













Развертывание смарт-контрактов

# Remix



DEPLOY & RUN  
TRANSACTIONS

ENVIRONMENT ⓘ

Remix VM (Cancun)

VM

ACCOUNT + ⓘ

0x5B3...eddC4 (100 ether)

GAS LIMIT

☒ Estimated Gas

☐ Custom

3000000

VALUE

0

Wei

CONTRACT

Ballot - contracts/3\_Ballot.sol

evm version: paris

Deploy

string[] proposalNames

☐ Publish to IPFS

At Address

Load contract from Address

Transactions recorded 0 ⓘ >

Pinned Contracts (network: vm-cancun)

> BALLOT AT 0X5E1...4EFF5 ⓘ ⓘ ⓧ

Deployed/Unpinned Contracts ⓘ

> STORAGE AT 0XE28...4157A (MEM ⓘ ⓘ ⓧ

> OWNER AT 0X93F...C96CC (MEMC ⓘ ⓘ ⓧ

# Web-интерфейс контракта

- **Красные** - payable-функции
- **Оранжевые** – функции, изменяющие состояние EVM,
- **Голубые** – view-функции и поля данных.
- **Светлый оттенок** - объекты, обращение к которым требует задания параметров

▼ FUND AT 0X049...A1FD3

Balance: 0.000000999999 ETH

Pinned at: 26.09.2024, 11:08:27

File path: default\_workspace/contracts/fund.sol

approve

claim uint256\_value ▼

enter

get

approvements

claimAddress

claimValue

manager

participants uint256 ▼

Low level interactions ⓘ

CALLDATA

Transact



# Ganache

- Ganache – персональный блокчейн для быстрой разработки распределенных приложений Ethereum.
- Позволяет разрабатывать, развертывать и тестировать приложения в безопасной среде.
- В программе реализованы ранние версии протокола Эфириума, использующие майнинг.
- Отсутствует возможность организации сетевого взаимодействия с другими узлами и использования каких-либо механизмов консенсуса.
- Версии Ganache доступны для Windows, Mac и Linux.




# Ganache

<https://archive.trufflesuite.com/ganache/>

При запуске можно воспользоваться быстрым стартом с базовыми настройками новой блокчейн-сети или создать кастомизированную сеть.



 Ganache

WORKSPACE


SERVER


ACCOUNTS & KEYS

CHAIN

ADVANCED

ABOUT

 CANCEL

 SAVE WORKSPACE

## WORKSPACE

WORKSPACE NAME

labored-meal

A friendly name for this workspace.

TRUFFLE PROJECTS

Link Truffle projects to this workspace by adding their `truffle-config.js` or `truffle.js` file to this workspace.

This will show useful contract and event data to better understand what's going on under the hood.

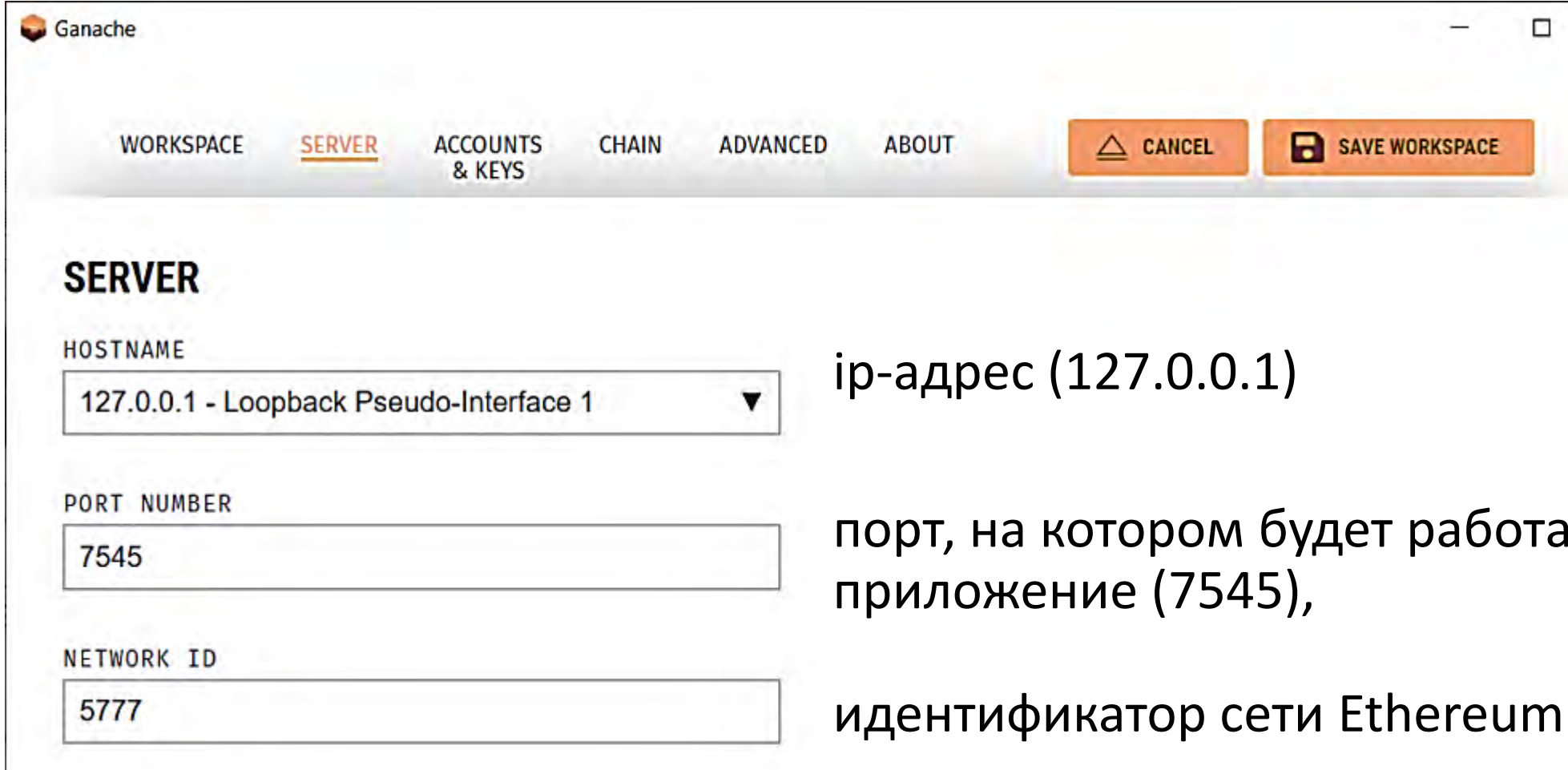
ADD PROJECT

REMOVE PROJECT

## Вкладка Workspace

Позволяет задать имя своей новой сети и подгрузить свой проект с сервисов Truffle.

# Вкладка Server



The image shows the 'SERVER' tab in the Ganache application. The window has a title bar with the Ganache logo and standard window controls. Below the title bar is a navigation bar with tabs: 'WORKSPACE', 'SERVER' (which is selected and underlined), 'ACCOUNTS & KEYS', 'CHAIN', 'ADVANCED', and 'ABOUT'. To the right of the tabs are two orange buttons: 'CANCEL' and 'SAVE WORKSPACE'. The main area of the window is titled 'SERVER' and contains three configuration fields: 'HOSTNAME' with a dropdown menu showing '127.0.0.1 - Loopback Pseudo-Interface 1', 'PORT NUMBER' with a text input containing '7545', and 'NETWORK ID' with a text input containing '5777'. To the right of the form, there are three lines of explanatory text in Russian: 'ip-адрес (127.0.0.1)', 'порт, на котором будет работать приложение (7545),', and 'идентификатор сети Ethereum (5777)'.

