Tap!

近畿大学産業理工学部情報学科 2年
嶋村颯人 / 秋本隼勢

NFTとは

NFT = Non Fungible Token

Non Fungible Token

=

非代替性トークン

NFT

・代替不可能

・同価値を示すトークンは他に存在しない

・ブロックチェーン上で発行

・高い耐改竄性と追跡性を有する

・参加ノードによる検証が可能

・トークンが移転されると自身の手元からは完全に消え、相手に渡る

→コピーと仕組みは全く異なる

Tap! 概要

NFTを用いた
「作品管理プラットフォーム」

NFTを用いた「作品管理プラットフォーム」

NFTにより唯一性を保証、所有者情報を明確に保持

「作品に対しNFTを発行し所有者情報・唯一性を保つ」

非Crypto的なNFTの活用

非Crypto的なNFTの活用

暗号通貨とNFTの切り離し

非Crypto的なNFTの活用

暗号通貨とNFTの切り離し

既存のNFTの発行には暗号通貨が必要であり参入ハードルが高まる原因に

あえて切り離すというNFTへの新しいアプローチを提案

ターゲットユーザー

「NFT気になるけどウォレットとかよく分からん！」

「NFT発行ってお金かかるの！？」

「暗号通貨...ハードル高いなぁ....」

ターゲットユーザー

・NFT発行のハードルの高さや参入の難しさ

・暗号通貨そのものへの抵抗意識

を課題とするユーザー

ユーザーのメリット

・無料でNFTが発行できる

・暗号通貨ウォレットの用意が必要ない

・ETHなどの暗号通貨を事前に購入する必要がない

NFTに触れる機会を提供

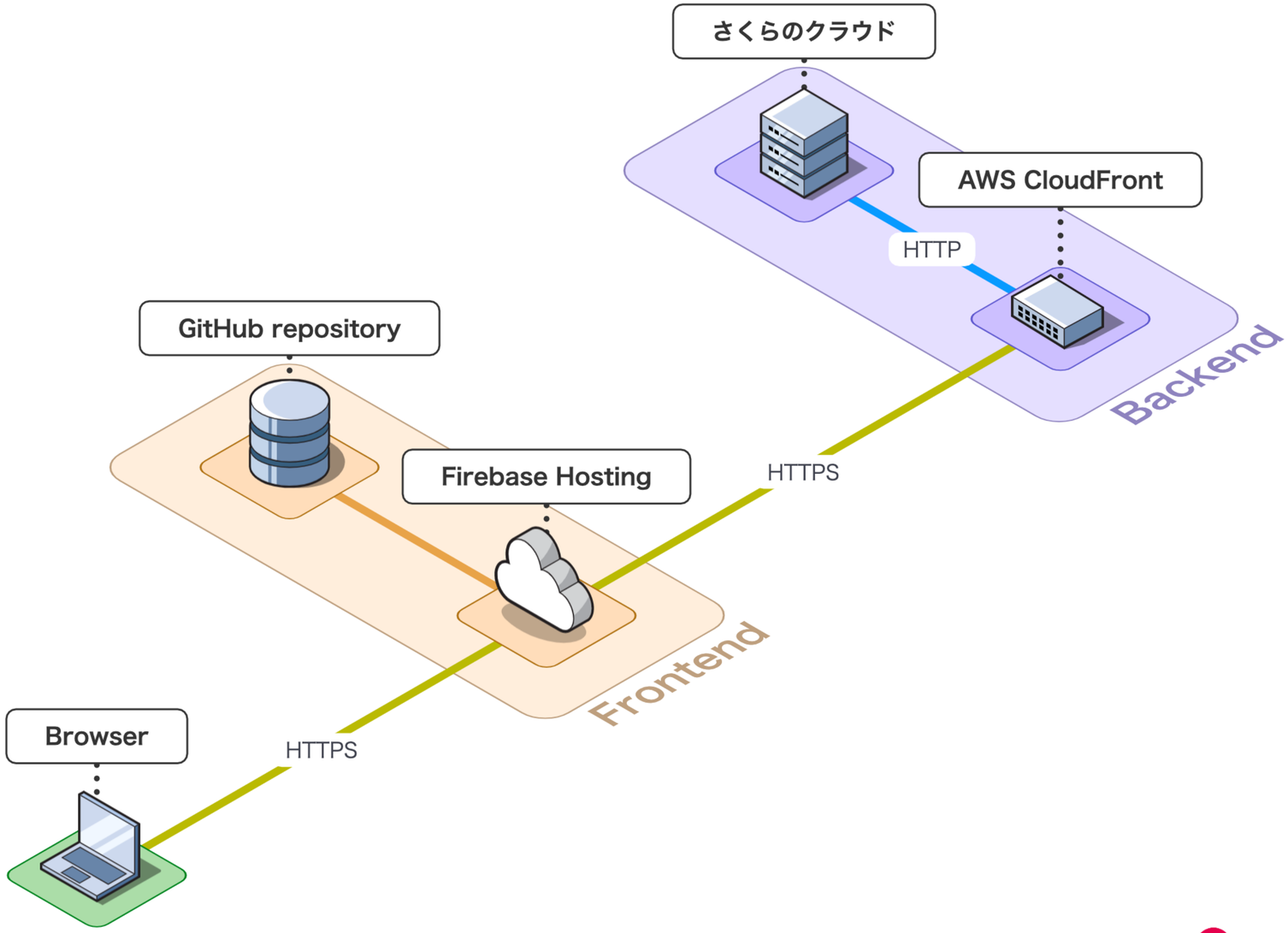
現状怪しすぎるNFT技術の核心について知ってほしい。

NFTアート・投機的側面以外の部分、NFT技術そのものについて

知り、体験してほしい。

システム構成

システム構成図


さくらのクラウド
AWS CloudFront
HTTP
Backend
GitHub repository
Firebase Hosting
HTTPS
Frontend
Browser
HTTPS
isoflow.io

