МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТ
Кафедра ВМСС

Типовой расчет

По дисциплине «Введение в технологию блокчейн»
по теме
возможность применения оракулов в смарт-контрактах в chainlink.

Выполнил

Студент Кондратьев А.С.
Бобровский И.А.

Группа А-08м-22

Москва, 2023

Что такое оракулы?

Оракулы обеспечивают мост между реальными и сетевыми смарт-контрактами, являясь источником данных, на которые могут полагаться смарт-контракты и на которые они действуют.
Оракулы играют решающую роль в раскрытии всего потенциала полезности смарт-контрактов. Без надежной связи с реальными условиями смарт-контракты не могут эффективно служить реальному миру.
Как децентрализованная система оракулов, Chainlink может предоставлять смарт-контракты с ответами от нескольких узлов или источников. Например, смарт-контракт может получать ответы от трех разных узлов Chainlink на вопрос «Какова была цена эфира в долларах США сегодня в 15:00?» Если большинство из этих узлов честны, то легко убедиться, что A верен: просто возьмите большинство значений из ответов, которые дают узлы. Пока два из трех ответов верны (и непротиворечивы), результатом будет правильное значение A. Даже если один оракул ошибется и даст неверный ответ, A все равно будет правильным.

Как смарт-контракты используют оракулов?

Оракулы чаще всего используются с потоками данных . Платформы DeFi, такие как AAVE и Synthetix, используют оракулы потока данных Chainlink для получения точных цен на активы в реальном времени в своих смарт-контрактах.

Потоки данных Chainlink являются источниками данных, агрегированных от многих независимых операторов узлов Chainlink . Каждый поток данных имеет адрес в цепочке и функции, которые позволяют контрактам читать с этого адреса.

Смарт-контракт в цепочке контрактов Avalanche (C-Chain), которые используют оффлайн. данные цепного рынка.

Для начала подключимся к тестовой сети Avalanche. Получим тестовую валюту для этой сети на сайте https://faucet.avax.network/. Chainlink Data Feeds это самый быстрый способ связать ваши смарт-контракты с реальными данными, такими как цены на активы, остатки резервов. Когда подключается смарт-контракт к реальным сервисам или данным вне сети создается гибридный смарт-контракт. Например, использование потоков данных Chainlink для подключения своих смарт-контрактов к данным о ценах на активы, таким как поток ETH/USD . Эти потоки данных используют данные,

агрегированные от многих независимых операторов узлов Chainlink. Мы начнем с импорта необходимого контракта Chainlink для ценовых каналов, который содержит интерфейс для получения данных из существующих предварительно агрегированных децентрализованных ценовых каналов. Чтобы использовать этот интерфейс, нам нужно знать, где находятся потоки цен. Это можно найти в документации Chainlink Avalanche Feeds . Мы используем адрес для фида AVAX/USD, поэтому мы можем просто инициализировать интерфейс ценового фида с этим адресом в качестве единственного параметра при создании контракта, например:. Как раз таки следующий контракт демонстрирует эти возможности

Сетевое имя: Avalanche FUJI C-Chain

Новый URL-адрес RPC: https://api.avax-test.network/ext/bc/C/rpc.

