
DApps Delivery Guide Documentation

Release 0.0.1

Sergii Bomko

Aug 14, 2022

Contents

1	Web2 epoch	3
2	Web3 epoch	5
2.1	Delivery tools	5
2.1.1	GitHub and GitHub Actions	5
2.1.2	Fleek	7
2.2	Hosting	7
2.2.1	IPFS	7
2.2.2	Ethereum Swarm	9
2.3	Domain names	9
2.3.1	ENS	9
2.3.2	CNS	17
2.4	References	28

The guide covers the delivery process of DApps.



CHAPTER 1

Web2 epoch

We use to use DevOps tools and Internet infrastructure for Web Apps delivery. Everybody knows a number of hosting providers and domain name providers, including CI/CD services that allow achieving our needs in Web App delivery. In the guide, we would cover how we can use Web2 infrastructure and tools for Web3 applications.

Web3 is our nearest future which is still under construction.

2.1 Delivery tools

2.1.1 GitHub and GitHub Actions



GitHub provides hosting for software development version control using Git. GitHub is the largest host of source code in the world [1].

A **decentralized application** (DApp, dApp, Dapp, or dapp) is a computer application that runs on a distributed computing system. DApps have been popularized by distributed ledger technologies (DLT) such as the Ethereum Blockchain, where DApps are often referred to as smart contracts.

Many DApps are open-sourced, most of them are hosted in GitHub. That is why it is important to have proper tooling for delivering DApps.



GitHub Actions

[GitHub Actions](#) enables everyone to create custom software development life cycle (SDLC) workflows directly in the GitHub repository.

Since GitHub introduced Actions, a single commit is needed to create a pipeline for a DApp. You do not need to use 3rd party services for that anymore.

Basic Pipeline for a DApp

Step 1: Create pipeline

Open GitHub repository -> Actions -> *New Workflow* -> *Set up a workflow yourself*

The screenshot shows the GitHub Actions workflow editor for the repository 'aquiladev / ddns-dapp'. The workflow file 'main.yml' is being edited. The workflow is named 'CI' and is triggered on 'push' and 'pull_request' events for the 'master' branch. It contains a single job named 'build' that runs on the 'ubuntu-latest' runner. The job steps include checking out the repository using 'actions/checkout@v2'. The right sidebar displays the 'Marketplace' with a search bar and a list of featured actions, including 'Setup Node.js for use with actions', 'Upload artifact', and 'Setup Go for use with actions'.

Step 2: Modify pipeline

1. Define a trigger of the pipeline (e.g. only on *master* branch):

```
on:
  push:
    branches:
      - master
```

2. Leave the definition of the job and run environment:

```
jobs:
  build:
    runs-on: ubuntu-latest
```

3. Define prep-steps:

```
steps:
  - uses: actions/checkout@v2

  - name: Setup Node
    uses: actions/setup-node@v1
    with:
      node-version: '10.x'

  - run: npm ci

  - run: npm run build
```

Then you need to commit your *main.yml* by pressing *Start commit* button.

2.1.2 Fleek



Fleek is a service that allows launch and maintains fast, modern websites hosted on IPFS.

Take a look on [documentation](#)

2.2 Hosting

We used to deploy Web Apps in hosting services. DApps can and should be deployed in **decentralised storage**.

2.2.1 IPFS



InterPlanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.

IPFS allows users to not only receive but host content, in a similar manner to BitTorrent. As opposed to a centrally located server, IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node who has it using a distributed hash table (DHT).

Upload to IPFS

There is a [GitHub Action](#) which allows to upload a DApp to IPFS on Marketplace.

Input parameters:

Parameter	Re-quired	Ser-vice	Description
path	Yes		Directory's path to upload.
service	No		Type of target service to upload. Supported services [<i>ipfs</i> , <i>pinata</i> , <i>infura</i>]. Default <i>ipfs</i>
timeout	No		Request timeout. Default <i>60000</i> (1 minute)
verbose	No		Level of verbosity [<i>false</i> - quiet, <i>true</i> - verbose]. Default <i>false</i>
host	No	ipfs	IPFS host. Default <i>ipfs.komputing.org</i>
port	No	ipfs	IPFS host's port. Default <i>443</i>
protocol	No	ipfs	IPFS host's protocol. Default <i>https</i>
headers	No	ipfs	IPFS headers as json object. Default <i>{}</i>
pinataKey	Yes*	pinata	Pinata Api Key. Required for pinata service.
pinataSecret	Yes*	pinata	Pinata Secret Api Key. Required for pinata service.
pinataPin-Name	No	pinata	Human name for pin.

In order to use it, you need to add step to *main.yml*:

```
- uses: aquiladev/ipfs-action@v0.1.1
  id: upload
  with:
    path: ./build
```

There will be a build artifact on a runner after [steps](#) (usually in directory *build* or *dist*). You need to pass the directory as a *path* parameter.

The step will have *hash* output — it is needed for later use. Token `${{ steps.upload.outputs.hash }}` can be used in next steps where *upload* is the id of current step.

Upload to IPFS Pinata pinning service

The same GitHub Action allows to upload a DApp to [Pinata](#) pinning service. Pinata simplifies immutable data with simple IPFS API and toolkit.

In order to use it, you need to add step to *main.yml*:

```
- uses: aquiladev/ipfs-action@v0.1.3
  id: pinata
  with:
    path: ./build
    service: pinata
    pinataKey: ${ secrets.PINATA_KEY }
```

(continues on next page)

(continued from previous page)

```
pinataSecret: ${ secrets.PINATA_SECRET }
pinataPinName: {pin_name}
```

The output of the upload action is similar to the previous example.

