Portfele EOS.IO i inteligentne kontrakty

W części 2 przedstawiłem EOS.IO, kiedy omówiłem bitcoin, altcoiny i różne mechanizmy konsensusu. W szczególności omówiłem, w jaki sposób EOS.IO jest przykładem altcoinów, które zamieniają się w tokeny; stworzyłeś producenta bloków EOS i byłeś w stanie stworzyć pełny węzeł zdolny do wydobywania tokenów EOS. Ethereum było początkiem tworzenia inteligentnych kontraktów blockchain i nauczyłeś się używać języka Solidity do pisania inteligentnych kontraktów i dappów. EOS.IO stworzył bardziej niezawodną architekturę niż Ethereum do inteligentnych kontraktów i rozwoju dappów. W tej części rozwinę blockchain EOS.IO i pokażę, jak zbudować inteligentny kontrakt EOS.IO, który można wykorzystać w zdecentralizowanych aplikacjach (dapps). Skonfigurujesz lokalne środowisko testnetowe i dowiesz się, jak skonfigurować narzędzia i biblioteki EOS.IO. Dowiesz się o portfelach EOS.IO oraz o tym, jak tworzyć, usuwać i tworzyć kopie zapasowe portfeli, a także wykonywać operacje, takie jak otwieranie, blokowanie i odblokowywanie portfela. Zakryję klucz do portfela pary i jak rozkręcić i ponownie rozkręcić lokalnego producenta bloków testnetowych. Dowiesz się o uprawnieniach oraz opcjach pojedynczego podpisu i multipodpisu. Aby lepiej zrozumieć inteligentne kontrakty EOS.IO, utworzysz inteligentny kontrakt „HelloWorld" i token kontraktu inteligentnego. Będziesz tworzyć konta, pisać kod smart kontraktów C++, kompilować kod i generować pliki WebAssembly i ABI, a także kontrakty Ricardian. Następnie dowiesz się, jak wdrażać inteligentne kontrakty i wchodzić z nimi w interakcje, a także wydawać tokeny i przekazywać tokeny innemu użytkownikowi. Na koniec połączysz się z publicznym producentem bloków testnetowych w celu przetestowania w bardziej realistycznym środowisku, a także połączysz się i publikujesz na producencie bloków mainnet. Uwaga EOS to natywna kryptowaluta (token), która zasila oprogramowanie EOS.IO. EOS.IO to w pełni dostosowany protokół architektury blockchain na skalę przemysłową, który umożliwia zdecentralizowane aplikacje, zapewniając dostęp do części tworzących blockchain. Pomyśl o EOS.IO jako o systemie operacyjnym blockchain, ponieważ emuluje prawdziwy komputer i umożliwia dostęp do zasobów, takich jak procesor, GPU, pamięć RAM i dysk twardy. EOS.IS nie pobiera opłat transakcyjnych podczas wykonywania milionów transakcji na sekundę. Token EOS jest tokenem użytkowym, a posiadanie tokena (stakowanie) zapewnia przepustowość i pamięć masową w łańcuchu bloków EOS.IO. Otrzymujesz zasoby proporcjonalnie do całkowitej stawki, którą posiadasz, do całkowitej stawki (posiadanie 1% tokenów EOS daje wykorzystanie do 1% całkowitej przepustowości EOS.IO). „Oprogramowanie EOS.IO wprowadza nową architekturę blockchain zaprojektowaną w celu umożliwienia pionowego i poziomego skalowania zdecentralizowanych aplikacji. Osiąga się to poprzez stworzenie działającej konstrukcji podobnej do systemu, na której można budować aplikacje. Oprogramowanie zapewnia konta, uwierzytelnianie, bazy danych , komunikację asynchroniczną i planowanie aplikacji w wielu rdzeniach procesora lub klastrach. Powstała technologia to architektura blockchain, która może ostatecznie skalować się do milionów transakcji na sekundę, eliminuje opłaty użytkowników oraz pozwala na szybkie i łatwe wdrażanie i utrzymanie zdecentralizowanych aplikacji w kontekście zarządzanego łańcucha bloków". -Biała księga EOS.IO block.one . Jak wspomniano w części 2, EOS.IO opiera się na konsensusie delegowanego dowodu udziału (DPoS). EOS.IO jest w stanie obsłużyć małe opóźnienia i dziesiątki milionów aktywnych użytkowników dziennie (z pominięciem Ethereum). Osiąga się to dzięki konsensusowi DPoS, a także EOS.IO działający jako wielowątkowy (działający na wielu rdzeniach komputera) i działający jako system operacyjny. Ten rodzaj skalowalności może umożliwić przyjęcie technologii blockchain przez duże firmy. EOS.IO oferuje wiele dodatkowych funkcji, takich jak:

• Bezpłatne transakcje o ograniczonej stawce

• Transakcje o niskim opóźnieniu (takie jak 0,25 sekundy czasu transmisji lub czas bloku 0.5)

• Odzyskiwanie skradzionych kluczy

• Równoległe wykonywanie aplikacji

• Transakcje atomowe z wieloma rachunkami

Zachęcam do zapoznania się z white paper EOS.IO i odwiedzenia GitHub

Strony z pełną listą funkcji.

- https://github.com/EOSIO/eos

- https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

Z finansowego punktu widzenia EOS został opracowany przez prywatną firmę o nazwie block.one i był w stanie zebrać zadziwiające 4 miliardy dolarów w początkowej ofercie monet (ICO) poprzez sprzedaż tokenów ERC-20. W chwili pisania tego tekstu cena EOS wynosi od 2 do 8 dolarów, a całkowita kapitalizacja rynkowa wynosi około 2 miliardy dolarów, co czyni EOS siódmą co do wielkości kryptowalutą pod względem kapitalizacji rynkowej. EOS oferuje kilka repozytoriów, które pomagają w tworzeniu kontraktów EOS.IO; są one wymienione na https://github.com/EOSIO i obejmują: eos, eosio.cdt, eosjs, demux-js i eosio.contracts. W tej części zainstalujesz biblioteki EOS i EOSIO.CDT. Biblioteka EOS to platforma inteligentnych kontraktów typu open source, a biblioteka EOSIO.CDT to zestaw narzędzi do budowania kontraktów EOS.IO. W chwili pisania tego tekstu platforma EOS.IO ma stromą krzywą uczenia się. Kod ciągle się zmienia, a dokumentacja i przykłady EOS.IO nie są aktualizowane na czas, więc czasami może się wydawać, że gonisz za ruchomym celem. Powoduje to, że kod czasami nie kompiluje się, polecenia nie działają oraz dokumentacja i przykłady zawierające kod i polecenia, które zostały przestarzałe. Podczas opracowywania kontraktu kilka razy łatwo się zakłopotać; jednak, gdy zrozumiesz EOS.IO, łatwo pokonasz te przeszkody.

Konfigurowanie środowiska testnetowego

Zanim przejdziemy do kodowania, zacznijmy od zainstalowania EOS.IO i EOSIO. CDT. Zbudujesz swoją wersję EOS.IO i skonfigurujesz lokalnego producenta bloków testnetowych. Następnie dowiesz się o narzędziach EOS.IO zwanych cleos, keosd i nodeos oraz o tym, jak je konfigurować oraz tworzyć i zarządzać portfelem z cleos. Te narzędzia i biblioteki są niezbędne do rozwoju.

Zainstaluj EOS.IO

Najłatwiejszym sposobem zainstalowania EOS.IO na macOS jest użycie Brew.

> brew tap eosio/eosio

> brew install eosio

Obecny EOS.IO to wersja 1.7.3. Polecam sprawdzić sekcję repozytorium i problemy na GitHub (https://github.com/eosio/eos) lub przeprowadzić wyszukiwanie w Google w przypadku napotkania błędów podczas instalacji lub budowania EOS.IO. Zobacz także https://github.com/EOSIO/eos/issues. Po zakończeniu instalacji zobaczysz komunikat na rysunku w Terminalu.

Następnie dodaj lokalizację plików binarnych EOS.IO do swojego środowiska, aby uruchamiać nodeos z dowolnego miejsca.

