Quickstart

This quickstart guide will help you set up and make calls on the IPFS network using the Infura endpoints.

info To access the IPFS network through Infura, you'll need to add a billing card to your account, even if you're using a free account. Refer to thebilling details for more information.

Prerequisites

Before you begin, ensure you have a validAPI key andAPI key secret .

Make calls

cURL

cURL generates the auth header and encodes your credentials behind the scenes.

Include the-u flag with the authentication information.

```
curl -X POST -F file=@myfile -u ":" "https://ipfs.infura.io:5001/api/v0/add"
```

JavaScript

Wrap JavaScript calls to IPFS with the InfuraAuthorization header.

```
xhr . setRequestHeader ( "Authorization" ,

"Basic "

+

btoa ( < API_KEY

+

":"

+

< API_KEY_SECRET

      ) ) ;
```

NodeJS

Change the and in the NodeJS example code below.

Save the following script to a file, e.g.index.js .

```
const https =

require ( "https" ) ;

const projectId =

"" ; const projectSecret =

"" ;

const options =

{ host :

"ipfs.infura.io" , port :

5001 , path :

"/api/v0/pin/add?arg=QmeGAVddnBSnKc1DLE7DLV9uuTqo5F7QbaveTjr45JUdQn" , method :

"POST" , auth : projectId +
```

":"

+ projectSecret , } ;

let req = https . request (options ,

(res)

=>

{ let body =

"" ; res . on ("data" ,

function

(chunk)

{ body += chunk ; }) ; res . on ("end" ,

function

( )

{ console . log (body) ; }) ; }) ; req . end ( ) ; In a terminal window, run the script withnode index.js

Output something like:

{"Pins":["QmeGAVddnBSnKc1DLE7DLV9uuTqo5F7QbaveTjr45JUdQn"]}

Python

Change theprojectId andprojectSecret in the Python example code below.

Save the following script to a file, e.g.index.py .

import requests

projectId

"" projectSecret =

"" endpoint =

"https://ipfs.infura.io:5001"

CREATE AN ARRAY OF TEST FILES

files

{ 'file' :

'myNFT.png' }

ADD FILE TO IPFS AND SAVE THE HASH

response1

requests . post (endpoint +

'/api/v0/add' , files = files , auth = (projectId , projectSecret)) print (response1) hash

= response1 . text . split (",") [1] . split (":") [1] . replace ('"' , '') print (hash)

READ FILE WITH HASH

params

{ 'arg' :

hash } response2 = requests . post (endpoint +

'/api/v0/cat' , params = params , auth = (projectId , projectSecret)) print (response2) print (response2 . text)

REMOVE OBJECT WITH PIN/RM

response3

requests . post (endpoint +

'/api/v0/pin/rm' , params = params , auth = (projectId , projectSecret)) print (response3 . json ( )) Run the script withpython index.py .

Output something like:

QmWtBbpKST49AQFLx8HAdwwjUu7HBP2wrtAH1x8df5qrVm myNFT.png {'Pins': ['QmWtBbpKST49AQFLx8HAdwwjUu7HBP2wrtAH1x8df5qrVm']}

kubo-rpc-client

Use the officialkubo-rpc-client JavaScript client library.

Install the library withnpm install --save kubo-rpc-client .

Save the following script to a file, e.g.index.mjs .

import

{ create }

from

'kubo-rpc-client'

const projectId =

"" ; const projectSecret =

"" ; const auth = "Basic "

+

Buffer . from (projectId +

":"

+ projectSecret) . toString ("base64") ;

const client =

create ({ host :

"ipfs.infura.io" , port :

5001 , protocol :

"https" , headers :

{ authorization : auth , } , }) ;

client . pin . add ("QmeGAVddnBSnKc1DLE7DLV9uuTqo5F7QbaveTjr45JUdQn") . then ((res)

=>

{ console . log (res) ; }) ; Run withnode index.mjs .

