

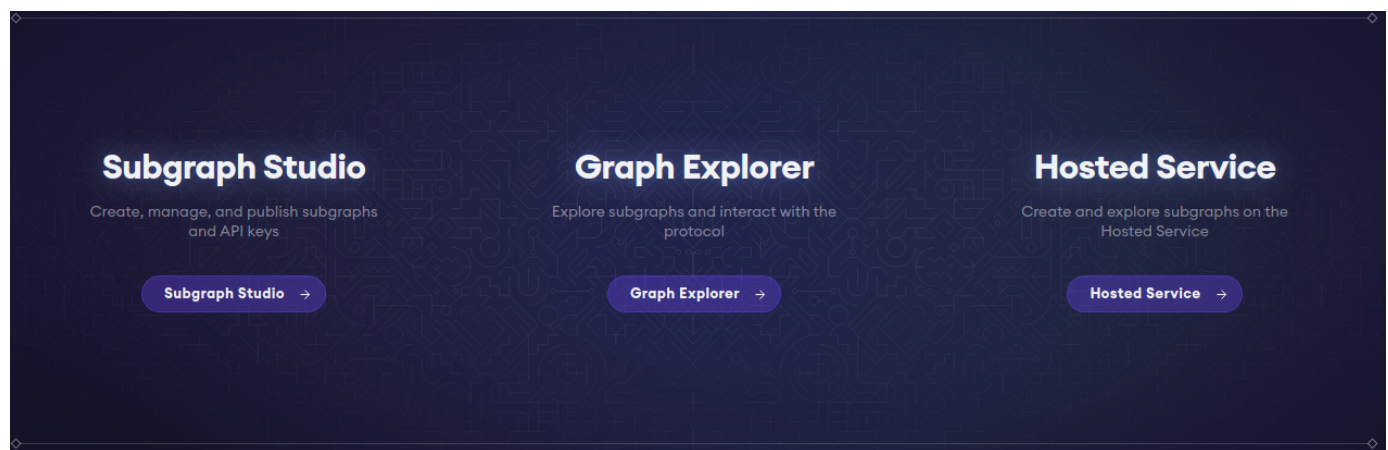
# DeFi data - Introduction to The Graph and Chainlink

## The Graph


The Graph is an indexing protocol for querying networks like Ethereum and IPFS. Anyone can build and publish open APIs, called subgraphs, making data easily accessible.

See The Graph Academy (<https://docs.thegraph.academy/>)

## Products



## Hosted Service

COMPOUND-FINANCE

# Compound V2

🔖 111

• Syncing (100%)

14.0M / 14.0M blocks

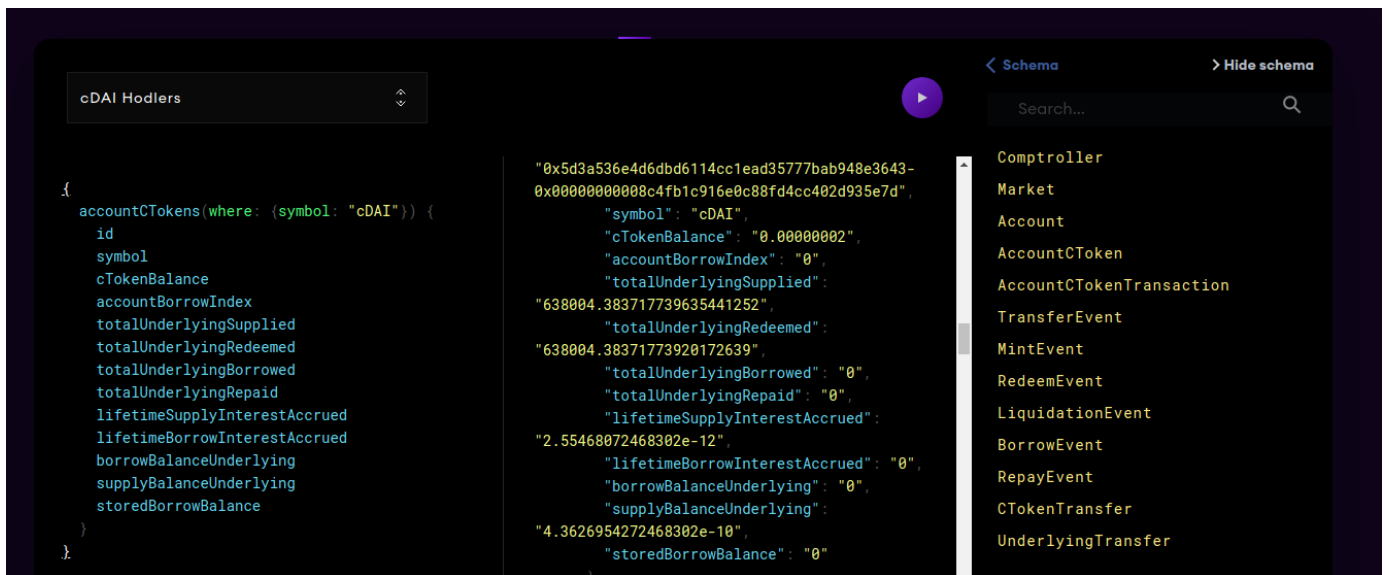
Compound is an open-source protocol for algorithmic, efficient Money Markets on the Ethereum blockchain.