> export PATH=$PATH:/usr/local/eosio/bin

Spowoduje to ustawienie zmiennej ścieżki w tej sesji terminala, ale chcesz ustawić zmienną środowiskową ścieżki na stałe, więc dodaj ją do pliku bash_profile, otwierając plik za pomocą vima lub ulubionego edytora tekstu.

> vim ~/.bash_profile

Następnie wstaw następujące wiersze:

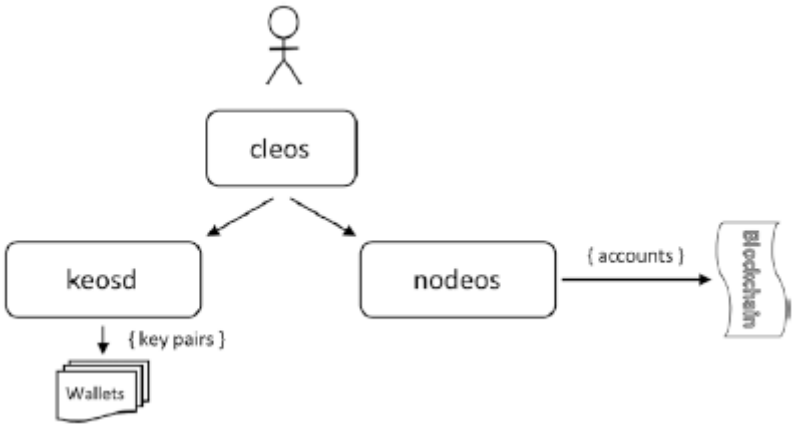
Setting PATH for EOSIO

PATH="/usr/local/eosio/bin:${PATH}"

Na koniec uruchom bash_profile, aby zatwierdzić zmiany.

> . ~/.bash_profil

EOS.IO jest dostarczany po wyjęciu z pudełka z wbudowanymi narzędziami i programami; są tutaj: /usr/local/eosio/. Rysunek przedstawia schemat architektury tych narzędzi.



• nodeos: jest to podstawowy demon EOS.IO, który umożliwia uruchomienie komponentu węzła blockchain. Nodeos można skonfigurować za pomocą wtyczek. Dodatkowo węzły można skonfigurować do uruchamiania producenta bloków w lokalnym środowisku programistycznym lub na dedykowanych punktach końcowych. Współdziała z łańcuchem bloków, tworząc bloki

• cleos: Jest to główne narzędzie wiersza poleceń dla interfejsów EOS.IO.It z REST API udostępnianym przez nodeos. Może również uzyskiwać dostęp do portfeli, gdy wchodzi w interakcję z keosd. Aby uzyskać listę poleceń cleos, po prostu uruchom następujące polecenie:

> cleos

• keosd: Jest to demon portfela do ładowania i zarządzania kluczami portfela. Robi to, ładując wtyczki związane z portfelem, takie jak interfejs HTTP i RPC API.

• eosio-launcher: To narzędzie pomoże Ci wdrożyć wielowęzłową sieć blockchain.

Zainstaluj EOSIO.CDT

Zainstalowałeś EOS.IO. Inną ważną biblioteką, której potrzebujesz, jest EOSIO.CDT (CDT to skrót od „contract development toolkit”). EOSIO.CDT to zestaw narzędzi służących do budowania kontraktów EOS.IO. Aby zainstalować bibliotekę, będziesz używać Brew.

> brew tap eosio/eosio.cdt

> brew install eosio.cdt

Najnowszym EOSIO.CDT w momencie pisania tego tekstu jest wersja 1.6.1. Uruchom aktualizację naparu, jeśli masz starszą wersję.

> brew upgrade eosio.cdt

Aby upewnić się, że instalacja przebiegła prawidłowo, uruchom polecenie eosio-cpp z argumentem help.

