JWT : JSON Web Tokens
(JAWT)

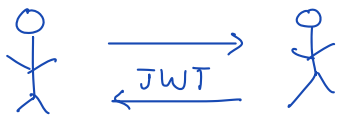
↓

very popular way for doing user authorization in web apps today.

↓

standard way for two parties to communicate securely.

open industry standard specification called RFC 7519, which outlines how a JWT should be structured, and how to use it for exchanging information or claims.



↳ secure communication

↓

other option for authorization is session tokens. They have one thing to blame HTTP. HTTP is a stateless protocol. Every interaction in HTTP needs to contain all the information needed for that interaction and nothing is remembered from before.

↓

This is okay incase of a static web app where the content doesn't change depending on the user. The issue comes in when the response from the server is dynamic and depends on who the user is. Need to tell the server the route alongwith identity, on each request.

↓

Doesn't align with actual experience. On logging in once, the website remembers us, and makes call with that

user identify.

↓

Multiple ways in which webapps manage and remember sessions. 2 of the popular ways use tokens. i.e. session tokens or JWT.

↓

example: customer has a support request with a customer care department. Calls them and informs them about the issues. The rep. tries some troubleshooting steps, but has to pass the request to another department, and asks the customer to call back tomorrow.

↓

cust. executive logs the ticket, and gives the customer a support ticket. Next time customer calls back, they don't have to go through the entire thing again.

↓

Next day, a diff. customer rep. can look at the ticket log and can straightaway start assisting the customer.

↓

kind of what is happening with authentication using session tokens. When the user authenticates, the server creates a session and keeps track of it itself. Creates a session id and gives it to the customer like the support ticket we saw above.

↓

subsequently client passes the same token to the server as a part of every request, and the server looks it up, and identifies who the client is, and can serve them accordingly.

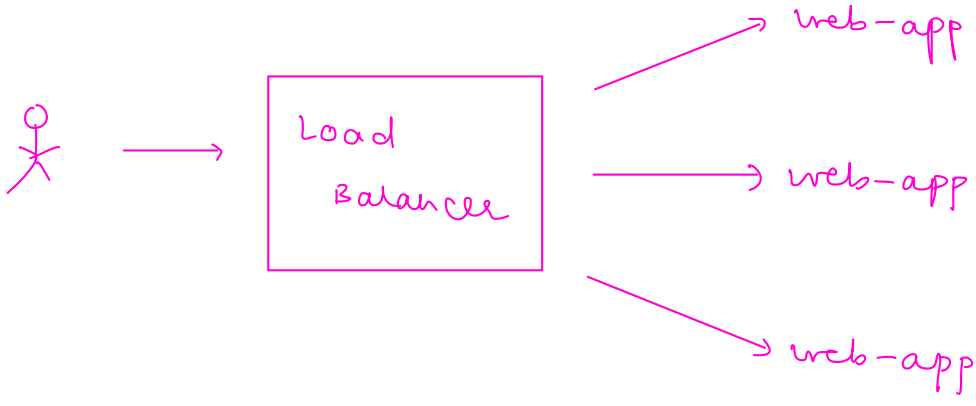
↓

How exactly the session id is passed depends on the implementation. Common approach to save the session id in a cookie, so that it automatically gets added to the cookie header on all subsequent requests.

↳ session id + cookies : most popular mechanism for authorization

↓ some problems

(i) approach assumes that there is always one monolithic web server app. modern web apps look different.



Load balancer decides which server to route the request to. Possible for initial request to be routed to different

server instance and another request in the session might go to some other instance.

↓

shared session cache (Redis cache) → one point of failure

↓

services opt to use the sticky session pattern, where the load balancer remembers which instance was handling the user. Scalability issues.

Problem also when multiple microservices working with each other. Session information carried over b/w different services.

↕ alternate model

Back to earlier example of customer support. Service folks don't remember state this time. No internet, phone, etc and the customer goes to the dept., and on explaining the issue, the service guy says, we'll work on it and please come back tomorrow.

↓

But don't want the customer to have to repeat the whole thing tomorrow. Instead of registering the case in the system, and giving the cust. case number.

↓

write down the details on a piece of paper, and ask them to bring it back next time.

↓

How does another cust. rep trust it ? Sign the piece of paper, and then that signature can be verified. This is the model implied in the JWT model. Instead of saving the information on a server / cache and returning the id, it returns the user information and other context as a token.

Everytime the customer makes a subsequent request, the client sends the whole JSON token with the request.

↓

Server verifies the token, and lets them in.

↓

JSON token exchanged over the web.

↓

The information is sent in a special signed format. All that JWT is. A way for a client and server to communicate & share information directly that has a meaning across multiple interactions.

Sessiontoken : reference tokens

Jwt : value tokens → can be sent as cookies, stored in localstorage.