利用技術

フロント

Firebase Hosting, Firebase Authentication

言語：React

担当：秋本

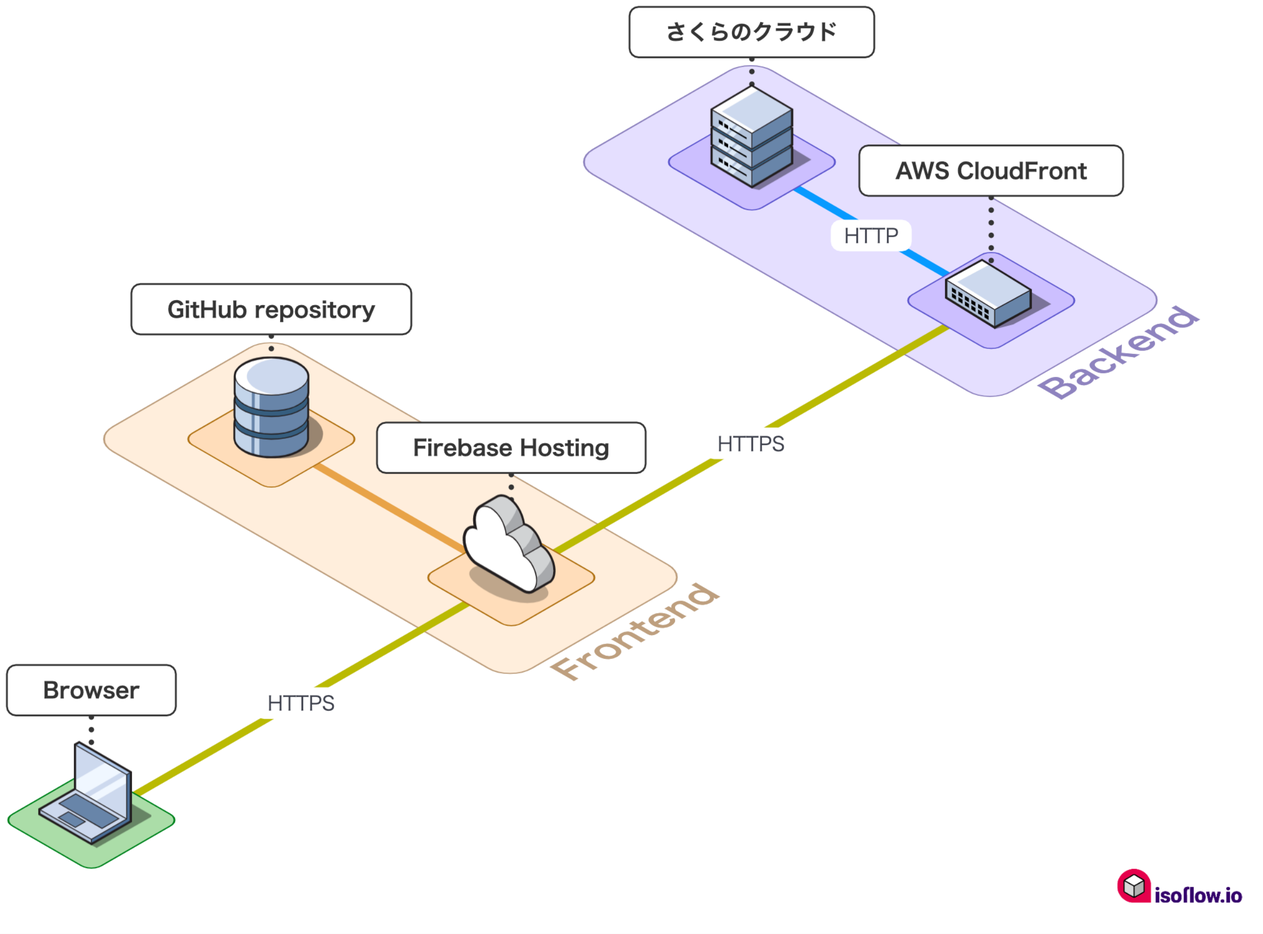
バック

CentOS, Docker, Nginx, Ruby on Rails,

Tapyrus, IPFS

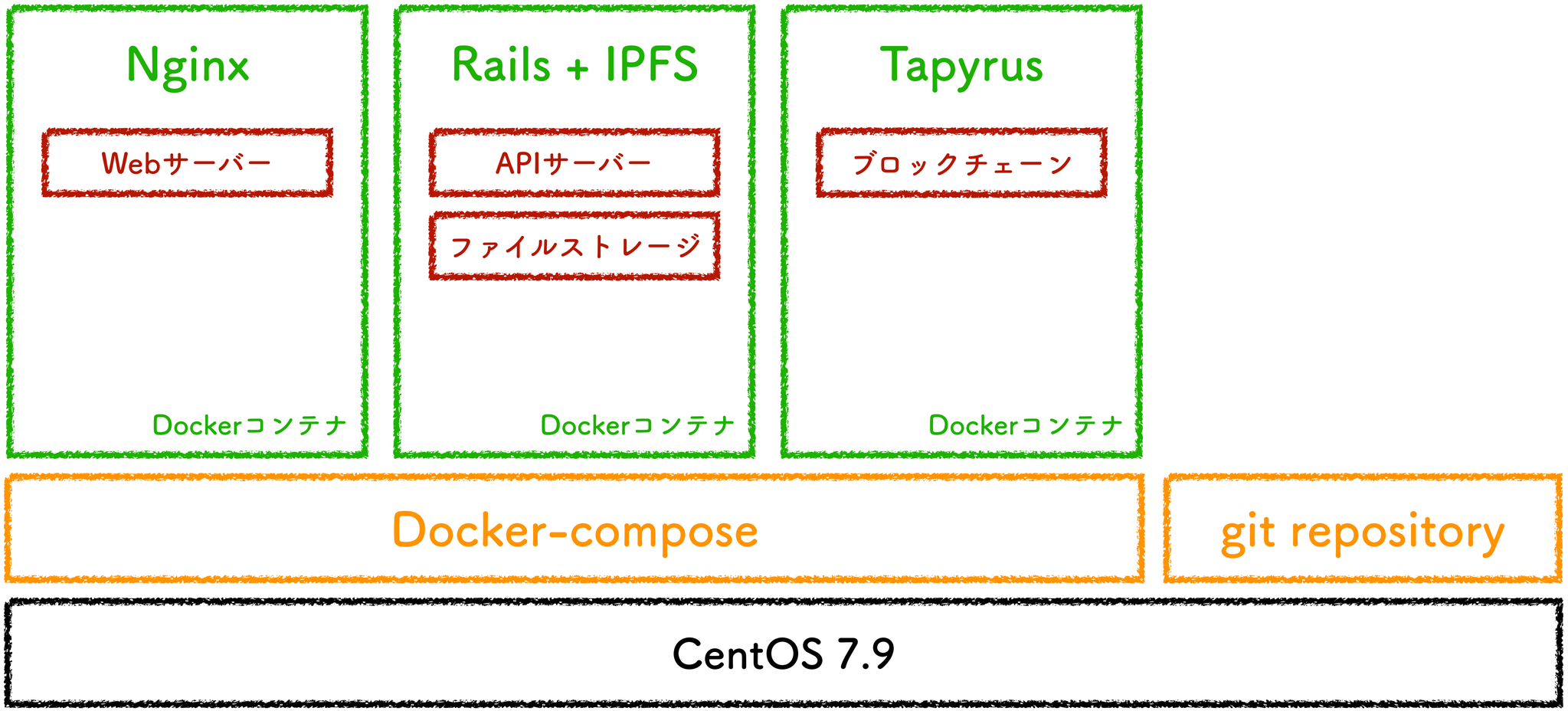
言語：Ruby

担当：嶋村



バックエンド構成

バックエンド構成


Nginx
Webサーバー
Dockerコンテナ
Rails + IPFS
APIサーバー
ファイルストレージ
Dockerコンテナ
Tapyrus
ブロックチェーン
Dockerコンテナ
Docker-compose
git repository
CentOS 7.9