> eosio-cpp --help

Jak pamiętasz, użyłeś Truffle i Remix do wygenerowania plików interfejsu binarnego aplikacji (ABI) Ethereum. W przypadku inteligentnych kontraktów EOS.IO używasz eosio-cpp, który jest kompilatorem generującym plik WebAssembly (.wasm), który jest ABI, który należy przesłać do łańcucha bloków dla inteligentnego kontraktu. eosio-cpp generuje również funkcje pomocnicze, które serializują/deserializują typy zdefiniowane w kodzie ABI na potrzeby opracowywania kontraktów inteligentnych. Więcej informacji o EOSIO.CDT znajdziesz na stronie GitHub:

https://github.com/EOSIO/eosio.cdt.

W przyszłości, jeśli chcesz usunąć EOSIO i EOSIO.CDT, uruchom następujące polecenia:

> brew remove eosio

> brew remove eosio.cdt

Uwaga eosio-cpp jest zamiennikiem eosiocpp, który jest przestarzały. Pierwotnie eosiocpp był częścią instalacji EOS.IO, ale teraz jest częścią CDT

Zbuduj EOS.IO

Dobrym sposobem na wizualne zrozumienie EOS.IO i narzędzi związanych z EOS.IO jest przyjrzenie się powyższemu rysunkowi.

Pliki konfiguracyjne keosd i nodeos

Domyślne porty dla keosd i nodeos wykorzystują ten sam port: 8888. Aby skonfigurować nodeos, zobacz ten plik konfiguracyjny:

> vim "/Users/[user]/Library/Application Support/eosio/nodeos/

config/config.ini"

W pliku config.ini wartą uwagi zmienną do zmiany jest ładowana lista wtyczek. Nie będziesz wprowadzać zmian, ale w miarę postępów w rozwoju może być konieczne wprowadzenie zmian. Podobnie jak w przypadku nodeos, możesz skonfigurować keosd, edytując ten plik konfiguracyjny:

> vim ~/eosio-wallet/config.ini

Po otwarciu pliku zwróć uwagę, że istnieje zmienna o nazwie httpserver-address, której można użyć do zmiany portu 8888 na wypadek, gdyby ten port był potrzebny do innego oprogramowania. Tutaj ustawmy go na dowolny port, który lubisz. Zmienna jest wykomentowana. Aby ustawić go na port 9000, zmień go z tego:

http-adres-serwera =

do następujących:

http-server-address = http://127.0.0.1:9000

Możesz użyć domyślnego portu; jednak dobrze jest wiedzieć, jak skonfigurować EOS.IO.

Twórz i zarządzaj portfelem za pomocą cleos

W poprzednim rozdziale przedstawiłem kilka programów i narzędzi wbudowanych w EOS.IO. Jak wspomniano, cleos zapewnia interfejs API REST, który jest udostępniany przez nodeos. Przewodnik referencyjny Cleos można znaleźć tutaj: https://developers.eos.io/eosio-cleos/reference. Aby znaleźć numer cleos --version, uruchom polecenie klienta --version. W momencie pisania tego tekstu możesz zbudować d4ffb4eb.

> cleos version client

d4ffb4eb

Jak wspomniano, aby uzyskać listę poleceń, wystarczy wpisać cleos. lub cleos

--help;.

> cleos –help

Jeśli nie pamiętasz konkretnej podkomendy, wpisz ją i uzyskaj listę podkomend w danych wyjściowych; na przykład polecenie get wyświetla listę podkomend, taką jak info dla informacji o producencie twojego bloku.

> cleos get

> cleos get info

Nie udało się połączyć z węzłami pod adresem http://127.0.0.1:8888/; czy Nodes działa? Zauważ, że ponieważ nie masz uruchomionego węzła, nie otrzymujesz żadnych wyników i pojawia się komunikat o błędzie; jednak w dalszej części tego rozdziału, gdy rozkręcisz nodeos, otrzymasz informacje o producencie bloku.