Network	Last updated	Created	Entities
mainnet	2 months ago	2 years ago	6,372,155

Github  
<https://github.com/graphprotocol/compound-v2-subgraph>

ID  
QmfKmZ7xft9PRydVGJRRWEdr7rbBcRoKbDyX7fGK7MCq5j

Queries (HTTP)  
<https://api.thegraph.com/subgraphs/name/graphprotocol/compound-v2>

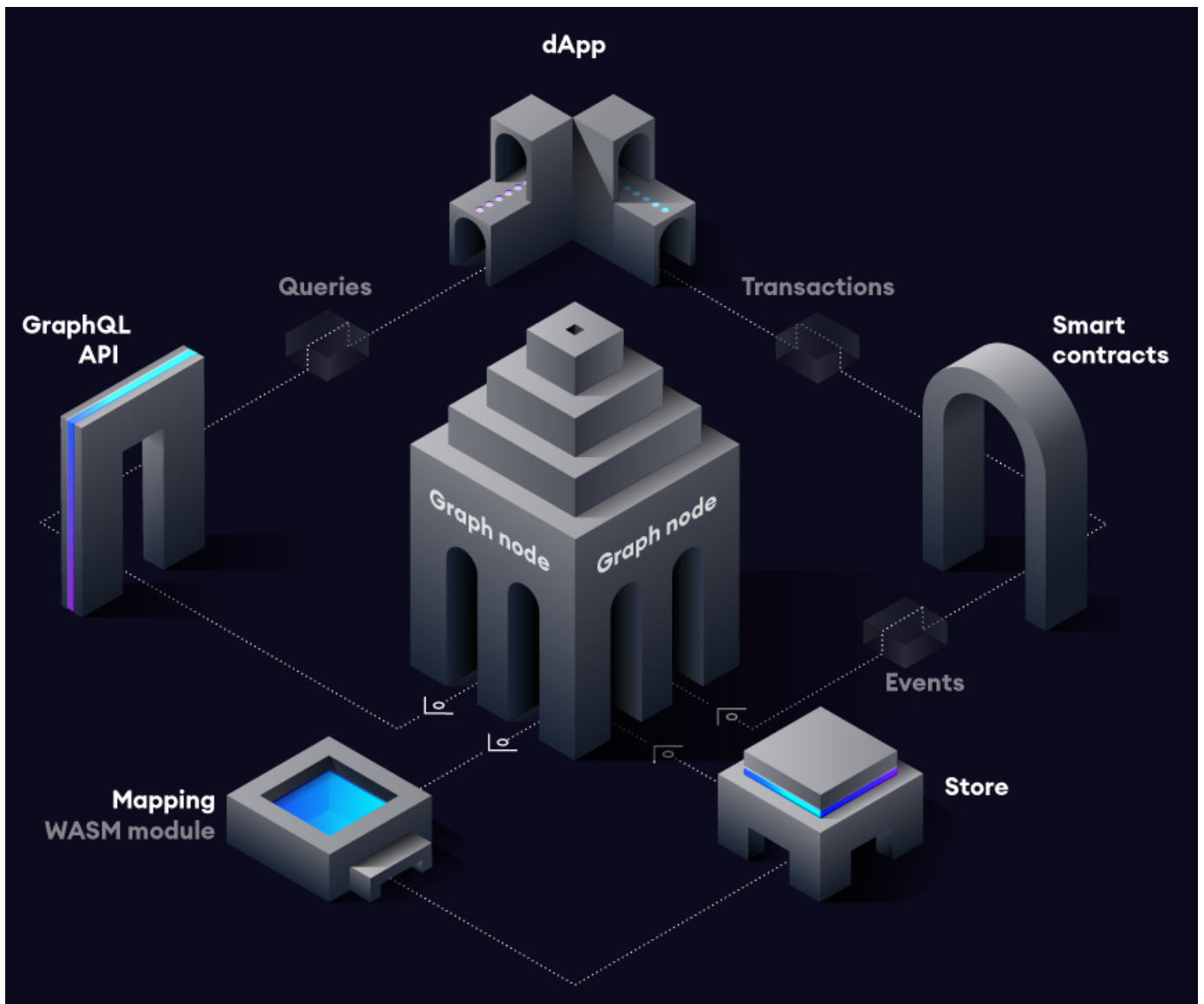


The Graph learns what and how to index Ethereum data based on subgraph descriptions, known as the subgraph manifest. The subgraph description defines the smart contracts of interest for a subgraph, the events in those contracts to pay attention to, and how to map event data to data that The Graph will store in its database.

Once you have written a subgraph manifest, you use the Graph CLI to store the definition in IPFS and tell the indexer to start indexing data for that subgraph.

This diagram gives more detail about the flow of data once a subgraph manifest has been deployed, dealing with Ethereum transactions:

## Data Flow



1. A decentralized application adds data to Ethereum through a transaction on a smart contract.
2. The smart contract emits one or more events while processing the transaction.
3. Graph Node continually scans Ethereum for new blocks and the data for your subgraph they may contain.
4. Graph Node finds Ethereum events for your subgraph in these blocks and runs the mapping handlers you provided. The mapping is a WASM module that creates or updates the data entities that Graph Node stores in response to Ethereum events.
5. The decentralized application queries the Graph Node for data indexed from the blockchain, using the node's GraphQL endpoint. The Graph Node in turn translates the GraphQL queries into queries for its underlying data store in order to fetch this data, making use of the store's indexing capabilities. The decentralized application displays this data in a rich UI for end-users, which they use to issue new transactions on Ethereum. The cycle repeats.

## Roles

### 1. Indexer (<https://thegraph.com/docs/en/indexing/>)

Indexers operate nodes and index and serve data through open APIs. They are required to provide a stake and earn rewards for their services.

