UNDERSTANDING HOW IPFS DEALS WITH FILES

CORE COURSE A

Alan Shaw
JS IPFS

Mikeal Rogers
IPLD

Steven Allen
Go IPFS

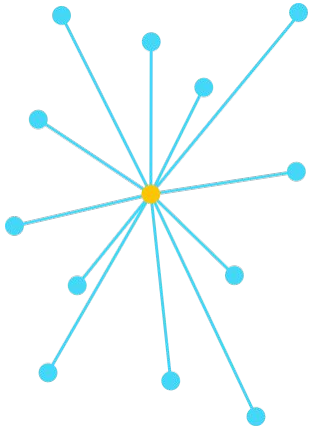
IPFS Camp

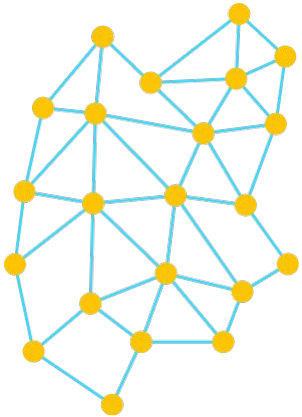
Core Course A:

WHY IMMUTABILITY?

Why Immutability?:

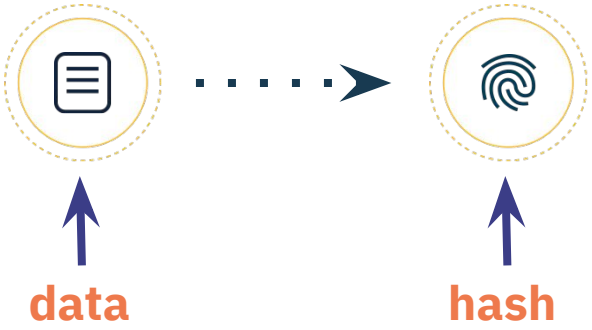
NO TRUST





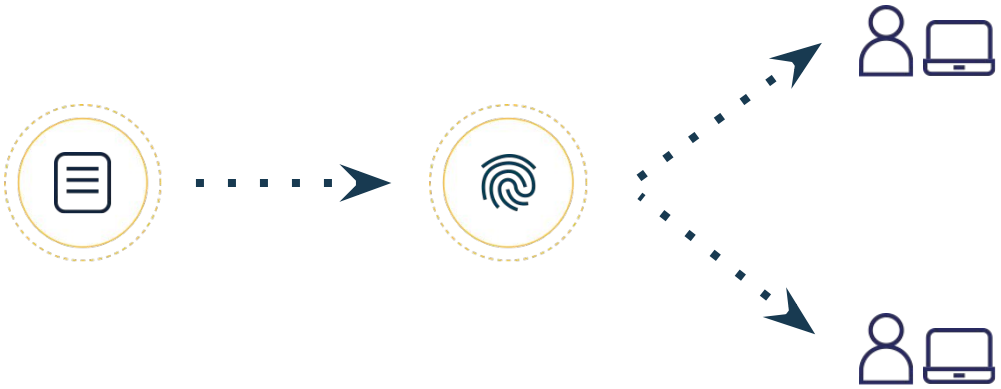
Why Immutability?:

INTEGRITY CHECKING



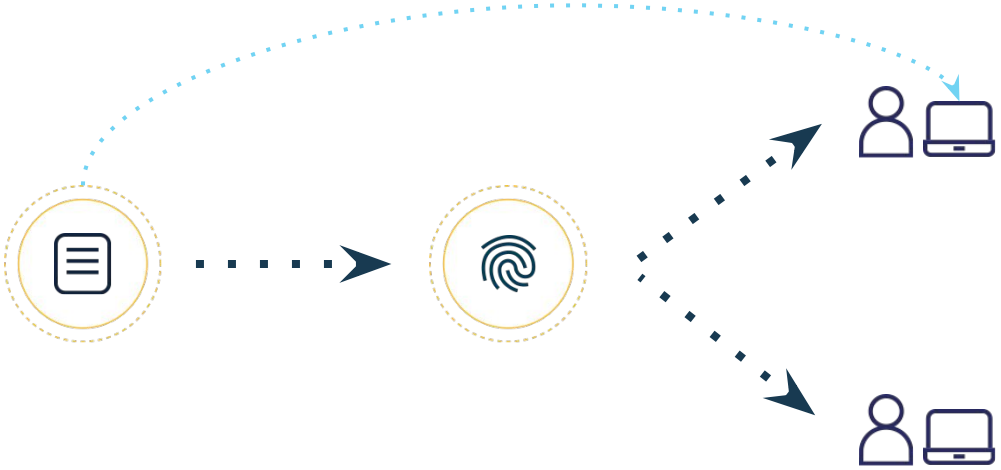
Why Immutability?:

INTEGRITY CHECKING



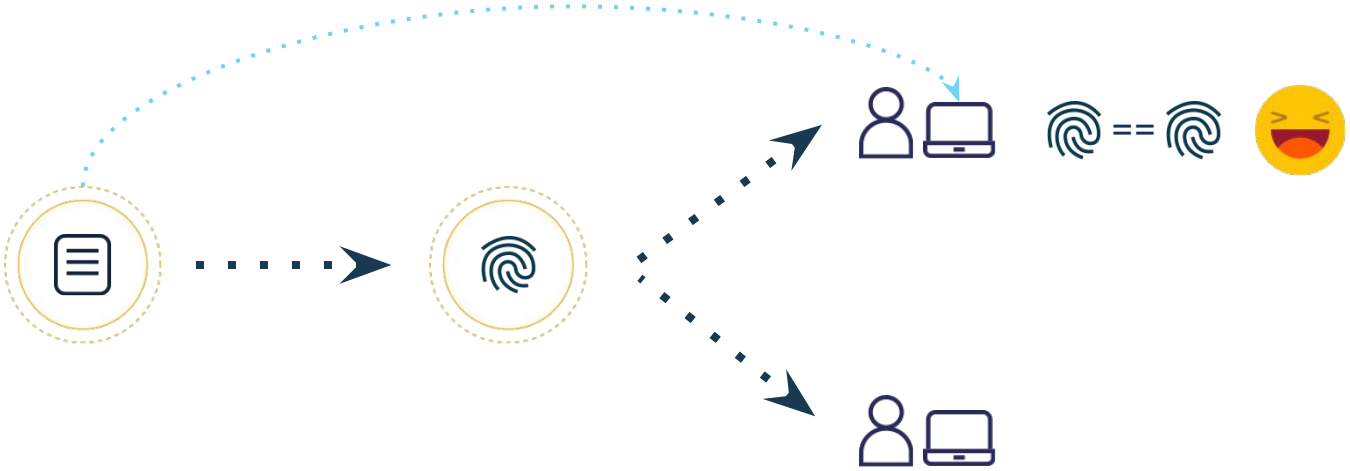
Why Immutability?:

INTEGRITY CHECKING



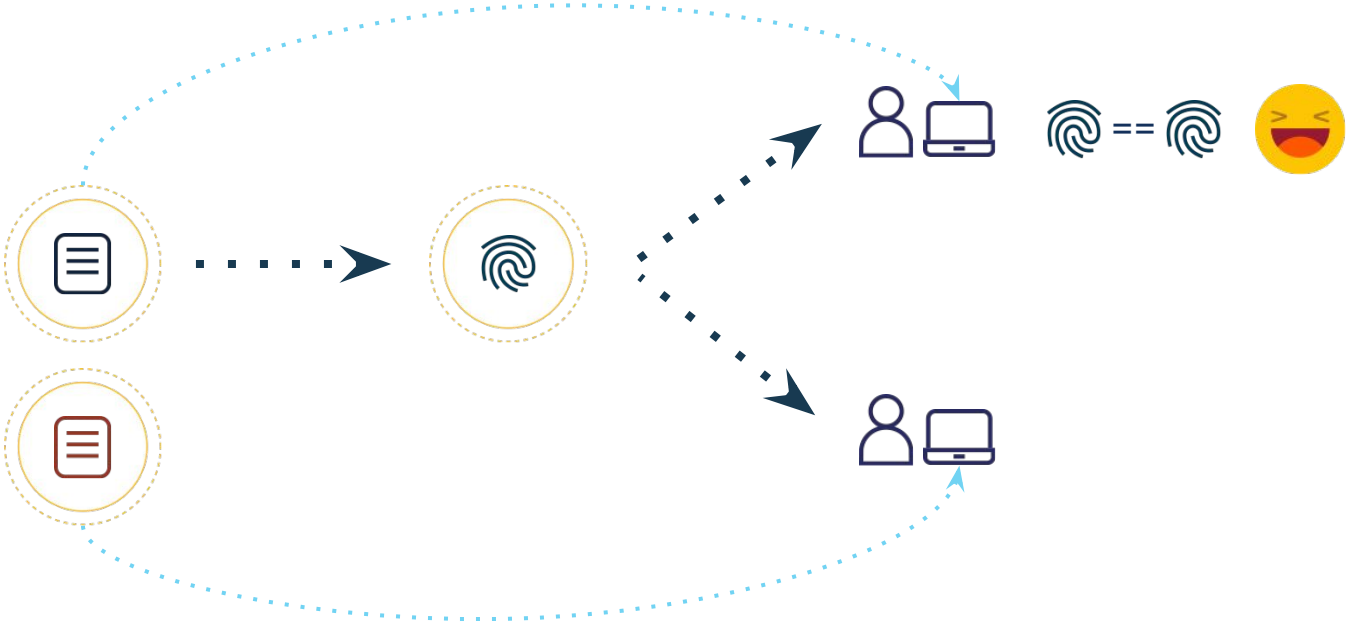
Why Immutability?:

INTEGRITY CHECKING



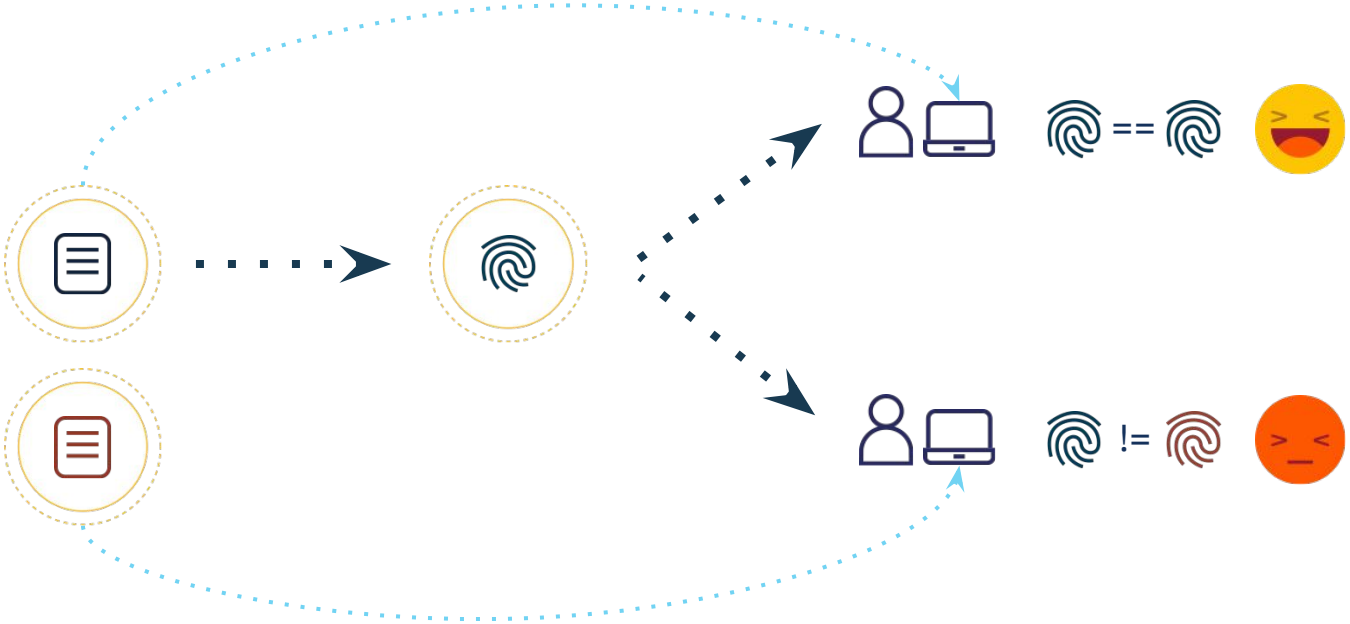
Why Immutability?:

INTEGRITY CHECKING



Why Immutability?:

INTEGRITY CHECKING



Why Immutability?:

IMMUTABLE CONTENT

1. Verifiability

Why Immutability?:

VERIFIABILITY

abc.com/poodle.jpg

24h

later

abc.com/poodle.jpg

Why Immutability?:

VERIFIABILITY

Location Addressing

abc.com/poodle.jpg

VS

Content Addressing

Why Immutability?:

IMMUTABLE CONTENT

1. Verifiability
2. Caching & Deduping

Why Immutability?:

CACHING & DEDUPING

24h

later

Why Immutability?:

CACHING & DEDUPING

VS

Why Immutability?:

IMMUTABLE CONTENT

1. Verifiability
2. Caching & Deduping
3. Fetch from anyone

Why Immutability?:

FETCH FROM ANYONE

abc.com/poodle.jpg

xyz.com/poodle.jpg

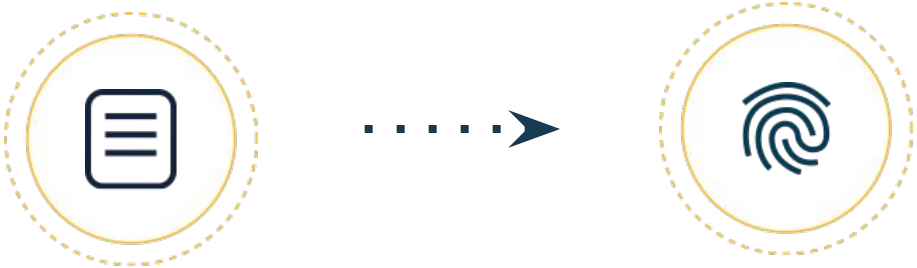
IPFS Camp

Core Course A:

ANATOMY OF A CID

Anatomy of a CID:

CRYPTOGRAPHIC HASH



110010010 (x n)

10100111 (8)

Anatomy of a CID:

HASHING ALGORITHMS

sha1
sha2-256
sha3-256
sha3-512
shake-256
keccak-512
blake2b-160
...

Anatomy of a CID:

HASHING ALGORITHMS

sha1
sha2-256
sha3-256
sha3-512
shake-256
keccak-512
blake2b-160
...

Anatomy of a CID:

WHICH ALGORITHM?

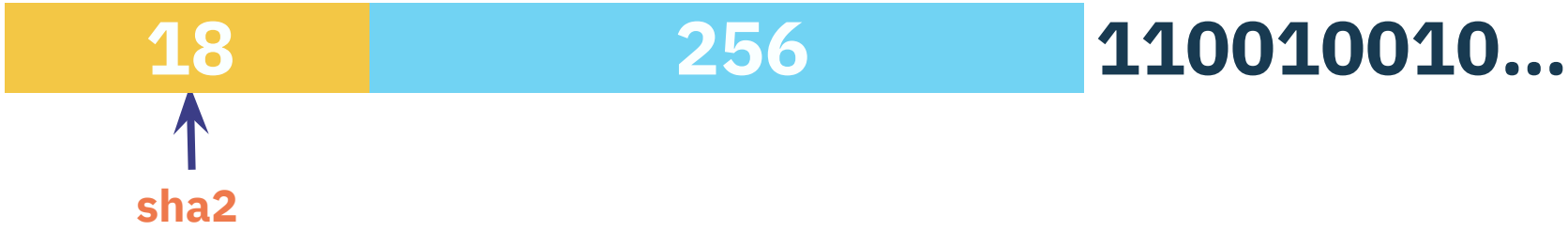
110010010... = ?

Anatomy of a CID:

MULTIHASH

Anatomy of a CID:

MULTIHASH



Anatomy of a CID:

MULTIHASH

00010010 10000000 00000010 110010010...

sha2 128 2

Anatomy of a CID:

HOW TO INTERPRET THE DATA?

CBOR?
PROTOBUF?
JSON?
...

Anatomy of a CID:

IPLD FORMAT

01110000 00010010 10000000 00000010 110010

dag-pb

sha2

128

2

Anatomy of a CID:

CID VERSION 😪

00000001 01110000 00010010 10000000 00000000

Version 1 | dag-pb | sha2 | 128 | 2

Anatomy of a CID:

CID STRING

QmbWqxBEKC3P8tqsKc98xmWNzrzDtRLMiMPL8wBuTGsMnR

bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi

Anatomy of a CID:

MULTIBASE

bafybeigdyrzt5...

Anatomy of a CID:

ALL TOGETHER!

Binary:

<cid-version><ipld-format><multihash>

String:

<base>**base(**<cid-version><ipld-format><multihash>**)**

Demo:

CID EXPLORER
cid.ipfs.io

Anatomy of a CID:

VERSION 0

QmbWqxBEKC3P...

Anatomy of a CID:

ALL TOGETHER! (v0)

Binary:

<multihash>

String:

base58btc(<multihash>)

Anatomy of a CID:

ALL TOGETHER! (v0)

<0><dag-pb><multihash>

<z><0><dag-pb>**base58btc**(<multihash>)

Core Course A:

MERKLE DAG

What is a DAG?

“Directed Acyclic Graph” is a special type of merkle tree where multiple nodes can point at the same child and circular references are impossible.

Merkle DAG:

Creating a DAG.