Output something like:

CID(QmeGAVddnBSnKc1DLE7DLV9uuTqo5F7QbaveTjr45JUdQn)

go-ipfs-api

- Use the official IPFSgo-ipfs-api
- GoLang API.
- Install withgo get -u github.com/ipfs/go-ipfs-api
- .
- Create a go module withgo mod init infura
- .
- Save the following script to a file, e.g.index.go
- , and include the Infuraauth
- header with thehttp.RoundTripper
- wrapper.

package main

import

("fmt" "net/http" "os" "strings"

ipfsApi "github.com/ipfs/go-ipfs-api"

// v0.2.0)

func

main ( )

{ projectId :=

"" projectSecret :=

""

shell := ipfsApi . NewShellWithClient ("https://ipfs.infura.io:5001" ,

NewClient (projectId , projectSecret)) cid , err := shell . Add (strings . NewReader ("Infura IPFS - Getting started demo.")) if err !=

nil

{ fmt . Println (err) os . Exit (1) }

fmt . Printf ("Data successfully stored in IPFS: %v\n" , cid) }

// NewClient creates an http.Client that automatically perform basic auth on each request. func

NewClient (projectId , projectSecret string)

* http . Client { return

& http . Client { Transport : authTransport { RoundTripper : http . DefaultTransport , ProjectId : projectId , ProjectSecret : projectSecret , } , } }

// authTransport decorates each request with a basic auth header. type authTransport struct

{ http . RoundTripper ProjectId string ProjectSecret string }

func

(t authTransport)

RoundTrip (r * http . Request)

(* http . Response ,

error)

{ r . SetBasicAuth (t . ProjectId , t . ProjectSecret) return t . RoundTripper . RoundTrip (r) } Run withgo run index.go .

Output something like:

CID(QmeGAVddnBSnKc1DLE7DLV9uuTqo5F7QbaveTjr45JUdQn)

go-ipfs-http-client

- Use the official IPFSgo-ipfs-http-client
- GoLang API.
- Install withgo get github.com/ipfs/go-ipfs-http-client
- .
- Create a go module withgo mod init infura
- .
- Save the following script to a file, e.g.index.go
- , and include the Infuraauth
- header with thehttp.RoundTripper
- wrapper.

package main

import

("context" "encoding/base64" "fmt" "net/http" "os" "strings"

ipfsFiles "github.com/ipfs/go-ipfs-files"

// v0.0.8 ipfsApi "github.com/ipfs/go-ipfs-http-client"

// v0.1.0)

func

main ( )

{ projectId :=

"" projectSecret :=

""

httpClient :=

& http . Client { } httpApi , err := ipfsApi . NewURLApiWithClient ("https://ipfs.infura.io:5001" , httpClient) if err !=

nil

{ fmt . Println (err) os . Exit (1) } httpApi . Headers . Add ("Authorization" ,

"Basic "

+

basicAuth (projectId , projectSecret))

content := strings . NewReader ("Infura IPFS - Getting started demo.") p , err := httpApi . Unixfs ( ) . Add (context . Background ( ) , ipfsFiles . NewReaderFile (content)) if err !=

nil

{ fmt . Println (err) os . Exit (1) }

fmt . Printf ("Data successfully stored in IPFS: %v\n" , p . Cid ( ) . String ( )) }

func

basicAuth (projectId , projectSecret string)

string

{ auth := projectId +

":"

+ projectSecret return base64 . StdEncoding . EncodeToString ([ ] byte (auth)) } Run withgo run index.go .

Example output:

Data successfully stored in IPFS: QmTHr95iiwSTA2USxx4g5kKnhqsNRixqohhwxjvdXmSrWn

Next Steps

Now that you have successfully made a call to the IPFS network, you can explore more functionalities and APIs provided by Infura. Here are some suggestions:

- Explore other IPFS APIs
- : Infura supports a wide range of APIs. You can find more information in the JSON-RPC API method documentation
- .
- Try out different networks
- : Infura supports multiple networks including Arbitrum, Linea, Polygon, Optimism, and more.
- Monitor your usage
- : Keep an eye on your usage on the Infura dashboard
- to ensure you're not hitting your rate limits.

Remember, the Infura community is here to help. If you have any questions or run into any issues, check out the Infura community for help and answers to common questions.

Last updatedonApr 19, 2024