2.2.2 Ethereum Swarm

Swarm - Decentralised data storage and distribution: Swarm is a peer to peer data sharing network in which files are addressed by the hash of their content. Similar to Bittorrent, it is possible to fetch the data from many nodes at once and as long as a single node hosts a piece of data, it will remain accessible everywhere. This approach makes it possible to distribute data without having to host any kind of server - data accessibility is location independent. Other nodes in the network can be incentivised to replicate and store the data themselves, obviating the need for hosting services when the original nodes are not connected to the network.

Upload to Swarm

There is a [GitHub Action](#) which allows to upload a DApp to Swarm on Marketplace.

In order to use it, you need to add step to *main.yml*:

```
- uses: aquiladev/swarm-action@v0.1
  id: upload
  with:
    path: ./build
```

There will be a build artifact on a runner after [steps](#) (usually in directory *build* or *dist*). You need to pass the directory as a *path* parameter.

The step will have *hash* output — it is needed for later use. Token `${ steps.upload.outputs.hash }` can be used in next steps where *upload* is the id of current step.

2.3 Domain names

2.3.1 ENS



[Ethereum Name Service \(ENS\)](#), a distributed, open, and extensible naming system based on the Ethereum blockchain. It maps human-readable names like *alice.eth* to machine-readable identifiers such as Ethereum addresses, content hashes, and metadata.

The [GitHub Action](#) allows updating *.eth* name during pipeline execution.

Registering .ETH name

This shortened guide will give you instructions of how to register a new .ETH name. For more detaild take a look at this [step-by-step tutorial](#) for registering .ETH name.

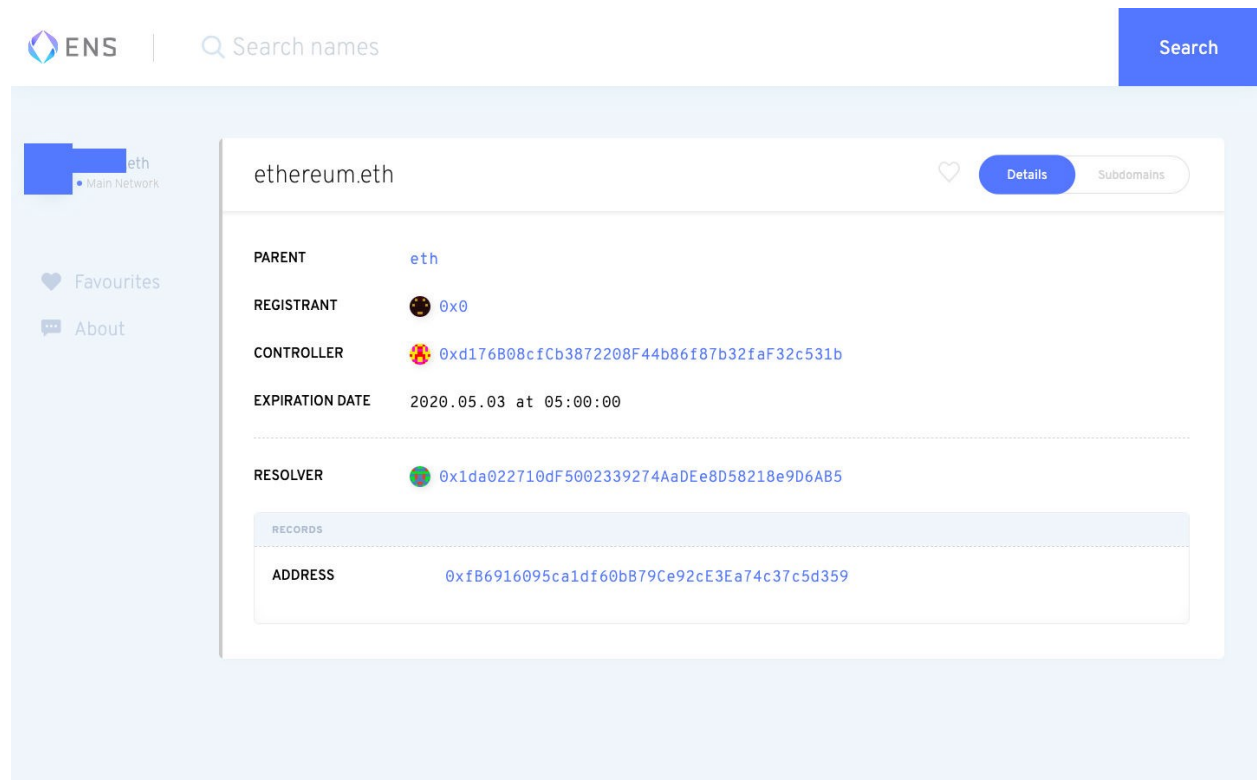
Step 1: Open your Ethereum-enabled browser

On desktop, this could be Chrome with the extension MetaMask, or the browser Brave with MetaMask (which comes built-in) enabled.