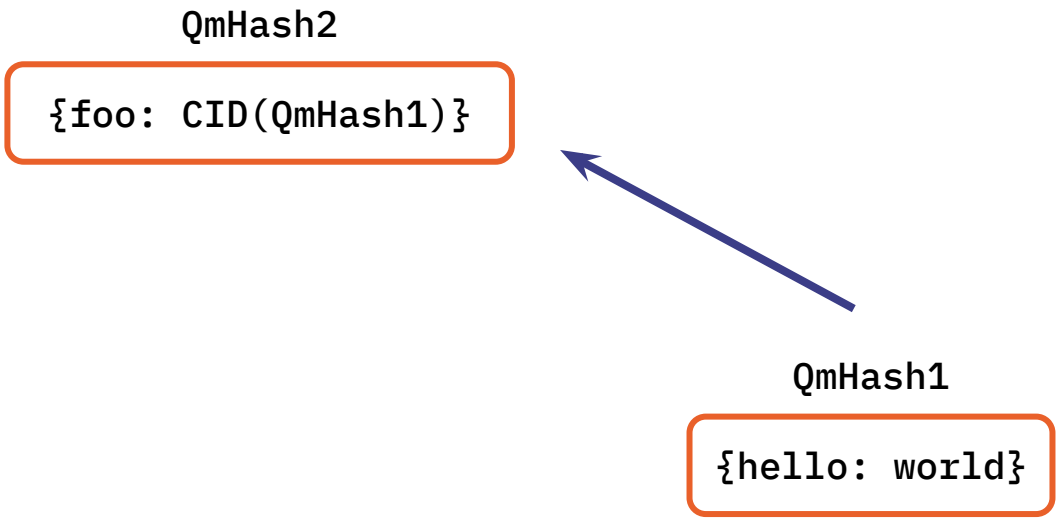
QmHash1

```
{hello: world}
```

Actual DAG structure may differ

Merkle DAG:

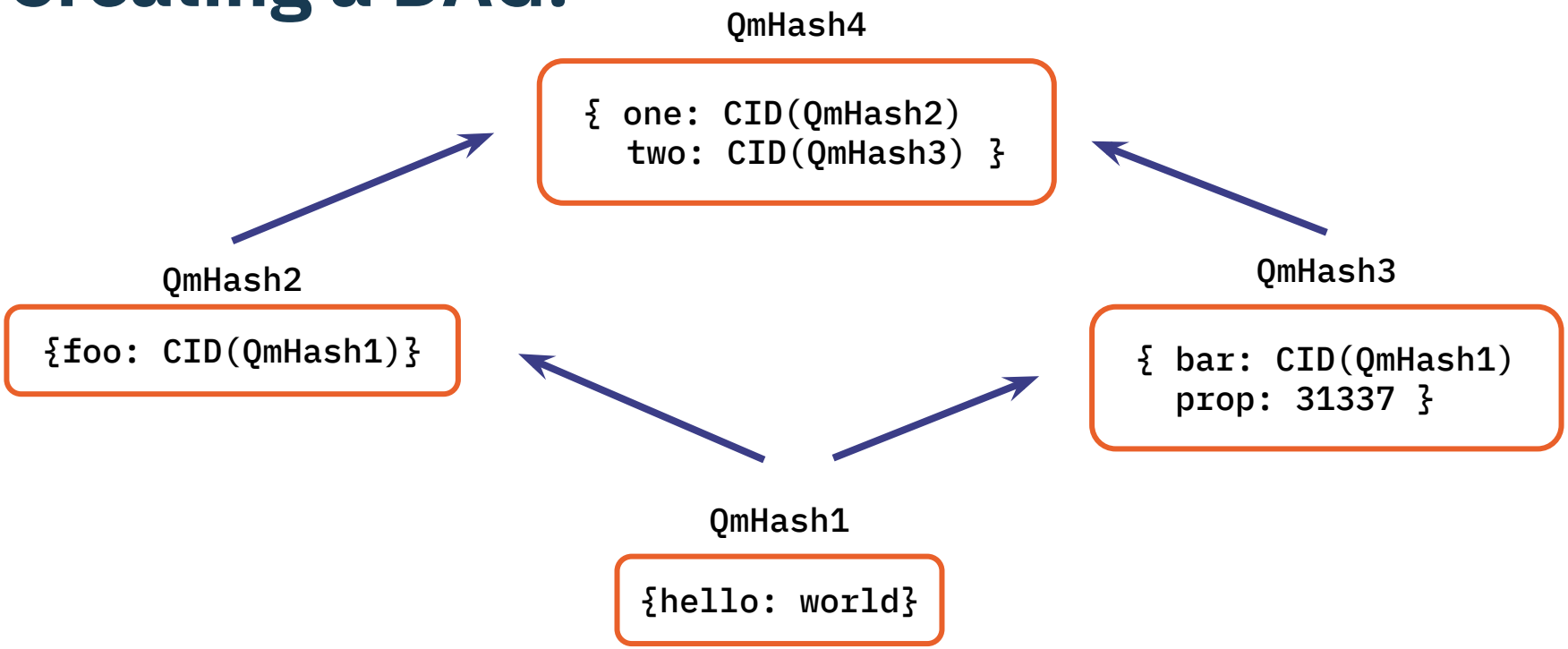
Creating a DAG.



Actual DAG structure may differ

Merkle DAG:

Creating a DAG.



Actual DAG structure may differ

Merkle DAG:
Reading a DAG.

Blocks

```
{ one: CID(QmHash2)
  two: CID(QmHash3) }
```

QmHash4

```
{ bar: CID(QmHash1)
  prop: 31337 }
```

QmHash3

```
{foo: CID(QmHash1)}
```

QmHash2

```
{hello: world}
```

QmHash1

Data Structure

```
{ one:
  { foo:
    { hello: world }
  },
  two:
  { bar: { hello: world },
    prop: 31337
  }
}
```

Actual DAG structure may differ

Merkle DAG:
Reading a DAG.

Blocks

```
{ one: CID(QmHash2)
  two: CID(QmHash3) }
```

QmHash4

```
{ bar: CID(QmHash1)
  prop: 31337 }
```

QmHash3

```
{foo: CID(QmHash1)}
```

QmHash2

```
{hello: world}
```

QmHash1

Paths

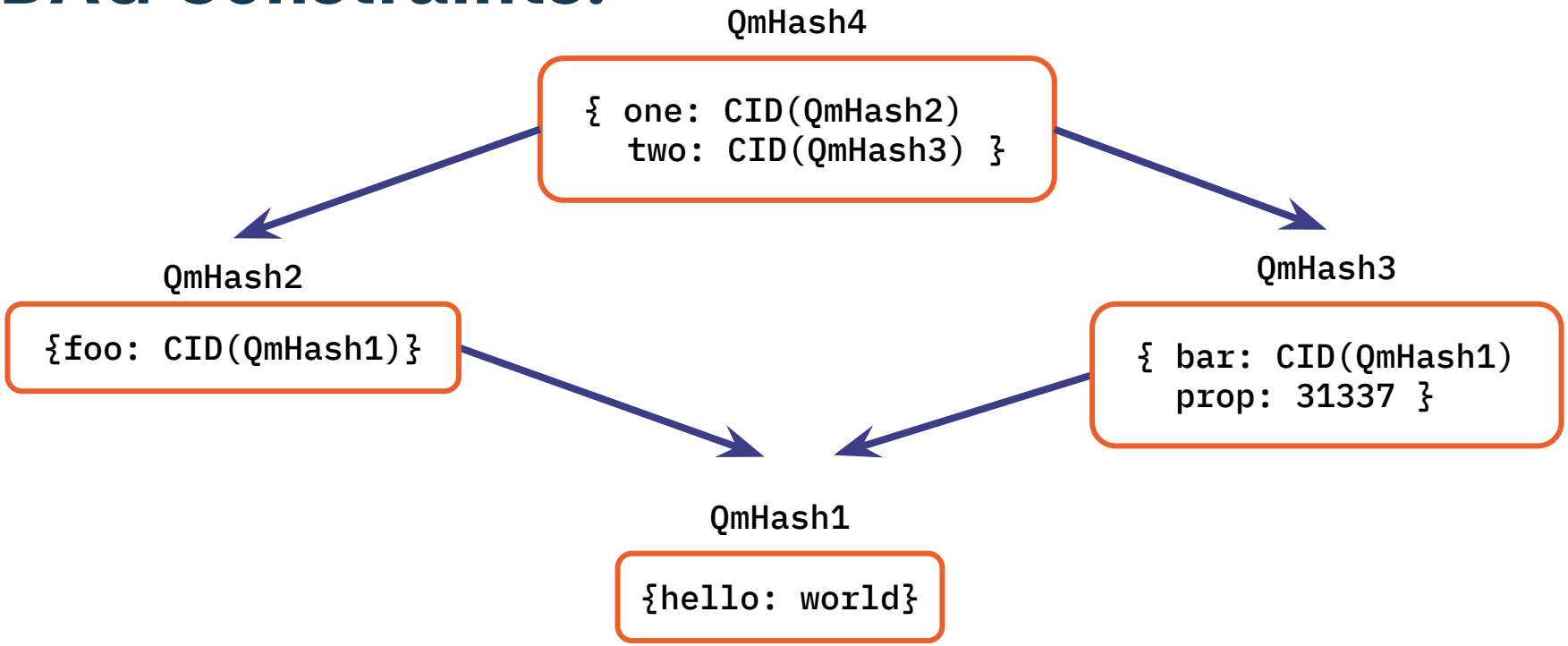
```
> QmHash4/one/foo/hello
“world”
> QmHash4/two/bar/hello
“world”
> QmHash4/two/prop
31337
```

```
> QmHash1/hello
“world”
> QmHash3/bar/hello
“world”
> QmHash3/prop
31337
```

Actual DAG structure may differ

Merkle DAG:

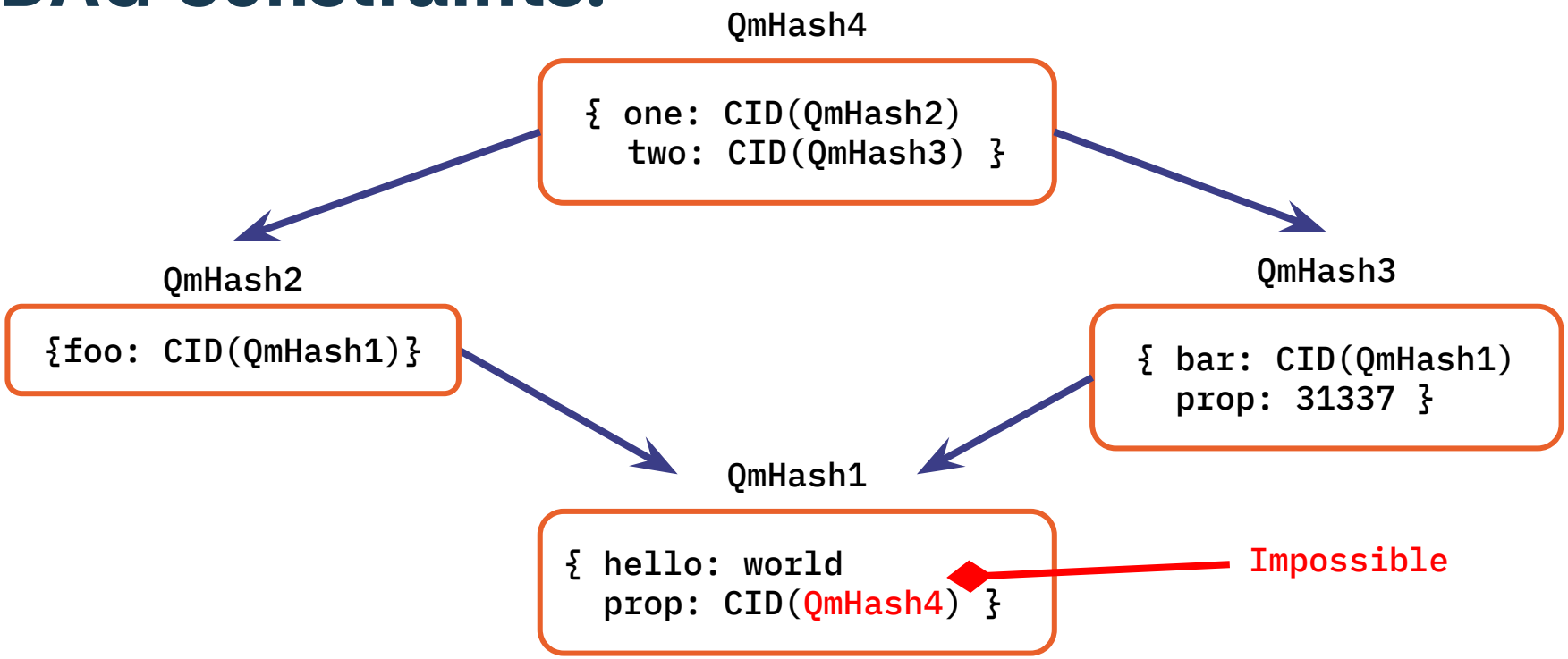
DAG Constraints.



Actual DAG structure may differ

Merkle DAG:

DAG Constraints.

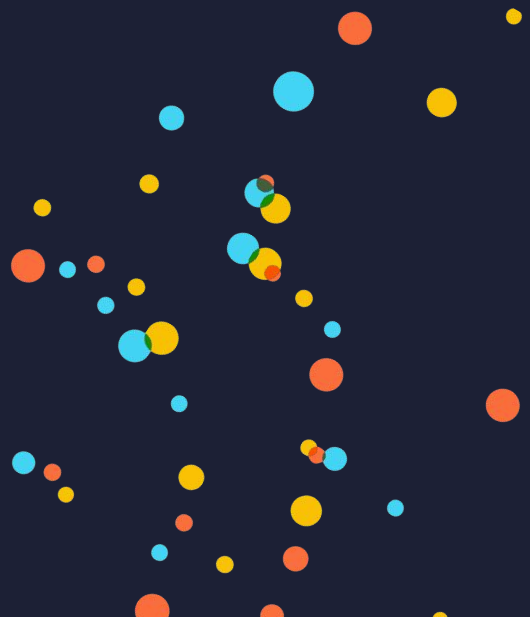


Actual DAG structure may differ

Merkle DAG:

DAG Constraints.

BLOCK SIZES



Merkle DAG:

Block size.

Too Big ☹️

*Cannot download from multiple peers.

*Larger orphaned blocks during mutation.

*Transport limitations.

*Less de-duplication.

Merkle DAG:

Block size.

Too Small ☹️

*More block requests.

*More encoding.

*More hashing.

*More “hops.”

Merkle DAG:

Block size.

Just Right

*It depends ;)

*Optimizing for writes or reads.

*Optimizing for initial creation or mutation.

*Transport performance.

TIME FOR A BREAK!

See you in 10 minutes!

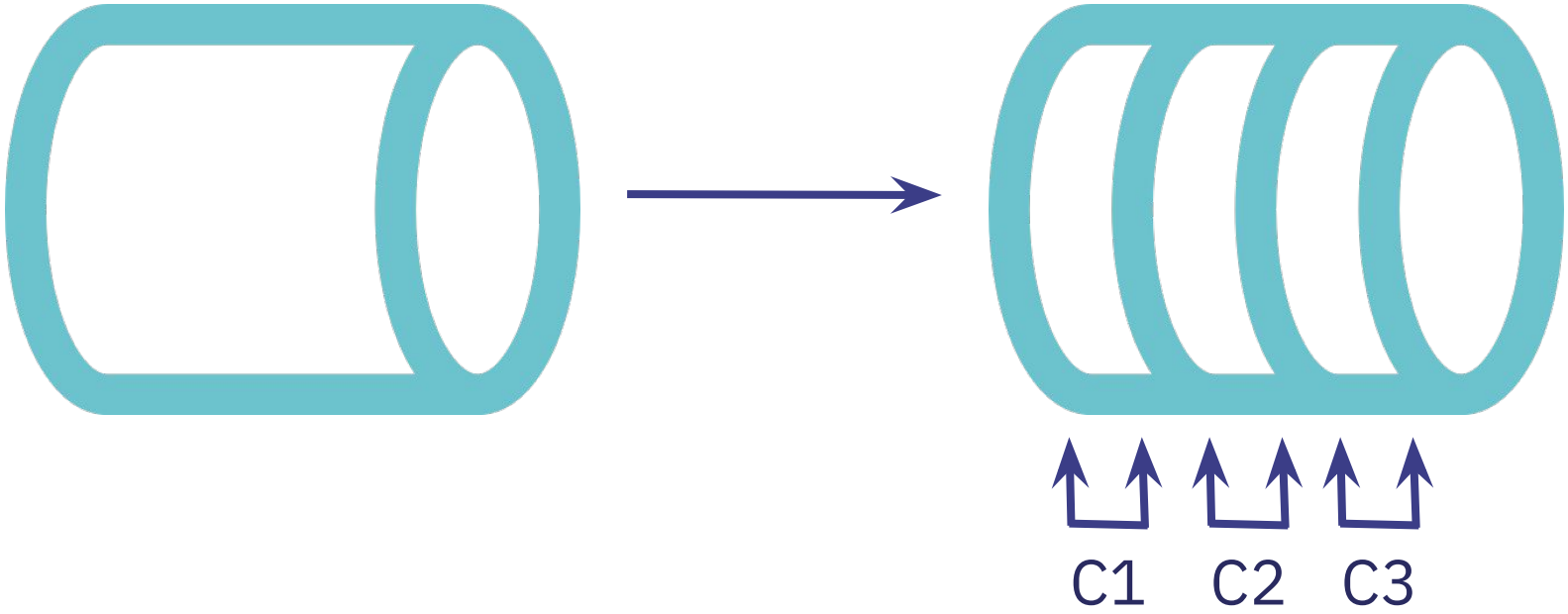
IPFS Camp

Core Course A:

IMPORTING FILES

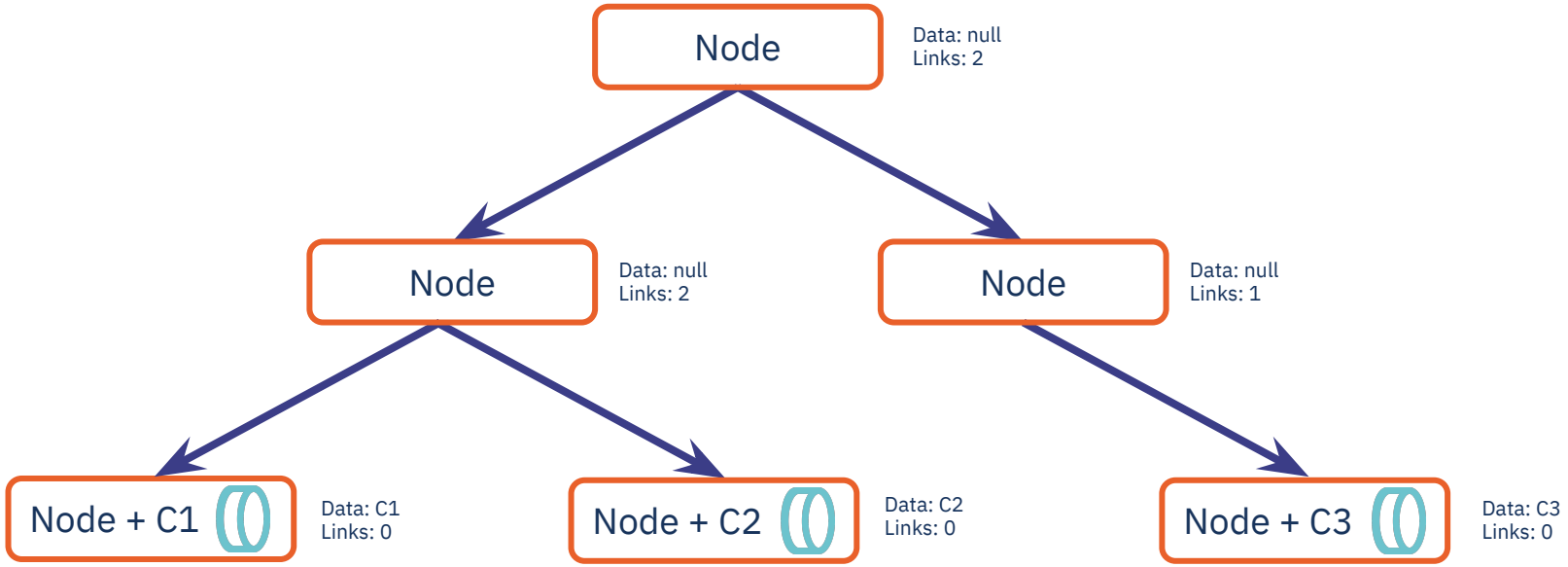
Importing files:

WHAT HAPPENS WHEN I USE IPFS ADD?