Tapyursについて

Tapyrusについて

ブロックチェーンにはTapyrusを採用

・BitcoinやEthereumなどのパブリックチェーンでは発行費がかかる

・プライベートチェーンでは中央集権的でNFTの目的を果たせない

↓

その中間に位置するコンソーシアム型を採用し

プラットーフォームがブロックを発行できるTapyrusに注目

今回はネットワークの構築は行わず、devノードを利用

IPFSについて

IPFSについて

ファイルストレージにはIPFSを採用

・P2Pネットワーク上で動作するハイパーメディアプロトコル

・ファイルをIPFSネットワーク上にアップロードし分散型ストレージのように利用可能

・コンテンツ毎にユニークなCIDが発行される

Rails APIと同じコンテナ内でノードを構築し運用

当初の課題

当初

Firebase

画像データ

ブロックチェーン

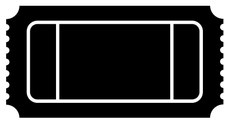
NFT

当初

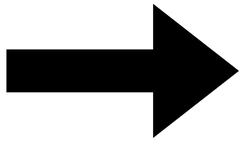
Firebase

画像データ

ブロックチェーン



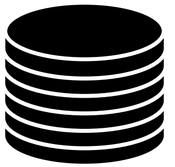
NFT



これらを
データベース上に保存

サーバー上

Database



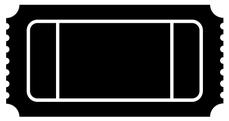
data	token_id
⋮	⋮
FirebaseのURI	NFTのID

当初

Firebase

画像データ

ブロックチェーン



NFT

