➡

Test, Deploy & Run Typescript functions

Testing Typescript Functions

To test your Typescript Function locally, run:

```

Copy npxhardhatw3f-runW3FNAME

```

Example:

```

Copy npxhardhatw3f-runoracle

```

Optional flags:

- --logs
- Show internal Web3 Function logs
- --debug
- Show Runtime debug messages
- --network [NETWORK]
- Set the default runtime network & provider.
-

Example:

```

Copy npxhardhatw3f-runoracle--logs--networkhardhat

```

Output:

```

Copy Web3FunctionBuildresult: ✓Schema:/Users/chuahsonglin/Documents/GitHub/Gelato/contract/w3f-template/web3-functions/oracle/schema.json ✓Builtfile:/Users/chuahsonglin/Documents/GitHub/Gelato/contract/w3f-template/.tmp/index.js ✓Filesize:2.47mb ✓Buildtime:947.91ms

Web3Functionuserargsvalidation: ✓currency:ethereum ✓oracle:0x71B9B0F6C999CBbB0FeF9c92B80D54e4973214da

Web3Functionrunninglogs:

    Last oracle update: 0 Next oracle update: 3600 Updating price: 1898

Web3FunctionResult: ✓Returnvalue:{ canExec:true, callData:[ { to:'0x71B9B0F6C999CBbB0FeF9c92B80D54e4973214da', data:'0x8d6cc56d0000000000000000000000000000000000000000000000000000000000000076a' } ] }

Web3FunctionRuntimestats: ✓Duration:1.35s ✓Memory:113.55mb ✓Storage:0.03kb ✓Rpccalls:3

```

Deploying Typescript Functions

To compile your Typescript Function and deploy it to IFPS, use

```

Copy npxhardhatw3f-deployW3FNAME

```

Example:

```

Copy npxhardhatw3f-deployoracle

```

Once uploaded, Gelato Nodes will pin the file on your behalf on IPFS. If the upload was successful, you should get the IPFS CID of your Typescript Function returned.

```

Copy ✓Web3Functiondeployedtoipfs. ✓CID:QmbQJC5XGpQUsAkLq6BqpvLtD8EPNDEaPqyFf4xK3TM6xj

```

Note: This CID will be different for every new Typescript Function version that you will deploy.

Creating Typescript Function Task

Before creating solidity function tasks, familiarize yourself with the availableTrigger Types ! ?

1. Selection of Function
2.
    - Navigate to theWhat to trigger
3.
    - section.
4.
    - Within theTypescript Function
5.
    - subsection, find theIPFS CID
6.
    - input box.
7. *
8. Function Details Input
9.
    - Input the CID you secured after deploying your Typescript function. Upon entry, you should see a message like "Typescript Function code imported," signifying a successful connection.
10. *
11. Network Configuration
12.
    - Scroll to theNetwork
13.
    - dropdown menu.
14.
    - Choose the blockchain network where the Typescript function should work, e.g., "Göerli."
15. *
16. Task Configuration
17.
    - If your Typescript function needs secret variables or API keys, securely enter them in theTask Secrets
18.
    - section. For every secret:
19.
    -
        - Key: Define the name of the variable or key, e.g., "API_KEY".
20.
    -
        - Value: Enter the associated secret value.
21.
    -
        - ClickSave
22.
    -
        - after each input to guarantee its safe storage.
23.
    - *
24. *
25.

Previous Callbacks Next Writing Solidity Functions Last updated4 months ago On this page *Testing Typescript Functions * Optional flags: * Deploying Typescript Functions * Creating Typescript Function Task