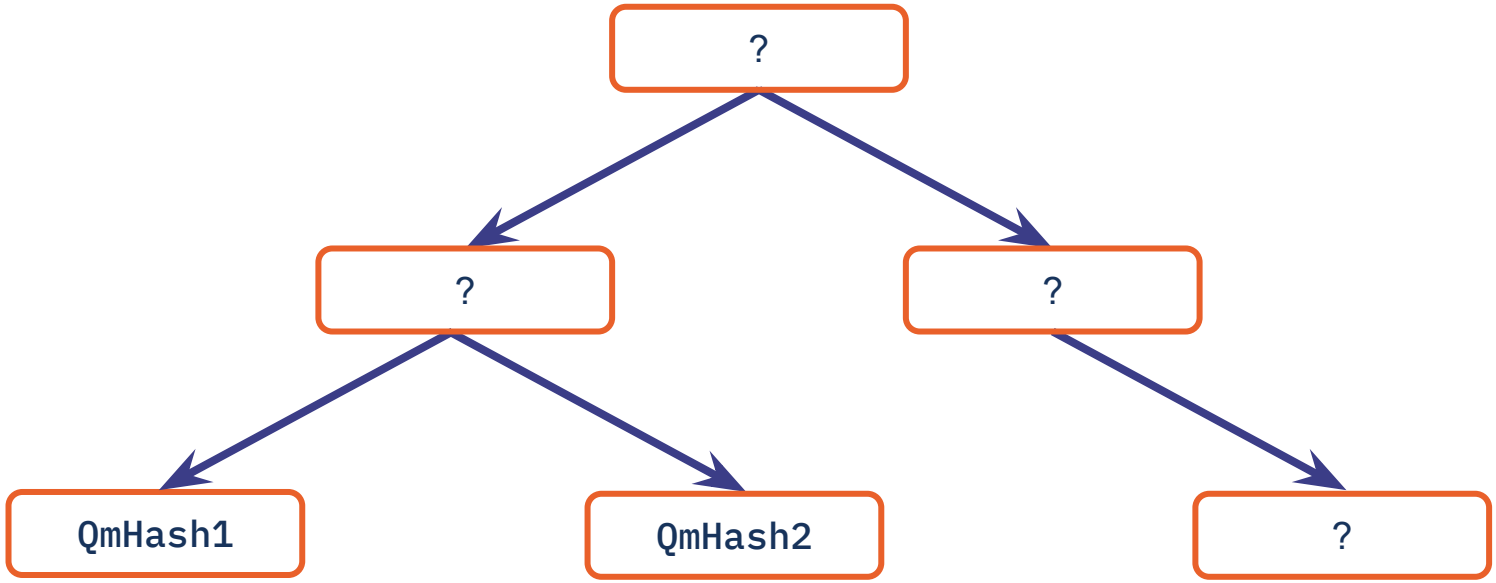
Database by Sebastian Belalcazar Lareo from the Noun Project

