

Home

Fundamentals

- What is Skyflow?
- Get started with Skyflow
- Explore what Skyflow can do
- Authenticate**
- Accounts and environments
- Security best practices
- Get data into Skyflow

Vaults and storage

Access controls

Tokenization

Connections

Secure workflows

Detect

Financial data

SDKs

Elements

References

Authenticate

To use Skyflow's [Management API](#), [Data API](#), or [SDKs](#), you need a JWT bearer token (recommended) or an API Key to authenticate your API calls. JWT Bearer tokens and API keys allow scoped and permission-sensitive access to your account and the vaults it has. JWT Bearer tokens are time-limited, and API keys are long-lived.

- Skyflow's bearer tokens match the RFC's [Authorization Bearer Token Header](#) specification.

Prerequisites

- Studio
- API

Sign in to your Skyflow account. If you don't have an account, [sign up for a free trial](#).

- The following content uses the [Quickstart vault template](#). Adapt your commands accordingly.

Create a service account

When generating tokens using a Skyflow SDK or Python script, you must create a service account. A service account is an identity for machine access to your vault. The service account's roles, and the policies attached to those roles, decide the level of access a service account has to a vault.

- You must have Vault Owner permissions to create a service account.

If you already have a service account, skip to the method you want to use to generate a bearer token.

- In Studio, find the vault you want to access and click **Open**.
- In the side navigation, click **Access**.

- Click **Service accounts**.
- Click **Add service account**.

- For **Name** enter a value. For example, "Authenticate".
- For **Service account admins**, select the admins of your service account.

- Click **Next**.
- For **Authentication**, select your authentication type.

- Unless you have a valid business need, use JWT bearer tokens to authenticate.

- Optional: To enforce context-aware authentication, select **Inject context_identifier in bearer token**.
- Click **Next**.

- For **Assignments**, select the resource and roles for which you want to assign to the service account.
- For roles, select the role for which you want to assign to the resource.

- Click **Create Service Account**.

- Your browser downloads a *credentials.json* file. Store this file in a secure location. You'll need it to generate bearer tokens.

Generate a bearer token

You can generate a bearer token with an SDK, Python script, or (if you're in a trial environment) through Skyflow Studio. In production environments, we recommend using Skyflow-provided SDKs.

Use an SDK

When you integrate your backend systems with one of Skyflow's SDKs, you can use service account credentials to generate bearer tokens.


Bearer tokens generated from SDKs are valid for 60 minutes and let you make API calls allowed by the policies associated with the service account.

Step 1: Install the SDK


Now that you have your *credentials.json* file, it's time to prepare the SDK in the language of your choice.

```
Go      Java      Node.js  Python
```

Make sure your project is using Go Modules:

```
bash |   
1 go mod init
```

Then reference skyflow-go in a Go program with import:

```
go |   
1 import (  
2     soutil "github.com/skyflowapi/skyflow-go/serviceaccount/util"  
3     Skyflow "github.com/skyflowapi/skyflow-go/skyflow/client"  
4     "github.com/skyflowapi/skyflow-go/skyflow/common"  
5     logger "github.com/skyflowapi/skyflow-go/commonutils/logger"  
6 )
```

Step 2: Generate the bearer token


With the SDK installed, you can generate bearer tokens by passing your *credentials.json* file into an appropriate language-specific function.

```
Go      Java      Node.js  Python
```

The Go SDK has two functions that can take *credentials.json* and return a bearer token:

- `GenerateBearerToken(filepath)` takes the path to *credentials.json* as input.
- `GenerateBearerTokenFromCreds(credentials)` takes the body of *credentials.json* as a string as input.

Example

```
go |   
1 package main  
2  
3 import (  
4     "fmt"  
5     soutil "github.com/skyflowapi/skyflow-go/serviceaccount/util"  
6 )  
7  
8 var bearerToken = ""  
9  
10 func GetSkyflowBearerToken() (string, error) {  
11  
12     filePath := "%PATH_TO_CREDENTIALS.JSON%"  
13     if soutil.IsExpired(bearerToken) {  
14         newToken, err := soutil.GenerateBearerToken(filePath)  
15         if err != nil {  
16             return "", err  
17         } else {  
18             bearerToken = newToken.AccessToken  
19             return bearerToken, nil  
20         }  
21     }  
22     return bearerToken, nil  
23 }
```

Once you have your bearer token, you can programmatically interact with Skyflow APIs. See [next steps](#).