Step 2: Navigate to the ENS Manager

In your browser, go to app.ens.domains

Step 3: Search for your desired .eth name



The screenshot shows the ENS Manager interface. At the top, there is a search bar with the text "Search names" and a blue "Search" button. Below the search bar, the domain "ethereum.eth" is displayed. To the right of the domain name are a heart icon, a "Details" button, and a "Subdomains" button. The main content area shows the following details for "ethereum.eth":

PARENT	eth
REGISTRANT	0x0
CONTROLLER	0xd176808cfCb3872208F44b86f87b32faF32c531b
EXPIRATION DATE	2020.05.03 at 05:00:00
RESOLVER	0x1da022710dF5002339274AaDEe8D58218e9D6A85

Below the details, there is a "RECORDS" section with a table showing the address for the domain:

ADDRESS
0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359

Step 4: Start the registration process

Click the blue *Request To Register* button on the bottom right. A box should pop up from MetaMask asking you to confirm the transaction. Click the *Confirm* button to confirm it.

Step 5: Wait

The waiting period is required to ensure another person hasn't tried to register the same name

Step 6: Finish registration

Once your wait time is over, orange text will appear that says *Click register to move to the 3rd step*. Click the blue next to it that says *Register*.

Step 7: Set Relosver

After registration is completed you need to set Resolver.

The screenshot shows the ENS website interface. The browser address bar displays 'app.ens.domains/name/ethereum.eth'. The page header includes the ENS logo, a search bar with the placeholder 'Search names or addresses', and a 'Search' button. On the left sidebar, there is a user profile for '0x989F400Cd...' on the 'Main Network', along with 'Favourites' and 'About' links. The main content area displays the details for the domain 'ethereum.eth'. It includes tabs for 'Register', 'Details' (selected), and 'Subdomains'. The domain information is as follows:

Field	Value	Action
PARENT	eth	
REGISTRANT	0x0ABa55c93cF7292f71067B0Ba0D8b464592895cA	Transfer
CONTROLLER	0x809FA673fe2ab515FaA168259cB14E2BeDeBF68e	Transfer
EXPIRATION DATE	2030.05.04 at 13:12	Renew
RESOLVER	0x1da022710dF5002339274AaDEe8D58218e9D6AB5	Set

Below the domain details, there is a 'RECORDS' section with a table:

Field	Value	Action
ADDRESS	0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359	Edit

At the bottom, there is a section for 'OTHER ADDRESSES'.

Setup pipeline with .ETH update

Requirements

1. Before setting up a pipeline the ENS name should be configured, it should have a resolver. Take a look on prev section.
2. Basic pipeline should be configured with step which provides IPFS hash

Pipeline step

Open and add the step to *main.yml*:

```
- uses: aquiladev/ddns-action@v0.1.1
  with:
    mnemonic: ${{ secrets.MNEMONIC }}
    rpc: ${{ secrets.RPC }}
    name: ddns-action.eth
    contentHash: ${{ steps.upload.outputs.hash }}
```

Parameters

- `${{ secrets.MNEMONIC }}` is a secret. The mnemonic phrase is needed for wallet recovery of an account which owns ENS name. It can be a private key of the account as well
- `${{ secrets.RPC }}` is a secret. RPC is a URL of Ethereum Mainnet node
- `ddns-action.eth` - ENS name which you want to update
- `${{ steps.upload.outputs.hash }}` is content hash. It came from [upload to IPFS step](#)

Secrets

In order to manage secrets in a repository you need to open Settings -> Secrets

The screenshot shows the GitHub interface for the repository 'ddns-dapp' by 'aquiladev'. The 'Settings' tab is selected, and the 'Secrets' section is active. The left sidebar contains navigation links: Options, Manage access, Branches, Webhooks, Notifications, Integrations & services, Deploy keys, Secrets (highlighted), Actions, Moderation, and Interaction limits. The main content area is titled 'Secrets' and explains that secrets are encrypted environment variables. It lists three existing secrets: 'ETHERSCAN_KEY', 'MNEMONIC', and 'RPC', each with a 'Remove' button. At the bottom, there is a link to 'Add a new secret'.

Pipeline

Eventually pipeline should look like:


```
name: CI
on:
  push:
    branches:
      - master

jobs:
  build:
    runs-on: ubuntu-latest

steps:
  - uses: actions/checkout@v2

  - name: Setup Node
    uses: actions/setup-node@v1
    with:
      node-version: '10.x'

  - run: npm ci

  - run: npm run build

  - name: Upload to IPFS
    uses: aquiladev/ipfs-action@v0.1.2-alpha
    id: upload
    with:
      path: ./build

  - name: Update ENS
    uses: aquiladev/ddns-action@v0.1.1
    with:
      mnemonic: ${ secrets.MNEMONIC }
      rpc: ${ secrets.RPC }
      name: ddns-action.eth
      contentHash: ${ steps.upload.outputs.hash }
```

Run pipeline

The pipeline will run immediately after commit (if you committed to master branch)

The screenshot shows the GitHub Actions interface for the repository `aquiladev/ddns-dapp`. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. Below the repository name, there are tabs for Code, Issues, Pull requests, Actions (selected), Projects, Wiki, Security, Insights, and Settings. The 'All workflows' section is active, displaying a list of workflow runs for the 'Update main.yml' workflow. The runs are filtered by the 'CI' event. The table shows the status of each run, the commit hash, the actor, and the time taken. The first run (CI #29) failed, while the second (CI #28) succeeded. The subsequent runs (CI #27, #26, #25) all failed.