Importing files:
DAG



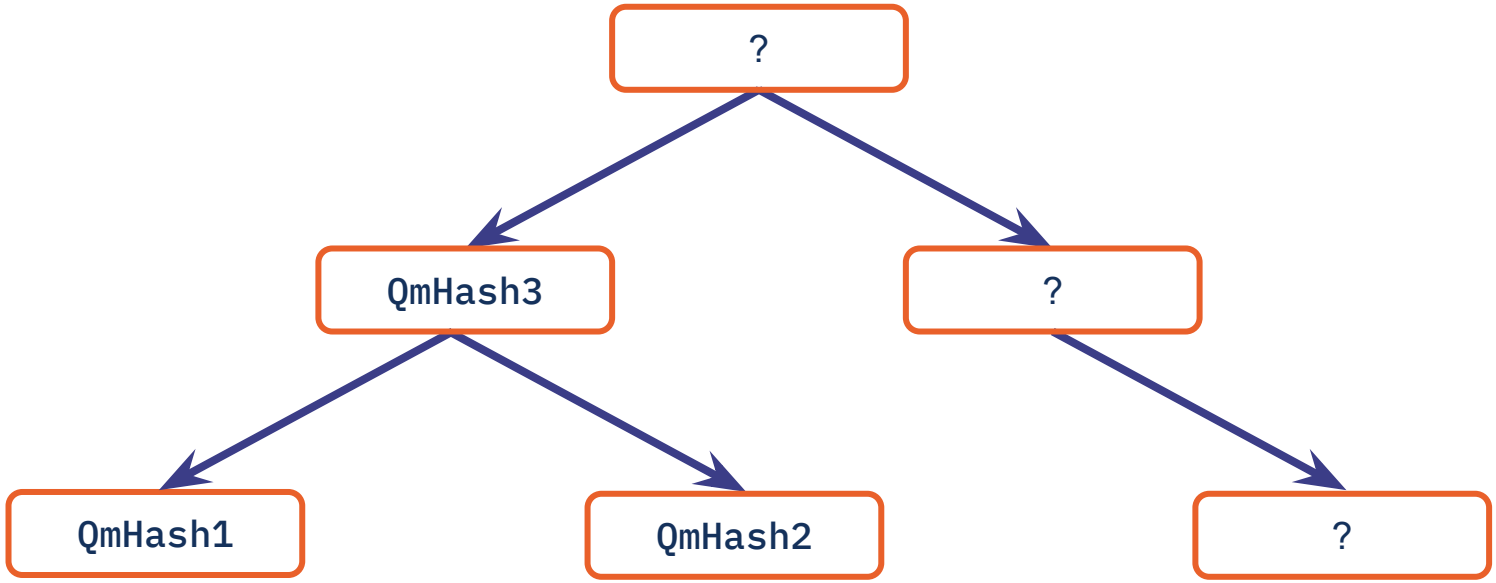
Actual DAG structure may differ

Importing files:
Calculating CIDs



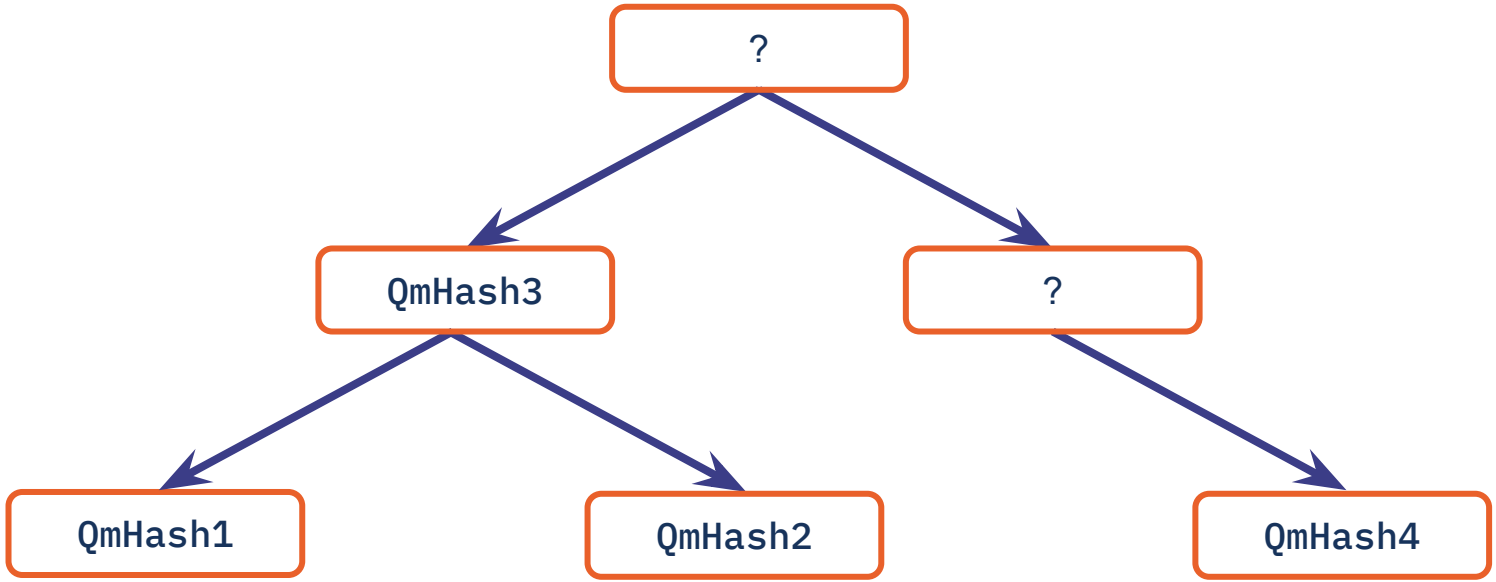
Actual DAG structure may differ

Importing files:
Calculating CIDs



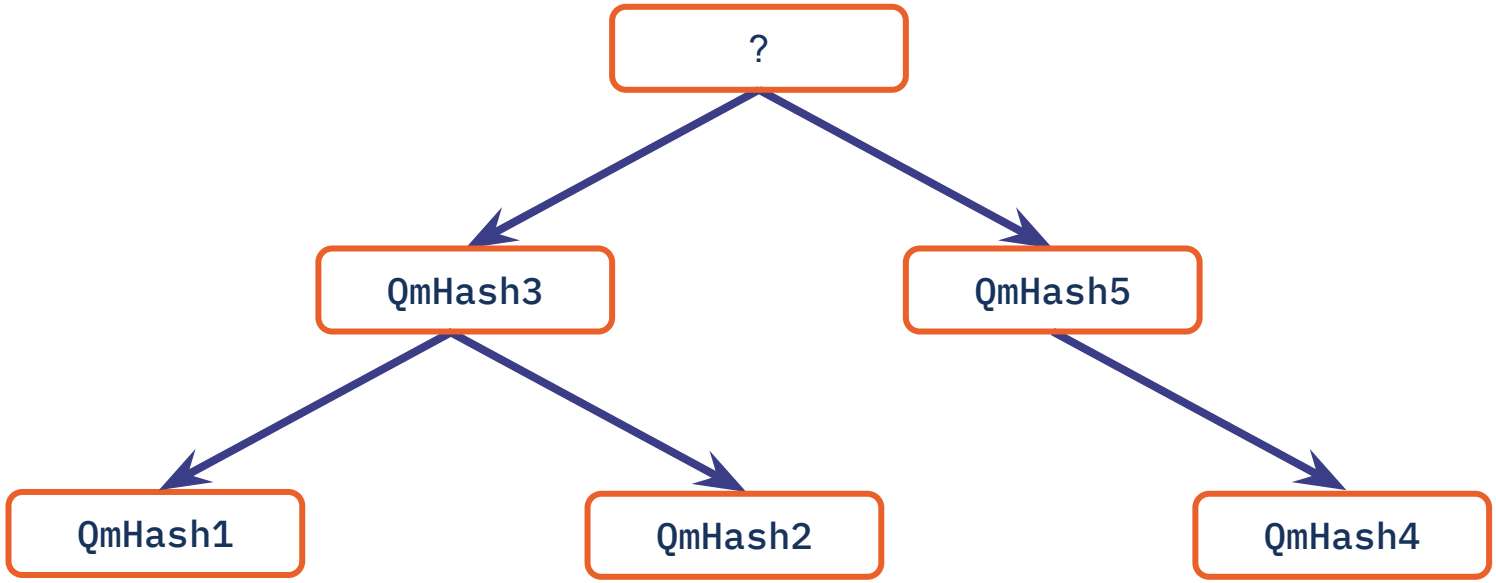
Actual DAG structure may differ

Importing files:
Calculating CIDs



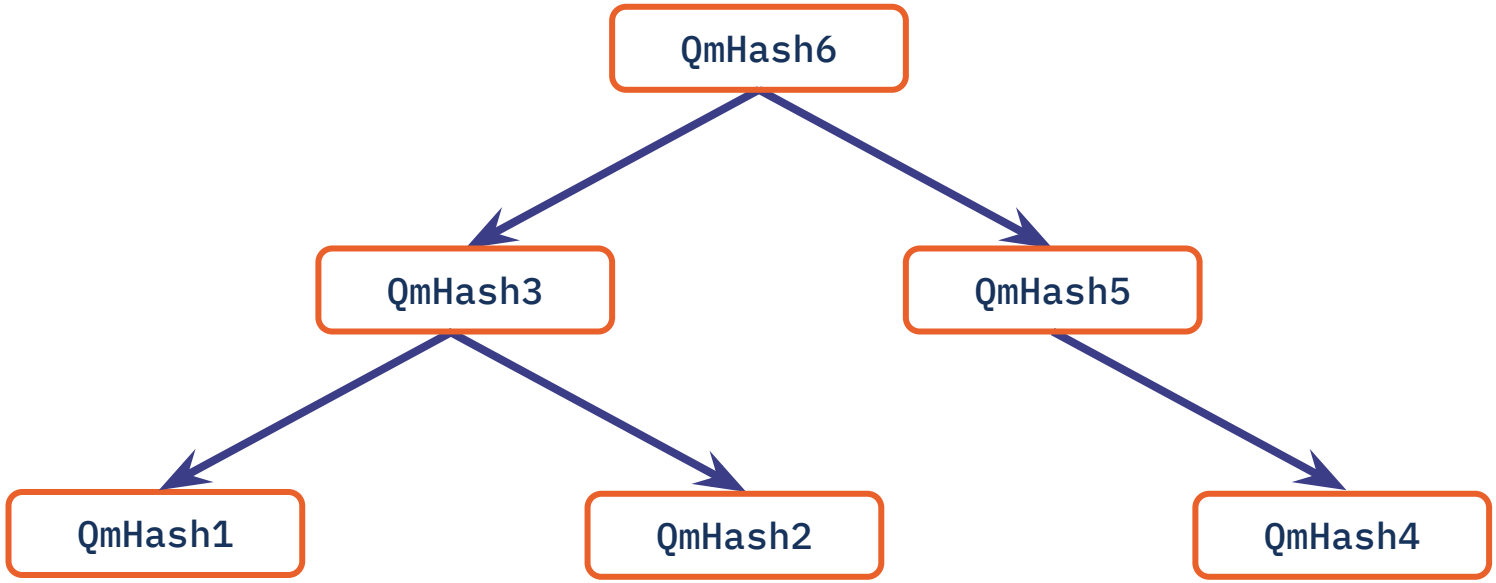
Actual DAG structure may differ

Importing files:
Calculating CIDs



Actual DAG structure may differ

Importing files:
Calculating CIDs



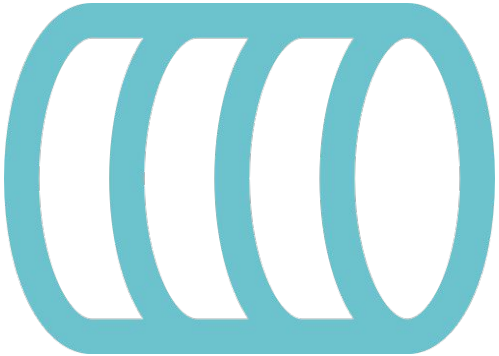
Actual DAG structure may differ

Demo:

BUILD A DAG
dag.ipfs.io

Importing files:

WHY VARY THE CHUNK SIZE?

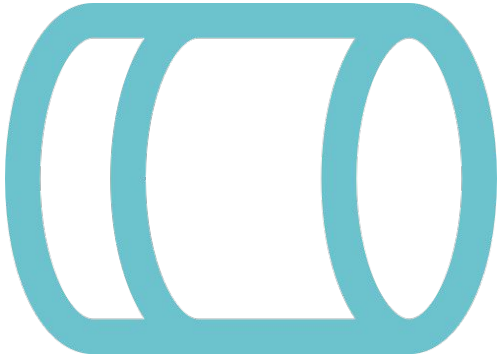


Large

vs

Small

vs



Smart

Database by Sebastian Belalcazar Lareo from the Noun Project

Demo:

DEDUPLICATION
dag.ipfs.io

Importing files:

UNIXFS

Node + UnixFS File + C1

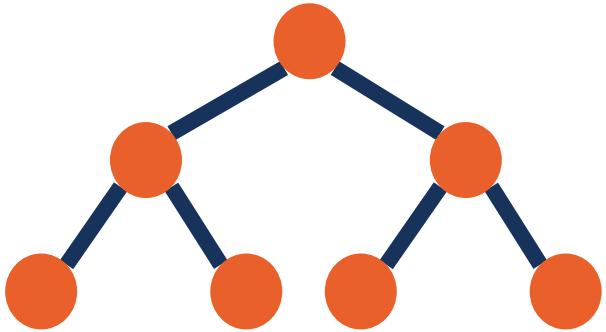
Data: UnixFSFile<C1>
Links: 0

Demo:

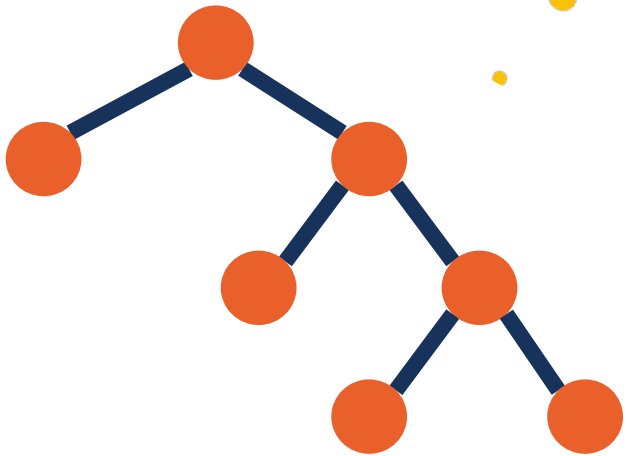
UNIXFS
dag.ipfs.io

Importing files:

DAG LAYOUTS



Balanced



Trickle

Actual DAG structure may differ

Demo:

DAG LAYOUTS
dag.ipfs.io

IPFS Camp

Core Course A:

MUTABLE FILE SYSTEM (MFS)

MFS:

WHAT IS IT?

