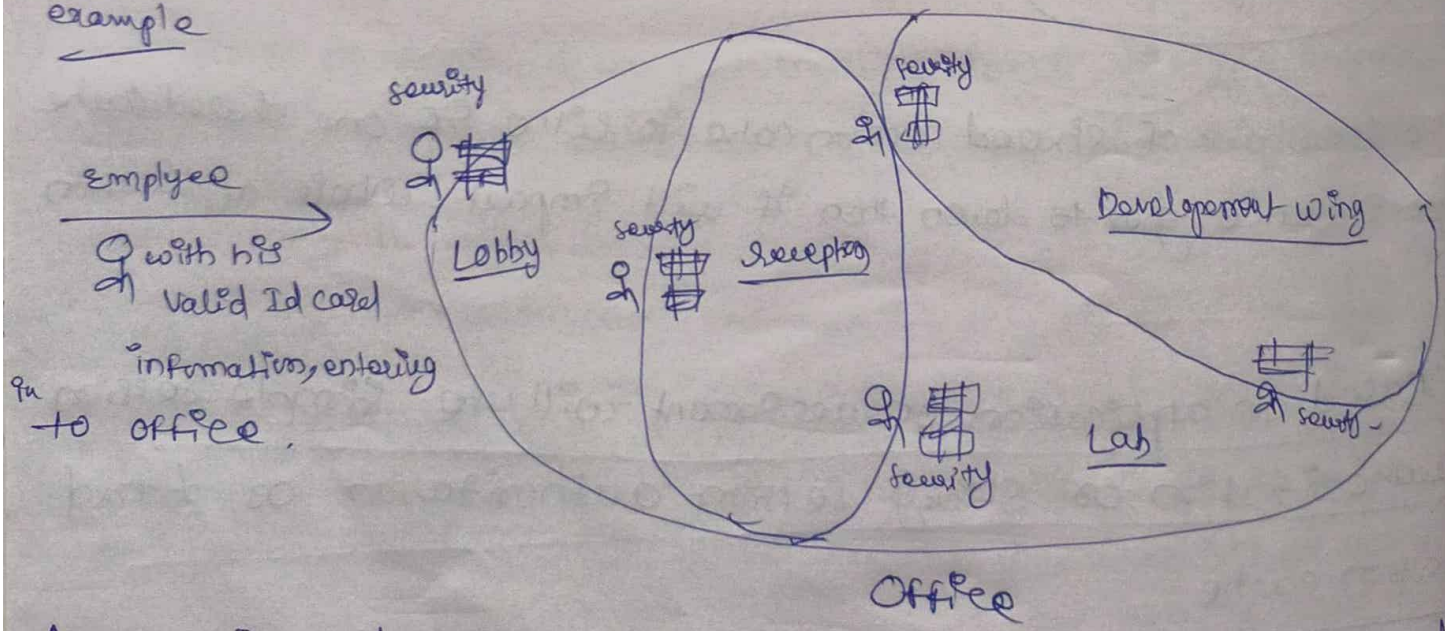
now days we are using micro service based applications, so here one micro service can interact/communicate with multiple services, so as we discussed in sticky session, shared session cache can't be suitable for this type of authorization because of drawbacks of respective authorization.

Ex:-



one service communicating multiple services.

real time example



As we discussed session based example, there security will check Id of worker in Id book then allow him into factory.

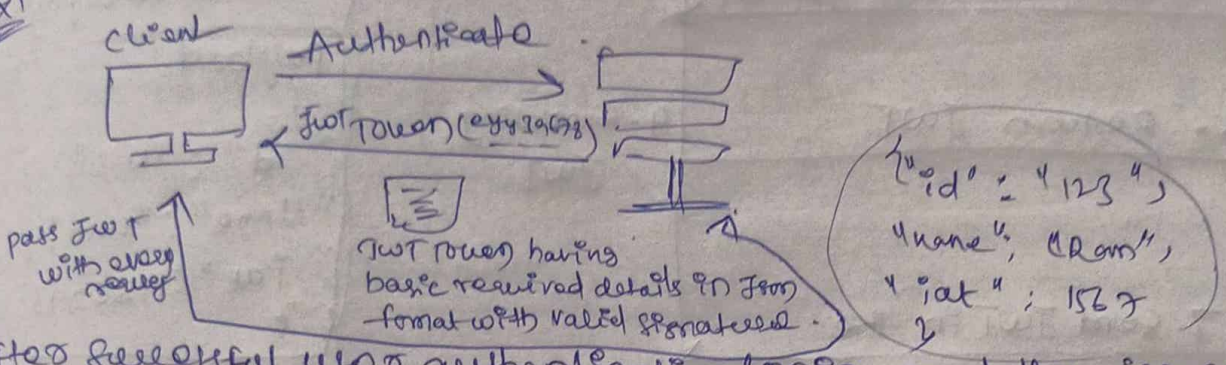
But here employee has all valid/required details in Id/access card, so security doesn't have any details to check, But just observe the valid/required details in card and then allow.