The indexer's node connects to

- an Ethereum endpoint
- a postgres database
- IPFS

### **Proof of Indexing**

POIs are used in the network to verify that an indexer is indexing the subgraphs they have allocated on. A POI for the first block of the current epoch must be submitted when closing an allocation for that allocation to be eligible for indexing rewards.

Disputes are possible during a certain time window. Network participants that open disputes are called Fishermen, they are required to give a deposit of 10000 GRT.

After the dispute is resolved the indexer may get their stake slashed, and the Fisherman may lose their deposit.

### 2. Delegator (<https://thegraph.com/docs/en/delegating/>)

Delegators delegate their tokens to an indexer, in return they earn a proportion of the query fees.

### 3. Curator (<https://thegraph.com/docs/en/curating/>)

For end consumers to be able to query a subgraph, the subgraph must first be indexed. Indexing is a process where files, data, and metadata are looked at, cataloged, and then indexed so that results can be found faster. In order for a subgraph's data to be searchable, it needs to be organized.

And so, if Indexers had to guess which subgraphs they should index, there would be a low chance that they would earn robust query fees because they'd have no way of validating which subgraphs are good quality.

Curators assess the web3 ecosystem and signal on the subgraphs that should be indexed by The Graph Network.

Curators make The Graph network efficient and signaling is the process that curators use to let Indexers know that a subgraph is good to index.

### 4. Developer

Create and publish subgraphs.

## **Defining a subgraph**

We can use Subgraph studio and the CLI to define a subgraph. You need to create an account (connect to a metamask account)

Subgraph studio allows us to create the metadata for the subgraph and gives the steps to

use the CLI to define the subgraph.

The CLI will step through the details to allow you create a subgraph from an existing contract (fetching the API from etherscan or falling back to a local ABI file).

