ЛЕКЦІЯ 7
ПРИНЦИПИ РОБОТИ З ETHEREUM

ПЛАН

1. Обліковий запис Ethereum .
2. Транзакції.
3. Консенсус.
4. Мітки часу.
5. Число nonce .
6. Час блоку.
7. Розгалуження.
8. Генезис.
9. Деномінація ефіру.
10. Віртуальна машина Ethereum .
11. Газ.
12. Виявлення вузлів.
13. Протоколи Whisper та Swarm .

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Ethereum - це децентралізована платформа, на якій можна запускати додатки у вигляді смарт-контрактів (**smart contract, розумний контракт**). Додаток може складатися з одного або кількох смарт-контрактів. Смарт-контракт Ethereum – це програма, яка виконується в мережі Ethereum та працює виключно так, як запрограмовано, без ризику простою, цензури, шахрайства та втручання третьої сторони.
Головна перевага платформи Ethereum для виконання смарт-контрактів полягає в тому, що контракти можуть легко взаємодіяти один з одним. Більше того, вам не треба турбуватися про інтеграцію протоколу консенсусу та інші речі - навпаки, вам лише потрібно написати логіку додатку.
Смарт-контракти для платформи Ethereum можуть бути написані на різних мовах програмування, зокрема Solidity, LLL Serpent.

Ethereum має внутрішню валюту, яка називається ефір (ether). Для того, щоб розвернути на платформі смарт-контракт або викликати його методи, нам потрібен ефір. Може існувати кілька екземплярів смарт-контракту або додатка, і кожен екземпляр ідентифікується на його унікальну адресу. Як рахунки користувачів, і смарт-контракти можуть зберігати ефір.

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Ethereum заснований на структурі даних блокчейну та протоколі консенсусу з доказом виконання роботи. Метод смарт-контракту може бути викликаний через транзакцію чи інший метод. У мережі є два типи вузлів: майнери та звичайні вузли. Звичайні вузли просто зберігають копію блокчейна, а майнер будують блокчейн, виробляючи блоки.

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Для створення облікового запису Ethereum нам потрібна пара асиметричних ключів. Ключі шифрування можуть генеруватися з урахуванням різних алгоритмів. Ethereum використовує криптографію на еліптичних кривих (Elliptic Curve Cryptography , ECC). Алгоритм ECC має різні параметри, від яких залежить швидкість та безпека. В Ethereum використана еліптична крива **secp256k1. П**

Відкритий та закритий ключі Ethereum є 256-бітовими числами. Оскільки процесор не може обробити настільки великі числа цілком, їх представляють у вигляді шістнадцяткового рядка з 64 символів.

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Облікові записи (акаунти) відіграють центральну роль у функціонуванні системи Ethereum .
Існує два типи облікових записів:

- ✓ **зовнішні (EOAs);**
- ✓ **контрактні**.

Облікові записи є ідентифікаторами зовнішніх користувачів, якими є як приватні особи, так і вузли видобутку або автоматизовані агенти.

Облікові записи використовують відкриті зашифровані ключі для підписання транзакцій, що дозволяють EVM ідентифікувати відправника трансакції.

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Створити обліковий запис у системі Ethereum – це використовувати гаманці, які підтримують Ethereum . Ще одним таким гаманцем виступає **Mist Ethereum** – програма, яка дозволяє користувачеві керувати та оперувати своїми обліковими записами. Версії програми гаманця доступні для Linux , Mac OS X та Windows .

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Завантаження

Розархівація завантаженої папки

Запуск файлу Ethereum-Wallet

Синхронізація з Ethereum з комп'ютером (запуск повного вузла geth , потрібно не менше 100 Гбайт на вінчестері)

Інструкція Ethereum-Wallet

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Коли ви вперше запустите гаманець Mist Ethereum Ви побачите обліковий запис. За умовчанням вона називатиметься **MAIN ACCOUNT** (ETHERBASE), а нижче ви побачите нульовий баланс і відкритий ключ (ваша публічна адреса).

Обліковий запис – це ключі, захищені паролем. Обліковий запис може містити Ether , захищені токени або монети та контракти на керування. Облікові записи не можуть відображати вхідні транзакції.

Для створення додаткових облікових записів просто натисніть ADD ACCOUNT на головному інтерфейсі програми та введіть пароль.

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Рекомендации по настройке: http://ethdocs.org/en/latest/account-management.html.

