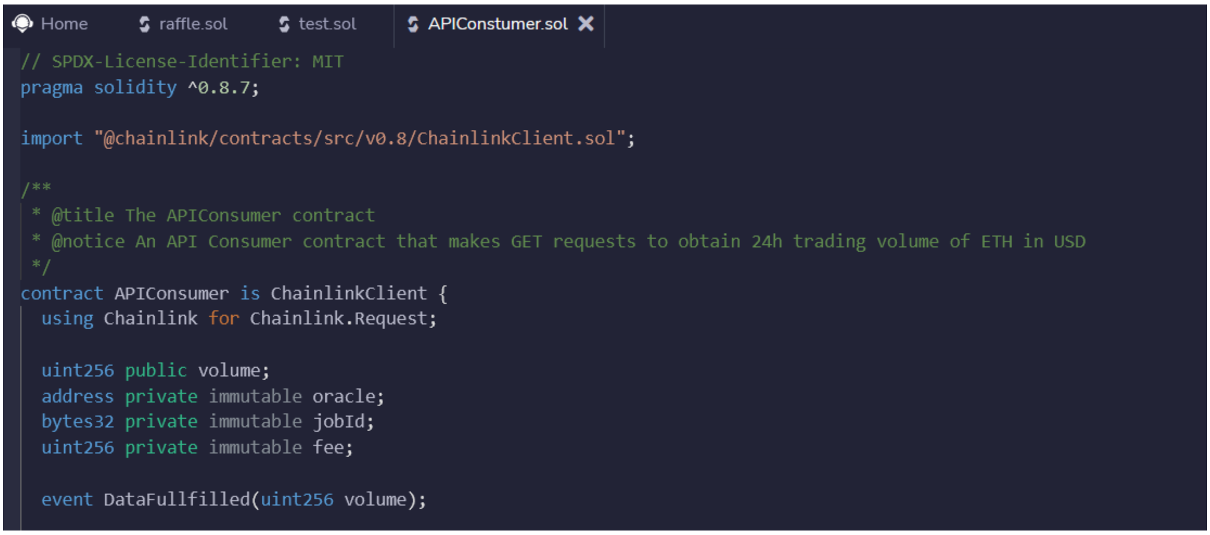
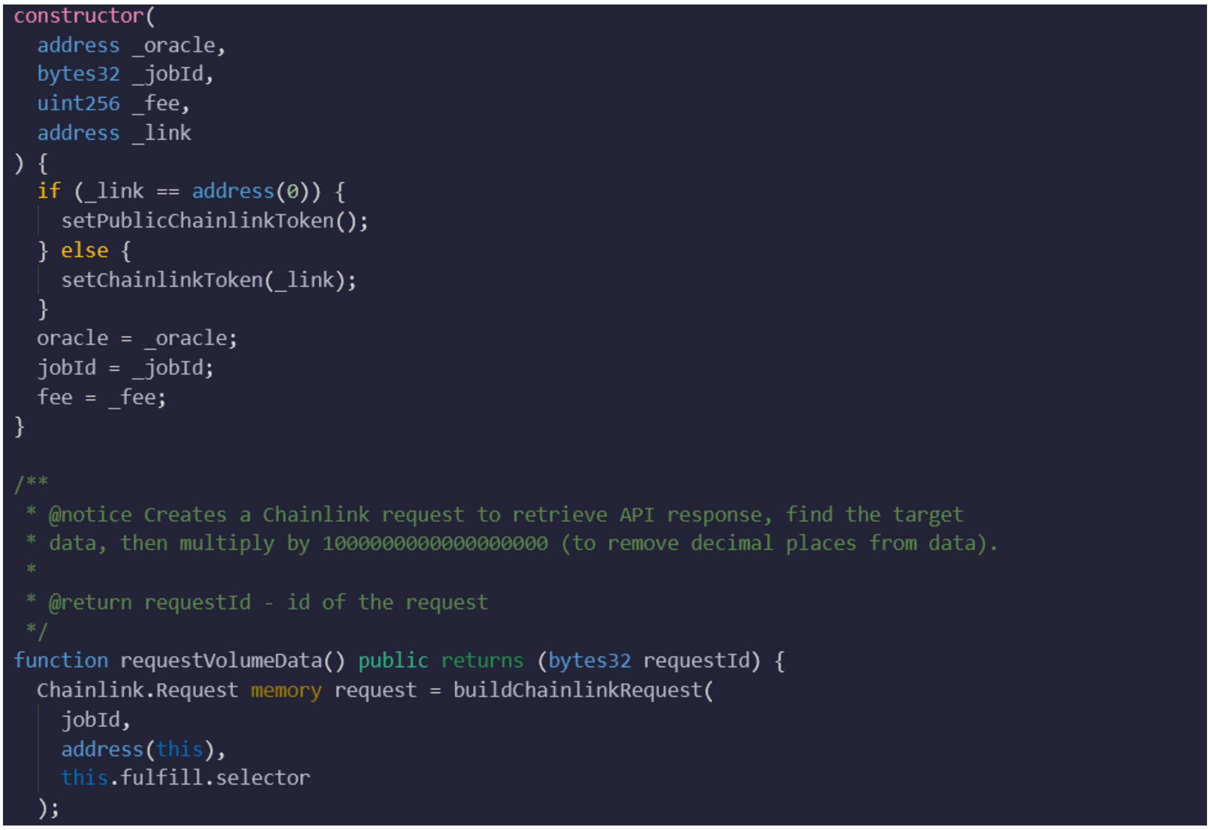
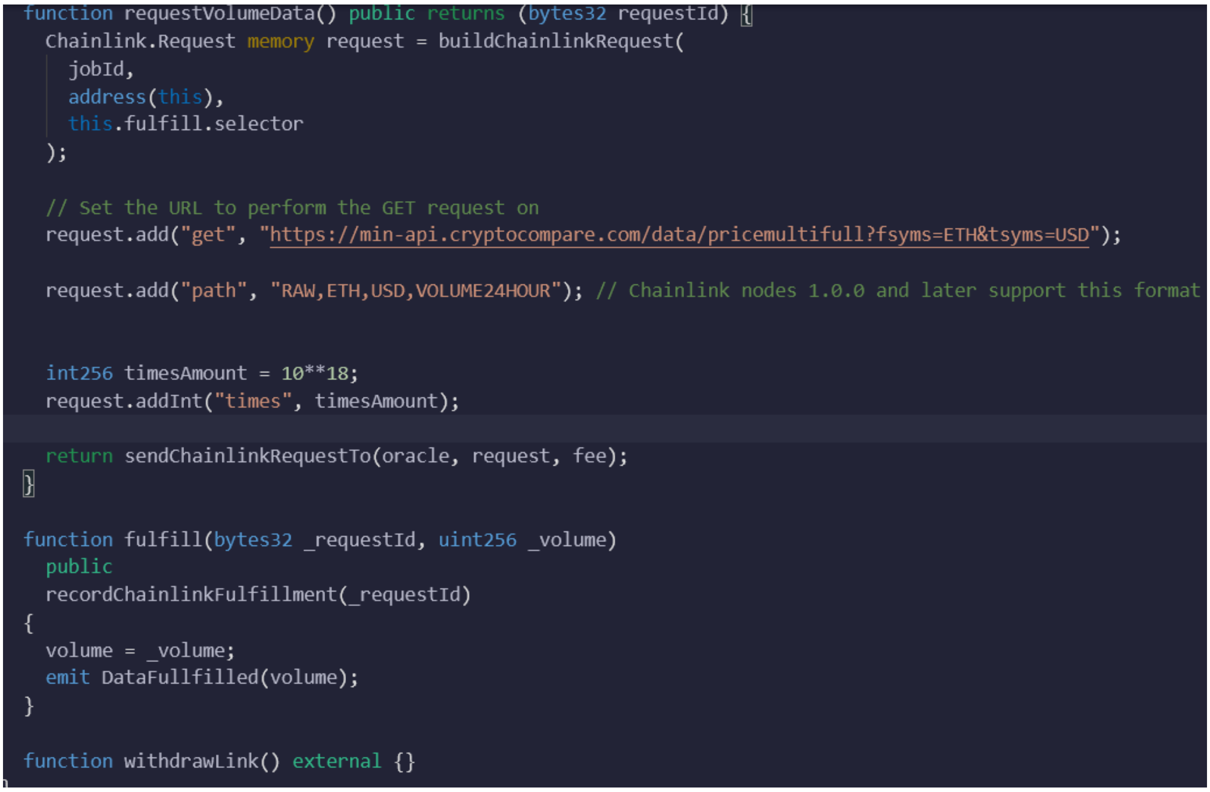


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

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

/**
 * @title The APIConsumer contract
 * @notice An API Consumer contract that makes GET requests to obtain 24h trading volume of ETH in USD
 */
contract APIConsumer is ChainlinkClient {
  using Chainlink for Chainlink.Request;

  uint256 public volume;
  address private immutable oracle;
  bytes32 private immutable jobId;
  uint256 private immutable fee;

  event DataFullfilled(uint256 volume);
```



```
constructor(
  address _oracle,
  bytes32 _jobId,
  uint256 _fee,
  address _link
) {
  if (_link == address(0)) {
    setPublicChainlinkToken();
  } else {
    setChainlinkToken(_link);
  }
  oracle = _oracle;
  jobId = _jobId;
  fee = _fee;
}

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



```
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");

  request.add("path", "RAW,ETH,USD,VOLUME24HOUR"); // Chainlink nodes 1.0.0 and later support this format


  int256 timesAmount = 10**18;
  request.addInt("times", timesAmount);

  return sendChainlinkRequestTo(oracle, request, fee);
}

function fulfill(bytes32 _requestId, uint256 _volume)
  public
  recordChainlinkFulfillment(_requestId)
{
  volume = _volume;
  emit DataFullfilled(volume);
}

function withdrawLink() external {}
```