Event	Status	Branch	Actor	Time
Update main.yml	Failed	master	CI #29: Commit 9b54b71 pushed by aquiladev	2 days ago, 4m 12s
Update main.yml	Success	master	CI #28: Commit 558181f pushed by aquiladev	3 days ago, 54m 27s
Update main.yml	Failed	master	CI #27: Commit cc3e4b0 pushed by aquiladev	3 days ago, 4m 49s
Update main.yml	Failed	master	CI #26: Commit dc0a6a9 pushed by aquiladev	3 days ago, 4m 55s
Update main.yml	Failed	master	CI #25: Commit d337db6 pushed by aquiladev	3 days ago, 4m 5s
Update main.yml	Failed	master		3 days ago

You can open all pipeline runs and check outputs

The screenshot shows a GitHub repository for 'aquiladev/ddns-dapp'. The 'Actions' tab is selected, displaying a workflow run titled 'Update main.yml' on the 'master' branch. The workflow is currently in a 'Failed' state. The left sidebar shows the workflow steps, with 'build' selected. The main panel shows the details of the 'build' job, which failed 2 days ago in 4m 1s. The job steps are listed with their durations: 'Set up job' (1m 20s), 'Run actions/checkout@v2' (1s), 'Request Block' (3s), 'Parse' (6s), 'Setup Node' (4s), 'Cache dependencies' (1s), 'Run npm ci' (45s), 'Build' (40s), and 'Upload to IPFS' (1m 0s). The 'Upload to IPFS' step is marked as failed. The logs for this step show the command 'Run aquiladev/ipfs-action@v0.1.2-alpha' and the output of the 'ipfs' command, which lists the hashes for the build assets. A yellow banner at the bottom of the logs indicates 'Annotations: 1 failure'.

How to access a DApp with .ETH name?

After a successful run of the pipeline, you should be able to access your updated DApp, but it takes some time on IPFS side to resolve newly uploaded content.

Check ENS record

In order to check updated IPFS hash in ENS, you need to open <https://app.ens.domains>

The screenshot shows the ENS domain management interface for the domain `ddns-action.eth`. The browser address bar shows `app.ens.domains/name/ddns-action.eth`. The ENS logo and a search bar are at the top. The domain name `ddns-action.eth` is displayed with a 'Registrant' label and a heart icon. Below this, there are tabs for 'Register', 'Details' (selected), and 'Subdomains'. A sidebar on the left shows the user's profile (0x989F400Cd...) and options for 'Favourites' and 'About'. The main content area shows the domain details: 'PARENT' is `eth`; 'REGISTRANT' is `0x989F400Cd59584415Aa44e676b75A45873b59834` with a 'Transfer' button; 'CONTROLLER' is the same address with a 'Set' button; 'EXPIRATION DATE' is '2021.02.25 at 18:27' with a 'Renew' button; and 'RESOLVER' is `0x4976fb03C32e5B8cfe2b6cCB31c09Ba78EBaBa41` with a 'Set' button. Below these details is a 'RECORDS' section with a '+' icon. It contains an 'ADDRESS' record with the value `0x989F400Cd59584415Aa44e676b75A45873b59834` and a 'Reverse record: not set' status. There is also a 'CONTENT' record with the value `ipfs://QmUNLLsPACcz1vLxQVvXqqLX5R1X345qqfHbsf67hvA3Nn`.

Access DApp

The easiest way to open your DApp through <http://eth.link>



2.3.2 CNS



Unstoppable Domains launching domain names on blockchain technology. The domain names can be used for both payments and websites. A domain name can't be taken down by the players that typically participating in the domain name system. Hence, the name Unstoppable Domains. Another value is that It can replace your cryptocurrency addresses with one single human-readable name.

Unstoppable Domains provides *.crypto* (aka CNS) domain names for the entire crypto community.

The [GitHub Action](#) allows updating *.crypto* name during pipeline execution.

Registering a .CRYPTO Name

This shortened guide will give you instructions of how to register a new *.CRYPTO* name. For more detaild take a look at this [step-by-step tutorial](#) for registering *.CRYPTO* name.

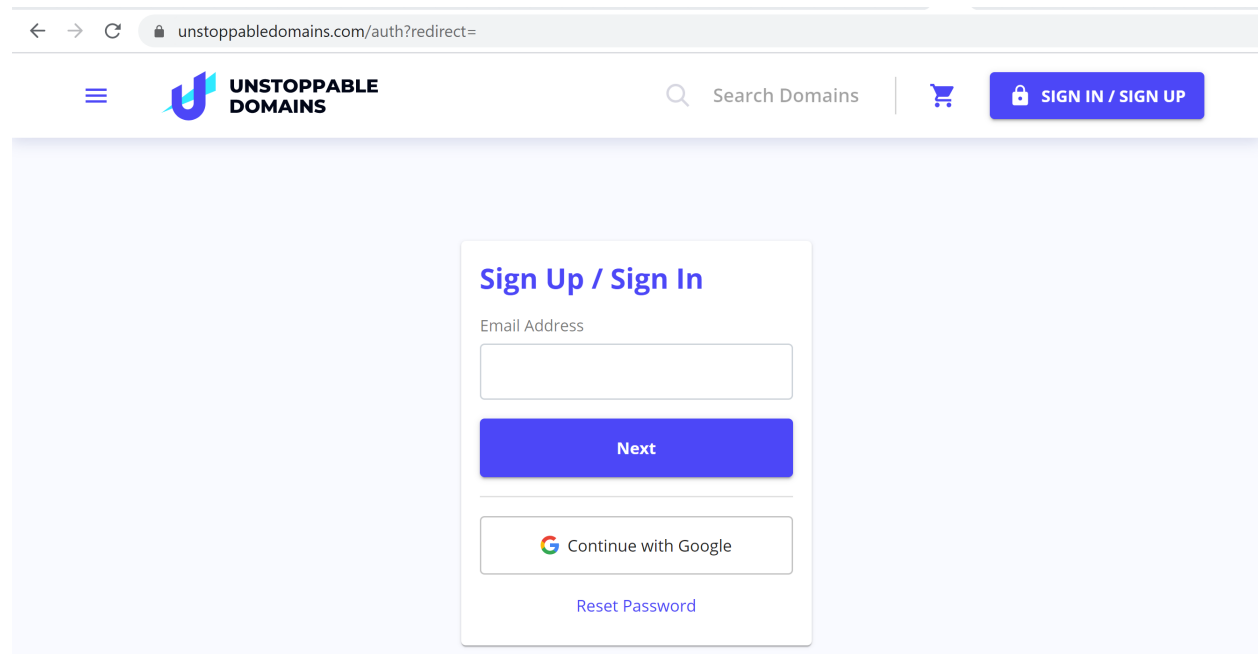
Step 1: Open your Ethereum-enabled browser