Інший спосіб створити обліковий запис – через командний рядок. Якщо ви завантажите Go Ethereum Client з GETH, ви зможете створити свій обліковий запис. Вам потрібно буде ввести пароль, який ви повинні зберігати, і в результаті ви отримаєте загальнодоступну адресу нового облікового запису. Ви можете створити стільки облікових записів, скільки захочете, ви також можете переглянути їх усі, набравши GETH Account List . Зверніть увагу, що тут ви також побачите обліковий запис, створений в Ethereum Wallet .

ОБЛІКОВИЙ ЗАПИС ETHEREUM

Для створення облікового запису Ethereum вам не потрібно підключатися до блокчейну і навіть до Інтернету: створення облікового запису відбувається повністю в автономному режимі.

Для створення облікового запису вам потрібна пара криптографічних ключів, відкритого та закритого.

ТРАНЗАКЦІЇ

Транзакція — це підписаний пакет даних, призначений для переміщення ефіру з одного рахунку на інший рахунок або контракт, виклику методу контракту або розгортання нового контракту.

Транзакція підписано цифровий підписом ECDSA (Elliptic Curve Digital Signature Algorithm, алгоритм цифровий підписи на еліптичних кривих).

Транзакція містить:

- покажчик на одержувача повідомлення,
- підпис відправника, що підтверджує його особу та наміри,
- кількість ефіру для передачі, максимальна кількість обчислювальних кроків, дозволених для виконання транзакції (так званий ліміт газу),
- вартість, яку відправник транзакції готовий заплатити за кожний обчислювальний крок (так звана ціна газу).

Якщо призначення транзакції полягає у викликі методу контракту , вона також містить вхідні дані , а якщо вона призначена для розгортання контракту , то може містити код ініціалізації .

Кількість газу та його ціна називаються збором за транзакцію. Щоб надіслати ефір або виконати метод контракту, вам необхідно транслювати транзакцію до мережі. Відправник повинен підписати транзакцію за допомогою закритого ключа.

КОНСЕНСУС

Вирішення завдання має вимагати витрачання значних обчислювальних ресурсів, що робить створення нового блоку важкою роботою.

Усі децентралізовані програми, які використовують доказ виконання роботи, не використовують абсолютно однаковий набір алгоритмів. Вони можуть відрізнятися в деталях щодо того, яке завдання має вирішити майнер , наскільки складним має бути завдання, скільки часу займає рішення і т.п.

КОНСЕНСУС

Будь - який учасник мережі може стати майнером . Кожен майнер вирішує завдання індивідуально. Перший майнер , який вирішив завдання, стає переможцем і отримує п'ять ефірів та збори від усіх транзакцій, що увійшли до блоку. Якщо ваш процесор потужніший, ніж у решти вузлів мережі, це не означає, що ви завжди будете переможцем, тому що параметри завдання не однакові для всіх майнерів . Однак, якщо ваш процесор потужніший, у вас більше шансів на успіх. Алгоритм доказу роботи схожий на лотерею, а обчислювальна потужність процесора відповідає кількості куплених лотерейних білетів. Безпека мережі залежить не від кількості майнерів , а від сукупної обчислювальної потужності мережі

Не існує обмеження щодо кількості блоків у блокчейні та кількості ефіру, який можна виробити.

КОНСЕНСУС

Щоб створити блок, майнер насамперед збирає необроблені (сирі) транзакції, які викладені в мережу, перевіряє їх та відкидає некоректні. Коректні транзакції повинні бути правильно підписані з використанням закритого ключа, рахунок відправника повинен містити достатню суму на балансі, щоб провести транзакцію, і т. д. Далі майнер створює блок, який має заголовок і вміст. Вміст складається зі списку транзакцій, які включені до блоку.

КОНСЕНСУС

Заголовок містить такі об'єкти:

- хеш попереднього блоку,
- номер блоку,
- одноразове число (nonce2),
- цільове число (target),
- мітку часу (timestamp),
- складність (difficulty),
- адреса майнера
- інші об'єкти.

Мітка часу містить час додавання блоку.

Nonce тут - це нічого не значуща кількість, яку майнер повинен знайти шляхом перебору, щоб отримати рішення задачі. Зазвичай завдання полягає в тому, щоб знайти таке число nonce , при якому хеш готового блоку менше цільового числа або дорівнює йому.

Ethereum використовує алгоритм хешування Ethash (Ethereum hash) : (https://github.com/ethereum/wiki/wiki/Ethash). Єдиний спосіб вирішення завдання полягає у переборі всіх можливих значень nonce .