This will create some configuration files :

### **subgraph.yaml**

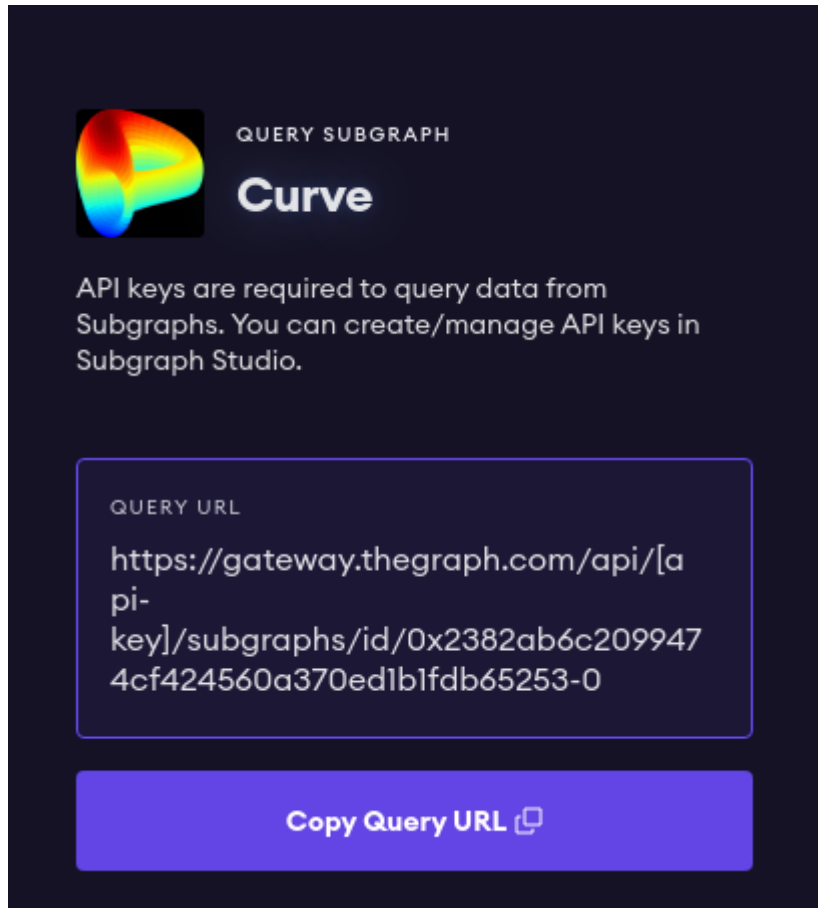
```
specVersion: 0.0.2
schema:
  file: ./schema.graphql
dataSources:
  - kind: ethereum
    name: VolcanoCoin
    network: rinkeby
    source:
      address: "0xe6190da9364067790EF1F402CC78D5CA5DF5D0be"
      abi: VolcanoCoin
    mapping:
      kind: ethereum/events
      apiVersion: 0.0.5
      language: wasm/assemblyscript
      entities:
        - Transfer
        - supplyChanged
      abis:
        - name: VolcanoCoin
          file: ./abis/VolcanoCoin.json
      eventHandlers:
        - event: Transfer(indexed address,uint256)
          handler: handleTransfer
        - event: supplyChanged(uint256)
          handler: handlesupplyChanged
      file: ./src/mapping.ts
```

### **schema.graphql**

```
type ExampleEntity @entity {
  id: ID!
  count: BigInt!
  param0: Bytes! # address
  param1: BigInt! # uint256
}
```