**SERVER**

HOSTNAME  
127.0.0.1 - Loopback Pseudo-Interface 1 ▼

PORT NUMBER  
7545

NETWORK ID  
5777

ip-адрес (127.0.0.1)

порт, на котором будет работать приложение (7545),

идентификатор сети Ethereum (5777)

# Вкладка Server

AUTOMINE



Блоки генерируются майнером при поступлении транзакций

ERROR ON TRANSACTION FAILURE



Обработка ошибок в транзакциях непосредственно в Ganache

**CHAIN FORKING**



Создание форка текущей цепочки с сохранением существующих аккаунтов, контрактов, транзакций и данных





WORKSPACE



SERVER

ACCOUNTS  
& KEYS

CHAIN

ADVANCED

ABOUT

 CANCEL SAVE WORKSPACE

## ACCOUNTS & KEYS

ACCOUNT DEFAULT BALANCE

100

TOTAL ACCOUNTS TO GENERATE

10

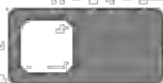
AUTOGENERATE HD MNEMONIC



vivid maximum fancy give mouse refuse pioneer couch

note: this mnemonic is not secure; don't use it on a public  
blockchain.

LOCK ACCOUNTS



Стартовый баланс пользователей

Количество аккаунтов

Генерировать мнемоническую фразу

Мнемоническая фраза, которая будет  
использоваться для создания аккаунтов

Блокировка аккаунтов



Ganache



WORKSPACE

SERVER

ACCOUNTS  
& KEYS

CHAIN

ADVANCED

ABOUT



CANCEL



SAVE WORKSPACE

## GAS

GAS LIMIT

6721975

Лимит газа для выполнения транзакций  
в блоке

GAS PRICE

20000000000

Стоимость единицы газа в wei

## HARDFORK

HARDFORK

Muir Glacier ▼

Petersburg

Constantinople

Byzantium

Istanbul

Muir Glacier

Используемая версия Ethereum

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBER

CURRENT BLOCK  
0

GAS PRICE  
20000000000

GAS LIMIT  
6721975000

HARDFORK  
ISTANBUL

NETWORK ID  
5777

RPC SERVER  
HTTP://127.0.0.1:7545

MINING STATUS  
AUTOMINING

WORKSPACE  
LABORED-MEAL

SWITCH

MNEMONIC

vivid maximum fancy give mouse refuse pioneer couch honey ivory cream illegal

HD PATH  
m/44'/60'/0'/0/account\_index

ADDRESS	BALANCE	TX COUNT	INDEX	
0xf9707ab072A6aee24aa00EC16b2A8bE1E6627Db1	100.00 ETH	0	0	
0x413CDc472A2477CB2A58149a84c135b55272d0F9	100.00 ETH	0	1	
0xAD988B986584976880681543B676A96C6DE5e8a8	100.00 ETH			
0x8198F0Ba23A121A8945B3e89987a454fd2CE9e32	100.00 ETH			

ACCOUNT INFORMATION

ACCOUNT ADDRESS

0xf9707ab072A6aee24aa00EC16b2A8bE1E6627Db1

PRIVATE KEY

d2e34728897a55420c0ae594392abebe350d8fea1d50bdb70375955b8e598383

Do not use this private key on a public blockchain; use it for development purposes only!

DONE

После создания сети доступны аккаунты в указанном количестве, их ключи, мнемоническая фраза.

Указано количество транзакций, сформированных каждым из пользователей.



Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUM

CURRENT BLOCK  
10

GAS PRICE  
2000000000

GAS LIMIT  
6721975

HARDFORK  
MERGE

NETWORK ID  
5777

RPC SERVER  
HTTP://127.0.0.1:7545

MINING STATUS  
AUTOMINING

WORKSPACE  
DIREFUL-DEER

SWITCH

BLOCK 10	MINED ON 2024-08-16 16:03:33	GAS USED 21000	1 TRANSACTION
BLOCK 9	MINED ON 2024-08-16 16:01:35	GAS USED 21000	1 TRANSACTION
BLOCK 8	MINED ON 2024-08-13 15:04:38	GAS USED 31881	1 TRANSACTION

В строке состояния  
указаны все параметры  
запущенной блокчейн  
сети: количество блоков,  
gas price, gas limit,  
hardfork, network ID, RPC  
server, имя сети.