Portfele EOS.IO

Portfele EOS.IO używają kluczy i oferują stan zablokowany (zaszyfrowany) oraz odblokowany (odszyfrowany) w celu ochrony kluczy. Polecenia blokady i odblokowania wymagają hasła o wysokiej entropii, które otrzymasz po utworzeniu portfela. Klucze portfela mogą być powiązane z kontem, aby zapewnić uprawnienia do tokenów konta, ale nie jest to konieczne do utworzenia portfela. Oprogramowanie portfela wykorzystuje cleos jako warstwę pośredniczącą między operacjami pobierania kluczy keosd a akcjami blockchain nodeos. Na przykład możesz użyć cleos, aby uzyskać dostęp do konta, ponieważ wymaga to wygenerowania podpisów z kluczy. Aby utworzyć domyślny portfel, po prostu uruchom polecenie Utwórz portfel. Użyj flagi --to-console, aby uzyskać klucz główny (hasło).

> cleos wallet create --to-console

Creating wallet: default

Save password to use in the future to unlock this wallet.

Without password imported keys will not be retrievable.

"[DEFAULT_MASTER_KEY]"

Upewnij się, że przechowujesz hasło. Teraz możesz sprawdzić, czy portfel został utworzony i uruchom polecenie listy portfeli, a będziesz mógł zobaczyć tablicę, która zawiera listę portfeli i zawiera domyślny portfel, który utworzyłeś

> cleos wallet list

Wallets:

[

"default ∗"

]

Zauważ, że po utworzeniu domyślnego portfela obok nazwy portfela znajduje się gwiazdka. Gwiazdka oznacza, że jest odblokowana. W następnej sekcji dowiesz się więcej o stanach blokowania i odblokowywania.

Usuń i wykonaj kopię zapasową portfeli

Aby usunąć utworzony portfel, musisz usunąć rzeczywisty plik portfela; znajduje się tutaj: ~/eosio-wallet.

> rm -rf ~/eosio-wallet

Uruchom polecenie listy portfeli, a zobaczysz, że tablica portfela jest pusta.

> cleos wallet list

"/usr/local/eosio/bin/keosd" launched

Wallets:

[]

Aby wykonać kopię zapasową portfela, skopiuj pliki portfela i przechowuj je w bezpiecznym miejscu.

Portfel EOS.IO z niestandardową nazwą

Do tej pory utworzyłeś domyślny portfel. Załóżmy teraz, że chcesz utworzyć kolejny portfel i nazwać go mywallet. Wszystko, co musisz zrobić, to użyć flagi -n lub --name. Wybierz nazwę i uważaj na ścisłe ograniczenia dotyczące nazw (dozwolone są tylko a–z i 1–5, z długością 12). Wybieram mój portfel.

> cleos wallet create -n mywallet --to-console

Creating wallet: mywallet

Save password to use in the future to unlock this wallet.

Without password imported keys will not be retrievable.

"[DEFAULT_MASTER_KEY]"

EOS.IO: otwieranie, blokowanie i odblokowywanie portfela

Kiedy tworzyłeś swój portfel, masz klucz główny o wysokiej entropii, który jest twoim hasłem. To hasło służy do szyfrowania (zablokowania) i odszyfrowania (odblokowania) pliku portfela. Aby zablokować i odblokować portfel, użyj następujących poleceń:

> cleos wallet lock -n mywallet

> cleos wallet unlock -n mywallet

password: [DEFAULT_MASTER_KEY]

password: Unlocked: mywallet

Polecenia zablokuj i odblokuj umożliwiają portfelowi ustawienie stanu szyfrowania i odszyfrowywania, który jest chroniony hasłem. To, co chronisz, to klucze do portfela. Aby odblokować domyślny portfel, po prostu uruchom następujące polecenie:

> cleos wallet unlock

Ponadto, aby wykonać operacje na portfelach, musisz najpierw otworzyć portfel. Po ponownym uruchomieniu keosd portfel zostanie zamknięty. Uruchom polecenie otwórz, aby otworzyć portfel w razie potrzeby.

> cleos wallet open

Open : defaut

Generowanie kluczy EOS.IO

Podobnie jak w innych blockchainach, EOS.IO przechowuje klucze w portfelu. Generujesz te klucze i przypisujesz je do konta EOS.IO. Istnieje wiele sposobów tworzenia kluczy. Będziesz tutaj używał cleos. Najpierw odtwórzmy domyślny portfel, na wypadek, gdybyś go wcześniej usunął.