Ex① JWT based authorization (JSON web Tokens)

→ These are stateless servers. (It won't share session details on servers)

→ It is only for authorization purpose (or user authorization)
only not for authentication.

Ex!



After successful user authentication login and the server will do the minimum basic required information will be stored in Json format.

```
{
"id" : "12345",
"name" : "Ram",
"iat" : 156729
}
```

iat = issued At (identifies the time at which the JWT was issued)

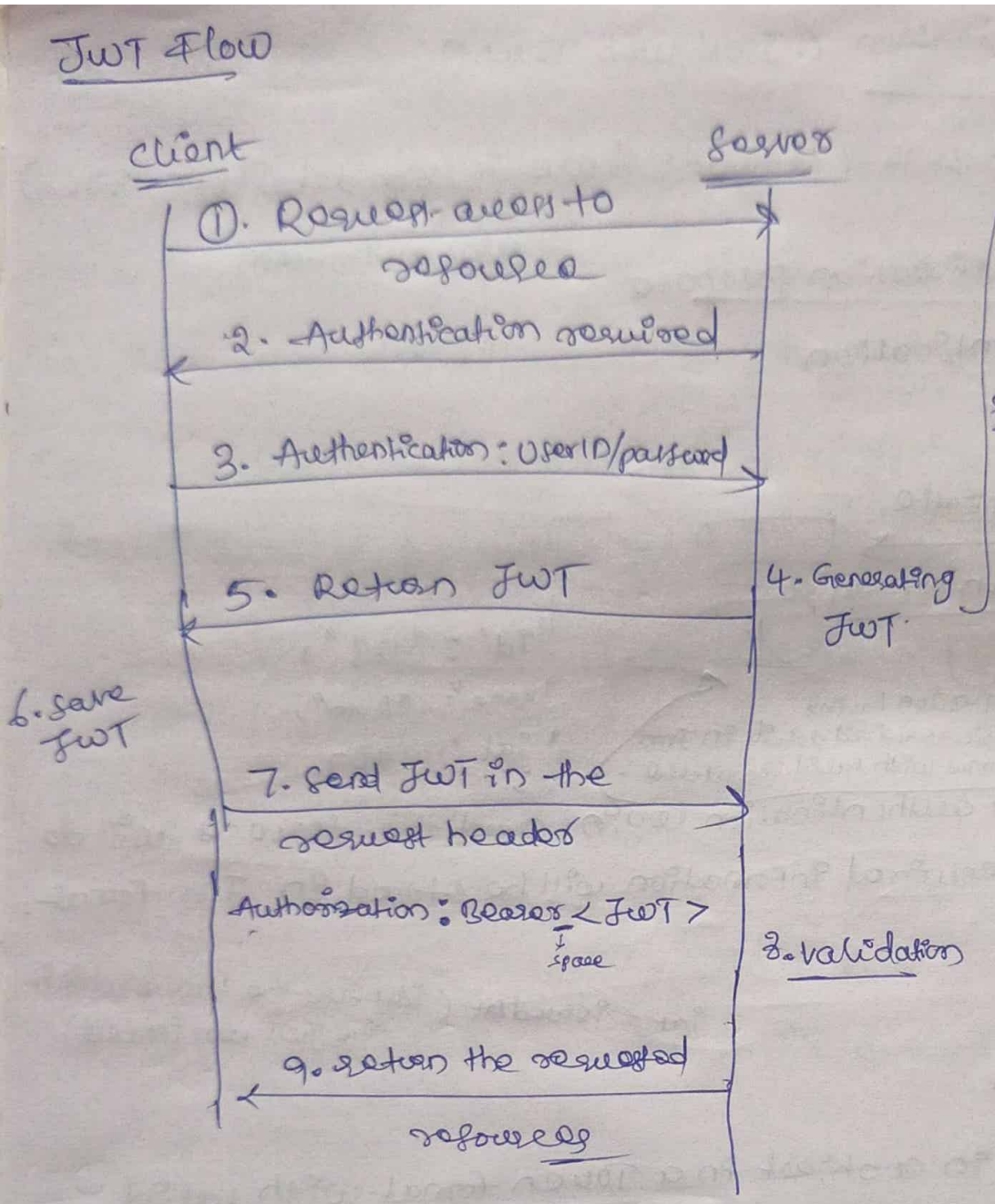
and It will store in a object in a Token format with valid signature then send to client.

⇒ It won't store any Json or session details in servers, It just (server)
send required information in Json format along with valid signature as a JWT token.
(eyy1296...)

⇒ client has to store JWT token in their respective places either local ds, or etc...

⇒ After successful authentication only server will generate authorization tokens.

JWT Flow



steps

- How server will generate JWT token
- (i) After successful authentication only
- (ii) It will take required details in JSON format as a payload

Payload
```
{
"id": "12345",
"Name": "Ram",
"iat": 156293
}
```

(iii) we can call every key-value pair in JSON as claims

(iii) Based on software application build It will prepare an Header having algorithm type and type of authorization

Header
```
{
"alg": "HS256",
"type": "JWT"
}
```

(iv) Then Both Header & payload convert to encoded (means Convert all strings, special characters to object which is like alphanumeric string) Base64Encode.