画像データとNFTは
データベース上で「紐付け」しているだけ

data	token_id
⋮	⋮
FirebaseのURI	NFTのID

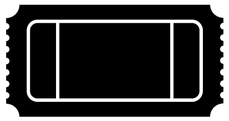
- サービスのデータベースに依存する形
- 「紐付けの改ざん」が可能

当初

Firebase

画像データ

ブロックチェーン



NFT

Firebase上の画像を差し替える
ことでも改ざん可能

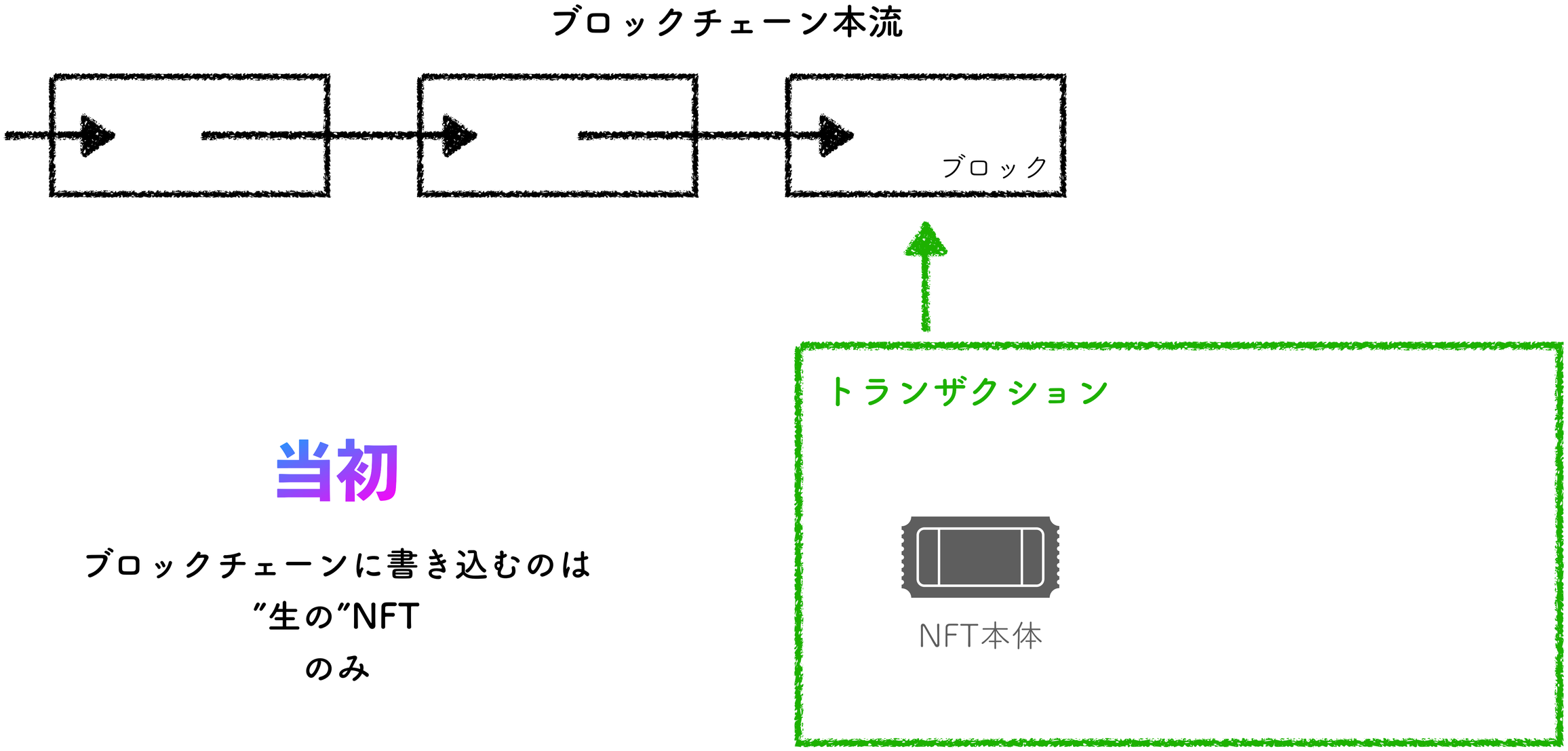
data	token_id
⋮	⋮
FirebaseのURI	NFTのID

フルオンチェーン化

フルオンチェーンとは

全部ブロックチェーンに
組み込んだろうぜ

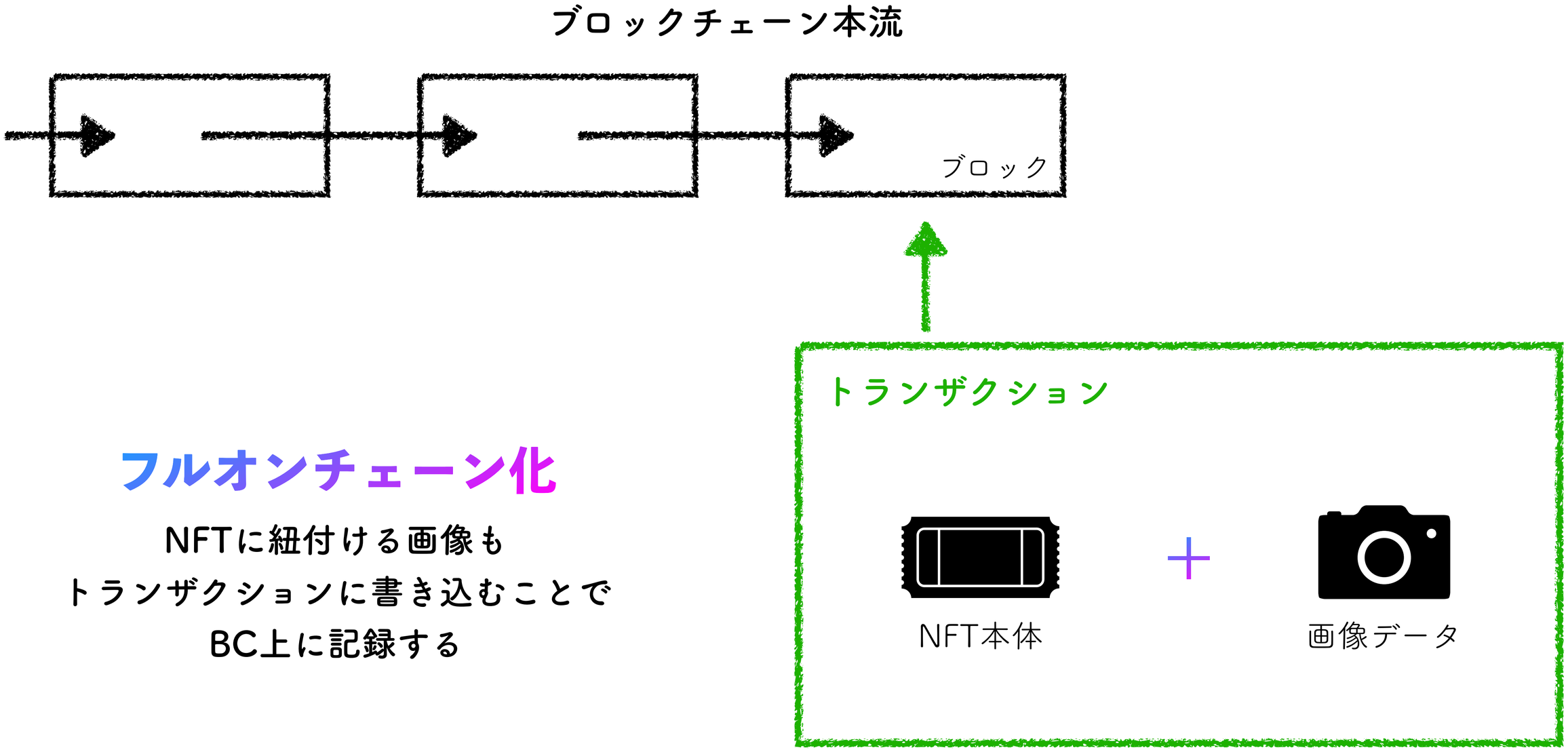
フルオンチェーンとは



当初

ブロックチェーンに書き込むのは
”生の”NFT
のみ

フルオンチェーンとは



フルオンチェーン化

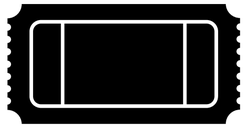
NFTに紐付ける画像も
トランザクションに書き込むことで
BC上に記録する

実装

実装

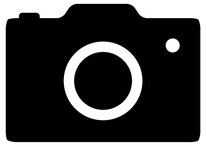
トランザクションサイズの制限

トランザクション



NFT本体

＋



画像データ

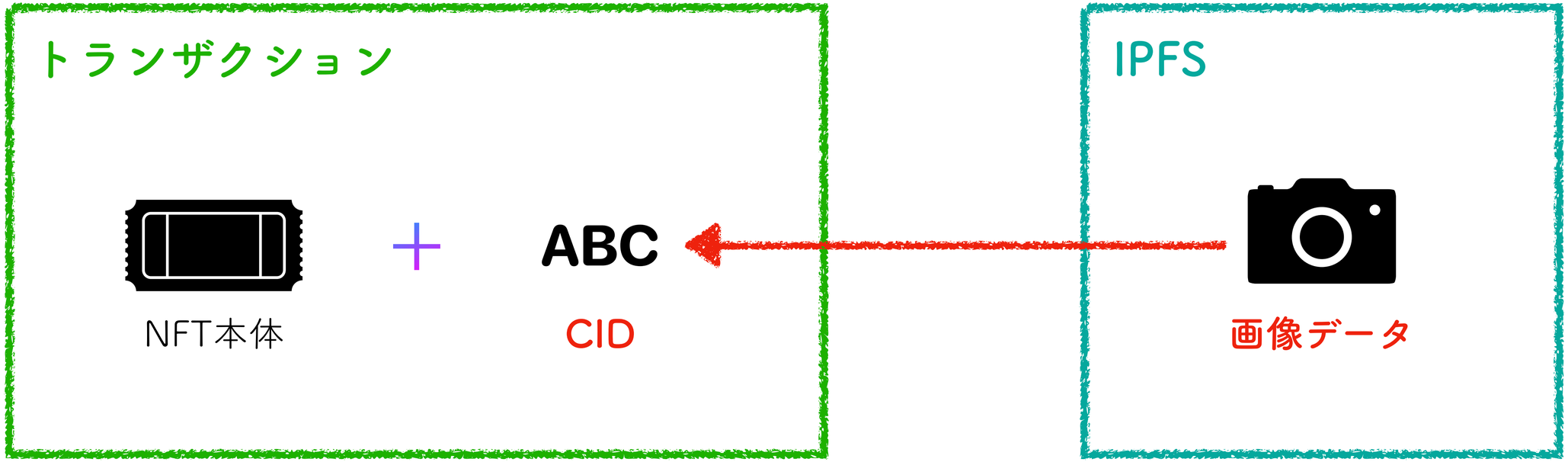
画像データを直接格納したいが

データサイズの制限が存在するため、

大きすぎる画像は入らない

実装