Цільове число - це 256-бітове число, яке обчислюється виходячи з різних факторів. Значення складності у заголовку є іншим уявленням цільового числа. Чим менше цільове число, тим більше часу потрібно для знаходження відповідного значення nonce і навпаки, чим більше цільове число, тим швидше знаходиться **nonce** .

ФОРМУЛА ДЛЯ ОБЧИСЛЕННЯ СКЛАДНОСТІ ЗАДАЧІ:

```
current_block_difficulty = previous_block_difficulty +
previous_block_difficulty / 2048 * max(1 - (current_block_timestamp -
previous_blocktimestamp) / 10, -99) + int(2 ** ((current_block_number /
100000) - 2))
```

Коли майнер вирішив завдання і приєднав до блокчейну новий блок, будь-який вузол мережі може перевірити, чи блокчейн є правильним. Для цього він перевіряє, чи коректні транзакції блоку, чи правильно вказана мітка часу, чи правильно знайдено число nonce щодо цільового числа target для всіх блоків, чи правильно майнер вказав величину своєї винагороди тощо.

Якщо вузол мережі отримує два дійсні блокчейни, то приймається версія, у якої сукупна складність блоків вище.

КОНСЕНСУС

Якщо вузол мережі замінює кілька транзакцій у блоці, він повинен потім обчислитичисло nonce для всіх наступних блоків. На той час, коли він знову знайде число nonce для наступних блоків, мережа створить ще багато нових блоків, тому мережа відкидає цей ланцюжок.

МІТКА ЧАСУ

Для формули, за якою обчислюється цільове число, потрібна поточна позначка часу. Крім цього, кожен блок містить позначку часу в заголовку. Ніщо не заважає майнеру використовувати фіктивну мітку часу, поки він видобуває новий блок, але зазвичай вони так не роблять, тому що не пройдуть перевірку мітки часу, і інші вузли не приймуть блок. Отже, майнер дарма витратить ресурси. Коли майнер публікує новий блок, його мітка часу перевіряється порівнянням, чи вона перевищує мітку часу попереднього блоку. Якщо майнер використовує мітку часу, яка більша, ніж поточна мітка, параметр складності (difficulty) буде меншим, тому що його значення обернено пропорційно значенню мітки часу. Отже, якщо інший майнер запропонує блок, у якого мітка часу збігається з поточною міткою, то мережа віддасть перевагу саме цьому блоку, тому що у нього складність вище. Якщо майнер використовує мітку часу, яка більша, ніж у попереднього блоку, але менша, ніж поточний час, то складність буде вищою, що вимагатиме більше часу на вирішення завдання. За час, поки видобувається блок, мережа зробить інші блоки, і цей блок буде відхилений, як такий, що має меншу складність. Ось чому майнери завжди використовують точні позначки часу, інакше вони нічого не запрацюють.

ЧИСЛО NONCE

Nonce - це 64-бітове беззнакове ціле число, є рішенням завдання.

Майнер продовжує послідовно збільшувати число, доки знайде рішення.

Хеш блоку, який видобуває майнер завжди різний для різних майнерів , тому що він залежить від таких параметрів, як мітка часу, адреса майнера та інших, які не можуть бути однаковими у всіх майнерів . Отже, це не перегони за рішенням. Скоріше це лотерея.

ЧАС БЛОКУ

```
current_block_difficulty = previous_block_difficulty +
previous_block_difficulty / 2048 * max(1 - (current_block_timestamp -
previous_blocktimestamp) / 10, -99) + int(2 ** ((current_block_number /
100000) - 2))
```

містить 10-секундну затримку, яка гарантує, що різниця часу майнінгу між батьківським та дочірнім блоком перебуватиме в межах 10–20 секунд.

Якщо обчислювальної потужності мережі недостатньо, пошук рішення (і проведення транзакції) може тривати досить довго. Іноді мережа може повезти, і тоді рішення буде знайдено дуже швидко.

ЧАС БЛОКУ

Якщо складність завдання низька, це загрожує безпеці блокчейну, тому що потужні майнери видобуватимуть блоки набагато швидше за слабкі майнери і можуть захопити управління децентралізованим додатком. Неможливо знайти постійне значення складності, яке гарантує стабільність мережі, тому що обчислювальна потужність мережі є непостійною.