Header
```
{
"alg": "HS256",
"type": "JWT"
}
```
↓ Base64encode

payload
```
{"id": "12345",
"name": "Ram",
"iat" 156293
```
↓ Base64code

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6IjEyMzQ1IiwibmFtZSI6IlJhbSIsImlhdCI6MTU2MjkzfQ.
abcdefghijklmn

Signature
HMACSHA256(
base64UrlEncoded(header) + "." + base64UrlEncode(payload),
my-server-secret-key
)

Generate Signature

JWT consists of ③ parts

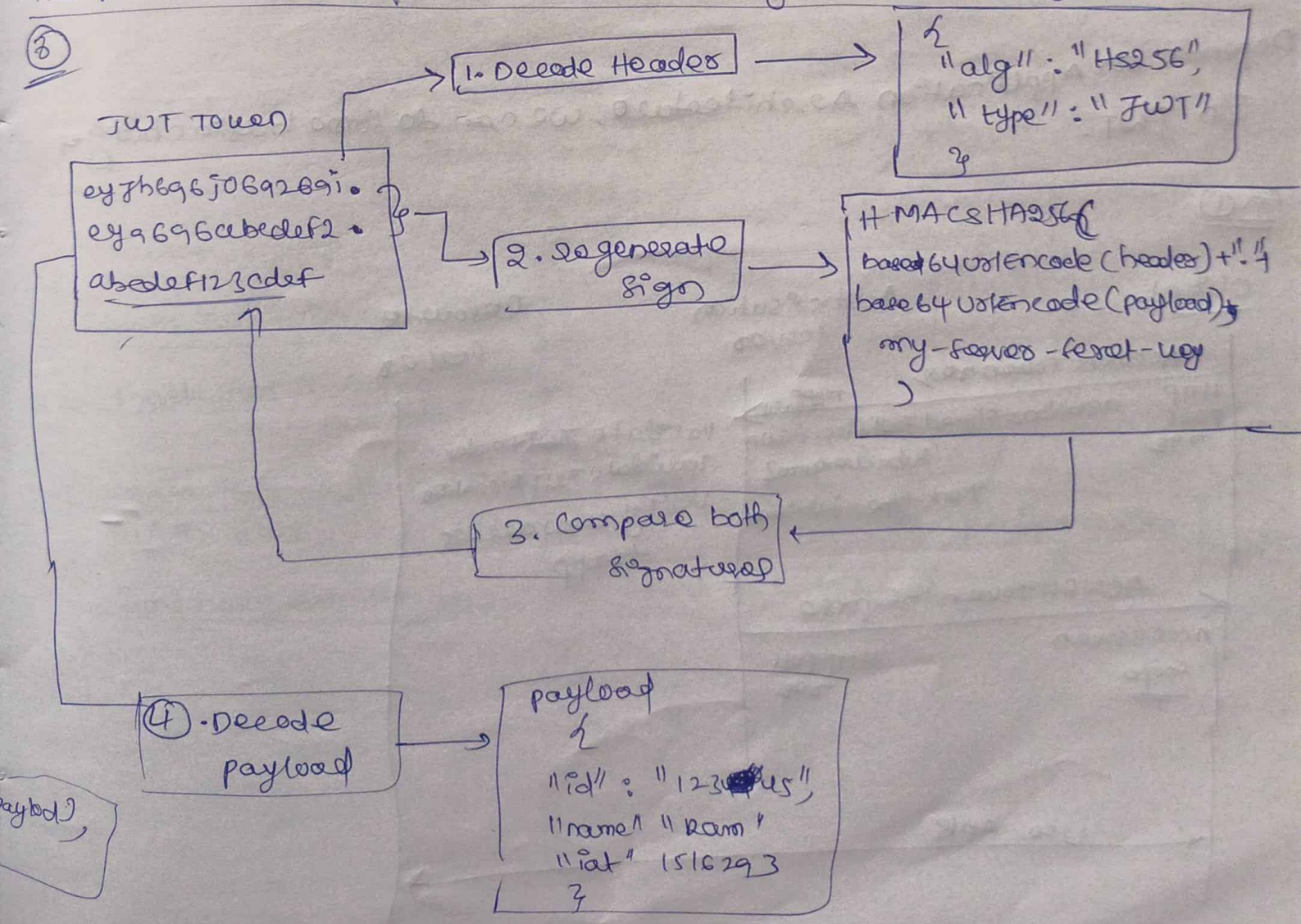
i) encoded value of Header as first

ii) then encoded value payload as second.

iii) signature which can use both encoded value of Header, payload and secret key of server then it will generate signature value as third value using Hashing.

this JWT send back to client and client will store some where. for every request, we needs to add this JWT token in request header as Authorization: Bearer <JWT> format.