トランザクションサイズの制限

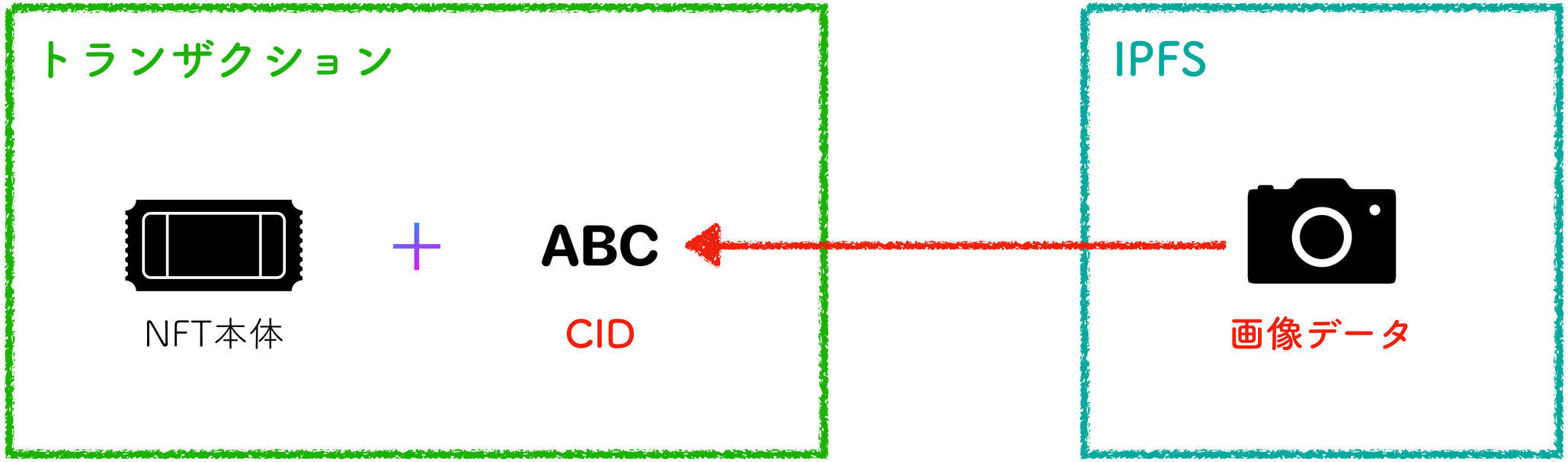


データをIPFS上に保存し

データを指し示すCIDを格納することに

実装

トランザクションサイズの制限



- IPFSでは画像ごとにユニークなCIDが発行されるので
  IPFS上での差し替えは不可能
- トランザクションデータも改ざん不可能

実装

Gluebyさん、

タイムスタンプ発行時にデータをTXに格納する機能は…　ある

NFT発行時にデータをTXに格納する機能は…　ない

タイムスタンプのコードを参考に
NFT発行時に任意データをTXに格納できるように
ライブラリを改造（Monky Patch）する

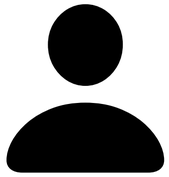
仕組みと流れ

トークン発行フロー

1. フロントで画像を受け取り、APIサーバーへbase64エンコードでユーザーIDと共に送信する。
2. APIサーバーでデータを受け取る。
3. base64からデコードしIPFSへアップロード、コンテンツIDを取得する。
4. そのコンテンツIDで過去にトークンが発行されていないか確認する。
5. ユーザーのウォレットを取得する。
6. ユーザー名義でトークンの発行を開始する。
7. 発行時に生成されるトランザクションにIPFSのアドレス（コンテンツID）を格納する。
8. データベースに発行されたトークンのIDとトランザクションID、IPFSのアドレスを保存する。
9. トークンIDをレスポンスとして返す。
10. フロントでレスポンスを受け取りトークンIDから画像を取得し表示する。