## **Querying the Data**

Graph Explorer provides endpoints




**QUERY SUBGRAPH**

**Curve**

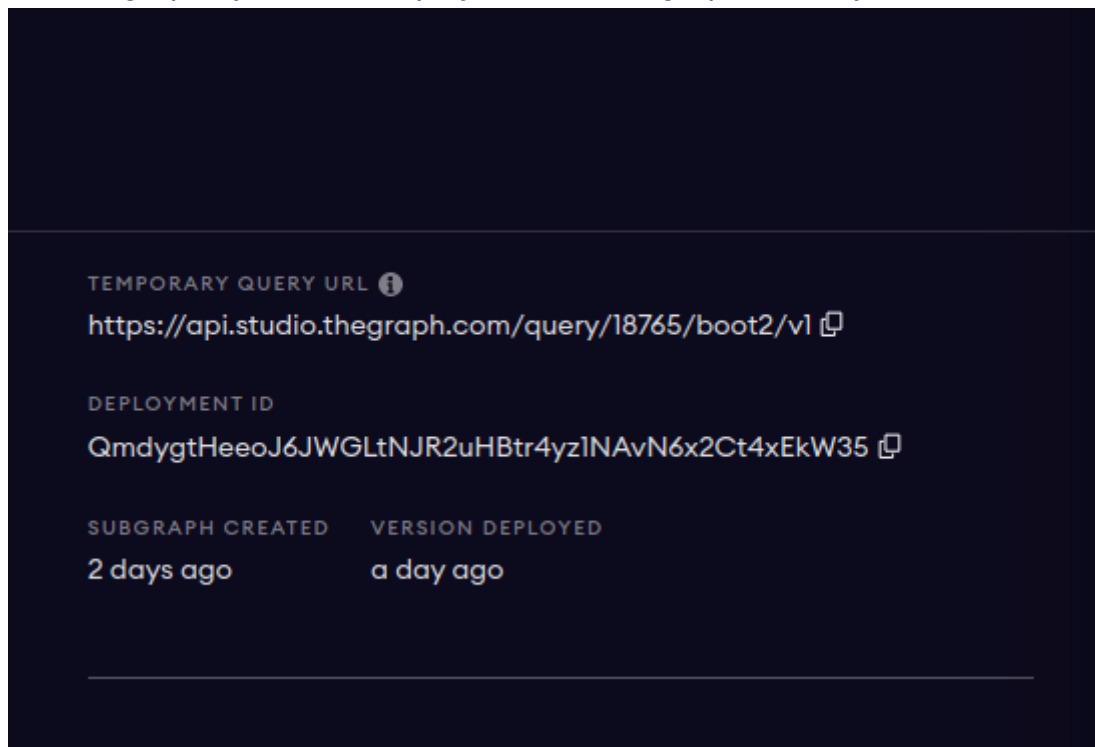
API keys are required to query data from Subgraphs. You can create/manage API keys in Subgraph Studio.


**QUERY URL**


```
https://gateway.thegraph.com/api/[api-key]/subgraphs/id/0x2382ab6c2099474cf424560a370ed1b1fdb65253-0
```

**Copy Query URL** 


For subgraphs you have deployed from Subgraph studio you can also see the endpoint



**TEMPORARY QUERY URL** 

```
https://api.studio.thegraph.com/query/18765/boot2/v1 
```

**DEPLOYMENT ID**

```
QmdygtHeeoJ6JWGLtNJR2uHBtr4yz1NAvN6x2Ct4xEkW35 
```

SUBGRAPH CREATED	VERSION DEPLOYED
2 days ago	a day ago

## UI Integration









Apollo Client (<https://www.apollographql.com/docs/react/>) is a Javascript library that allows you to integrate GraphQL queries into your UI.

# Oracles - Chainlink

## Products

### Products

Chainlink offers several products on blockchains and layer-2 networks. Select a product to get started.

 <b>Data Feeds</b> Get data and asset prices in your Smart Contracts. 	 <b>Verifiable Random Numbers (VRF)</b> Use Chainlink VRF to consume randomness in your smart contracts. 	 <b>Call External APIs</b> Request & Receive data from any API using the Chainlink contract library. 	 <b>Chainlink Keepers</b> Automation for smart contract functions and maintenance. 
---	--	---	--

## Chainlink Architecture

### Cross-chain enabled use cases



#### Cross-chain yield harvesting

Optimized yield harvesting by farming and aggregating across multiple blockchain networks that provide the highest returns on assets.



#### Cross-chain collateralized loans

Collateral can be deposited in a smart contract on a source blockchain, while allowing users to borrow a token on a higher-throughput destination chain.



#### Low-cost transaction computation

Transaction data can be processed on a high-throughput chain where the results of the computation are bridged to a higher-cost trusted chain for settlement.



#### New kinds of DeFi applications

Cross-chain services open-up a new category of DeFi applications that can be built by developers for the multi-chain ecosystem.

## Decentralised Data Model - Data Aggregation

See Documentation (<https://docs.chain.link/docs/architecture-decentralized-model/>)

Each data feed is built and funded by the community of users who rely on accurate, up-to-date data in their smart contracts. As more users rely on and contribute to a data feed, the quality of the data feed improves. For this reason, each data feed has its own properties depending on the needs of its community of users.