Use Studio

You can generate bearer tokens through Studio for short-term use. Bearer tokens generated in Studio are valid for 24 hours and let you make API calls allowed by the policies associated with your account.

- In Studio, click your account icon and choose **Generate API Bearer Token**.
- Click **Generate Token**.

Studio copies the token onto your clipboard.

Use a Python script

In production environments, generate bearer tokens using Skyflow-provided SDKs.


However, you can use this Python script to test generating bearer tokens on your local machine. To execute the script, make sure you have the *credentials.json* file, downloaded during the service account creation.

- This guide uses Homebrew to run Python installation commands. Adapt your Python installation accordingly.


Step 1: Prepare your environment

From your terminal, run the following commands to install python and the appropriate libraries.

Install Python version 3.5 or later.

```
bash |   
1 brew install python
```

Install the following libraries:


```
bash |   
1 pip3 install PyJWT  
2  
3 pip3 install requests  
4  
5 pip3 install cryptography
```

Step 2: Install the Python bearer token script

Now that you have your *credentials.json* file, it's time to prepare the Python script for generating a bearer token. To get started, copy, and paste the following `getBearerToken.py` script into your IDE.

```
python |   
1 import json  
2 # Requests lib installation: 'pip install requests'  
3 # PyJWT lib installation:  
4 # 'pip install pyjwt[cryptography]>=2.0.0' or  
5 # 'pip install cryptography; pip install pyjwt>=2.0.0'  
6 import jwt  
7 import requests  
8 import time  
9  
10  
11 def getSignedJWT(credsFile):  
12     # credsFile is the filepath to your credentials.json file  
13     # Load the credentials.json file into an object called creds  
14     fd = open(credsFile)  
15     creds = json.load(fd)  
16     fd.close()  
17  
18     # Create the claims object with the data in the creds object  
19     claims = {  
20         "iss": creds["clientId"],  
21         "key": creds["keyID"],  
22         "aud": creds["tokenURI"],  
23         "exp": int(time.time()) + (3600), # JWT expires in Now + 80 minutes  
24         "sub": creds["clientId"],  
25     }  
26     # Sign the claims object with the private key contained in the creds object  
27     signedJWT = jwt.encode(claims, creds["privateKey"], algorithm='RS256')  
28     return signedJWT, creds  
29  
30  
31 def getBearerToken(signedJWT, creds):  
32     # Request body parameters  
33     body = {  
34         'grant_type': 'urn:ietf:params:oauth:grant-type:jwt-bearer',  
35         'assertion': signedJWT,  
36     }  
37     # Request URI (= https://api.skyflow.dev/v1/auth/sa/oauth/token)  
38     tokenURI = creds["tokenURI"]  
39  
40     # Send the POST request using your favorite Python HTTP request lib  
41     r = requests.post(tokenURI, json=body)  
42     return r.text  
43  
44  
45 jwtToken, creds = getSignedJWT('%PATH_TO_CREDENTIALS.JSON%')  
46 bearerToken = getBearerToken(jwtToken, creds)  
47 print(bearerToken)
```

Locate the `jwtToken`, `creds` parameter and enter the full path to your *credentials.json* file.

```
python |   
1 jwtToken, creds = getSignedJWT('%PATH_TO_CREDENTIALS.JSON%')
```


Save this file as *getBearerToken.py* to a secure location. You'll need it to execute the script.

Step 3: Generate a bearer token

From your terminal, navigate to the folder with the *getBearerToken.py* script and run the following command to generate a bearer token.

```
bash |   
1 python3 getBearerToken.py
```

Skyflow validates the JWT assertion and returns a bearer token.

```
javascript |   
1 {  
2     "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczov  
3     "tokenType": "Bearer"  
4 }
```

Once you have your bearer token, you can programmatically interact with Skyflow APIs. See [next steps](#).

Get a bearer token for a client-side SDK


Skyflow's client-side SDKs doesn't have direct methods to generate bearer tokens. Bearer token generation typically involves sensitive operations that shouldn't happen in client-side environments (like browsers) because of security concerns.

Bearer tokens are usually generated on a server, where you can securely store your application's credentials and use them to authenticate with the Management API to retrieve a token. You then pass the token to the client-side application, which can use a client-side SDK to interact with Skyflow.