On desktop, this could be Chrome with the extension MetaMask, or the browser Brave with MetaMask (which comes built-in) enabled.

Step 2: Navigate to the Unstoppable Domains web site

In your browser, go to unstoppabledomains.com

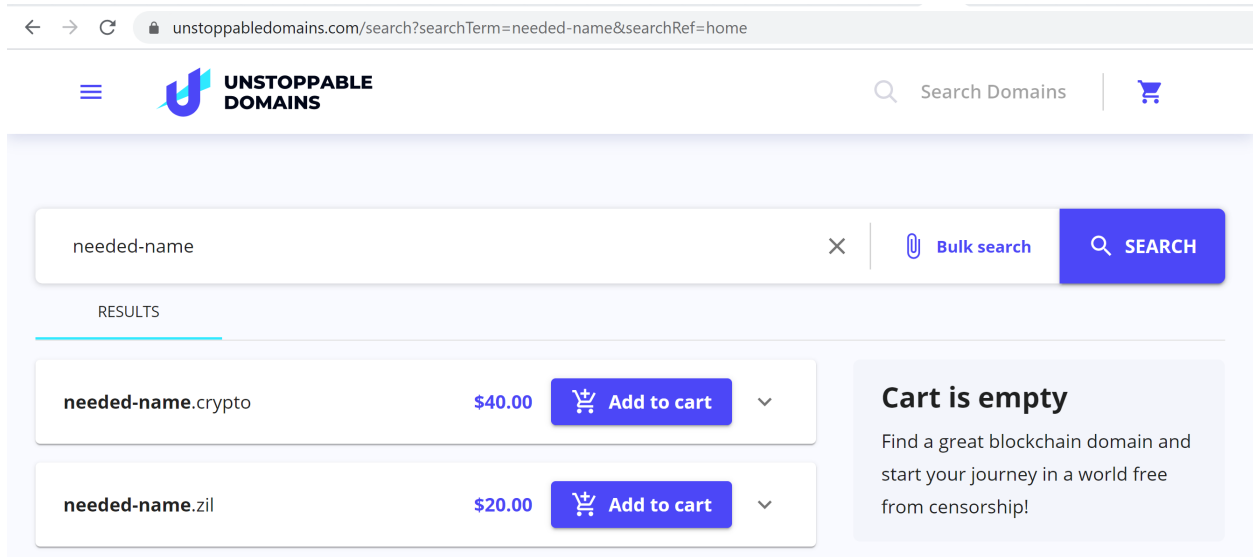
Step 3: Pass sign up process



The screenshot shows a web browser window with the URL `unstoppabledomains.com/auth?redirect=`. The page header includes the Unstoppable Domains logo, a search bar labeled "Search Domains", and a "SIGN IN / SIGN UP" button. The main content area features a "Sign Up / Sign In" form with an "Email Address" input field, a "Next" button, a "Continue with Google" button, and a "Reset Password" link.

Step 4: Search for your desired .crypto name

Login and search for *.crypto* name



The screenshot shows the Unstoppable Domains search interface. The browser address bar displays `unstoppabledomains.com/search?searchTerm=needed-name&searchRef=home`. The page header includes the Unstoppable Domains logo, a search bar with the text "Search Domains", and a shopping cart icon. The search results section shows two domains: `needed-name.crypto` priced at \$40.00 and `needed-name.zil` priced at \$20.00. Each domain has an "Add to cart" button. A "Cart is empty" message is displayed on the right, encouraging users to find a blockchain domain.