Each data feed is updated by a decentralized oracle network. Each oracle operator is rewarded for publishing data. The number of oracles contributing to each feed varies. In order for an update to take place, the data feed contract must receive responses from a minimum number of oracles or the latest answer will not be updated. You can see the

minimum number of oracles for the corresponding feed ()

Each oracle in the set publishes data during an aggregation round. That data is validated and aggregated by a smart contract, which forms the feed's latest and trusted answer.

### **Deviation Threshold**

A new aggregation round starts when a node identifies that the off-chain values deviate by more than the defined deviation threshold from the on-chain value. Individual nodes monitor one or more data providers for each feed.

### **Heartbeat Threshold**

A new aggregation round starts after a specified amount of time from the last update.

### **Contracts involved**

#### **Consumer**

Consumer contracts simply reference the correct AggregatorV3Interface and call one of the exposed functions.

#### **Proxy**

Proxy contracts are on-chain proxies that store the most up-to-date Aggregator for a particular data feed.

This creates a level of indirection allowing aggregators to be upgraded

### **Aggregators**

Aggregators are the contracts that receive periodic data updates from multiple Oracles. They aggregate and store data on-chain so that consumers can retrieve it and act upon it within the same transaction.

### **Example Consumer Contract**



```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract PriceConsumerV3 {

    AggregatorV3Interface internal priceFeed;

    /**
     * Network: Kovan
     * Aggregator: ETH/USD
     * Address: 0x9326BFA02ADD2366b30bacB125260Af641031331
     */
    constructor() {
        priceFeed =
            AggregatorV3Interface(0x9326BFA02ADD2366b30bacB125260Af641031331);
    }

    /**
     * Returns the latest price
     */
    function getLatestPrice() public view returns (int) {
        (
            uint80 roundID,
            int price,
            uint startedAt,
            uint timeStamp,
            uint80 answeredInRound
        ) = priceFeed.latestRoundData();
        return price;
    }
}

```

## Chainlink client

### Finding existing jobs

You will want to find existing jobs (<https://docs.chain.link/docs/listing-services/>)

From for example the chainlink market (<https://market.link/>)

The screenshot shows the Chainlink Market website. The header is dark blue with the Chainlink logo and navigation links: Home, Feeds, Nodes, Adapters, Jobs, Metrics, and Data Providers. There are also links for Sign in and Settings. The main section is titled "Data Provider Nodes" and includes a paragraph explaining that independent data providers leverage the Chainlink network to make their data accessible on-chain directly through their own Chainlink nodes. Below this are two buttons: "Become a Data Provider" and "Request New Data". The section "Latest Data Provider Nodes" features six cards, each with a logo, a title, and a description:

- Metaverse Valuation Node**: Metaverse LAND Valuations. MetaGameHub DAO makes NFTs more accessible. This is achieved through the implementation of an AI valuation algorithm that estimates the fair price of NFTs, primarily LANDs from The Sandbox and Decentraland. The valuations are brought on-chain enabling new ways of collateralizing NFTs.
- AccuWeather**: Forecast and Historical Weather Data. AccuWeather combines its premier global weather data with a history of innovation and meteorological expert insight to improve lives and businesses.
- Solipay**: Authorized User Data. Solipay believes that every person should get a say in the sale of their personal data.
- CRD Network**: KYC-as-a-Service. CRD Network bridges DeFi and CeFi by offering KYC, crypto custody, and banking services.
- The Associated Press**: Economic Data, Election Data, & News. The Associated Press is an independent global news operation dedicated to factual reporting.
- Venrai**: OFAC Sanctions Compliance Solutions. Venrai is a leading digital asset compliance solutions firm that delivers key OFAC sanctions data on-chain to DeFi platforms.

