

Hardhat Starter Kit

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "@chainlink/contracts/src/v0.8/ChainlinkClient.sol";
5
6 /**
7  * @title The APIConsumer contract
8  * @notice An API Consumer contract that makes GET requests to obtain 24h trading volume of ETH in USD
9  */
10 contract APIConsumer is ChainlinkClient {
11     using Chainlink for Chainlink.Request;
12
13     uint256 public volume;
14     address private immutable oracle;
15     bytes32 private immutable jobId;
16     uint256 private immutable fee;
17
18     event DataFullfilled(uint256 volume);
19
20     /**
21      * @notice Executes once when a contract is created to initialize state variables
22      *
23      * @param _oracle - address of the specific Chainlink node that a contract makes an API call from
24      * @param _jobId - specific job for :oracle: to run; each job is unique and returns different types
25      * @param _fee - fee to pay for the API call / data request
26      */
27     *
28     * Network: Rinkeby
29     * Oracle: 0xc57b33452b4f7bb189bb5afae9cc4aba1f7a4fd8
30     * Job ID: 6b88e0402e5d415eb946e528b8e0c7ba
31     * Fee: 0.1 LINK
32     */
33     constructor(
34         address _oracle,
35         bytes32 _jobId,
36         uint256 _fee,
37         address _link
38     ) {
39         if (_link == address(0)) {
40             setPublicChainlinkToken();
41         } else {
42             setChainlinkToken(_link);
43         }
44         oracle = _oracle;
45         jobId = _jobId;
46         fee = _fee;
47     }
48
49     /**
50      * @notice Creates a Chainlink request to retrieve API response, find the target
51      *
52      * @dev This function is called by the Chainlink node to retrieve the API response and find the target
53      *
54      * @param _request - Chainlink request object
55      * @param _response - API response
56      * @return - bool
57      */
58     function _handleRequest(Chainlink.Request memory _request, bytes memory _response) internal returns (bool) {
59         // Parse the response and find the target
60         // ...
61         return true;
62     }
63 }
```

```
54  */
55  function requestVolumeData() public returns (bytes32 requestId) {
56      Chainlink.Request memory request = buildChainlinkRequest(
57          jobId,
58          address(this),
59          this.fulfill.selector
60      );
61
62      // Set the URL to perform the GET request on
63      request.add("get", "https://min-api.cryptocompare.com/data/pricemultifull?fsyms=ETH&tsyms=USD");
64
65      // Set the path to find the desired data in the API response, where the response format is:
66      // {"RAW":
67      //   {"ETH":
68      //     {"USD":
69      //       {
70      //         "VOLUME24HOUR": xxx.xxx,
71      //       }
72      //     }
73      //   }
74      // }
75      // request.add("path", "RAW.ETH.USD.VOLUME24HOUR"); // Chainlink nodes prior to 1.0.0 support this
76      request.add("path", "RAW,ETH,USD,VOLUME24HOUR"); // Chainlink nodes 1.0.0 and later support this f
77
78      // Multiply the result by 1000000000000000000 to remove decimals
79
80
81
82
83
84  }
85
86  /**
87   * @notice Receives the response in the form of uint256
88   *
89   * @param _requestId - id of the request
90   * @param _volume - response; requested 24h trading volume of ETH in USD
91   */
92  function fulfill(bytes32 _requestId, uint256 _volume)
93  public
94  recordChainlinkFulfillment(_requestId)
95  {
96      volume = _volume;
97      emit DataFullfilled(volume);
98  }
99
100  /**
101   * @notice Withdraws LINK from the contract
102   * @dev Implement a withdraw function to avoid locking your LINK in the contract
103   */
104  function withdrawLink() external {}
105  }
106
```

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.7;
3
4  import "@chainlink/contracts/src/v0.8/interfaces/KeeperCompatibleInterface.sol";
5
6  /**
7   * @title The Counter contract
8   * @notice A keeper-compatible contract that increments counter variable at fixed time intervals
9   */
10 contract KeepersCounter is KeeperCompatibleInterface {
11     /**
12      * Public counter variable
13      */
14     uint256 public counter;
15
16     /**
17      * Use an interval in seconds and a timestamp to slow execution of Upkeep
18      */
19     uint256 public immutable interval;
20     uint256 public lastTimeStamp;
21
22     /**
23      * @notice Executes once when a contract is created to initialize state variables
24      *
25      * @param updateInterval - Period of time between two counter increments expressed as UNIX timestamp

```

Home APIConsumer.sol KeepersCounter.sol X

```

26     */
27     constructor(uint256 updateInterval) {
28         interval = updateInterval;
29         lastTimeStamp = block.timestamp;
30
31         counter = 0;
32     }
33
34     /**
35      * @notice Checks if the contract requires work to be done
36      */
37     function checkUpkeep(
38         bytes memory /* checkData */
39     )
40     public
41     override
42     returns (
43         bool upkeepNeeded,
44         bytes memory /* performData */
45     )
46     {
47         upkeepNeeded = (block.timestamp - lastTimeStamp) > interval;
48         // We don't use the checkData in this example. The checkData is defined when the Upkeep was regist
49     }

```

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
3
4 import "../KeepersCounter.sol";
5
6 contract KeepersCounterEchidnaTest is KeepersCounter {
7     constructor() KeepersCounter(8 days) {}
8
9     function echidna_test_perform_upkeep_gate() public view returns (bool) {
10         return counter == 0;
11     }
12 }
13
```