How validation takes place at server for every request having JWT token



i) It will decode Header and analyze/note down which alg used

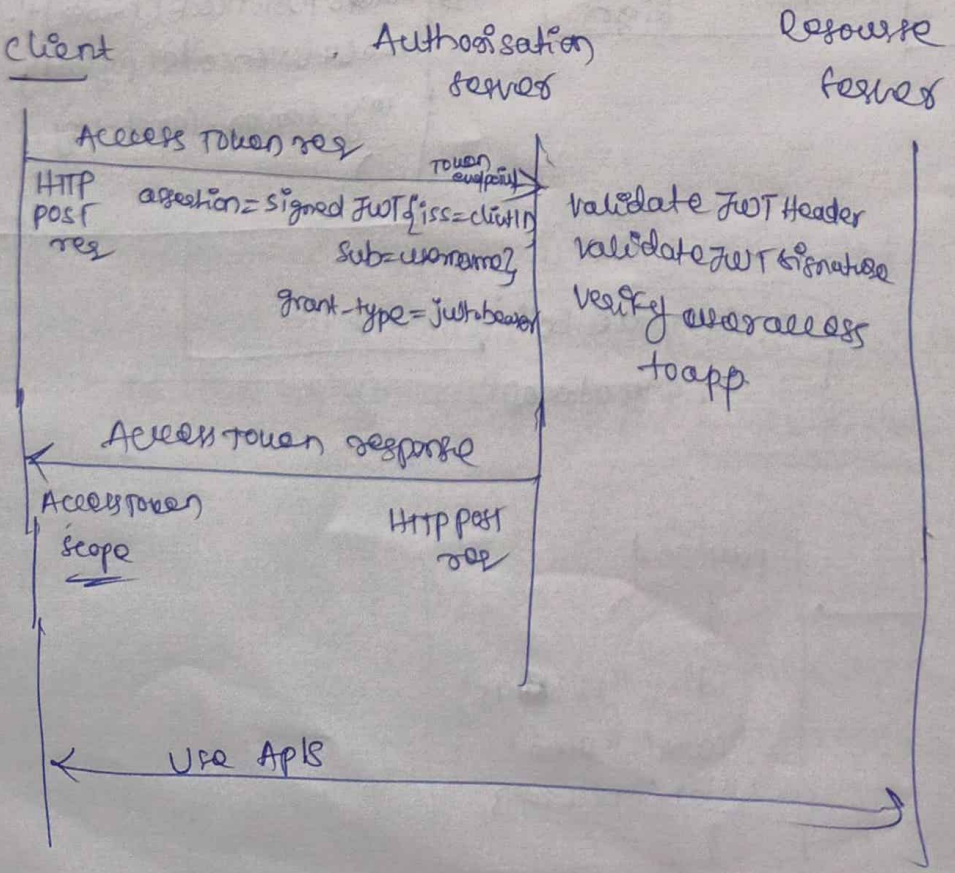
ii) Again re-generate signature using encoded value of both header and payload, my-secret-key.

iii) It compares both signatures, if both signatures are same then it will execute step ④ (Decode the payload) and ~~Observe~~ user details and give access)

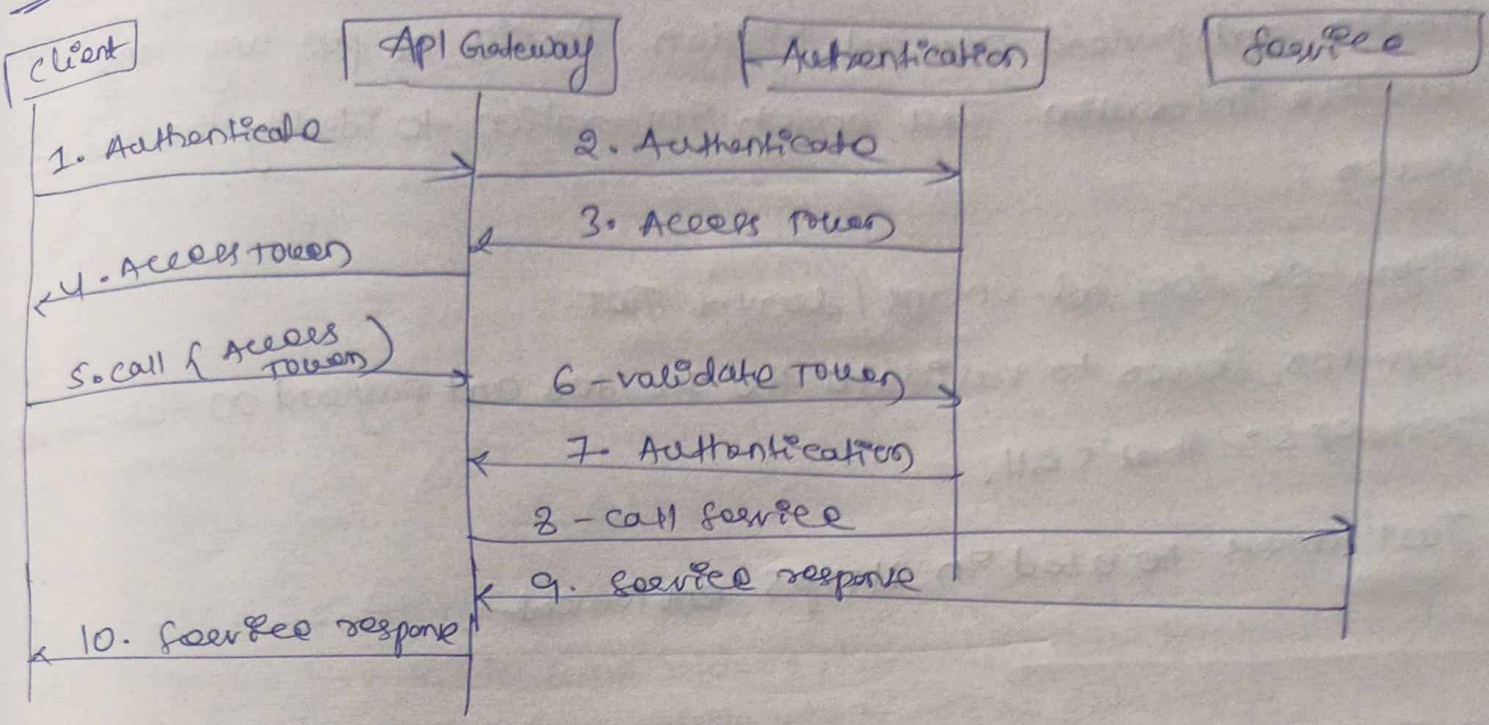
if both are different then server think someone has tampered our signature and sending something into then throws error