needed-name

RESULTS

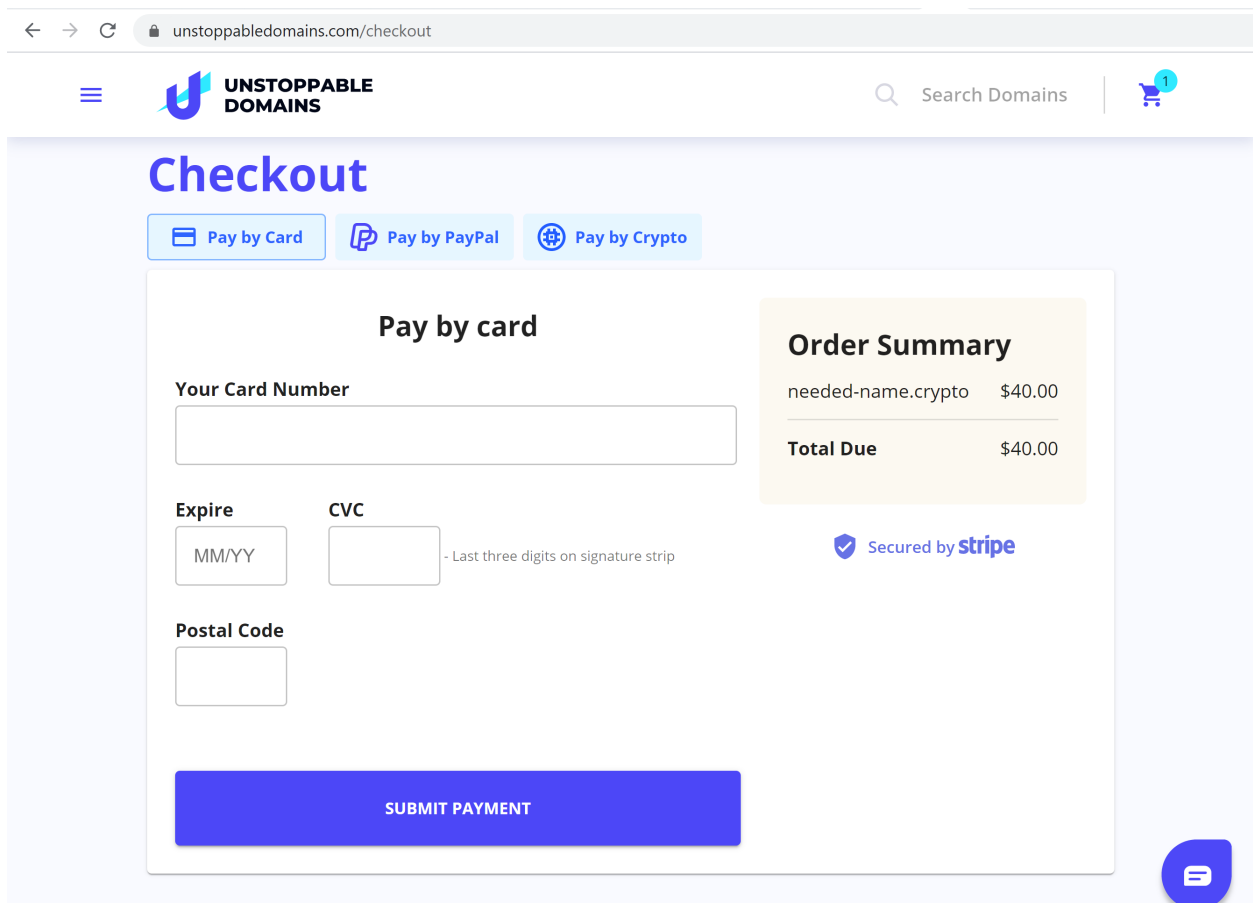
needed-name.crypto \$40.00 Add to cart

needed-name.zil \$20.00 Add to cart

Cart is empty
Find a great blockchain domain and start your journey in a world free from censorship!

Step 5: Checkout

Add `.crypto` name to a cart and go to Checkout page. Choose a suitable payment type and complete payment



The screenshot shows the Unstoppable Domains checkout page. The browser address bar displays `unstoppabledomains.com/checkout`. The page header includes the Unstoppable Domains logo, a search bar with the text "Search Domains", and a shopping cart icon with a notification badge. The main heading is "Checkout". Below the heading are three payment options: "Pay by Card", "Pay by PayPal", and "Pay by Crypto". The "Pay by Card" option is selected. The "Pay by card" section contains fields for "Your Card Number", "Expire" (MM/YY), "CVC", and "Postal Code". An "Order Summary" box on the right shows the domain `needed-name.crypto` for \$40.00 and a "Total Due" of \$40.00. A "Secured by stripe" logo is also present. A large blue "SUBMIT PAYMENT" button is at the bottom.