> cleos wallet create --to-console

Creating wallet: default

Save password to use in the future to unlock this wallet.

Without password imported keys will not be retrievable.

"[DEFAULT_MASTER_KEY]"

Uruchomiona lista portfeli powinna pokazywać dwa portfele.

> cleos wallet list

Wallets:

[

"default",

"mywallet ∗"

]

Następnie, aby utworzyć dwie pary kluczy publiczny/prywatny, uruchom polecenie Utwórz klucz.

> cleos create key --to-console

Private key: [PRIVATE_KEY_1]

Public key: [PUBLIC_KEY_1]

> cleos create key --to-console

Private key: [PRIVATE_KEY_2]

Public key: [PUBLIC_KEY_2]

Jak zauważyłeś, dwukrotnie uruchomiłeś polecenie tworzenia klawisza. To nie jest literówka; musisz mieć dwa klucze: jeden dla aktywnego użytkownika i jeden dla właściciela. Więcej o tej koncepcji dowiesz się po utworzeniu konta. Polecenie, które uruchomiłeś, wyprowadza pary kluczy publicznych i prywatnych. Zauważ, że klucz publiczny zaczyna się od słowa kluczowego EOS. Te arbitralne pary kluczy same w sobie nie mają znaczenia, ponieważ nie mają uprawnień (nie należą do żadnego portfela ani konta). Aby przypisać te pary kluczy do portfela, możesz je zaimportować do swojego portfela.

> cleos wallet import --private-key [PRIVATE_KEY_1]

imported private key for:

[PRIVATE_KEY_1]

imported private key for: [key]

> cleos wallet import --private-key [PRIVATE_KEY_2]

imported private key for: [PRIVATE_KEY_2]

W wyniku polecenia otrzymałeś komunikat potwierdzający z wiersza poleceń, że pary kluczy zostały dodane. Możesz jednak również potwierdzić, że pary kluczy zostały dodane, wywołując polecenie kluczy portfela.

> cleos wallet keys

[PUBLIC_KEY_1, PUBLIC_KEY_2]

Dodatkowo możesz poprosić o wyświetlenie par kluczy.

> cleos wallet private_keys --password [DEFAULT_MASTER_KEY]

[[PUBLIC_KEY_1, PRIVATE_KEY_2],[PUBLIC_KEY_1, PUBLIC_KEY_2]]

W poprzednim poleceniu przekazałeś argument -password zamiast czekać, aż linia poleceń poprosi o podanie hasła głównego. Na koniec musisz zaimportować specjalne konto rodzica EOS.IO. To specjalne konto nadrzędne służy do ładowania węzłów EOS.IO. Bez tego klucza prywatnego nie będziesz w stanie utworzyć swojego konta. Konta EOS.IO wymagają konta rodzica, aby utworzyć kolejne; w ten sposób EOS.IO przydziela zasoby i chroni przed spamem i hakerami.

> cleos wallet import --private-key

5KQwrPbwdL6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3

imported private key for:

EOS6MRyAjQq8ud7hVNYcfnVPJqcVpscN5So8BhtHuGYqET5GDW5CV

Rozkręć węzeł z węzłami

Transakcje są dołączone do bloku i potrzebujesz producenta bloku, aby móc przekazać te transakcje do sieci. Możesz pominąć tworzenie węzła EOS (nodeos), jeśli łączysz się bezpośrednio z publiczną siecią testową lub siecią główną; jednak lepiej najpierw uruchomić smart kontrakty w lokalnej sieci testowej przed przekazaniem kodu do publicznej sieci testowej lub sieci głównej. W tym momencie powinieneś przyzwyczaić się do tego procesu, tak jak zrobiłeś to samo, gdy opracowałeś inteligentną umowę dla Ethereum. Relacja blockchain IO. Aby uruchomić własnego jednowęzłowego lokalnego producenta bloków blockchain, w osobnym terminalu, uruchom nodeos.