## Basic Process for a GET request

Inherit from the basic contract

(<https://github.com/smartcontractkit/chainlink/blob/master/contracts/src/v0.6/ChainlinkClient.sol>)

This includes a struct called Chainlink.Request containing

- The oracle address
- The job id
- The fee
- Adapter parameters
- The callback function signature

## Example Contract

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/ChainlinkClient.sol";

/**
 * Request testnet LINK and ETH here: https://faucets.chain.link/
 * Find information on LINK Token Contracts
 * and get the latest ETH and LINK faucets
 * here: https://docs.chain.link/docs/link-token-contracts/
 */

/**
 * THIS IS AN EXAMPLE CONTRACT WHICH USES HARDCODED VALUES FOR CLARITY.
 * PLEASE DO NOT USE THIS CODE IN PRODUCTION.
 */
contract APIConsumer is ChainlinkClient {
    using Chainlink for Chainlink.Request;

    uint256 public volume;

    address private oracle;
    bytes32 private jobId;
    uint256 private fee;

    /**
     * Network: Kovan
     * Oracle: 0xc57B33452b4F7BB189bB5AfaE9cc4aBa1f7a4FD8 (Chainlink Devrel
     * Node)
     * Job ID: d5270d1c311941d0b08bead21fea7747
     * Fee: 0.1 LINK
     */
    constructor() {
        setPublicChainlinkToken();
        oracle = 0xc57B33452b4F7BB189bB5AfaE9cc4aBa1f7a4FD8;
        jobId = "d5270d1c311941d0b08bead21fea7747";
        fee = 0.1 * 10 ** 18; // (Varies by network and job)
    }

    /**
     * Create a Chainlink request to retrieve API response, find the target
     * data, then multiply by 1000000000000000000
     * (to remove decimal places from data).
     */
    function requestVolumeData() public returns (bytes32 requestId)
    {
        Chainlink.Request memory request
        = buildChainlinkRequest(jobId, address(this),
            this.fulfill.selector);

        // Set the URL to perform the GET request on
        request.add("get",
            "https://min-api.cryptocompare.com/data/pricemultifull
            ?fsyms=ETH&tsyms=USD");
    }

```

```

// Set the path to find the desired data in the API response,
// where the response format is:
// {"RAW":
//   {"ETH":
//     {"USD":
//       {
//         "VOLUME24HOUR": xxx.xxx,
//       }
//     }
//   }
// }
request.add("path", "RAW.ETH.USD.VOLUME24HOUR");

// Multiply the result by 1000000000000000000 to remove decimals
int timesAmount = 10**18;
request.addInt("times", timesAmount);

// Sends the request
return sendChainlinkRequestTo(oracle, request, fee);
}

/**
 * Receive the response in the form of uint256
 */
function fulfill(bytes32 _requestId, uint256 _volume)
public recordChainlinkFulfillment(_requestId)
{
    volume = _volume;
}

// function withdrawLink() external {}
- Implement a withdraw function to avoid locking your LINK in the contract
}

```

API Reference (<https://docs.chain.link/docs/chainlink-framework/>)

## VRF functions

Chainlink VRF follows the Request & Receive Data cycle.

To consume randomness, your contract should inherit from `VRFConsumerBase` and define two required functions:

- `requestRandomness`, which makes the initial request for randomness.
- `fulfillRandomness`, which is the function that receives and does something with verified randomness.

## Example Random Number contract

(<https://remix.ethereum.org/#url=https://docs.chain.link/samples/VRF/RandomNumberConsumer.sol>)

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/VRFConsumerBase.sol";

/**
 * THIS IS AN EXAMPLE CONTRACT WHICH USES HARDCODED VALUES FOR CLARITY.
 * PLEASE DO NOT USE THIS CODE IN PRODUCTION.
 */

/**
 * Request testnet LINK and ETH here: https://faucets.chain.link/
 * Find information on LINK Token Contracts and get the latest ETH and LINK fau
 */

contract RandomNumberConsumer is VRFConsumerBase {

    bytes32 internal keyHash;
    uint256 internal fee;

    uint256 public randomResult;

    /**
     * Constructor inherits VRFConsumerBase
     *
     * Network: Kovan
     * Chainlink VRF Coordinator address: 0xD3782915140c8f3b190B5D67eAc6dc5760
     * LINK token address: 0xa36085F69e2889c224210F603D836748e7d
     * Key Hash: 0x6c3699283bda56ad74f6b855546325b68d482e983852a7a82979cc4807b6
     */
    constructor()
        VRFConsumerBase(
            0xD3782915140c8f3b190B5D67eAc6dc5760C46E9, // VRF Coordinator
            0xa36085F69e2889c224210F603D836748e7dC0088 // LINK Token
        )
    {
        keyHash = 0x6c3699283bda56ad74f6b855546325b68d482e983852a7a82979cc4807b
        fee = 0.1 * 10 ** 18; // 0.1 LINK (Varies by network)
    }

    /**
     * Requests randomness
     */
    function getRandomNumber() public returns (bytes32 requestId) {
        require(LINK.balanceOf(address(this)) >= fee, "Not enough LINK - fill c
        return requestRandomness(keyHash, fee);
    }

    /**
     * Callback function used by VRF Coordinator
     */
    function fulfillRandomness(bytes32 requestId, uint256 randomness) internal
        randomResult = randomness;
    }
}