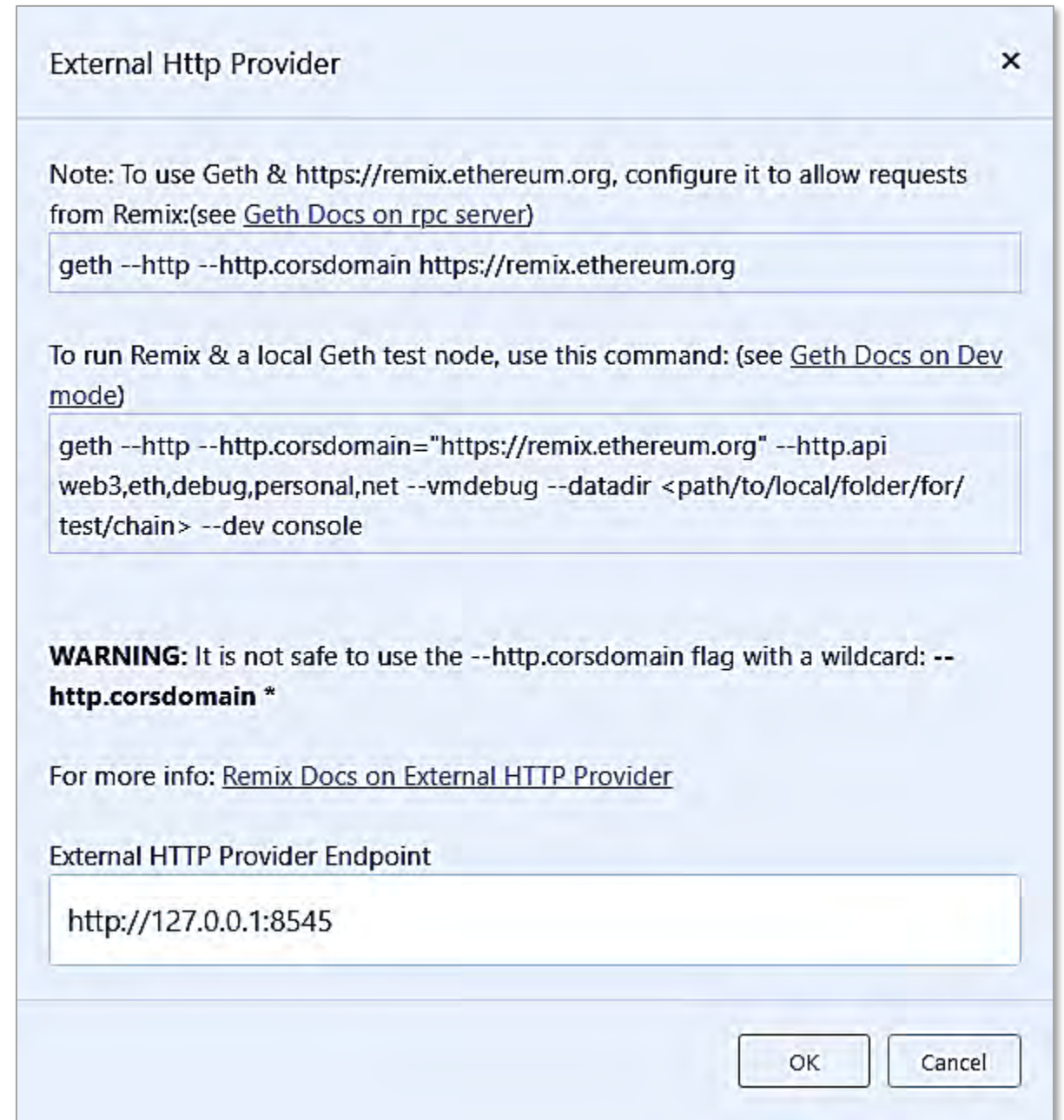
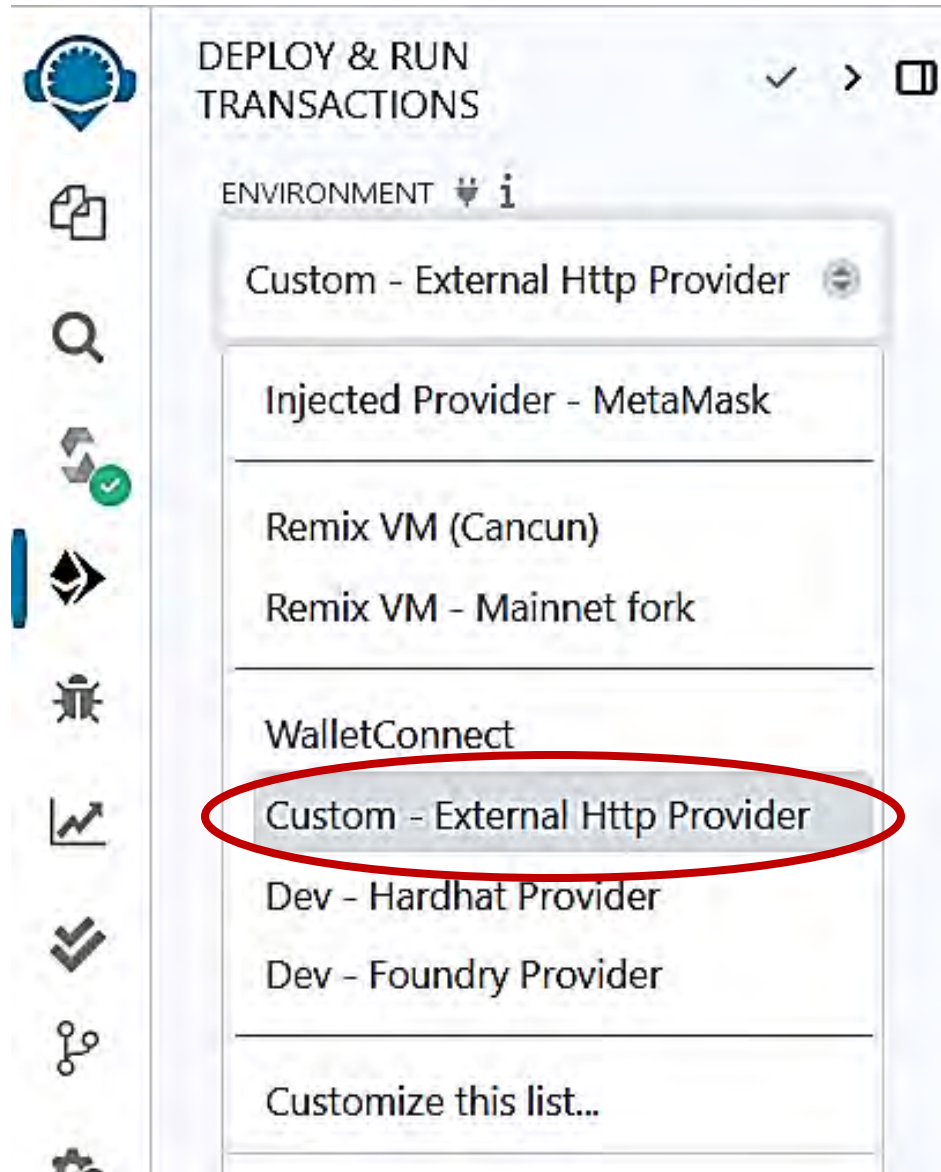
BLOCK 7	MINED ON 2024-08-13 15:04:12	← BACK	BLOCK 9
BLOCK 6	MINED ON 2024-08-13 15:03:43	<div><div>GAS USED</div><div>GAS LIMIT</div><div>MINED ON</div><div>BLOCK HASH</div></div> <div>210006721972024-08-16 16:01:350xbf7d364049cd7188276322268004a752ad8c94ee52533a7fa6294168fa1e8fed</div>	
BLOCK 5	MINED ON 2024-08-13 15:03:22	<div><div>TX HASH</div><div>FROM ADDRESS</div><div>TO ADDRESS</div><div>GAS USED</div><div>VALUE</div></div> <div>0x7f19b241f82e3694b2072af2e97f102b01fc27e7a9858a4d378ee93af9f4ba7a0xD76545Ce1498F0081472BE28EFfac0Be74a1913d0x4bccF21013dF35A0bD5aA05394cD4bE7221332cc21000100000000000000000000</div> <div>VALUE TRANSFER</div>	

Указывается вся  
информация о блоках в  
цепочке, в т.ч. genesis-  
блоке, информация о  
транзакциях.





# Интеграция с Remix



# Metamask

- Криптокошелёк Metamask – популярное клиентское приложение, позволяющее подключаться к широкому спектру блокчейн-сетей, включая тестовые сети, и создавать транзакции по переводу средств.
- Metamask доступен для мобильных ОС iOS и Android, а также для ПК на базе Windows. Приложение можно загрузить с официального сайта

<https://metamask.io/>

- Версия для ПК устанавливается как дополнение (add-on) к браузеру и позволяет взаимодействовать с приложениями, выполняемыми на сайтах, в том числе со средой программирования Remix.

**Давайте приступим к делу**  
MetaMask, которому доверяют миллионы, – это безопасный кошелек, предоставляющий всем доступ к миру web3.



☒ Я соглашаюсь с Условия  
использования MetaMask

Создать новый кошелек

Импорт существующего кошелька

1 Создать пароль 2 Безопасный кошелек 3 Подтвердите секретную фразу для восстановления

## Создать пароль

Этот пароль разблокирует ваш кошелек MetaMask только на этом устройстве. MetaMask не может восстановить этот пароль.

Новый пароль (мин. 8 знаков)

Показать

.....

Надежность пароля: **Сильный**

Подтвердить пароль



.....

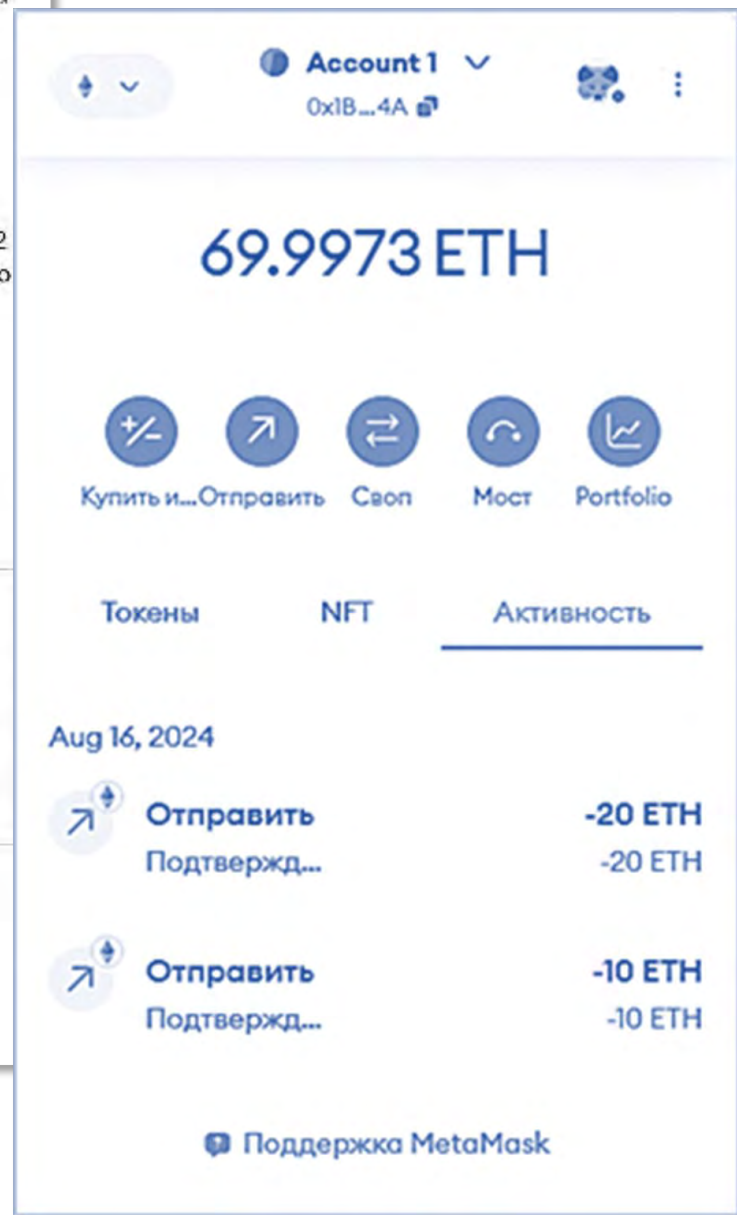
☒ Я понимаю, что MetaMask не может  
восстановить этот пароль для меня.  
[Подробнее](#)