Що станеться, якщо два майнери дістануть наступний блок приблизно в один і той самий час? Обидва блоки будуть цілком коректними, але блокчейн не може містити два блоки з однаковим номером, і два майнери не можуть отримати винагороду. Хоча це часта проблема, вона має просте рішення. Мережею буде прийнято блок, у якого більша складність. Відповідно, відкинутий блок буде названо застарілим блоком.

ЧАС БЛОКУ

Але у чому полягає проблема, яку створюють застарілі блоки?
1. Вони гальмують підтвердження транзакції. Якщо два майнери виробили блоки приблизно одночасно, вони можуть містити різний набір транзакцій, і якщо наші транзакції містяться в одному з них, ми не можемо вважати їх підтвердженими, тому що блок можуть визнати застарілим. Нам доведеться чекати, доки не буде знайдено кілька нових блоків. З вини застарілих блоків середній час підтвердження транзакції не дорівнює середньому часу вироблення блоку.

ЧАС БЛОКУ

2. Застарілі блоки знижують безпеку блокчейну.
Ми знаємо, що безпека мережі визначається сукупною обчислювальною потужністю всіх майнерів , що працюють у мережі. Коли обчислювальна потужність зростає, складність завдання зростає, щоб гарантувати, що блоки не виробляються швидше, ніж за встановлений середній час. Отже, більш висока складність означає вищу безпеку, тому що втручання зловмисного вузла блокчейн вимагатиме наявності у зловмисника значної потужності хешування. Якщо два блоки запропоновані приблизно одночасно, вони ділять мережу навпіл, де кожна половина перевіряє свою версію блокчейна. Але буде прийнято лише одну версію, а друга половина мережі даремно витратить обчислювальну потужність виконання безкорисної роботи. Це виглядає так, начебто знизилася корисна обчислювальна потужність мережі, тому час вироблення наступного блоку буде скорочено за рахунок зниження складності. Зниження складності зменшує загальний рівень безпеки мережі. Якщо застарілих блоків багато, це дуже погано впливає на безпеку блокчейна.

ЧАС БЛОКУ

Ethereum бореться із проблемою застарілих блоків за допомогою протоколу під назвою GHOST (Greedy Heaviest Observed SubTree), використовуючи модифіковану версію цього протоколу.

Існує спеціальна формула, за якою обчислюється винагорода майнер за прийом застарілого блоку в переробку. Залишок винагороди переходить до блоку-племінника (nephew block), який вбирає в себе блок-дядечка (uncle block):

```
(uncle_block_number + 8 - block_number) * 5 / 8
```

РОЗГАЛУЖЕННЯ

Якщо трапляється конфлікт між вузлами, що підтвердили справжність блокчейну, і виникає більше одного справжнього блокчейна в мережі, причому всі вони підтверджені кількома майнерами , то кажуть, що відбулося розгалуження (forking). Прийнято розрізняти три види розгалуження: просте (regular fork), м'яке (soft fork , софтфорк) та жорстке (hard fork , хардфорк). Просте розгалуження виникає, якщо два або більше майнер виробили блок майже одночасно. Проблема вирішується за рахунок того, що складність одного блоку більша, ніж у інших.

НАЙВІДОМІШІ ХАРДФОРКИ ETHEREUM

- Frontier (30.07.2015 р.);
- Frontier Thawing (07.09.2015 р.);
- Homestead (14.03.2016 р.);
- Ethereum Classic (20.07.2016 р.);
- Tangerine Whistle (18.10.2016 р.);
- Spurious Dragon (22.11.2016 р.);
- Byzantium (16.10.2017 р.);
- Constantinople (29.02.2019 р.);
- Istanbul (08.12.2019 р.);
- Muir Glacier (02.01.2020 р.);
- Beacon Chain (01.12.2020 р.);
- Berlin (15.04.2021 р.);
- London (05.09.2021 р.);
- Merge (Bellatrix і Paris) - "Злиття" (15.09.2022 р.).

- https://tangem.com/ru/blog/post/overview-of-ethereum-forks/

РОЗГАЛУЖЕННЯ

Після зміни вихідного коду сайту можуть виникнути конфлікти версій вузлів. Залежно від типу конфлікту може знадобитися обов'язкове оновлення вузлів у майнерів , сукупна обчислювальна потужність яких перевищує 50% потужності мережі, або оновлення абсолютно всіх вузлів мережі. Перший тип конфлікту називається м'яким розгалуженням, а другий жорстким. Прикладом м'якого розгалуження є оновлення програмного забезпечення (реалізації протоколу), у якому старі блоки/транзакції втрачають валідність. Але якщо у майнерів , які оновили свої вузли, сукупна потужність понад 50%, то їхня нова версія блокчейна матиме більш високу складність і зрештою буде прийнята всією мережею. Прикладом жорсткого розгалуження є оновлення вихідного коду, у якому змінюється механізм розрахунку винагороди, й у вирішення конфлікту все майнери мають оновитися. Ethereum з моменту запуску пройшов через кілька жорстких та м'яких розгалужень.