Depends on Application Architecture, we can do some basic changes in JWT.

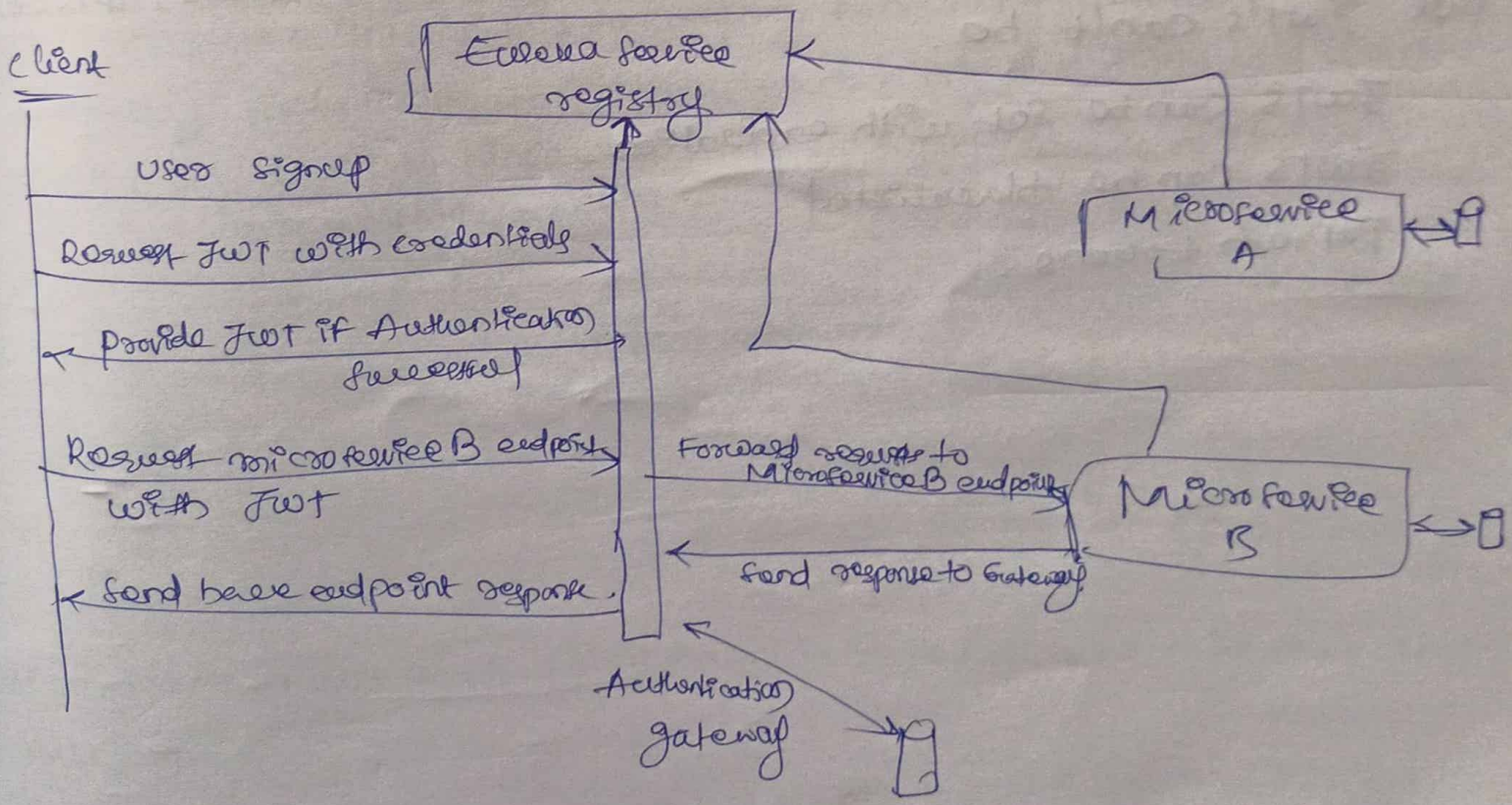
Ex ①



Ex ②


Client
Api Gateway
Authentication
Service
1. Authenticate
2. Authenticate
3. Access Token
4. Access Token
5. call { Access Token }
6 - validate Token
7. Authentication
8 - call service
9. service response
10. service response