Создать новый кошелек

# Metamask

- Для формирования мнемонической фразы и восстановления по ней закрытого ключа аккаунта Metamask использует стандарт BIP39, первоначально предложенный для Биткоин-кошельков, но впоследствии ставший распространённым методом авторизации и для других криптовалют.
- После создания и проверки мнемонической фразы Metamask автоматически подключится к основной сети Эфириума и выведет окно аккаунта, в котором отображается адрес, баланс средств и перечень проведённых транзакций

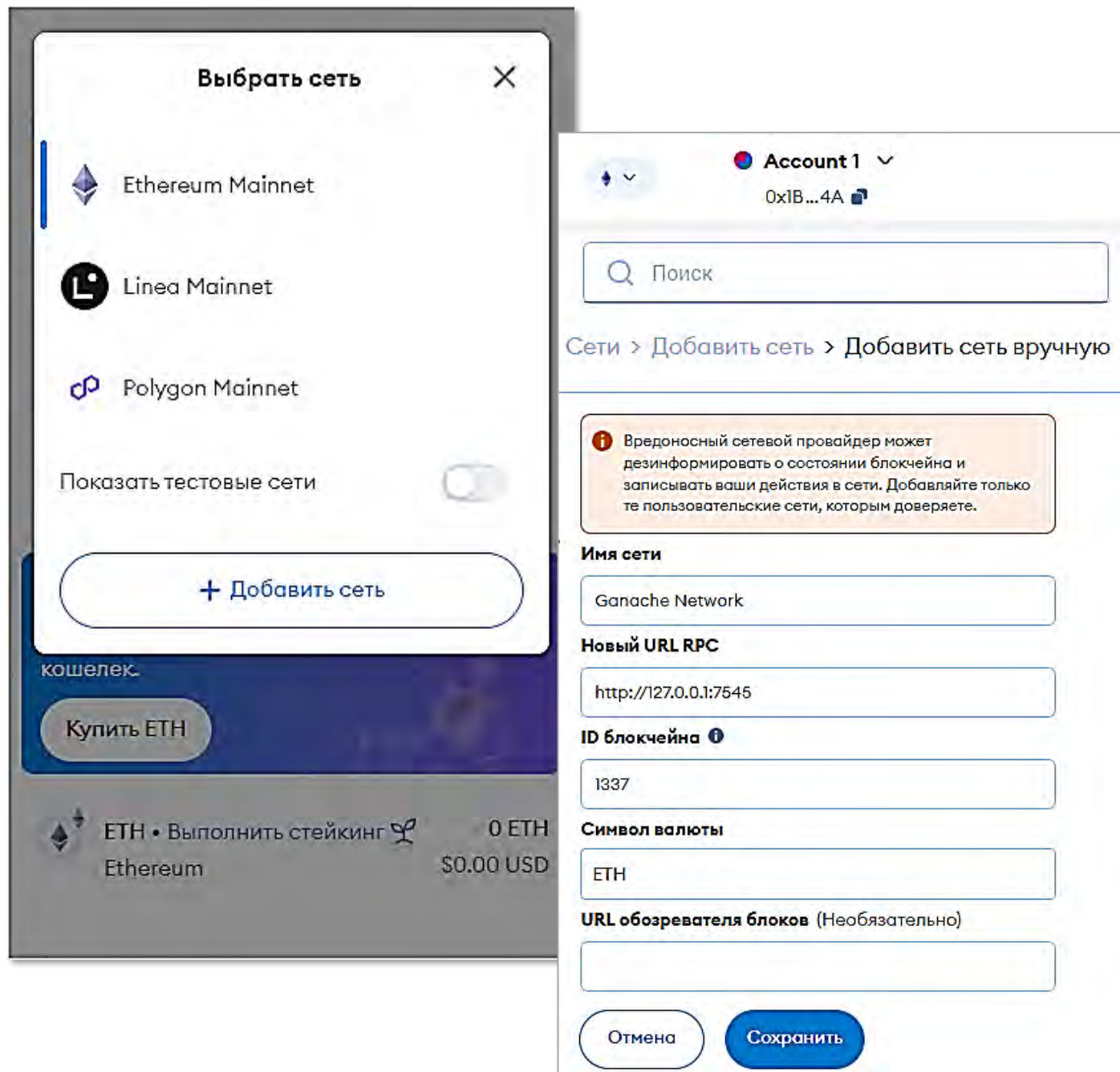
The image shows the Metamask setup screen for creating a new account. At the top, there are three numbered steps: 1. Создать пароль (Create password), 2. Безопасный кошелек (Secure wallet), and 3. Подтвердите секретную фразу для восстановления (Confirm secret phrase for recovery). The main heading is "Запишите секретную фразу для восстановления" (Write down the secret phrase for recovery). Below this, it says "Запишите эту секретную фразу для восстановления из 12 слов и сохраните ее в надежном месте, доступном только вам." (Write down this secret phrase for recovery from 12 words and save it in a safe place accessible only to you). There are three tips: "Советы: Сохранить в менеджере паролей, Храните в банковской ячейке, Запишите и храните в нескольких секретных местах." (Tips: Save in password manager, Store in a bank safe, Write down and store in several secret places). Below the tips, there are 12 input fields, each containing the word "secret". At the bottom, there are two buttons: "Скрыть сид-фразу" (Hide seed phrase) and "Скопировать в буфер обмена" (Copy to clipboard). A large blue button labeled "Далее" (Next) is at the bottom.





