Authentication

When testing APIs, handling authentication correctly is important to ensure that your tests can interact with secured endpoints effectively.

Authentication method supported by postman

1. No Auth
2. API Key
3. Basic Auth
4. Bearer Token
5. JWT bearer
6. OAuth 1.0
7. OAuth2.0

# No Auth

Postman will not shoot authorization details with a request unless you define an authtype. By default the No Auth type is selected as authtype in Postman, however if it is not selected you can choose No Auth from the Authorization tab > Type dropdown list, If your request does not need authorization.

# API Key

With **API key auth**, you ship a key-fee pair to the API both inside the request headers or question parameters. In the request Authorization tab, select API Key from the Type listing. Enter your key name and price, and pick out either Header or Query Params from the Add to dropdown listing.

Example site

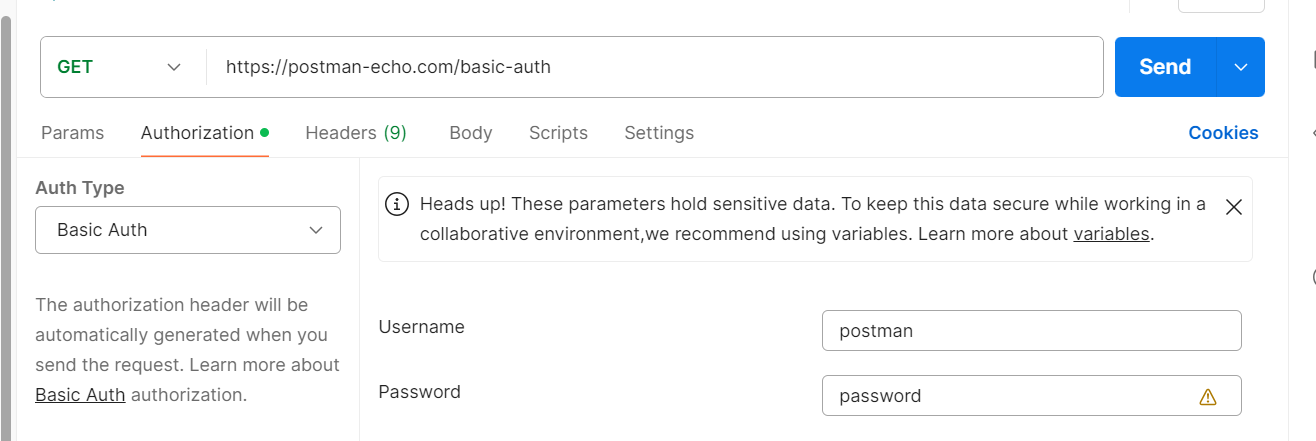
<https://api-ninjas.com/api/emoji>

# Basic Auth

A Basic Access Authentication is the most simple and basic type of authorization available. It requires just a username and password for checking the authorization of any person

The username and password are sent as header values in the ***Authorization header***.

1. Enter the endpoint ***https://postman-echo.com/basic-auth in*** *GET request*.
2. Send the request
3. Request will give 401 error for unauthorized
4. Now, we will provide the username and password for the request in the authorization tab.



1. Click on Send

# Bearer Token

Bearer tokens enable requests to authenticate using an access key, such as a JSON Web Token (JWT). The token is a text string, included in the request header. Select Bearer Token from the Auth Type dropdown list in the request Authorization tab. In the Token field, enter your API key value.

Example:

<https://simple-books-api.glitch.me/api-clients/>

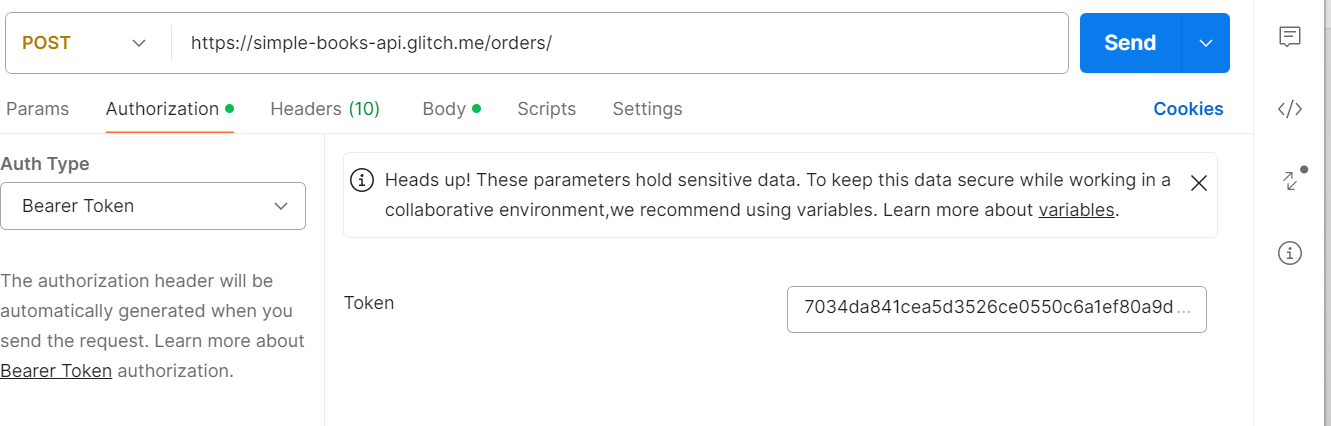
{

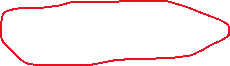
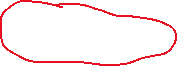
   "clientName": "Postman",

   "clientEmail": "user123333@gmail.com"

}

Add the access token generated as a bearer token





# JWT Token

Postman also supports generating JWT bearer tokens to authorize requests. You can enter a payload in an editor, and JWT tokens are generated and added to the request. Select JWT Bearer from the Auth Type dropdown list in the request Authorization tab.

* **Add JWT token to** - Select **Request Header** or **Query Param** to specify how the JWT token will be added to your request.
* **Algorithm** - Select an algorithm to use for the JWT token. Supported algorithms include:
  + **HS** - HMAC with SHA
  + **RS** - RSA (RSASSA-PKCS1-v1\_5) with SHA
  + **ES** - ECDSA with SHA
  + **PS** - RSA (RSASSA-PSS) with SHA
* **Secret** - The secret that’s used with the HMAC-SHA algorithm.
* **Secret Base64 encoded** - Select if the secret is encoded in the base-64 format.
* **Payload** - Enter the payload data for your JWT token, in JSON format.

# OAuth 1.0 and OAuth 2.0

OAuth 1.0 and OAuth 2.0 are both authorization frameworks that allow third-party applications to access HTTP services on behalf of a user without exposing the user's credentials.

Security: OAuth 2.0 is more secure than OAuth 1.0 because it uses SSL/TLS for communication, and has expiration times for access tokens.

With OAuth 2.0, you first retrieve an access token for the API, then use that token to authenticate future requests. Access tokens are typically short-lived, but the authorization server can also provide a long-lived refresh token. A client application can use the refresh token to automatically refresh the access token.

An example OAuth 2.0 flow:

* A client application makes a request for the user to authorize access to their data.
* If the user grants access, the application then requests an access token from the service provider, passing the access grant from the user and authentication details to identify the client.
* The service provider validates these details and returns a short-lived access token and a long-lived refresh token.
* The client uses the access token to request the user data from the service provider.
* When the access token expires, the client can use the refresh token to automatically request a new access token.