Ex ③ :-


Client
Eureka service registry
User signup
Request JWT with credentials
Provide JWT if Authentication successful
Request microservice B endpoint with JWT
Send back endpoint response.
Forward request to Microservice B endpoint
Send response to Gateway
Microservice A
Microservice B
Authentication gateway

Note:-

① Header and payload are not hidden. So do not put any any sensitive information. Just enough information to identify the user on server.

② Signature does not encrypt / decrypt JWT.
Just for server to validate the Header and payload on subsequent requests; that's all.

③ JWT must be used in HTTPS enviement.

Are JWTs better than session IDs?

Sessions can be invalidated / terminated on the servers; if needed
But JWT's can't be.

JWTs can be set with expirsation.
JWTs can be blacklisted
Refresh tokens.

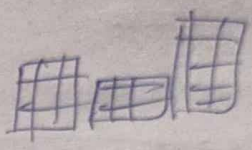
OAuth 2.0 and OIDC

(Open Authorization)

Ex:-

Visitor Badge

Video & photography crew



Company Conference Hall

Employee.

(Employee's Badge)

=> 1) Employee is organizing an event in company.

2) Employee has to do some video & photography But instead he planned to take a person from outside who knows all photography.

3) Employee can't provide his Employee's badge to photographer as he has access multiple areas, security issues.

4) If Employee informs this to company then company will provide ~~visitor~~ temporary visitor badge for photographer for limited time and limited access (only access to conference hall) on behalf/delegate to Employee. Or In which the meeting will ~~Products~~)

Ex

acrobat.adobe.com

User.

Google drive pdf file

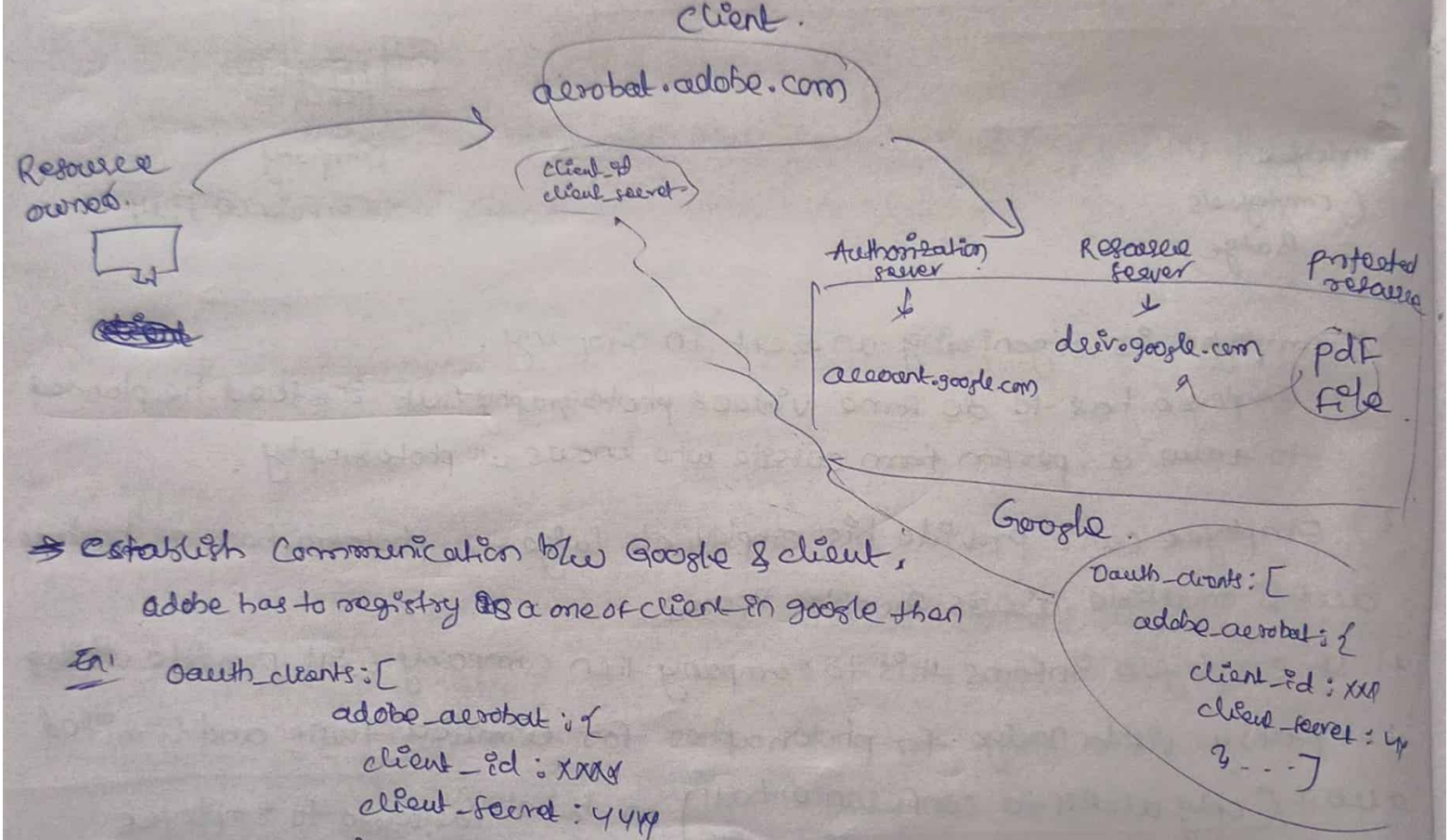
1) If user wants to edit the pdf which is present in google drive using third-party tool adobe.com.