Here's a general outline of the steps you would take to generate a bearer token and use it with a client-side SDK:

- Generate a bearer token with a server-side SDK:** Create a backend service with a [server-side SDK](#) to handle authentication. This service uses your Skyflow credentials to authenticate and retrieve a bearer token. Make sure to store your credentials securely, and don't expose them to the client.
- Pass the token to the client:** Provide an endpoint in your backend service that the client can call to retrieve the bearer token. The client calls this endpoint to get the token when needed.
- Initialize Skyflow client:** In your client application, use the client-side SDK to initialize a Skyflow client. You need to pass a helper function to the `getBearerToken` parameter of the initialization method. This function should make an API call to your backend service to retrieve the bearer token.

Here's a sample JavaScript implementation of the `getBearerToken` function in your client code:

```
javascript |   
1 const getBearerToken = () => {  
2     return new Promise((resolve, reject) => {  
3         // Make an API call to your backend service to get the bearer token.  
4         fetch(`${pathToYourBackendService}/endpoint`)  
5         .then(response => response.json())  
6         .then(data => {  
7             if (data && data.accessToken) {  
8                 resolve(data.accessToken); // Resolve the promise with the token.  
9             } else {  
10                reject('Failed to retrieve access token'); // Reject the promise if th  
11            }  
12        })  
13        .catch(error => {  
14            reject(error); // Reject the promise on network errors or other issues.  
15        })  
16    });  
17 };
```

You would initialize the Skyflow client like this:

```
javascript |   
1 const skyflowClient = Skyflow.init({  
2     vaultId: 'your-vault-id',  
3     vaultUrl: 'your-vault-url',  
4     getBearerToken: getBearerToken,  
5     options: {  
6         logLevel: Skyflow.LogLevel.ERROR, // Optional  
7         env: Skyflow.Env.PROD, // Optional  
8     }  
9 });
```

Remember that generating and handling bearer tokens requires careful security considerations, as these tokens provide access to your Skyflow vault. Always keep your credentials secure, and never expose them to the client-side.

Enable API key-based authentication

- This feature is in beta. Contact your Skyflow representative to enable it.

API key-based authentication is an alternate but less secure method of authenticating service accounts.

When using API key-based authentication, remember the following practices:


- API keys should only be the preferred authentication mechanism if a specific use case warrants its need. For example, partners already authenticate via your SDKs, and you want them to refrain from authenticating again with Skyflow.
- API keys don't expire. In some instances, this is a convenience in terms of usability. However, API keys reduce security, and the onus is on you and your partners or customers to protect the API key and monitor its usage securely.
- Only use API key-based authentication to insert records into a Skyflow vault. Don't use API key-based authentication to read or delete data.

Create a service account with API key-based authentication

When configuring your authentication method for a service account, you can use either JWT bearer tokens or API keys. You can generate API keys by calling the Management API to retrieve a token or create a service account. After configuring your service account, you can create, rotate, disable, or delete the key with the Management API.

- You can't generate bearer tokens with a service account configured to use API keys or create API keys with a service account configured to use bearer tokens.

Call the Service Accounts API with `apiKeyEnabled` set to true.

```
bash |   
1 curl -s -X POST "$MANAGEMENT_URL/v1/serviceAccounts" \  
2     -H "Authorization: $TOKEN" \  
3     -H "content-type: application/json" \  
4     -d '{  
5         "resource": {  
6             "id": ""$VAULT_ID"",  
7             "type": ""$VAULT_TYPE"",  
8         },  
9         "serviceAccount": {  
10            "id": ""$SERVICE_ACCOUNT_ID"",  
11            "name": ""$SERVICE_ACCOUNT_NAME"",  
12            "displayName": ""$SDISPLAY_NAME"",  
13            "description": ""$DESCRIPTION""  
14        }  
15        "clientConfiguration": {  
16            "enforceContextID": true,  
17            "enforceSignedDataTokens": true  
18        }  
19        "apiKeyEnabled": true  
20    }'
```

Next steps

You can now use your bearer token to interact with Skyflow APIs.

If you're new to Skyflow, see [Get started with Skyflow](#). Otherwise, see the various ways you can use Skyflow APIs:

- [Data API](#)
- [Management API](#)
- [SDKs](#)
- [Postman](#)

In this article

Prerequisites

Create a service account

Generate a bearer token

Use an SDK

Use Studio

Use a Python script

Get a bearer token for a client-side SDK

Enable API key-based authentication

Create a service account with API key-based authentication

Next steps