ГЕНЕЗИС

Перший блок блокчейна називають генезисом (genesis block) або блоком прабатьком. Він має нульовий номер. Це єдиний блок у блокчейні , який не посилається на попередній блок. Він не містить транзакції, тому що до цього моменту не випущено жодного ефіру. Два вузли мережі можуть встановити зв'язок між собою, якщо мають однаковий генезис, тобто синхронізація блоків між вузлами може відбуватися тільки в тому випадку, якщо вони зберігають однаковий блок-прабатько. Інший генезис, що має більш високу складність, не може замінити менш складний генезис. У багатьох мережах код блоку-прабатька вписаний у вихідний код клієнтської програми.

ДЕНОМІНАЦІЯ ЕФІРУ

- 1 Ether = 1000000000000000000 Wei;
- 1 Ether = 1000000000000000 Kwei;
- 1 Ether = 1000000000000 Mwei;
- 1 Ether = 1000000000 Gwei;
- 1 Ether = 1000000 Szabo;
- 1 Ether = 1000 Finney;

- 1 Ether = 0.001 Kether;
- 1 Ether = 0.000001 Mether;
- 1 Ether = 0.000000001 Gether;
- 1 Ether = 0.000000000001 Tether.

ВІРТУАЛЬНА МАШИНА ETHEREUM

Віртуальна машина Ethereum (Ethereum Virtual Machine, EVM) — це середовище виконання байт-коду контрактів. EVM працює на кожному вузлі мережі. Усі вузли мережі виконують усі транзакції, на які вказують смарт-контракти, тому кожен вузол виконує одні й ті самі обчислення та зберігає однакові значення. Тран акції, які тільки переміщують ефір, також вимагають деяких обчислень, таких як визначення достатності балансу на рахунку або віднімання суми з балансу.

Кожен вузол виконує транзакції та зберігає остаточний стан з кількох причин. Наприклад, існує смарт-контракт, який зберігає імена та дані гостей, запрошених на вечірку. Щоразу, коли додають нового гостя, нова транзакція транслюється у мережу. Будь-який вузол може ознайомитись із даними кожного запрошеного на вечірку. Для цього достатньо прочитати фінальний стан договору.

Для будь-якої транзакції необхідні обчислення та сховище у мережі. Отже, мають існувати витрати на транзакцію, інакше вся мережа буде переповнена транзакційним спамом, а у майнерів не буде стимулу включати транзакції блок, і вони почнуть генерувати порожні блоки. Кожна транзакція вимагає різних ресурсів для обробки та зберігання, отже, різні транзакції повинні мати різну вартість

ГАЗ

Газ (gas) - це одиниця виміру обчислювальних кроків. Кожна транзакція має містити ліміт газу та величину винагороди за газ (за одиничне обчислення). Майнер має можливість вибирати транзакції та збирати винагороду. Якщо кількість витраченого газу менша або дорівнює ліміту газу, то транзакція виконується. Якщо витрати газу перевищують ліміт, то всі зміни скасовуються, крім валідності транзакції, але винагорода за транзакцію майнером може бути отримана

Майнери самостійно встановлюють вартість газу (ціну за поодиноке обчислення на своєму вузлі). Якщо транзакція пропонує меншу ціну газу, ніж запросив майнер , то майнер відкидає транзакцію. Вартість газу вимірюється в одиницях Wei . Іншими словами, майнер може відмовитись включати транзакцію до блоку, якщо запропонована вартість газу менша, ніж йому потрібно.

ГАЗ

Транзакційні витрати впливають на максимальну кількість ефіру, який може бути переведений з одного рахунку на інший. Наприклад, якщо на рахунку зберігається п'ять ефірів, то неможливо ці п'ять ефірів повністю перекласти на інший рахунок, тому що після переказу на вихідному рахунку нічого не залишиться для оплати комісії за транзакцію. Якщо транзакція викликає метод контракту, і метод відправляє кілька монет або викликає інший метод контракту, комісія за транзакцію віднімається від рахунку, який викликав метод.