2) We can do it manually like i) download pdf file to local pc from google drive
ii) upload pdf file in third party site.
iii) After editing, again download to local pc.
iv) again upload this to google drive

But no one used this.

3) Instead that we need to establish the communication b/w two services then only will achieve it like it will pick file from google drive directly by giving trusted access using OAuth services

→ OAuth provider is used to Authorize one service with other services.

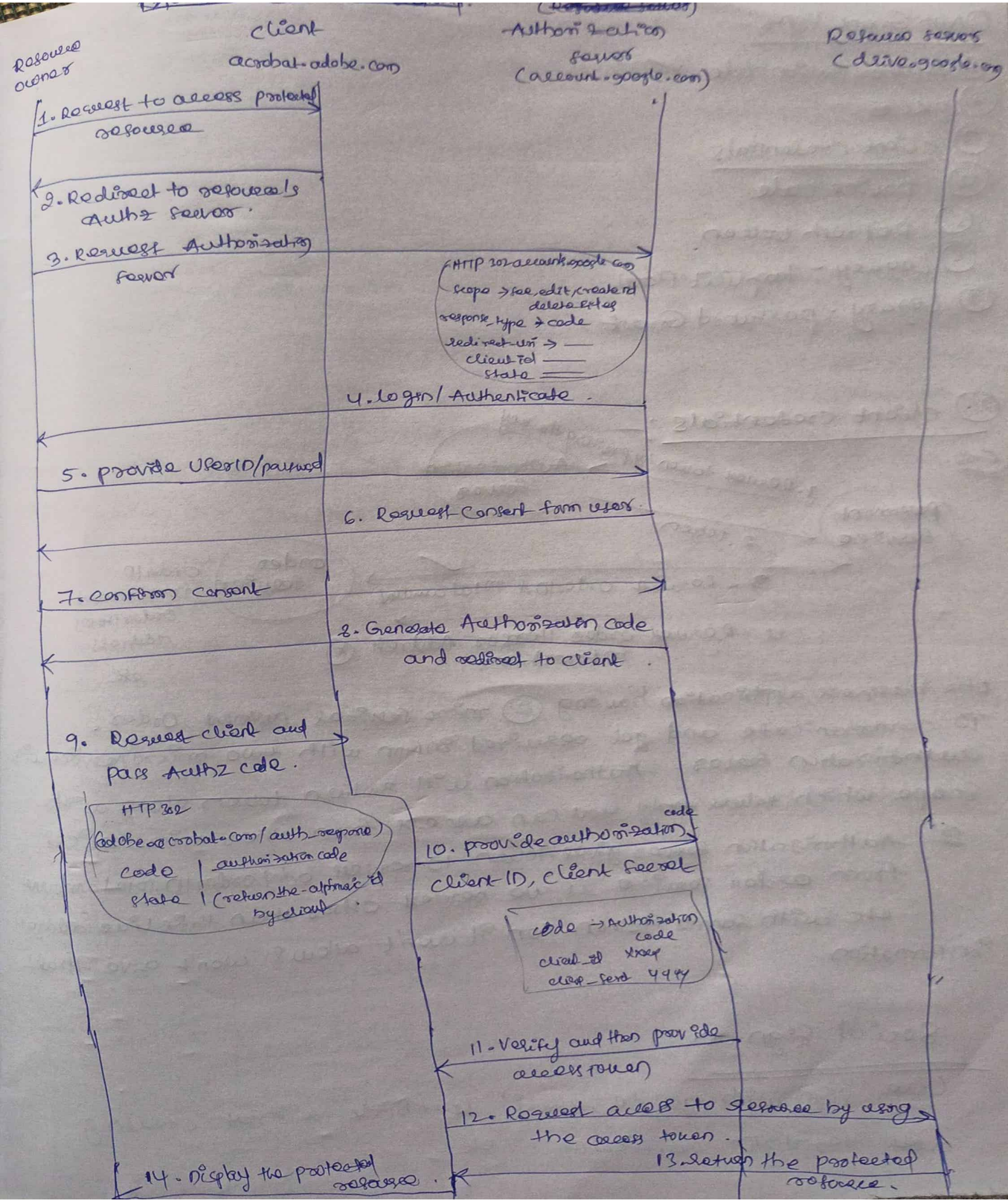


⇒ Establish communication b/w Google & client, adobe has to registry as a one of client in google then

Eg: Oauth_clients: [
adobe_aerobat: {
client_id: xxxx
client_secret: yyyy
}, ...]

