**Drawbacks of IPFS:**

* IPFS is not really optimized for private data at all.
* Currently, IPFS does not provide a built-in solution for storing private data. Encryption can be used to store and/or transfer private data over IPFS.
* Content has to be pinned for it to be accessible and saved, but even then, it's not permanent.
* IPFS cannot guarantee data availability upon being requested.
* As of today (10-07-2019) project is still in Alpha, lots of development is happening, API might change.

**Advantages of IPFS:**

* Open source.
* Peer-to-peer (p2p) file sharing system.
* IPFS is a shared public filesystem.

**Decentralized file storage methods(modern):**

Bluzelle

Filecoin

Storj

BigchainDB

**Cloud storage link:**

[**https://www.techradar.com/news/the-best-cloud-storage#best-free-cloud-storage**](https://www.techradar.com/news/the-best-cloud-storage#best-free-cloud-storage)

**S3**

**Drawbacks of S3:**

* For any serious support, they require the “AWS Support Plan,” which it’s billed separately (around 29 USD per month).

**Advantages of S3:**

* AWS S3 is a cheaper solution to store such a huge volume of data.
* Amazon S3 supports three different forms of encryption.
* Around 0,025 USD per GB/month with the Standard class, and 0,004 USD per GB/month with Glacier class. Prices also go down from time to time depending on the market
* It’s “developer-friendly.” They offer SDKs for many programming languages (e.g., Python, Javascript) and the terminal (aws-cli).
* Amazon S3 provides data transfer your data over automatic encryption and SSL once it is uploaded.
* Different storage classes for each necessity:
  + Standard (frequent use)
  + Infrequent Access Storage (infrequent use)
  + Glacier (long-term storage)
* New users receive for free 5GB each month during the first year.
* Amazon S3 is the best option as you can find documentation and tutorials everywhere on the internet.

### **How to use it with Javascript**

**1.Install the library**

npm install aws-sdk

**2.Authenticate**

var accessKeyId = 'accessKeyId';

var secretAccessKey = 'secretAccessKey';

var region = 'us-east-1';

var s3 = new AWS.S3({

accessKeyId: accessKeyId,

secretAccessKey: secretAccessKey,

region: region

})

Full documentation link:

<https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html#constructor-property>

S3 comes with a bunch of features to encrypt your data at rest.

**Data at rest** refers to the inactive data stored physically on a disk.

**Data in use** refers to the data is in memory.

**Data in transit** refers to the data is on the network.

**S3 with Encryption**

S3 provides two types of encryption:

1. **Server Side Encryption(SSE):**
2. **Client Side Encryption(CSE):** Client-side encryption means that you encrypt the data before you send it to AWS. It also means that you decrypt the data that you retrieve from AWS.

Is it okay if AWS technically sees your raw data? If yes, server-side encryption is the right option for you. If not, go with client-side encryption

Encrypt most of our data using AWS KMS (Client-Side) encryption where we manage the key and the data is secure on the servers.

**aws s3 cp path/to/local.file s3://bucket-name/sse-aes --sse AES256**

**aws s3 cp path/to/local.file s3://bucket-name/sse-kms --sse aws:kms**

**aws s3 cp path/to/local.file s3://bucket-name/sse-kms-cmk –sse aws:kms –sse-kms-key-id KMS\_KEY\_ID**