```

```
// function withdrawLink() external {} - Implement a withdraw function to a
}
```

## Keepers

Chainlink Keepers provide users with a decentralized network of nodes that are incentivized to perform all registered jobs (or Upkeeps) without competing with each other.

Example Keeper contract in Remix

(<https://remix.ethereum.org/#url=https://docs.chain.link/samples/Keepers/KeepersCounter.sol>)

The contract needs to implement the KeeperCompatibleInterface

It uses the following functions

- checkUpkeep - Checks if the contract requires work to be done.
- performUpkeep - Performs the work on the contract, if instructed by checkUpkeep().

## Interoperability with Chainlink

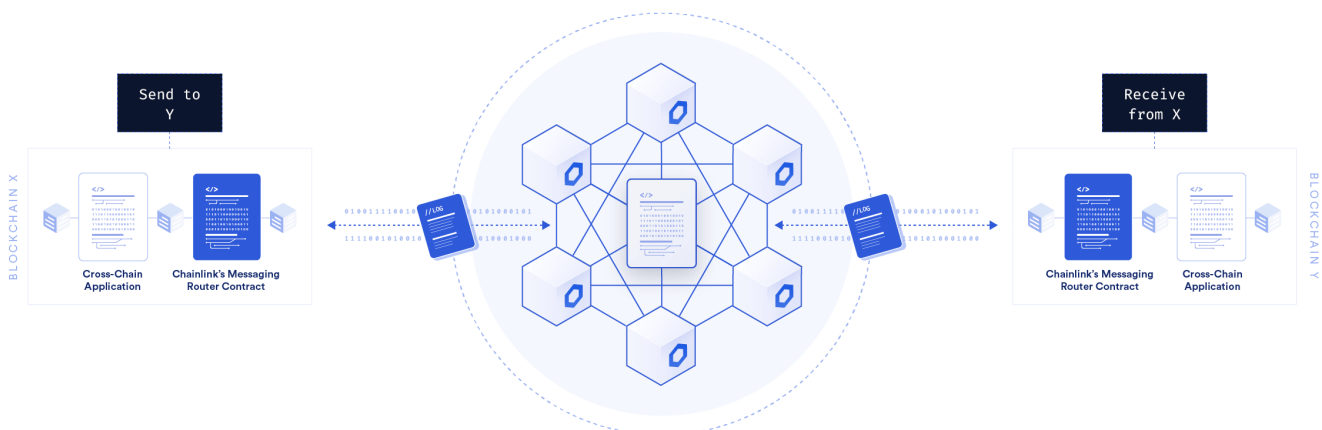
### Cross-Chain Interoperability Protocol (CCIP)

A smart contract from source chain invokes Chainlink's Messaging Router, which will leverage Chainlink DONs to securely send the message to the destination chain, where another Messaging Router validates it and sends it to the destination smart contract.

<https://chain.link/cross-chain> (<https://chain.link/cross-chain>)

<https://chain.link/cross-chain#cross-chain-interoperability-protocol> (<https://chain.link/cross-chain#cross-chain-interoperability-protocol>)

### Send messages between Chain X and Y



### Programmable Bridge Token

<https://chain.link/cross-chain#programmable-token-bridge> (<https://chain.link/cross-chain#programmable-token-bridge>)

## Cross-chain enabled use cases



### Cross-chain yield harvesting

Optimized yield harvesting by farming and aggregating across multiple blockchain networks that provide the highest returns on assets.



### Cross-chain collateralized loans

Collateral can be deposited in a smart contract on a source blockchain, while allowing users to borrow a token on a higher-throughput destination chain.



### Low-cost transaction computation

Transaction data can be processed on a high-throughput chain where the results of the computation are bridged to a higher-cost trusted chain for settlement.



### New kinds of DeFi applications

Cross-chain services open-up a new category of DeFi applications that can be built by developers for the multi-chain ecosystem.

## Chainlink Anti Fraud Network