OAuth flow (① Authentication code grant flow)

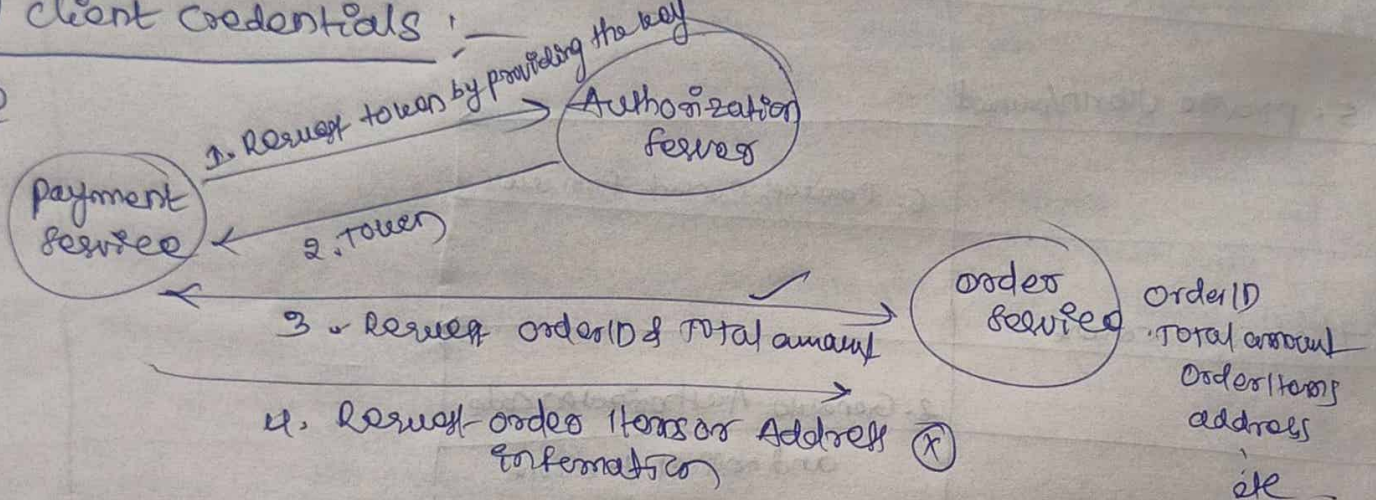
→ Here two separate business applications. Those google is separate
aerobat.adobe is separate


Resource owner
Client
acrobat.adobe.com
Authorization server
(account.google.com)
Resource server
(drive.google.com)
1. Request to access protected resources
2. Redirect to resource's Authz server.
3. Request Authorization server
HTTP 302 accounts.google.com
scope → see, edit, create and delete files
response_type → code
redirect_uri → ——
client_id ——
state ——
4. Login / Authenticate.
5. provide UserID/password
6. Request Consent form user.
7. confirm consent
8. Generate Authorization code and redirect to client.
9. Request client and pass Authz code.
HTTP 302
(adobe.acrobat.com/auth_response)
code | authorization code
state | (return the alphanumeric id by client)
10. provide authorization code
client-ID, client secret
code → Authorization code
client_id xxxx
client_secret yyyy
11. Verify and then provide access token
12. Request access to resource by using the access token.
13. Return the protected resource.
14. Display the protected resource.

Auth Grant Types:-

1. Authorization code ✓
2. PKCE
3. client credentials
4. Device code
5. Refresh token
6. Legacy: Implicit Flow
7. Legacy: password Grant.

2. client credentials :-



one business application having (3) micro services, payment, Order. To communicate and get secured info with two micro service will authorization server. Authorization will return token along with scope which values only you can access from other microservices.

Authorization server gives token to request only orderID, Total amount from order service. If we request other than this like address etc with same token then it won't allow & won't give that information.

Social Sign on

Like sign with google, email, facebook. without creating registration, verification --etc.

For example if we are creating application / sign on to application

Adobe Acrobat

New user? Creating an account

Email address

Continue

or

Continue with Google

Continue with facebook

Continue with apple

→ without user registration
User Authentication
User verification

Instead will use other service like google, facebook. It means if the user is valid for google/apple (if user have account in google, facebook) then it is valid for us also. So it just can use that authentication part from other valid service like google

If we want to use this kind of applications, then they built new application on OAuth 2 & openId