/pics/pug-pony.jpg

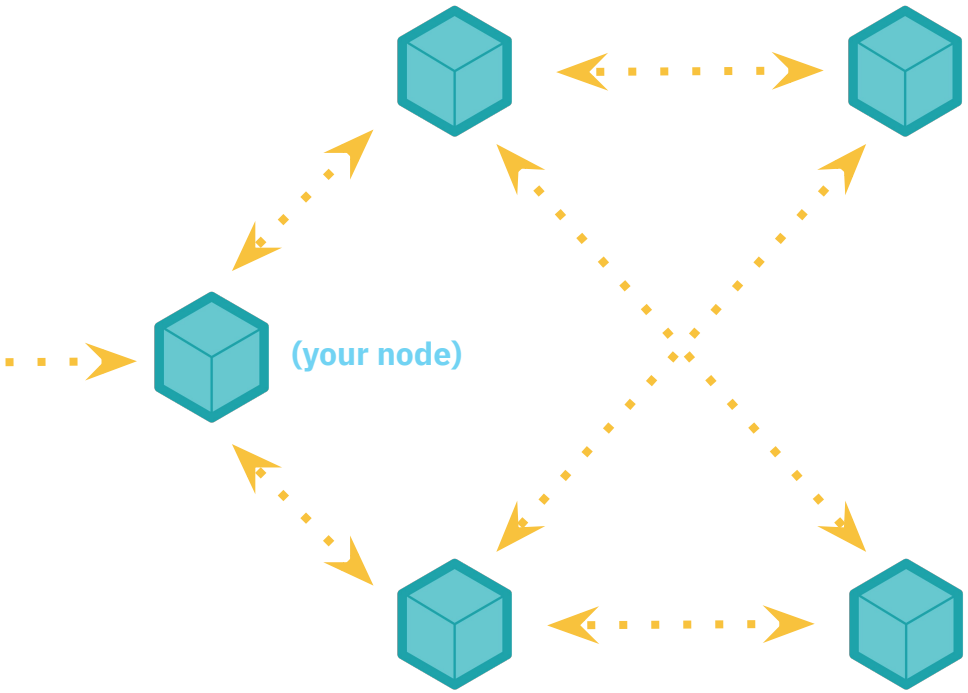
QmU8pC4jDQh2wZzFqM3WUZFw1t6Z2WbWXSogxMnurBFCtj

MFS:

PITFALLS

/ipfs/QmHash/pics/cat.gif

(an IPFS path)

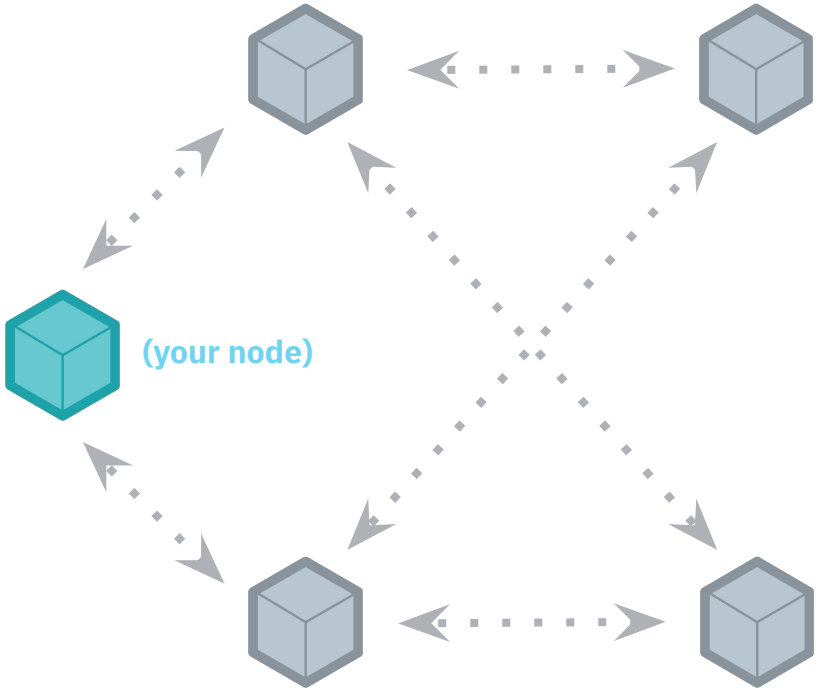


MFS:

PITFALLS

/path/to/business/doc.txt

(an MFS path)



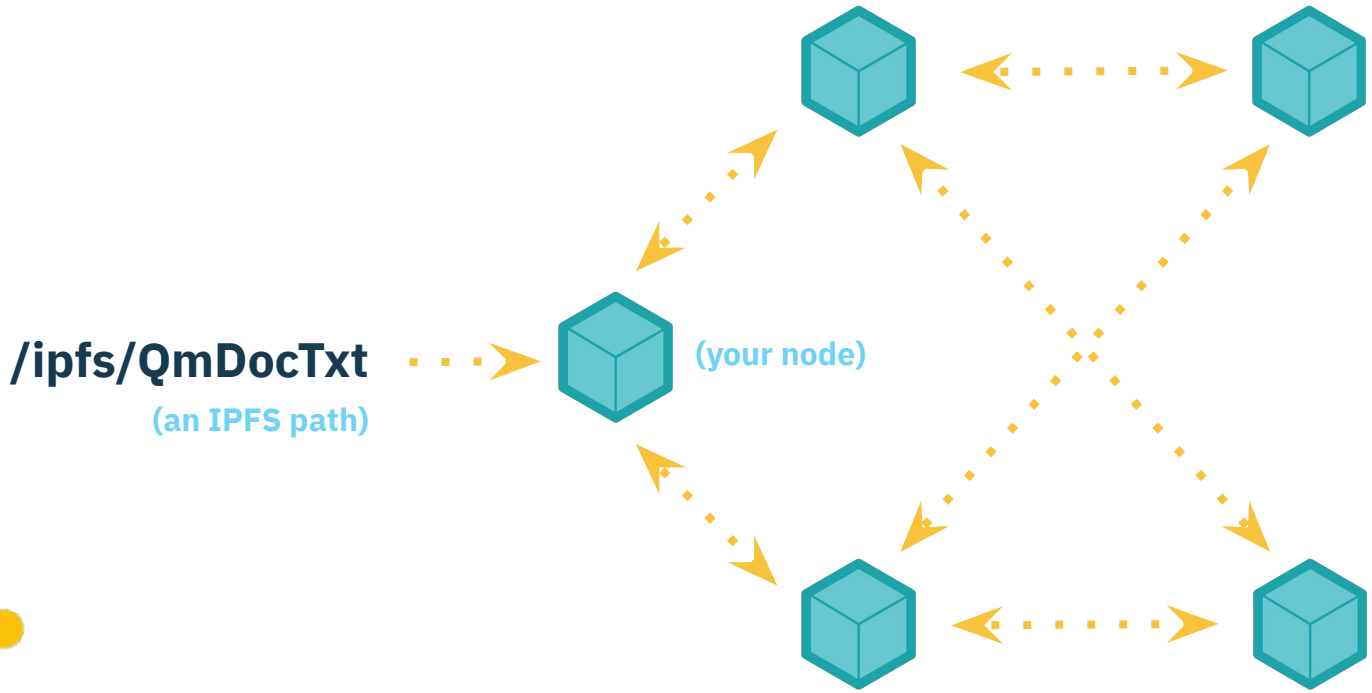
MFS:

PITFALLS

```
ipfs files stat /path/to/business/doc.txt
                ···➢ QmDocTxt
```

MFS:

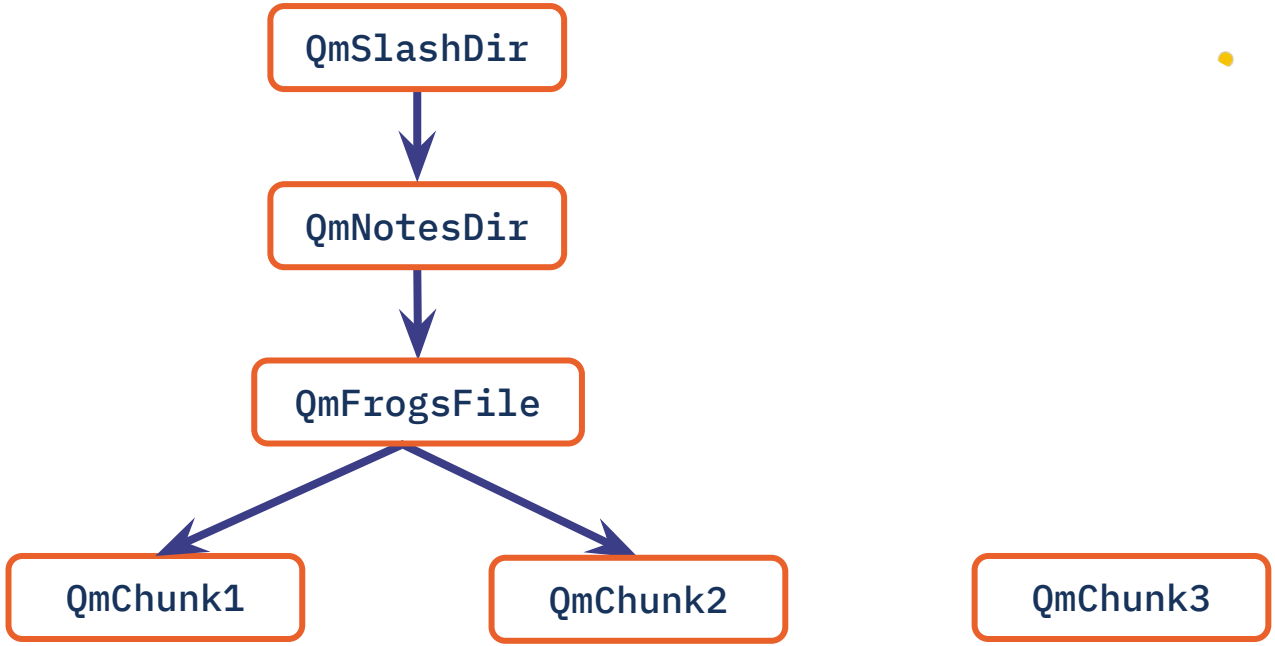
PITFALLS



MFS:

ADDING CONTENT

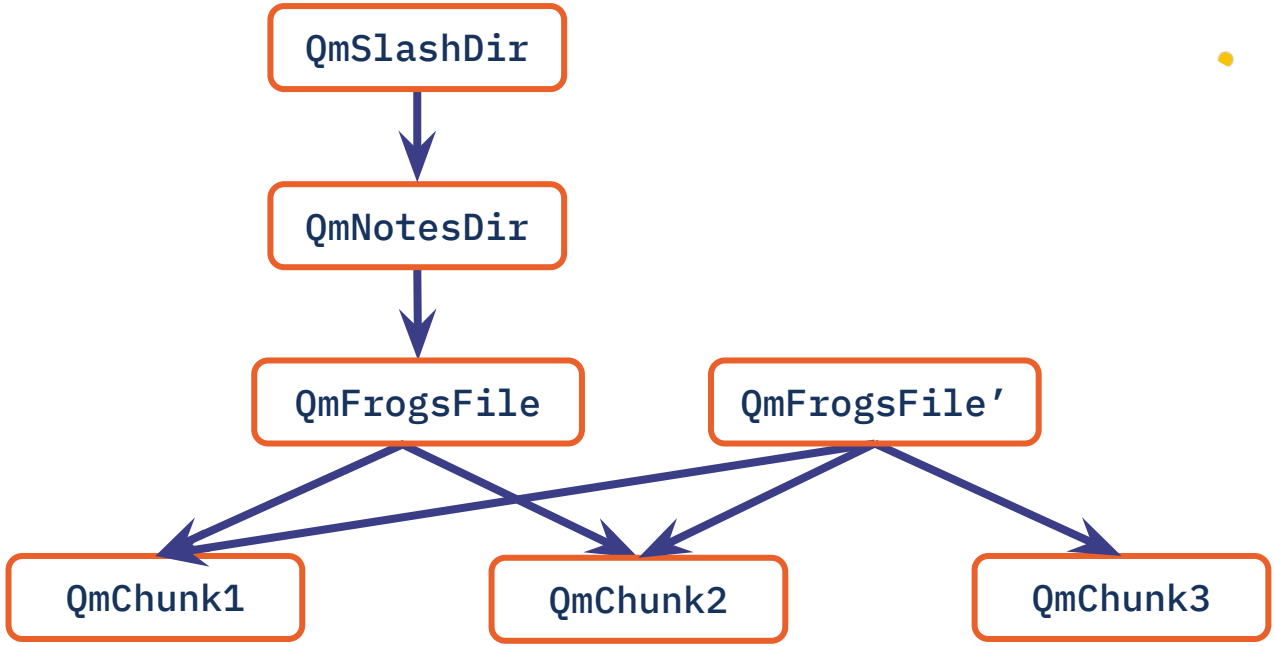
/notes/frogs.txt



MFS:

ADDING CONTENT

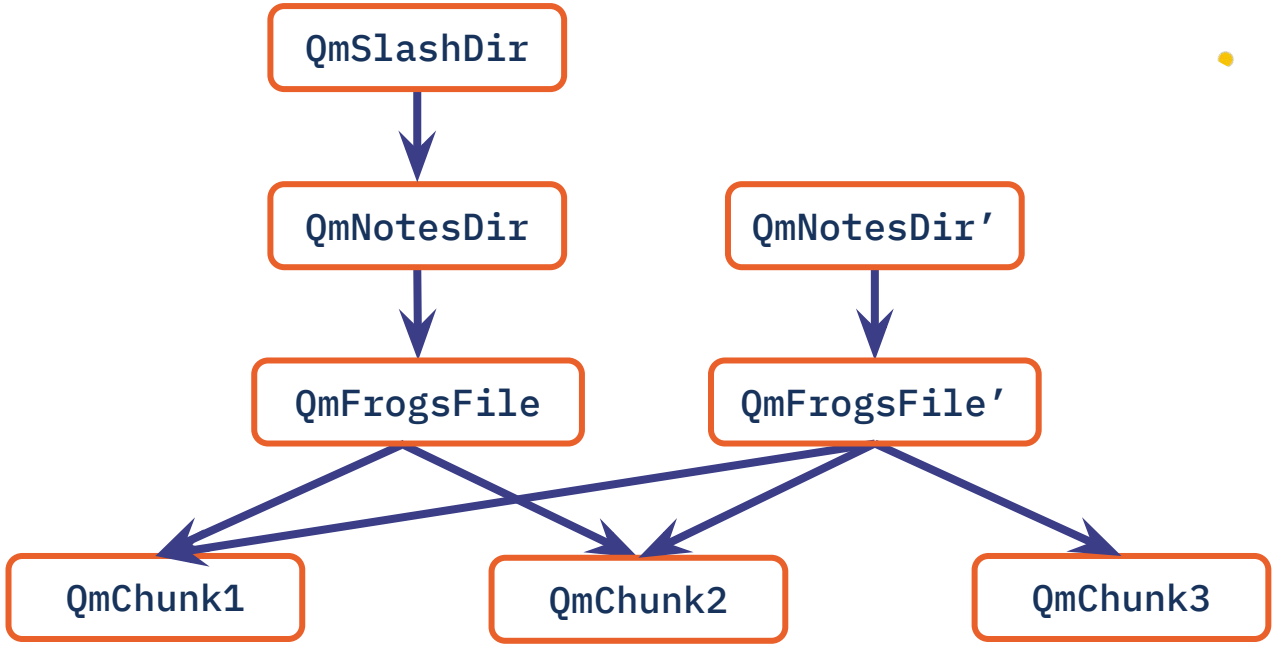
/notes/frogs.txt



MFS:

ADDING CONTENT

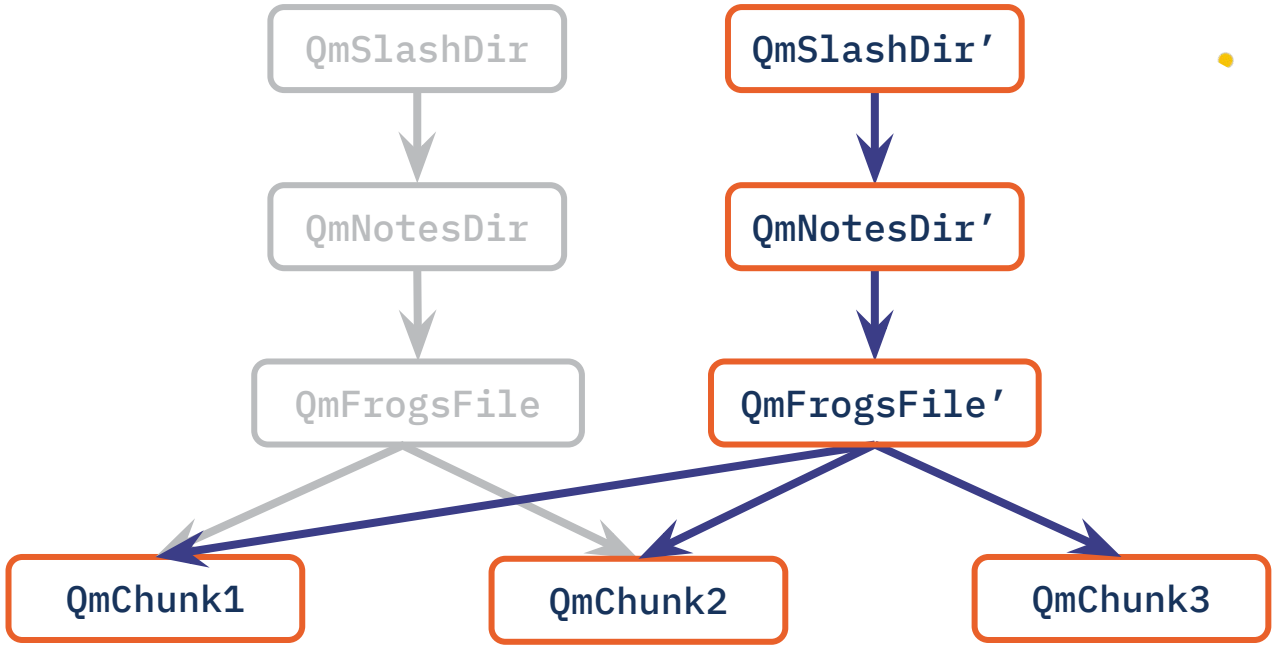
/notes/frogs.txt



MFS:

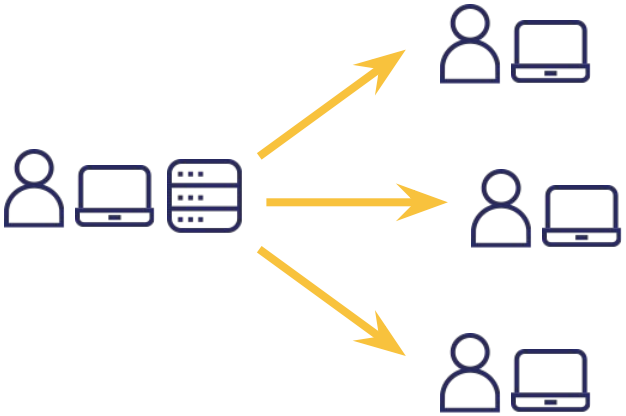
ADDING CONTENT

/notes/frogs.txt



MFS:

SHARING CONTENT



Exercise:

MFS TUTORIAL
proto.school

YOU HAVE COMPLETED

UNDERSTANDING HOW IPFS DEALS WITH FILES

CORE COURSE A

IPFS Camp

That's all, folks.

Thank you!