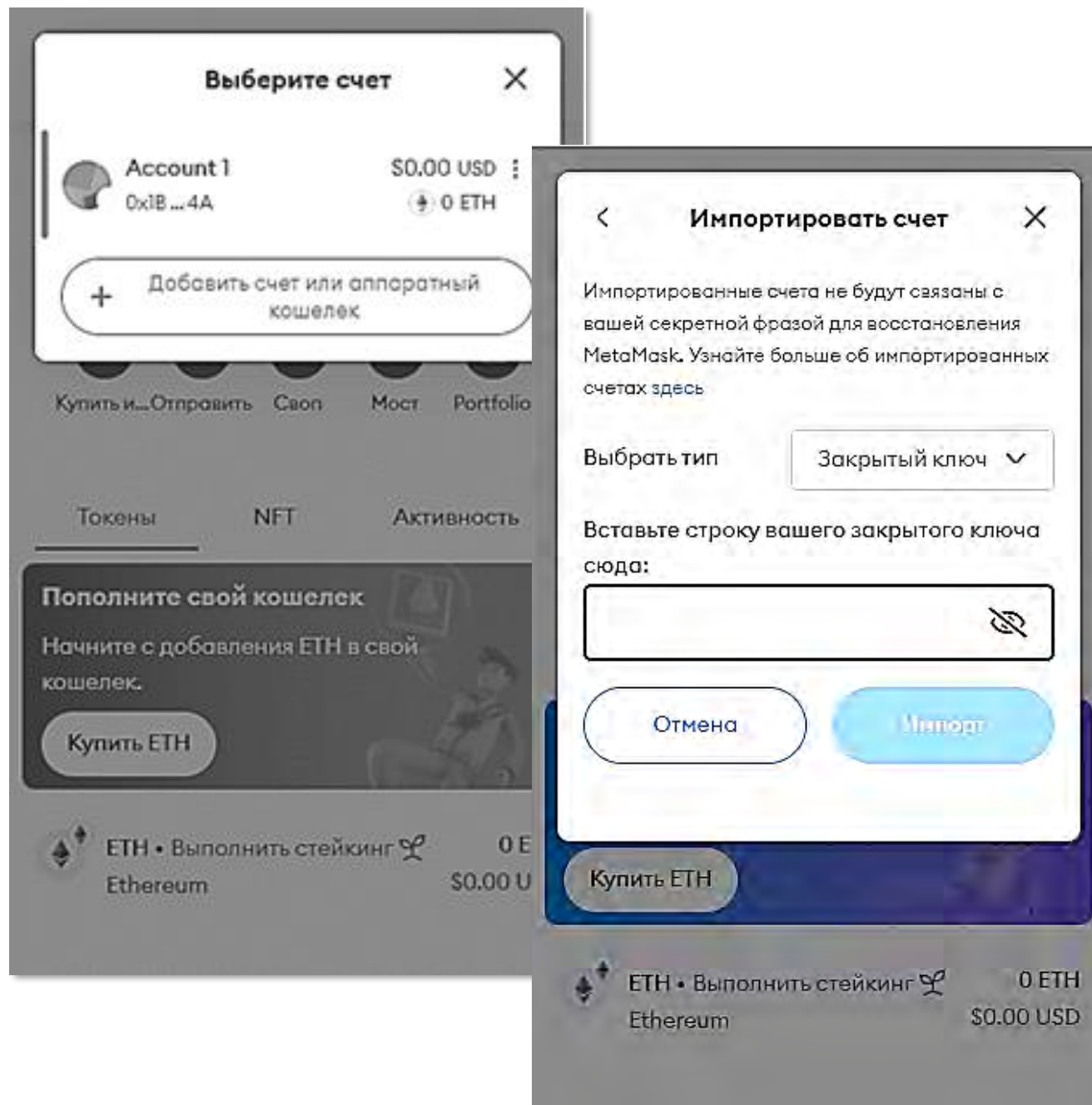
# Metamask

- Metamask поддерживает различные Эфириум-подобные сети, в т.ч. тестовые.
- Выбрать сеть можно, щёлкнув по значку сети в левом верхнем углу окна. Приложение выведет список используемых сетей. Добавить другие поддерживаемые сети можно, нажав кнопку «Добавить сеть».
- Если необходимо подключиться к сети, которой нет в списке, например к локально запущенному блокчейну Ganache, можно ввести параметры подключения, выбрав пункт «Добавить сеть вручную».



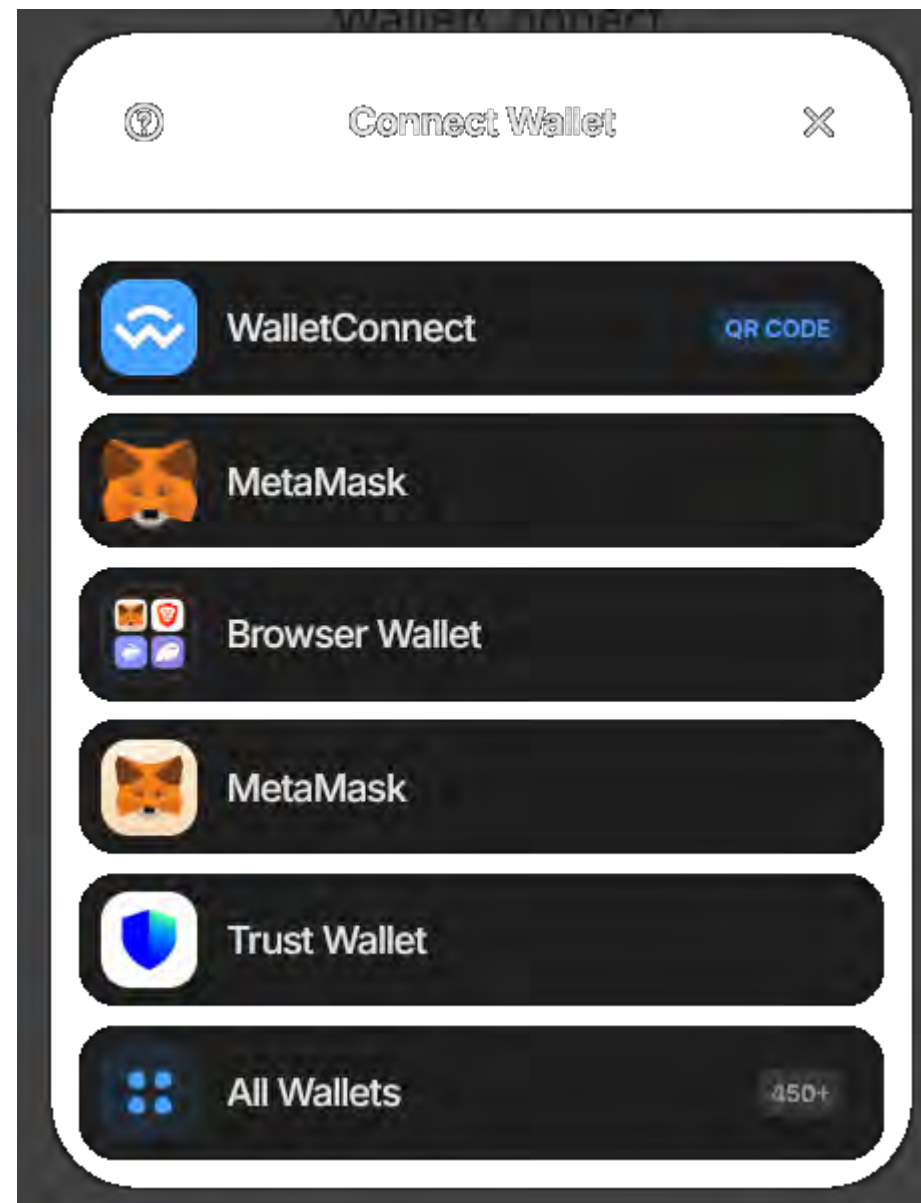
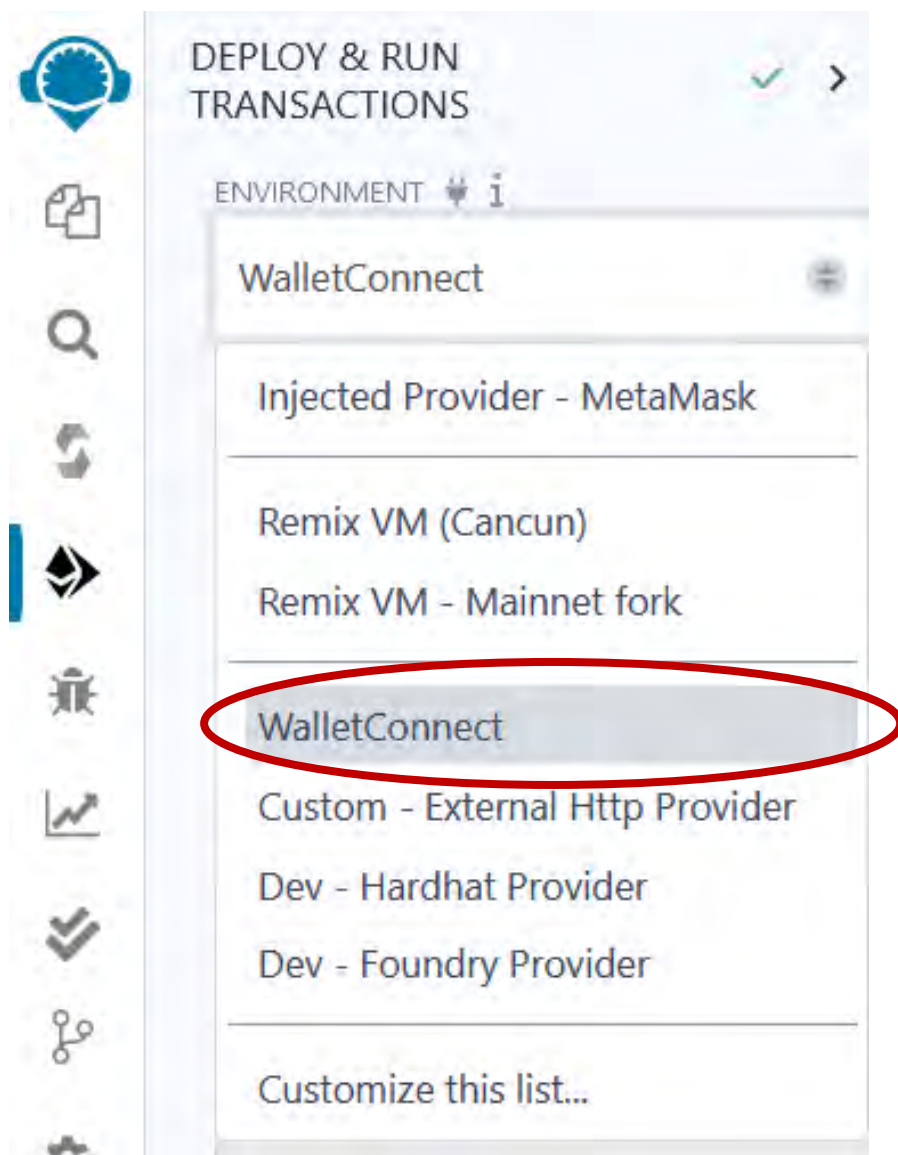
# Metamask