Checkout

Pay by Card Pay by PayPal Pay by Crypto

Pay by card

Your Card Number

Expire MM/YY CVC - Last three digits on signature strip

Postal Code

Order Summary

needed-name.crypto \$40.00

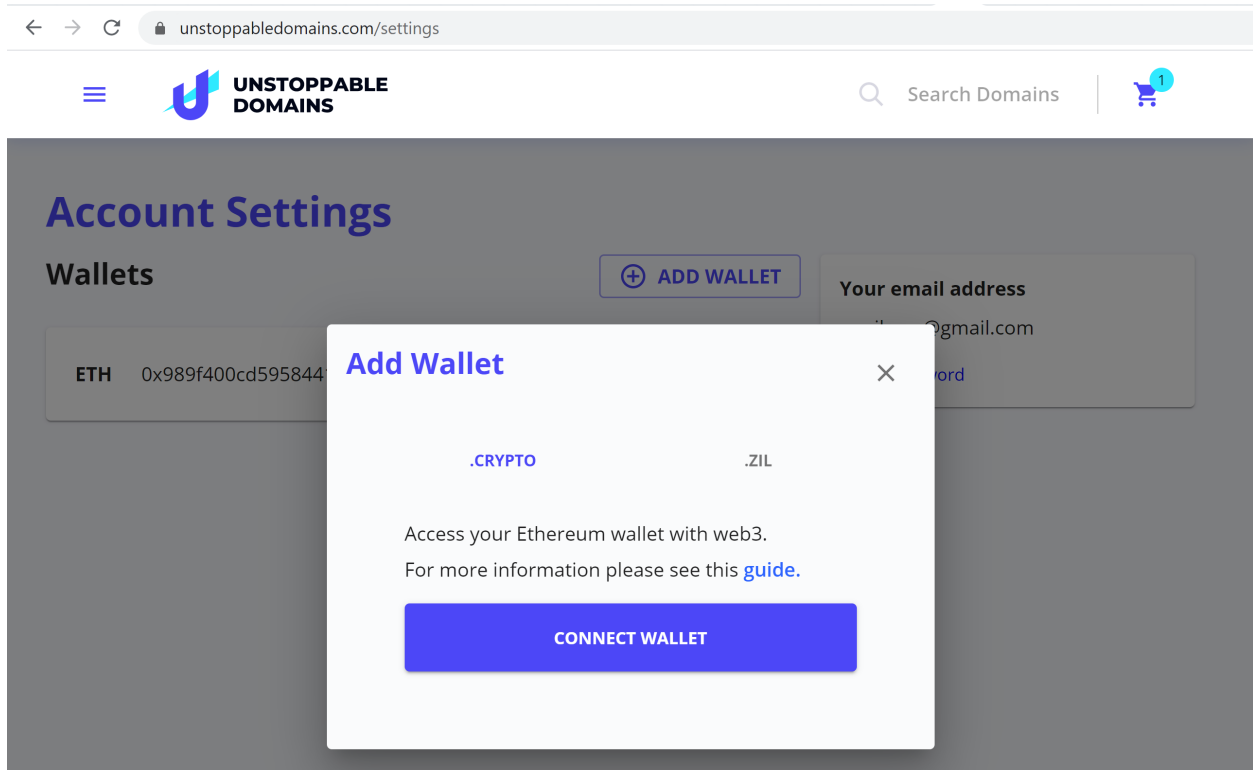
Total Due \$40.00

Secured by stripe

SUBMIT PAYMENT

Step 6: Connect Wallet to your account

Go to Account Settings page and click *Add wallet* button. You need to connect your MetaMask account to the web site in an appeared popup.



Step 7: Claim your domain name

Go to My Domains page, select domain name and click *Claim*

The screenshot shows the 'My Domains' interface on the Unstoppable Domains website. The header includes the site logo, a search bar, and a shopping cart icon with a '1' badge. The main heading is 'My Domains'. Below it, there's a control bar with a checked checkbox labeled 'Unselect', a 'Sort by: Name' dropdown, and a search icon. A button 'Claim selected domains (1)' is present. To the right, a notification states 'You selected 1 unclaimed pre-order' with a prominent blue 'CLAIM' button. Below the control bar, there are filters for '.crypto' and '.zil' domains, and a 'Status: All' dropdown. The domain list table has one entry: 'ddns-action.crypto' with a checked checkbox and the status 'Unclaimed'. At the bottom of the table, it shows 'Rows per page: 50' and '1-1 of 1'. A blue button '+ GET A NEW DOMAIN' is at the bottom center.

The process takes time, there is manual approval. You will need to wait for a while.

Step 8: Manage domain name

After all you will be able to manage your domains. Go to My Domains page and click *Manage* button for the domain name.

unstoppabledomains.com/manage?domain=ddns-action.crypto&page=settings

UNSTOPPABLE DOMAINS

Search Domains

ddns-action.crypto

Manage Transfer Website Back

Owner
0x989f400cd59584415aa44e676b75a45873b59834

Enter record information about your domain to be displayed when your domain is searched

✓ SAVE CHANGES

Add Cryptocurrency Addresses

Add cryptocurrency addresses from any wallet you control

Currency	Address
BTC	Enter your wallet address
ETH	0x989f400cd59584415aa44e676b75a45873b59834
ZIL	Enter your wallet address

Send cryptocurrency to a blockchain domain in these wallets right now:

- Coinomi
- Trust
- Ambo
- Atomic
- Guarda
- Coinrequest
- Viewblock
- Moonlet

Step 9: Set Resolver

In order to do this, you need to add at least one cryptocurrency address (e.g. owner of the name).

Setup pipeline with .CRYPTO update

Requirements

1. Before setting up a pipeline the CNS name should be configured, it should have a resolver. Take a look on prev section
2. Basic pipeline should be configured with step which provides IPFS hash

Pipeline step

Open and add the step to *main.yml*:

```
- uses: aquiladev/ddns-action@v0.1.1
  with:
    mnemonic: ${ secrets.MNEMONIC }
```

(continues on next page)

(continued from previous page)

```
rpc: ${ secrets.RPC }
name: ddns-action.crypto
contentHash: ${ steps.upload.outputs.hash }
```

Parameters

- `${ secrets.MNEMONIC }` is a secret. The mnemonic phrase is needed for wallet recovery of an account which owns CNS name. It can be a private key of the account as well
- `${ secrets.RPC }` is a secret. RPC is a URL of Ethereum Mainnet node
- `ddns-action.crypto` - CNS name which you want to update
- `${ steps.upload.outputs.hash }` is content hash. It came from [upload to IPFS step](#)

Secrets

In order to manage secrets in a repository you need to open Settings -> Secrets

The screenshot shows the GitHub interface for the repository 'ddns-dapp' by user 'aquiladev'. The 'Settings' tab is selected, and the 'Secrets' sub-tab is active. The left sidebar contains navigation links: Options, Manage access, Branches, Webhooks, Notifications, Integrations & services, Deploy keys, Secrets (highlighted), Actions, Moderation, and Interaction limits. The main content area is titled 'Secrets' and explains that secrets are encrypted environment variables. It lists three existing secrets: 'ETHERSCAN_KEY', 'MNEMONIC', and 'RPC', each with a 'Remove' button. At the bottom, there is a link to 'Add a new secret'.