トークン発行フロー

フロント



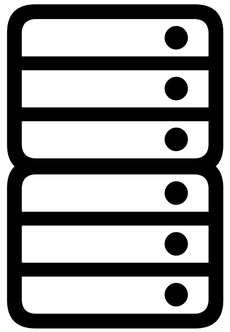
ユーザー

トークン発行フロー

トークン発行フロー

フロント

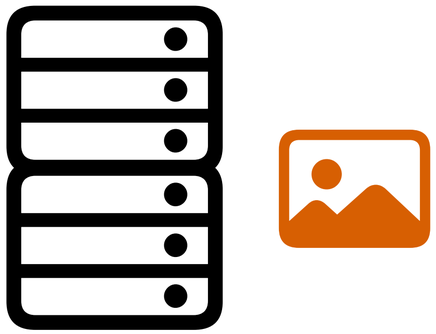
バックエンド



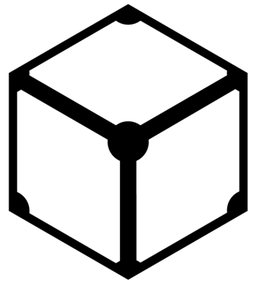
トークン発行フロー

トークン発行フロー

バックエンド



IPFS



ファイルストレージ

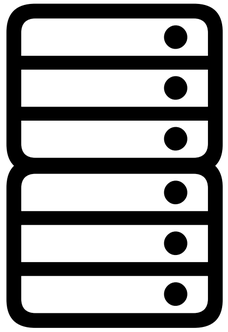
Tapyrus

ブロックチェーン

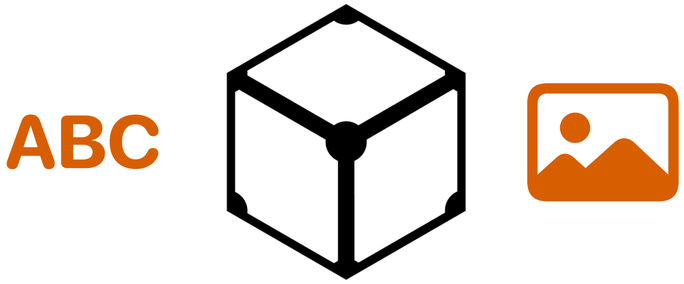
トークン発行フロー

トークン発行フロー

バックエンド



IPFS

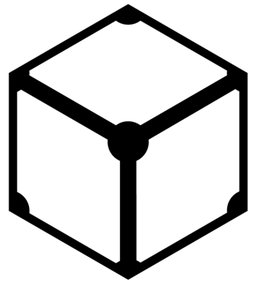


コンテンツIDを発行

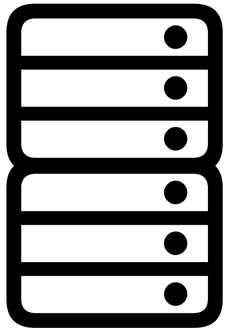
Tapyrus

トークン発行フロー

IPFS



バックエンド



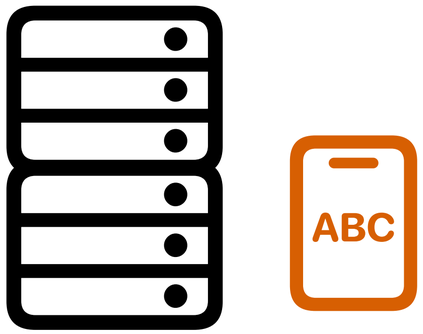
ABC

コンテンツIDを取得

Tapyrus

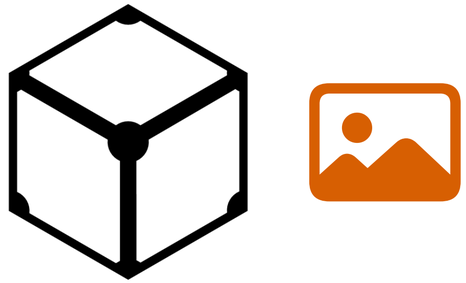
トークン発行フロー

バックエンド



