Hash verification

You may want to verify the authenticity and integrity of the application, deployed on IPFS. It can be done by CID (hash) verifying. In order to do so, you will need to download the source code of the application and build it locally. See the detailed instructions below.

Steps

info Lido Ethereum Staking Widget is taken as example here.

Prerequisites

You will need these tools installed in your system:

- Node.js 20+
- Yarn package manager v1 (classic)

Alternatively, you can use Docker to set up a building environment; read the sections below for the instructions.

1. Clone the repository

The repo for Ethereum Staking Widget is here:https://github.com/lidofinance/ethereum-staking-widget

2. Git checkout a commit, matching the IPFS version

You need togit checkout the specific commit, matching the release of an app you want to verify. This way, you can be sure that the app will not include any other changes, which affect the CID. There are several ways to do it.

Method 1 – using git tags

Each released version has its own git tag, one can use it for git checkout.

1. Open the app in your browser and check the right side of its footer.
2. There will be a version number, which is actually a link to a Releases page on GitHub.
3. Rungit fetch --all --tags --prune
4. to fetch all tags.
5. Rungit checkout tags/
6. , where
7. is the version from step 1.

Method 2 – searching on the GitHub Release page

1. Open the Releases page of the project's repository on GitHub. For Ethereum Staking Widget it ishere
2. .
3. Search manually for the latest release, where IPFS pinning happened.
4. Look for the commit hash near the release information.
5. Rungit checkout
6. , where
7. is the commit hash from the previous step.

3. Set up the project

Without Docker

1. Add ENV variables as instructed in README.
2. Removenode_modules
3. directory if the project was set up earlier.
4. Install node modules usingyarn install --frozen-lockfile
5. .
6. Follow other instructions described in the project's README.

Using Docker

If you have problems with setting up the environment or if it is your preference, you can use Docker to set up and build the project.

Steps for Docker 1. Configurebuild-info.json 2. as instructed inthis step 3. . 4. Createverification.Dockerfile 5. file in the project's root with this content:

build env

FROM node:20-alpine as build

WORKDIR /app

RUN apk add --no-cache git=~2 COPY package.json yarn.lock ./

RUN yarn install --frozen-lockfile --non-interactive --ignore-scripts && yarn cache clean

COPY . . RUN NODE_NO_BUILD_DYNAMICS=true yarn typechain && yarn build-ipfs

public/runtime is used to inject runtime vars; it should exist and user node should have write access there for it

RUN rm -rf /app/public/runtime && mkdir /app/public/runtime && chown node /app/public/runtime

final image

FROM node:20-alpine as base

WORKDIR /app RUN apk add --no-cache curl=~8 COPY --from=build /app /app 1. Add ENV variables as instructed in the project's README. 2. Run these commands:

docker build --no-cache -t verification:0 -f verification.Dockerfile . docker create --name verification-container verification:0 docker cp verification-container:/app/out /Users{Name}{Path-to-project}/dockerbuild-verification docker rm verification-container 1. Run further steps fromstep 6 2. of this instruction.

4. Configure build-info.json

Thebuild-info.json file is located in the project's root,here is the link . It must contain information about the version of the application, which is currently deployed to IPFS. You can take this information from the latest GitHub action in which IPFS pinning happened:

1. Open the app's repo, follow the "Actions" tab.
2. On the left, in the navigation, find the workflow for IPFS releasing, for the Ethereum Staking Widget it is called "IPFS Release
3. ".
4. Open the latest successful workflow and look for the "prepare-for-ipfs summary" title or the JSON data which looks like this:{ "branch": "main", "commit": "56ab68d", "version": "0.0.1" }
5. Copy the data to your localbuild-info.json

5. Build the IPFS version

Run a suitable npm script to build the IPFS version. In case of Ethereum Staking Widget, it isyarn build-ipfs .

6. Create a CAR file and get its CID (hash)

For Next.js applications the build files will be in theout directory. The following command generates a CAR file from theout directory with build files, and it will display the IPFS hash in the console:

npx ipfs-car pack ./out --output ./out.car

7. Get CID (hash) of the application deployed to IPFS

You will need to get the hash of the latest released CAR file. It can be found on the Releases page of the repository under the "Assets" collapsible block. Download the CAR file and run the following command:

npx ipfs-car roots ipfs_source_code.car It will show CID roots found in the CAR header. The CID (hash) must be the same as in the previous step. Edit this page Previous Security Next Lido IPFS applications