Pipeline

Eventually pipeline should look like:

```
name: CI
on:
  push:
    branches:
      - master

jobs:
  build:
    runs-on: ubuntu-latest

steps:
  - uses: actions/checkout@v2

  - name: Setup Node
    uses: actions/setup-node@v1
    with:
      node-version: '10.x'

  - run: npm ci

  - run: npm run build

  - name: Upload to IPFS
    uses: aquiladev/ipfs-action@v0.1.2-alpha
    id: upload
    with:
      path: ./build

  - name: Update CNS
    uses: aquiladev/ddns-action@v0.1.1
    with:
      mnemonic: ${ secrets.MNEMONIC }
      rpc: ${ secrets.RPC }
      name: ddns-action.crypto
      contentHash: ${ steps.upload.outputs.hash }
```

Run pipeline

The pipeline will run immediately after commit (if you committed to master branch)

github.com/aquiladev/ddns-dapp/actions

Search or jump to... Pull requests Issues Marketplace Explore

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Workflows [New workflow](#)

All workflows

Filter workflows

CI

Event	Status	Branch	Actor
Update main.yml	Failed	master	CI #29: Commit 9b54b71 pushed by aquiladev
Update main.yml	Successful	master	CI #28: Commit 558181f pushed by aquiladev
Update main.yml	Failed	master	CI #27: Commit cc3e4b0 pushed by aquiladev
Update main.yml	Failed	master	CI #26: Commit dc0a6a9 pushed by aquiladev
Update main.yml	Failed	master	CI #25: Commit d337db6 pushed by aquiladev
Update main.yml	Failed	master	

You can open all pipeline runs and check outputs

The screenshot shows a GitHub repository named 'ddns-dapp' by 'aquiladev'. The 'Actions' tab is selected, displaying a workflow run titled 'Update main.yml' on the 'master' branch. The workflow is currently in a failed state. The left sidebar shows the workflow steps, with 'build' selected. The main area shows the details of the 'build' step, which failed 2 days ago in 4m 1s. The step log shows the following commands and their durations:

- Set up job: 1m 20s
- Run actions/checkout@v2: 1s
- Request Block: 3s
- Parse: 6s
- Setup Node: 4s
- Cache dependencies: 1s
- Run npm ci: 45s
- Build: 40s
- Upload to IPFS: 1m 0s

The 'Upload to IPFS' step failed. The log shows the command 'Run aquiladev/ipfs-action@v0.1.2-alpha' and the output of the 'build' step, which includes the IPFS hashes for the assets:

```
1 ► Run aquiladev/ipfs-action@v0.1.2-alpha
9 build/asset-manifest.json QmXvrJQmB375vDrmVnMo124JmfqHv64zLZxS6MRuTNVDBg
10 build/favicon.ico QmNrcCAVt4GyCvQbg1J22nPFhTWcn9HYLSg46t8WLzFPo
11 build/index.html QmeyrjGKvWV2BhsacZz7FfKacB1KoaAMtRWpZfhmRkuvJB
12 build/logo192.png QmYwV2BhsacZz7FfKacB1KoaAMtRWpZfhmRkuvJB
13 build/logo512.png QmYwV2BhsacZz7FfKacB1KoaAMtRWpZfhmRkuvJB
14 build/manifest.json QmYwV2BhsacZz7FfKacB1KoaAMtRWpZfhmRkuvJB
```

An error message at the bottom of the log indicates 'Annotations: 1 failure'.

How to access a DApp with .CRYPTO name?

Check CNS record

In order to check updated IPFS hash in CNS, you need to open My Domains page, click *Manage* button for the domain name. Then you need to open *Website* tab.

The screenshot shows the Unstoppable Domains management page for the domain **ddns-action.crypto**. The browser address bar shows the URL: `unstoppabledomains.com/manage?domain=ddns-action.crypto&page=settings`. The page header includes the Unstoppable Domains logo, a search bar, and a shopping cart icon. Below the domain name, there are three tabs: **Manage** (active), **Transfer**, and **Website**, along with a **Back** button. The main content area is divided into two columns. The left column contains a section titled "Use a template to quickly launch your site" with a "CHOOSE A TEMPLATE" button and a note about selecting a template to launch the site in minutes. Below this is an "IPFS" section with a "How to launch your website on IPFS" link, an "IPFS Hash" input field containing the hash `QmSZ1DncBVxHyS85vFdwPpLTRJrcAG5K1UfWCbHZ2TRxE3`, and a "Redirect to Traditional Domain" section with an "Enter redirect URL" input field. The right column contains a section for entering record information about the domain to be displayed when searched, with a "SAVE CHANGES" button. Below this is a section for sending cryptocurrency to a blockchain domain in specific wallets, listing **Coinomi**, **Trust**, **Ambo**, **Atomic**, and **Guarda**.

Access DApp

1. You should install [Chrome Extension](#). The Extension allows Chrome browser to handle `.crypto` domain names.
2. Type `.crypto` (e.g. `ddns-action.crypto`) domain name in Chrome browser and you will be redirected to the DApp

← → ↻ gateway.ipfs.io/ipfs/QmSZ1DncBVxHyS85vFdwPpLTRJrcAG5K1UfWCbHZ2TRxE3/

12613 block(s) to Ethereum Mainnet since last update

2.4 References

1. [ENS update automation](#)
2. [CNS update automation](#)
3. [DApp's pipeline \(IPFS + ENS\)](#)

Contents:

[Keyword Index](#), [Search Page](#)