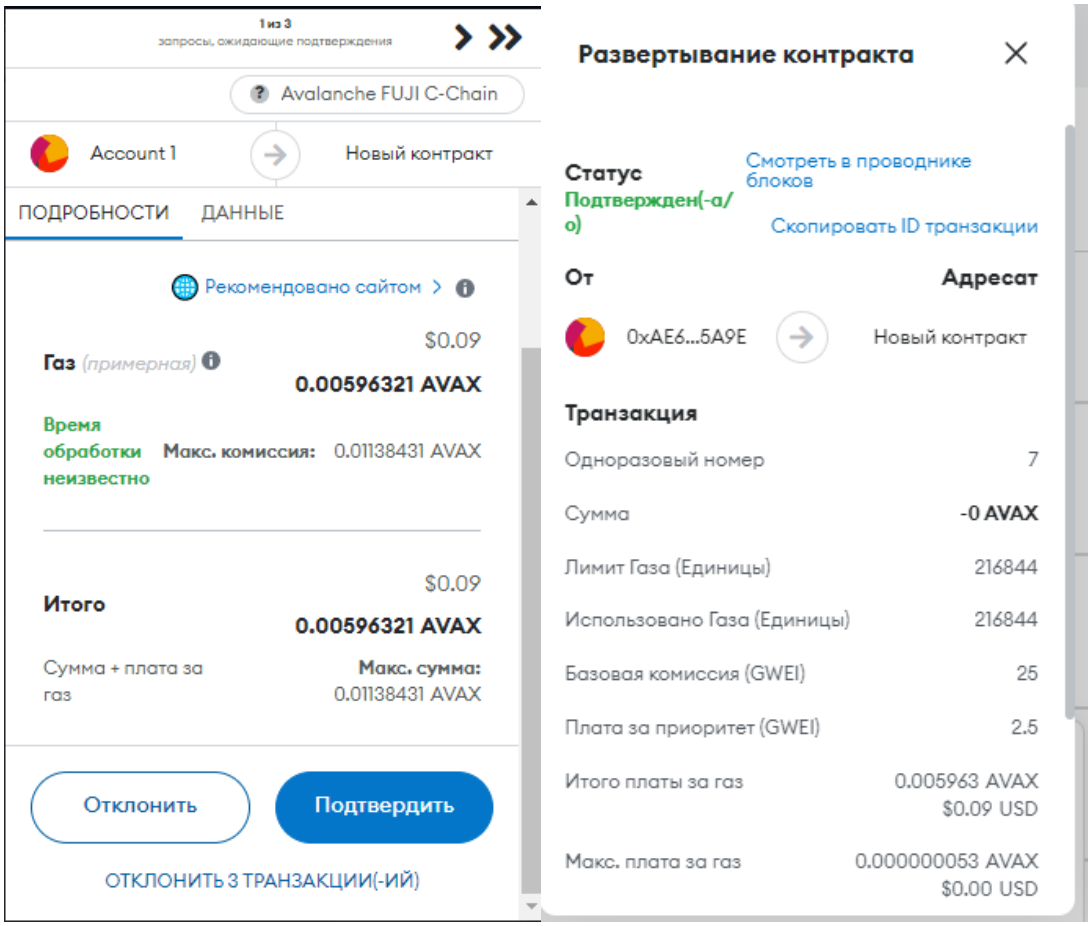
ID цепочки: 43113

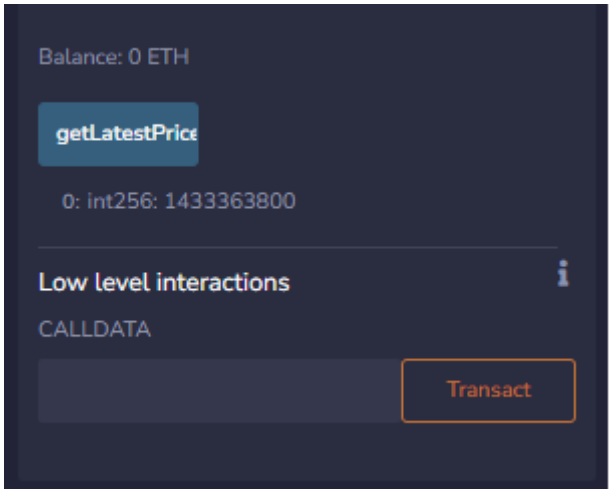
Символ: AVAXC

Проводник: https://cchain.explorer.avax-test.network

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

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

contract AvaxLinkFeeds {

    AggregatorV3Interface internal priceFeed;
    //   AVAX/USD
    //   Address: 0x5498BB86BC934c8D34FDA08E81D444153d0D06aD
    //   URL: https://docs.chain.link/docs/avalanche-price-feeds/
    constructor() {    154244 gas 129800 gas
        priceFeed = AggregatorV3Interface(0x5498BB86BC934c8D34FDA08E81D444153d0D06aD);
    }