ВИЯВЛЕННЯ ВУЗЛІВ

Щоб вузол став частиною мережі, він повинен встановити з'єднання з рештою вузлів мережі, завдяки чому зможе транслювати транзакції/блоки та слухати трансляцію транзакцій/блоків. Вузол не потребує з'єднання з усіма існуючими вузлами мережі. Навпаки, вузол з'єднується лише з кількома вузлами. У свою чергу, кожен із таких вузлів з'єднується з кількома іншими вузлами. Так формується мережа взаємозалежних вузлів.

Але яким чином вузол може знайти в мережі інші вузли, якщо не існує центрального сервера, з яким будь-хто міг би обмінятися інформацією? Ethereum використовує на вирішення цієї проблеми власний протокол виявлення вузлів, заснований на протоколі Kademlia. Відповідно до цього протоколу, в мережі є спеціальні вузли, які називають завантажувальними (bootstrap node). Завантажувальний вузол формує список вузлів, які підключалися до нього протягом певного часу. Завантажувальний вузол не зберігає блокчейн. Коли вузол користувача підключається до мережі, він спочатку з'єднується із завантажувальним вузлом і отримує список вузлів, які з'єднувалися з ним протягом заздалегідь заданого періоду часу. Потім новий вузол з'єднується та синхронізується з вузлами зі списку.

ВИЯВЛЕННЯ ВУЗЛІВ

Можуть існувати різні екземпляри Ethereum , тобто різні мережі, кожна з яких має ідентифікатор. Дві основні мережі Ethereum називаються mainnet (головна мережа) та testnet (тестова мережа). Мережа mainnet служить для торгівлі ефіром на біржах, а мережа testnet використовується розробниками для тестування. Все, що ми вивчили досі, відноситься до блокчейну мережі mainnet .

ПРОТОКОЛИ WHISPER ТА SWARM

Whisper (шепіт) і Swarm (рій) — це, відповідно, децентралізований протокол зв'язку та децентралізована платформа для зберігання даних, створені розробниками Ethereum.

Whisper допомагає вузлам спілкуватися один з одним. Він підтримує широкомовні повідомлення, зв'язок користувач-користувач, шифровані повідомлення та багато іншого. Він призначений передачі великого обсягу даних.

Ви можете прочитати більше про протокол Whisper за адресою: https://github.com/ethereum/wiki/wiki/Whisper та ознайомитись з оглядом прикладів коду за адресою: https://github.com/ethereum/wiki/wiki/Whisper-Overview.

ПРОТОКОЛИ WHISPER ТА SWARM

Swarm схожа на Filecoin і відрізняється, в основному, технічними деталями та заохоченням. Filecoin не сплачує за зберігання, тоді як Swarm виплачує винагороду - це збільшує доступність інформації. Ви можете запитати, як реалізовано заохочення у Swarm ? Чи має вона внутрішню валюту? Насправді у Swarm немає для виплати винагороди іншої валюти, крім ефіру. Існує смарт-контракт Ethereum , який відстежує заохочення. Очевидно, що смарт-контракт не може спілкуватися зі Swarm. Навпаки, Swarm може звертатися до контракту. Загалом, ви сплачуєте сховищу через смарт-контракт, і платіж зараховується сховищу після настання заданої дати. Ви також можете повідомити контракт про зникнення файлу, і контракт стягне штраф із відповідного сховища. Ви можете дізнатися більше про відмінності між Swarm та IPFS/ Filecoin за адресою: https://github.com/ethersphere/go-ethereum/wiki/IPFS-&-SWARM та отримати вихідний код смарт-контракту за адресою: https:// github.com/ethersphere/go-ethereum/blob/bzz-config/bzz/bzzcontract/swarm.sol .

ГЕТ

Geth (або Go-Ethereum) - це реалізація вузлів Ethereum , Whisper і Swarm . Geth може бути використаний як компонент усіх трьох протоколів або лише для будь-якого з них. Причина використання Geth полягає в потребі зробити так, щоб усі три децентралізовані програми разом виглядали як єдине ціле, а клієнт мав доступ до всіх трьох програм через один вузол мережі. Geth - це консольна програма, яка написана мовою Go . Воно доступне всім основних операційних систем. Поточна версія Geth не підтримує Swarm і підтримує лише деякі функції Whisper .

HTTPS://DOCS.SOLIDITYLANG.ORG/EN/V0.8.19/
HTTPS://WWW.YOUTUBE.COM/WATCH?V=8A8-7KS26YY

- Дякую за увагу!