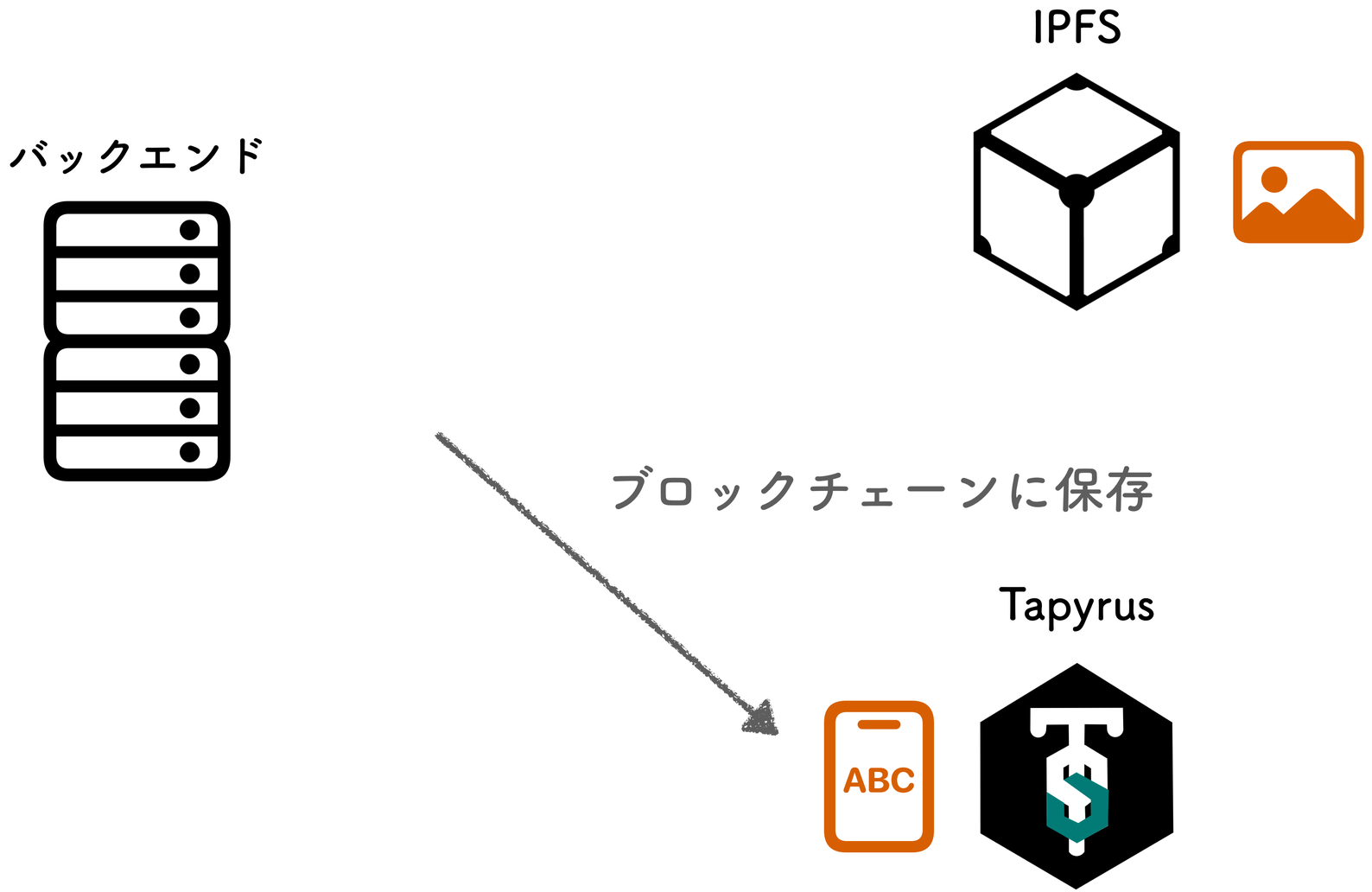
コンテンツIDを埋め込んだ
NFTを作成

IPFS



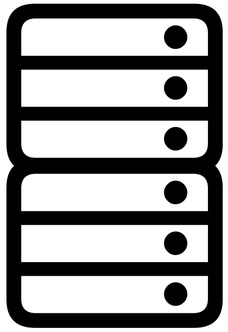
Tapyrus

トークン発行フロー

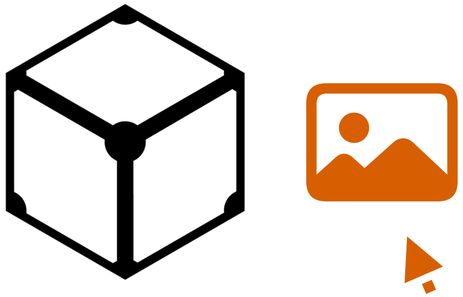


トークン発行フロー

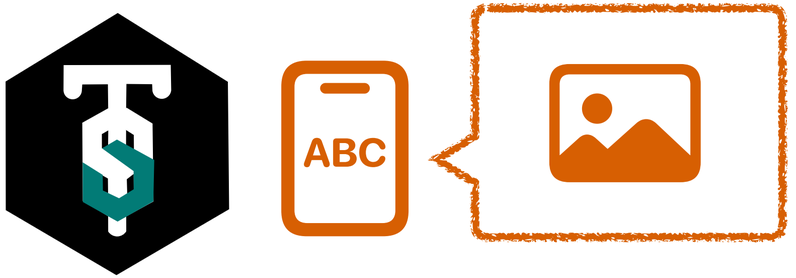
バックエンド



IPFS



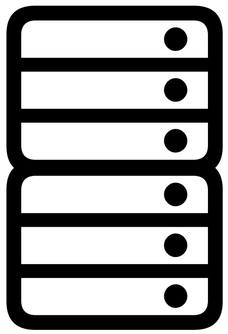
Tapyrus



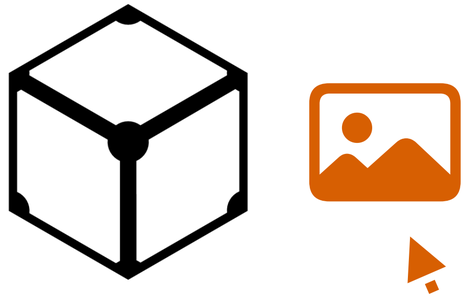
IPFS上に保存された
画像データを指し示す

トークン発行フロー

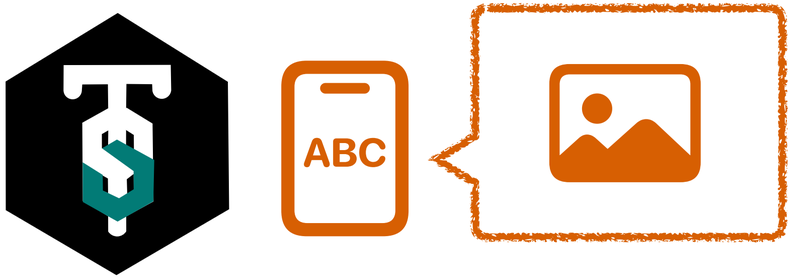
バックエンド



IPFS



Tapyrus



IPFS上に保存された
画像データを指し示す

Tap!のサーバー上で保管されることはなく
データの安全性・分散性が保たれる

仕組み・機能

この他にも

- **ユーザーの作成**
- **ユーザーのウォレットの変更***
- **ユーザーの退会***
- **トークンの移転***
- **トークンの焼却**

の機能を搭載（*印はフロント未実装）

ご清聴ありがとうございました