Retrieve Data

Learn how to retrieve data for the Filecoin network with different approaches as well as incentivize data retrievability.

Retrieve data using retrieval clients

To retrieve data stored on the Filecoin network, the basic process involves making retrieval requests to Service Providers (SPs) who initially stored the data, using either the Content ID (CID) or the storage deal ID.

A programmatic option is to utilize Filecoin retrieval clients, which handle the intricate retrieval process behind the scenes. By simply providing a Content ID (CID), retrieval clients can efficiently return your data either via the command-line interface (CLI) or through the programmable method.

Ingredients

With a given CID, you can use any of the following retrieval clients to retrieve content.

- Lassie
- : optimizes for most efficient available retrieval protocols.
-
    - go-car
-
    - : a content addressable archive utility.
- *
- Saturn
- : a Web3 CDN in Filecoin's retrieval market.
-

Instructions

1. retrieving content with Lassie
2.

Lassie is designed to fetch content in the content-addressed archive (CAR) form. In most cases, you will require additional tooling, such as the go-car library, to work with CAR files effectively.

The Lassie command line interface (CLI) provides the simplest method for retrieving content from the Filecoin/IPFS network. By using thelassie fetch command and passing the CID as an argument, you can easily retrieve the desired content.

```

Copy lassie fetch-o-|car extract

```

For example,

```

Copy lassie fetch -p bafybeic56z3yccnla3cutmvqsn5zy3g24muupcsjtoyp3pu5pm5amurjx4 | car extract

```

Lassie can also serve as a go library within your Golang application when programmatically retrieving content from the network. To utilize Lassie in your code, you need to install the dependency and import it into your program following the instructionshere .

The following example demonstrates how to use the Lassie library to fetch a CID.

```

Copy packagemain

import( "context" "fmt" "os"

"github.com/filecoin-project/lassie/pkg/lassie" "github.com/filecoin-project/lassie/pkg/storage" "github.com/filecoin-project/lassie/pkg/types" "github.com/ipfs/go-cid" trustlessutils"github.com/ipld/go-trustless-utils" )

// main creates a default lassie instance and fetches a CID funcmain() { ctx:=context.Background()

// Create a default lassie instance lassie, err:=lassie.NewLassie(ctx) iferr!=nil{ panic(err) }
```

// Prepare the fetch rootCid:=cid.MustParse("bafybeic56z3yccnla3cutmvqsn5zy3g24muupcsjtoyp3pu5pm5amurjx4")// The CID to fetch store:=storage.NewDeferredStorageCar(os.TempDir(), rootCid)// The place to put the CAR file request, err:=types.NewRequestForPath(store, rootCid,"", trustlessutils.DagScopeAll,nil)// The fetch request iferr!=nil{ panic(err) }

// Fetch the CID stats, err:=lassie.Fetch(ctx, request) iferr!=nil{ panic(err) }

// Print the stats fmt.Printf("Fetched%dblocks in%dbytes\n", stats.Blocks, stats.Size) }

```

1. retrieving content with Saturn
2.

Saturn Javascript Client is still a work in progress and not recommended for use in production yet. The following code example demonstrates how to use the Saturn in the Javascript program to fetch a CID.

```

Copy import{ Saturn }from'@filecoin-saturn/js-client'; // initialize a Saturn instance constsaturn=newSaturn({ clientKey:"...",// Key used for verification // ... other options });

// fetch content using the client with given CID constcidPath='https://samplepath/ipfs/{cid}'; constoptions={ fallbackLimit:5, raceNodes:true, };

(async()=>{ try{ forawait(constchunkofsaturn.fetchContentWithFallback(cidPath,options)) { // Process each chunk of data console.log(chunk); } }catch(error) { console.error('Error fetching content:',error); } })();

```

For quick retrieval of existing datasets with the methods above, check out the Filecoin Dataset Explorer .

Previous Store Data Next Privacy & Access Control

Last updated14 days ago
```