- В Metamask можно добавлять существующие аккаунты, например, предустановленные аккаунты в сети Ganache.
- После ввода закрытого ключа Metamask получит доступ к соответствующему аккаунту Ganache, от имени которого можно подписывать и отправлять транзакции.
- Выполнить перевод средств можно, нажав в окне аккаунта кнопку «Отправить» и введя адрес получателя и сумму. Metamask сформирует и подпишет соответствующую транзакцию и отправит её в Ganache для записи в блокчейн.





# Интеграция с Remix



# Geth

- Geth – консольный полноценный клиент исполнения сети Ethereum, со всеми необходимыми настройками, позволяет создавать частную сеть, а также работать с реальной сетью Ethereum.
- Все релизы по ссылке

<https://geth.ethereum.org/downloads/>

- Для создания частной сети, поддерживающей майнинг, нужна версия **<= 1.11.6**. Версии начиная с 1.12.0 работают на консенсусе PoS и требуют дополнительно установки клиента консенсуса.
- Распаковываем, получаем свою рабочую папку

# Geth

- Создание пользователя

```
geth account new --datadir <data-directory>
```

- Если просто запустим geth или запустим через cmd, то будет скачиваться весь блокчейн настоящего Ethereum.
- Скачивание будет происходить в папку  
C:\Users\<имя\_пользователя>\AppData\Roaming\Ethereum\chaindata\<сеть>
- В зависимости от выбранной сети будет заполняться папка **geth** для настоящего Ethereum, или например папка **rinkeby** – тестовая сеть.

# Генезис-блок

- Генезис-блок формируется вручную в формате json

```
{  
  "config": {  
    "chainId": 15,  
    "homesteadBlock": 0,  
    "eip150Block": 0,  
    "eip155Block": 0,  
    "eip158Block": 0,  
    "eip160Block": 0,  
    "byzantiumBlock": 0,  
    "constantinopleBlock": 0  
  },  
  "nonce": "0x00000000000000000001",  
  "difficulty": "100",  
  "gasLimit": "8000000",  
  "alloc": {  
    "0xc63830c325ae17a7668a3eea2a294791a5f7980d": {  
      "balance": "10000000000000000000000000"  
    }  
  }  
}
```

# Инициирование блокчейна

- Для того, чтобы начать работу с новой цепочкой, нужно загрузить генезис-блок командой:

```
geth.exe --datadir <data_dir> init <path_to_json>
```



# Запуск приватной сети

```
geth --http --http.addr "127.0.0.1" --http.port  
8545 --datadir <data_dir> --http.api "eth,  
miner, net, web3, admin, personal"  
--networkid 15 --allow-insecure-unlock  
--http.corsdomain "*" console  
--rpc.enableddeprecatedpersonal  
--nodiscover
```

# Параметры запуска geth

- `--http` – удалённый вызов процедур (RPC) по протоколу HTTP
- `--http.addr <ip_addr> (127.0.0.1)` – ip-адрес, на котором будет запущен клиент
- `--http.port <port> (8545)` – порт, на котором работает клиент
- `--http.api <namespaces_list>` – пространства имён geth, которые будут подключены
- `--networkid <network_id> (15)` – идентификатор сети
- `--datadir <data_dir>` – папка с цепочкой
- `--http.corsdomain <addr_list> ("*")` – адреса, с которых возможно использование cross-origin resource sharing
- `--allow-insecure-unlock` – возможность разблокировки аккаунтов в клиенте
- `--nodiscover` – не проводить поиск других узлов
- `console` – запуск в интерактивном режиме

# Пространства имён geth

Наименование	Назначение
<b>admin</b>	Команды администрирования узла
<b>eth</b>	Команды работы с блокчейном
<b>miner</b>	Команды работы с майнером (для Proof-of-Work – версий)
<b>net</b>	Команды работы с сетью
<b>personal</b>	Команды работы с аккаунтами пользователей
<b>web3</b>	Вспомогательные команды для web3-клиентов

- Перечень полей данных и функций в пространстве имён можно получить, введя его название в консоли

# Примеры команд geth

`personal.newAccount(<пароль>)` – создание нового аккаунта;

`personal.importRawKey(<ключ>, <пароль>)` – импорт существующего аккаунта;

`personal.unlockAccount(<адрес>)` – разблокировка аккаунта для выполнения транзакций в клиенте;

`eth.accounts()` – список известных узлу аккаунтов;

`eth.getBalance(<адрес>)` – баланс аккаунта;

`eth.getBlock(<номер>)` – вывод содержимого блока;

`eth.sendTransaction(<транзакция>)` – отправка транзакции;

`miner.setEtherbase(<адрес>)` – задание адреса для выплаты вознаграждения майнеру;

`miner.start(<кол-во потоков>)` – запуск майнера на узле;

`miner.stop()` – остановка майнера на узле;

# Тестовые сети

- **Тестовые сети** - блокчейны, имитирующие функционирование основной системы, но проводящие все операции в изолированном от неё реестре. Они дают возможность разработчикам без риска тестировать свои приложения и смарт-контракты перед их развертыванием в основной сети Эфириума.
- Тестовые сети используют те же адреса кошельков, что и основная сеть, однако криптовалюта тестовых сетей не торгуется на рынке. Она может быть куплена у владельца тестовой сети или получена бесплатно через **интернет-краны** (faucet), функционирование которых поддерживается заинтересованными в развитии сети организациями.



# Sepolia

- **Sepolia** – тестовая сеть, созданная разработчиками Эфириума в 2021 году.
- Работает на базе консенсуса Proof-of-Authority и поддерживает все базовые функции Эфириума.
- Криптовалюта Sepolia (Sepolia ETH) может быть получена бесплатно из интернет-кранов, например:

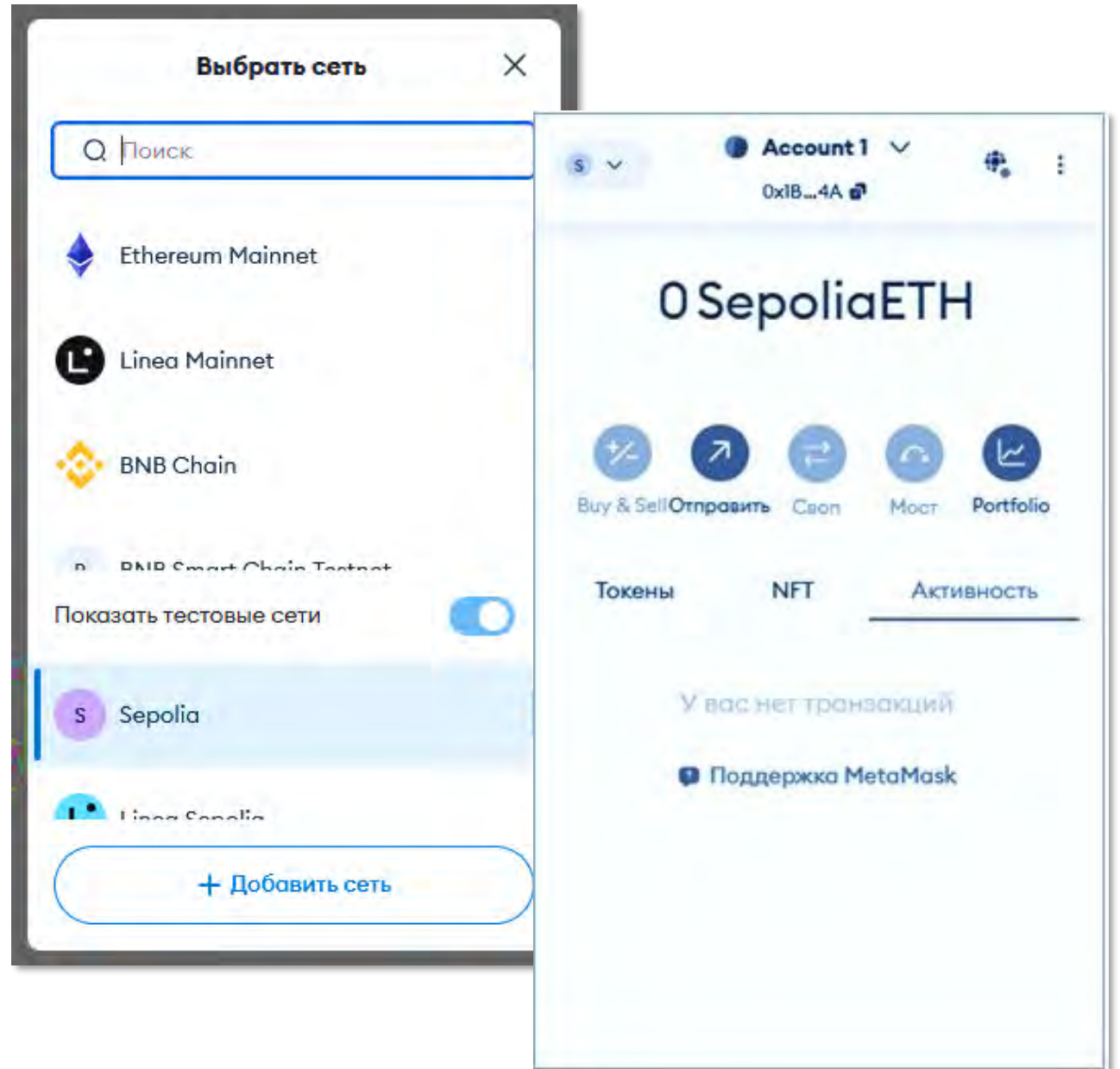
<https://cloud.google.com/application/web3/faucet/ethereum/sepolia>

- Текущие адреса узлов, предоставляющих доступ к API сети:

<https://sepolia.org/>

# Подключение к тестовой сети

- Подключение происходит через кошелёк, как и к основной сети Ethereum.
- Далее среда разработки может быть подключена к сети через WalletConnect либо Injected Provider.



# Внешний интерфейс смарт-контрактов: библиотека web3

- Библиотека содержит класс `Web3`, свойства и методы которого представляют интерфейс блокчейн-платформы. Для их использования объект `Web3` должен быть подключен к **провайдеру** – узлу, предоставляющему доступ к этим функциям:
  - **тестовый провайдер** – встроенный в библиотеку локальный эмулятор блокчейн-сети, содержащий несколько аккаунтов с определённым количеством средств и мгновенно включающий транзакции в блоки. Подключение к эмулятору производится при помощи метода `EthereumTesterProvider()`;
  - **локальный узел**, организованный с использованием запущенного на компьютере клиента выполнения Эфириума (например, `geth`). Доступ к локальному узлу может производиться как с использованием интернет-протокола HTTP, так и протокола межпроцессного взаимодействия IPC при помощи методов `HTTPProvider(<адрес>)` и `IPCProvider(<путь к ipc-файлу>)` соответственно;
  - **удалённый узел** – узел сети Эфириум, предоставляющий API для приложений. Доступ к нему может организовываться с помощью интернет-протоколов HTTP (метод `HTTPProvider(<адрес>)`) либо WebSocket (метод `WebSocketProvider(<адрес>)`).
- Проверить успешность подключения к провайдеру можно при помощи метода `is_connected()`
- Основные функции API узла представлены свойствами и методами объекта `Web3.eth`

# Свойства объекта Web3.eth

Наименование	Описание
account	Объект Account, обеспечивающий интерфейс для управления аккаунтами Ethereum
accounts	Список Ethereum-адресов, управление ключами которых осуществляет узел
block_number	Номер последнего блока в цепочке
chain_id	Идентификатор цепочки
coinbase	Текущий coinbase-адрес, на который перечисляется вознаграждение узлу
default_account	Адрес Ethereum, используемый в качестве from-адреса по умолчанию для всех транзакций
default_block	Номер блока по умолчанию, который будет использоваться для любых методов, принимающих идентификатор блока
gas_price	Текущая цена единицы газа
hashrate	Текущий хешрейт узла
max_priority_fee	Текущая максимальная комиссия за приоритет транзакции
mining	Статус майнинга узла

# Методы объекта Web3.eth

Наименование	Описание
<code>contract(address=None, contract_name=None, ContractFactoryClass= Contract, **contract_factory_ kwargs)</code>	Возвращает объект Contract, используемый для доступа к функциям контракта по адресу address
<code>get_balance(account, block)</code>	Возвращает баланс адреса account в блоке block
<code>get_block(block, full_transactions=False)</code>	Возвращает содержимое блока block. Если full_transactions=False, возвращаются только хеши транзакций, в противном случае – полностью их содержимое
<code>get_block_number()</code>	Возвращает номер последнего блока
<code>get_raw_transaction (hash)</code>	Возвращает содержимое транзакции с хешем hash в необработанном виде
<code>get_transaction(hash)</code>	Возвращает транзакцию с хешем hash
<code>get_transaction_count (account, block)</code>	Возвращает количество транзакций, отправленных с адреса account до блока block
<code>replace_transaction (hash, new_transaction)</code>	Заменяет ожидающую включения в блокчейн транзакцию с хешем hash транзакцией new_transaction
<code>send_transaction (transaction)</code>	Подписывает и отправляет в сеть транзакцию. Параметр transaction – словарь, содержащий значения полей транзакции
<code>sign(account, data=None, hexstr=None, text=None)</code>	Подписывает данные, задаваемые одним из параметров data, hexstr или text закрытым ключом адреса account
<code>sign_transaction (transaction)</code>	Подписывает транзакцию с использованием закрытого ключа пользователя
<code>wait_for_transaction_ receipt(hash, timeout=120, poll_latency=0.1)</code>	Ожидает включения в блок транзакции с хешем hash в течение периода времени timeout. При успехе возвращает подтверждение, в противном случае генерирует исключение.