    function getLatestPrice() public view returns (int) {    infinite gas
        (
            uint80 roundID,
            int price,
            uint startedAt,
            uint timeStamp,
            uint80 answeredInRound
        ) = priceFeed.latestRoundData();
        return price;
    }
}
```

Контракт использует внутренние фиды для получения данных. В этом примере используется курс AVAX/USD.





Ссылка на все адресы фидов https://docs.chain.link/data-feeds/price-feeds/addresses/?network=avalanche

Пример использования технологии оракул на тестовой сети Sepolia.

Код программы:

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

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

contract APIConsumer is ChainlinkClient, ConfirmedOwner {
    using Chainlink for Chainlink.Request;

    uint256 public volume;
    bytes32 private jobId;
    uint256 private fee;

    event RequestVolume(bytes32 indexed requestId, uint256 volume);

    constructor() ConfirmedOwner(msg.sender) {
        setChainlinkToken(0x779877A7B0D9E8603169DdbD7836e478b4624789);
        setChainlinkOracle(0x6090149792dAAeE9D1D568c9f9a6F6B46AA29eFD);
        jobId = "ca98366cc7314957b8c012c72f05aeeb";
        fee = (1 * LINK_DIVISIBILITY) / 10;
    }

    function requestVolumeData() public returns (bytes32 requestId) {
        Chainlink.Request memory req = buildChainlinkRequest(
            jobId,
            address(this),
            this.fulfill.selector
        );

        req.add(
            "get",
            "https://min-api.cryptocompare.com/data/price?fsym=ETH&tsyms=USD"
        );

        req.add("path", "USD");

        int256 timesAmount = 100;
        req.addInt("times", timesAmount);

        return sendChainlinkRequest(req, fee);
    }

    function fulfill(
        bytes32 _requestId,
        uint256 _volume
```

```
    ) public recordChainlinkFulfillment(_requestId) {
        emit RequestVolume(_requestId, _volume);
        volume = _volume;
    }

    function withdrawLink() public onlyOwner {
        LinkTokenInterface link = LinkTokenInterface(chainlinkTokenAddress());
        require(
            link.transfer(msg.sender, link.balanceOf(address(this))),
            "Unable to transfer"
        );
    }
}
```

Описание кода:

- requestVolumeData – Отправка запроса на оракул, в нашем случае на следующий endpoint;

```
{"USD":1826.22}
```

Рисунок 1 Ответ в формате json

- fulfill – функция, которая передается как callback, ее вызывает оракул, когда заканчивает получение и обработку данных;
- withdrawLink – функция, для снятия link с баланса контракта.

Принцип работы на тестовой сети Sepolia:

1) Деплой контракта

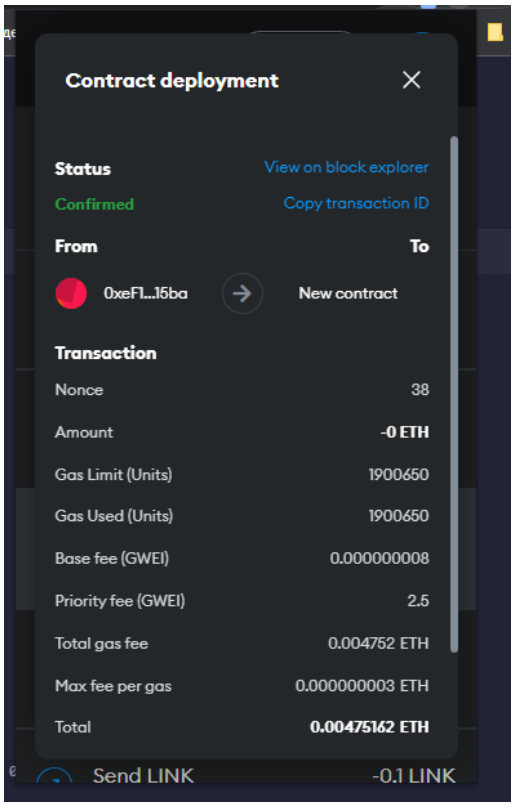


Рисунок 2 Детали деплоя контракта

2) Проверяем, что значение контракта 0:

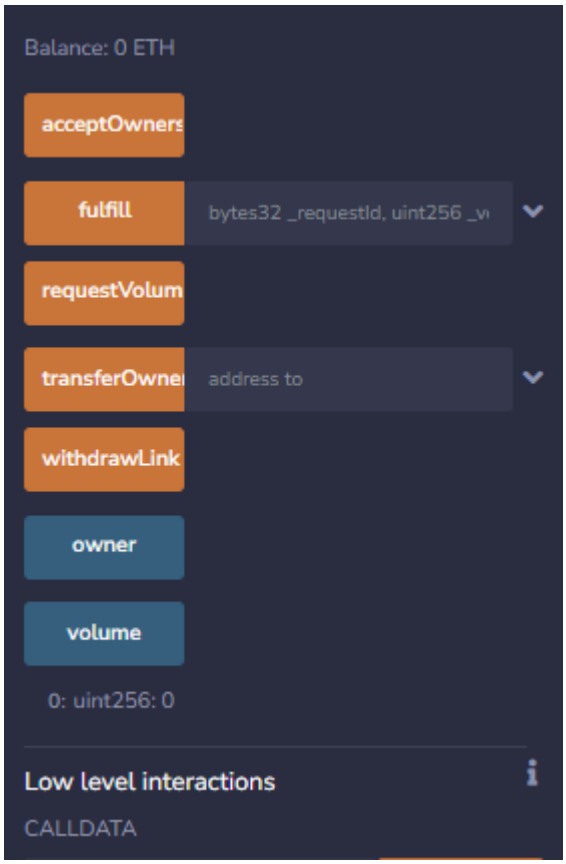


Рисунок 3 Проверка изначального значения в контракте

3) Спонсируем контракт в размере 0.1 Link

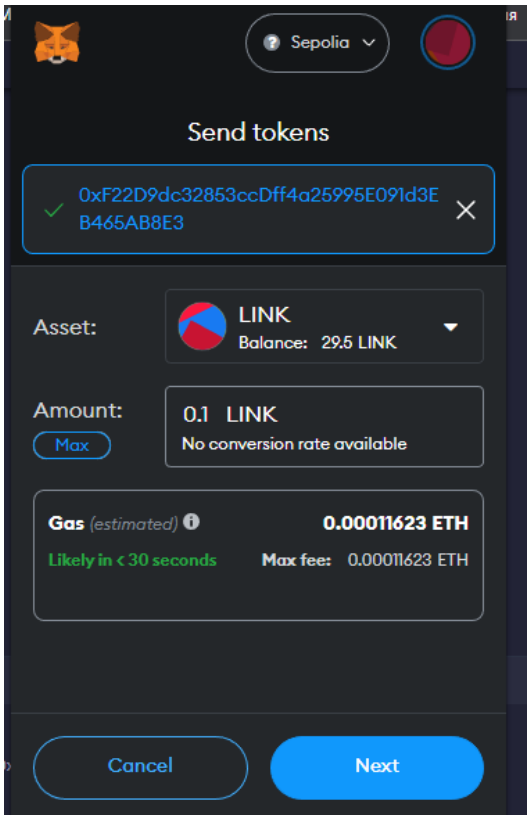


Рисунок 4 Спонсирование контракта

4) Отправляем запрос на оракул

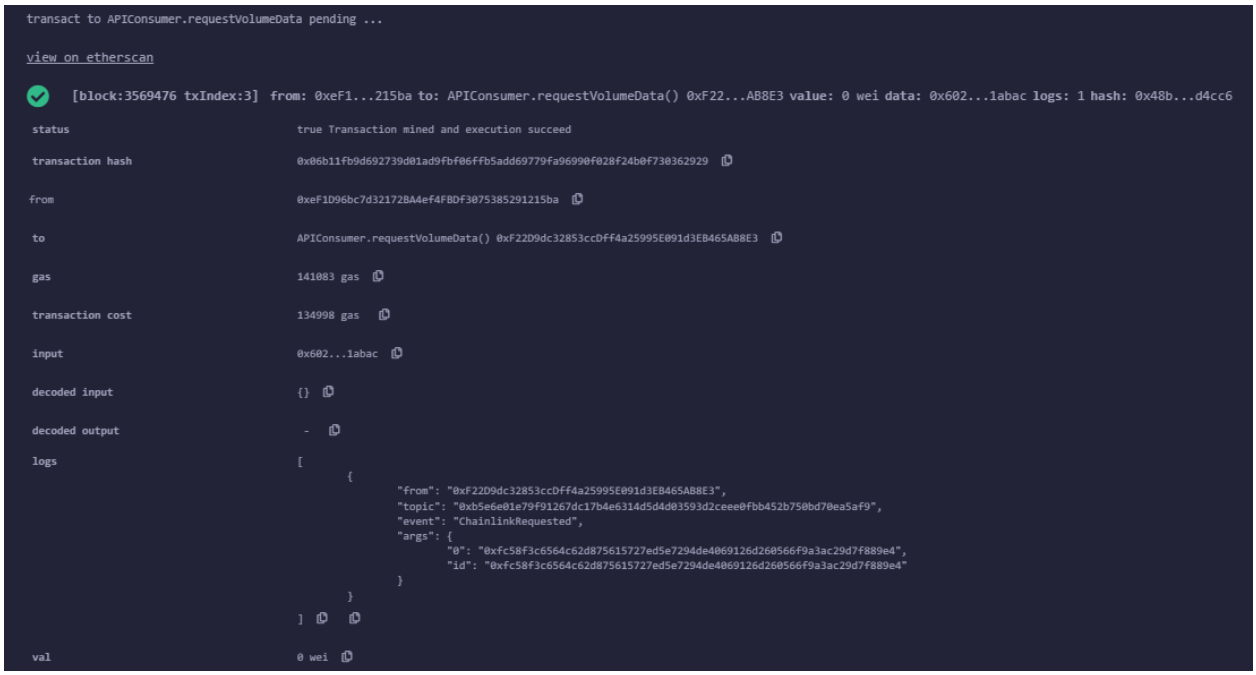


Рисунок 5 Лог транзакции запроса на оракул

5) Ждем пару секунд и проверяем, что значение изменилось

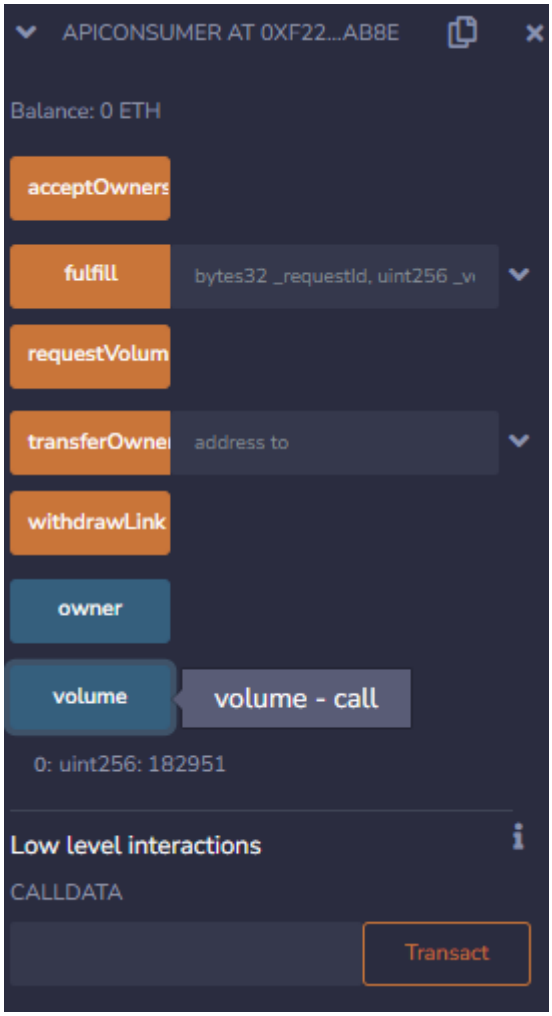


Рисунок 6 Проверка получения результата от оракула

Чтобы запустить узел Chainlink, у вас должен быть доступ к работающему узлу Ethereum с открытым подключением к веб-сокету. Любая сеть на основе Ethereum будет работать после того, как вы настроите идентификатор цепочки. Подробная инструкция содержится по адресу https://github.com/smartcontractkit/chainlink.

Список источников

- https://github.com/smartcontractkit/chainlink - github проекта в котором хранятся все библиотеки и инструменты по работе с chainlink.
- https://blog.chain.link/town-crier-and-chainlink/ - информация по поводу включения town-crier в chainlink. И вводный их обзор
- https://faucet.avax.network/ сайт для получения валюты для тестовой сети Fuji (C-chain)