# Пример: выполнение транзакции

```
from web3 import Web3
w3 = Web3(Web3.HTTPProvider('https://rpc2.sepolia.org/'))      # подключаемся к сети Sepolia
print(w3.is_connected())
account_1 = w3.eth.account.create()                          # Создаём аккаунты
account_2 = w3.eth.account.create()
dict_transaction = {                                         # Заполняем информацию
    'chainId': w3.eth.chain_id,                               # о транзакции
    'from': account_1.address,
    'to': account_2.address,
    'value': 1000000000000000000,
    'data': b'',
    'nonce': w3.eth.get_transaction_count(account_1.address),
    'gasPrice': w3.eth.gas_price,
    'gas': 21000,
}
signed_tx = w3.eth.account.sign_transaction(dict_transaction,  # Подписываем транзакцию
                                             account_1.key)
tx_hash = w3.eth.send_raw_transaction(signed_tx.rawTransaction)  # Отправляем в блокчейн
```



# Пример: выполнение функций смарт-контракта

```
import json

with open("abi.json", "r", encoding="utf-8") as f:
    abi = json.loads(f.read())

contract = w3.eth.contract(CONTRACT_ADDRESS, abi=abi)
contract.functions.function_1().call()

dict_transaction = {
    'from': account_1.address,
    'nonce': w3.eth.get_transaction_count(account_1.address),
}

transaction = contract.functions.function_2(1
    ).build_transaction(dict_transaction)

signed_tx = w3.eth.account.sign_transaction(transaction,
account_1.key)

tx_hash = w3.eth.send_raw_transaction(signed_tx.rawTransaction)
```

# Загружаем json-файл  
# с abi контракта

# Создаём контракт  
# Вызов функции без транзакции

# Заполняем информацию  
# о транзакции

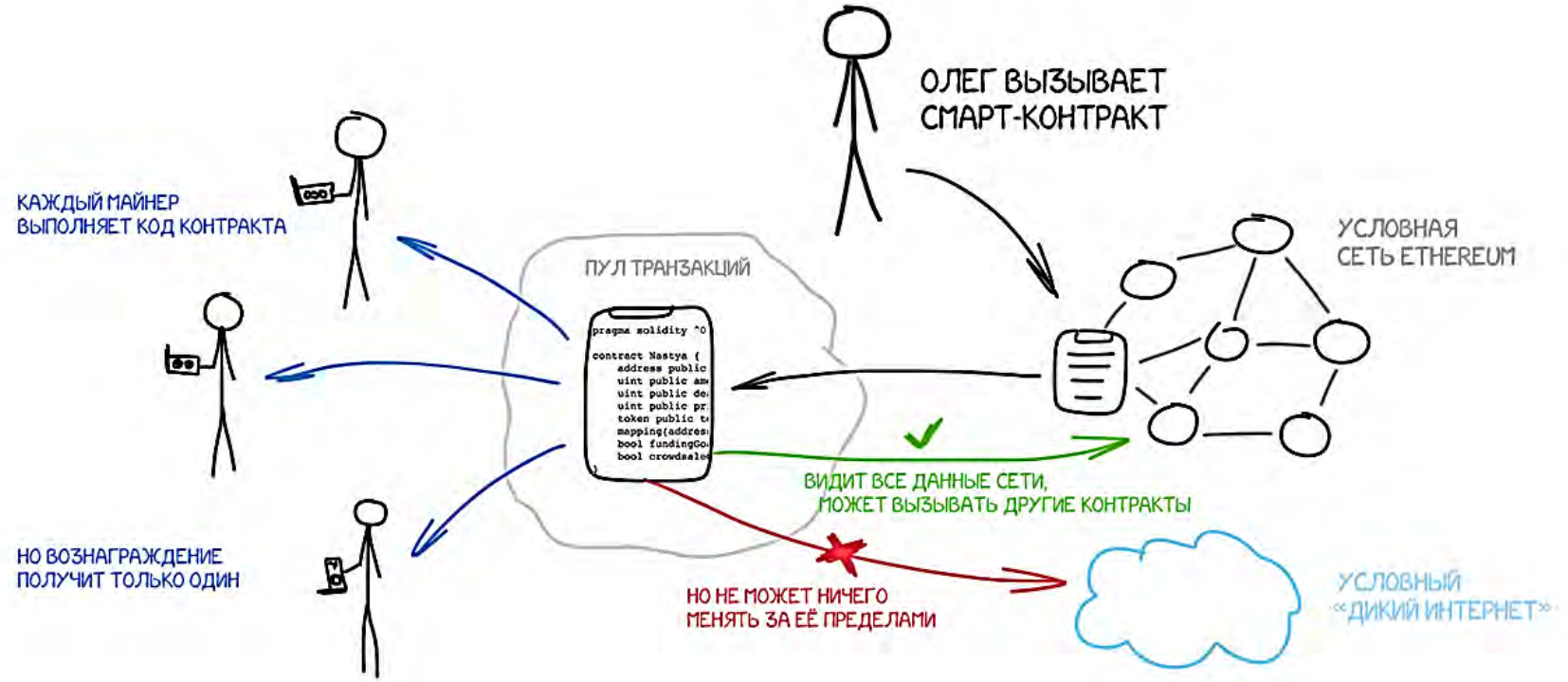
# Вызов функции через  
# транзакцию

# Подписываем транзакцию

# Отправляем в блокчейн

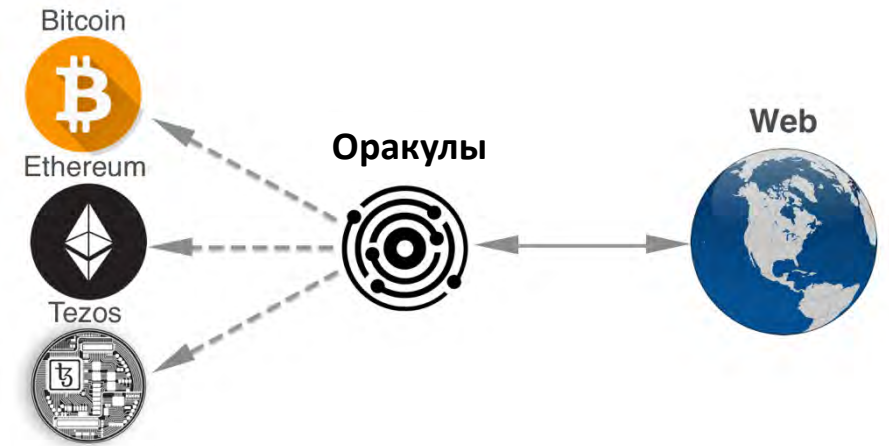
# Доступ смарт-контрактов к внешним данным

- Смарт-контракты, работающие на платформе Эфириум, не имеют непосредственного доступа к источникам данных, находящимся вне блокчейна.
- **Не гарантируется**, что внешняя информация будет одинаковой во всех узлах (различное время выполнения транзакции, ограничения на доступ к данным).
- **Не гарантируется**, что внешняя информация будет достоверна (внешний контроль, взлом).



# Оракулы

- **Оракулы** – специальные приложения, которые получают, проверяют и передают внешнюю информацию в смарт-контракты, гарантируя её достоверность и непротиворечивость
  - **входные** - передают данные из внешних источников для использования в контрактах;
  - **выходные** - передают данные из блокчейна во внешние приложения;
  - **вычислительные** - выполняют вычислительные задачи вне блокчейна и предоставляют смарт-контрактам их результаты.
- Представляют собой гибридные смарт-контракты, включающие в себя как компоненты, размещённые в блокчейне, так и вне его:
  - **контракт оракула** - часть, размещённая в блокчейне. Получает запросы от других контрактов, передаёт их внешним модулям, передаёт результаты обратно клиентским контрактам;
  - **узел оракула** - автономный компонент службы оракула. Извлекает информацию из внешних источников, производит необходимую обработку и инициирует транзакции для передачи запрошенных данных в блокчейн.



# Примеры сетей оракулов

- **Chainlink (<https://chain.link/>)** – децентрализованная платформа, обеспечивающая защищенные от несанкционированного доступа входные, выходные и вычислительные оракулы для поддержки продвинутых смарт-контрактов на любом блокчейне.
- **Witnet (<https://witnet.io/>)** – не требующий разрешений, децентрализованный и устойчивый к цензуре оракул, предоставляющий сильные криптографические гарантии подлинности данных, который позволяет смарт-контрактам реагировать на события реального мира.
- **UMA Optimistic Oracle (<https://uma.xyz/>)** – платформа для поставки в смарт-контракты произвольных внешних данных, построенная на основе игры эскалации с экономическими стимулами. Споры разрешаются путём голосования участников платформы при помощи токенов UMA.
- **Tellor (<https://tellor.io/>)** – прозрачный и не требующий разрешений протокол децентрализованных оракулов, позволяющий передавать в смарт-контракты любые необходимые для их работы внешние данные.
- **Paralink Network (<https://paralink.network/>)** – платформа децентрализованных оракулов для децентрализованных финансов и других приложений, работающая на основе блокчейна с несколькими цепочками в экосистеме Polkadot.