> nodeos -e -p eosio --plugin eosio::chain_api_plugin --plugin

eosio::history_api_plugin

To polecenie uruchamia producenta bloku i powinno wyświetlić proces na konsoli.

info 2019-04-28T19:03:34.811 thread-0 block_log.cpp:134

open] Log is nonempty

info 2019-04-28T19:03:34.820 thread-0 block_log.cpp:161

open] Index is nonempty

info 2019-04-28T19:03:34.878 thread-0 http_plugin.cpp:422

plugin_initialize] configured http to listen on

127.0.0.1:8888

...

...

...

Jak widać, konsola pokazuje, że twoja sieć lokalna zaczyna produkować bloki. Zauważ, że użyte polecenie ustawia wtyczki, a także ustawiasz flagę --contracts-console. Ta flaga jest niezbędna, aby móc zobaczyć komunikaty drukowane na konsoli w trybie programistycznym.

Uwaga : Możesz również ustawić flagę --contracts-console wewnątrz pliku config.ini zamiast przekazywać ten argument z nodeos za każdym razem.

Jak pamiętasz, poprzednio uruchamiałeś polecenie cleos get info i nie uzyskiwałeś żadnych wyników, ponieważ nie miałeś uruchomionego producenta bloków; teraz, jeśli uruchomisz to samo polecenie w nowym terminalu, możesz obserwować informacje o swoich blokach.

> cleos get info

{

"server_version": "d4ffb4eb",

"chain_id": "cf057bbfb72640471fd910bcb67639c22df9f92470936cd

dc1ade0e2f2e7dc4f",

"head_block_num": 73699,

"last_irreversible_block_num": 73698,

"last_irreversible_block_id": "00011fe2a80bf11315396c85e70860

122dddc24ac083911fba31f7ee2d64eb3e",

"head_block_id": "00011fe36fab1fc2d4885067e1391c72782895d43f14

cf7970ac282ddef17d67",

"head_block_time": "2019-04-28T19:04:06.500",

"head_block_producer": "eosio",

"virtual_block_cpu_limit": 200000000,

"virtual_block_net_limit": 1048576000,

"block_cpu_limit": 199900,

"block_net_limit": 1048576,

"server_version_string": "v1.5.1-dirty"

}

Ponowne rozkręcenie węzła lokalnego Testnet (nodeos).

Jeśli chcesz wyczyścić historię producenta bloku, usunąć wszystkie bloki i ponownie rozkręcić lokalną sieć testową, użyjesz tak zwanej twardej powtórki, używając następujących flag:

--delete-all-blocks --delete-state-history --hard-replay

Te argumenty wyczyszczą konta w lokalnej sieci testowej, a także bloki. Kompletne polecenie będzie wyglądało następująco:

> nodeos -e -p eosio --plugin eosio::chain_api_plugin --plugin

eosio::history_api_plugin --delete-all-blocks --delete-statehistory

--hard-replay --contracts-console

Konta EOS.IO

Konta EOS.IO posiadają czytelną dla człowieka nazwę, która jest przechowywana w EOS. Łańcuch bloków we/wy. Aby utworzyć konto w sieci głównej, ktoś, kto ma konto EOS.IO, musi je dla Ciebie utworzyć. Powodami tego regulowanego procesu są zapobieganie spamowi i hakerom oraz alokacja zasobów. Domyślnie konto posiada dwie natywne nazwy/uprawnienia.

- Właściciel: służy do odzyskiwania innych uprawnień, co jest przydatne w przypadku naruszenia uprawnień.

- Aktywny: jest używany do zmian na kontach wysokiego poziomu, takich jak transfer środków lub głosowanie na producentów bloków.

Podczas tworzenia konta testnet zaimportowałeś specjalny klucz konta nadrzędnego EOS.IO do ładowania początkowego. Każda nazwa uprawnień wymaga „rodzica”. Uprawnienie nadrzędne ma mieć możliwość wprowadzania zmian w ustawieniach uprawnień dla wszystkich swoich podrzędnych. EOS.IO zapewnia specjalny klucz nadrzędny konta dla lokalnej sieci testowej, który został zaimportowany w celu utworzenia konta. Aby transakcja była ważna i podpisana, każde nazwane uprawnienie wymaga spełnienia warunków, takich jak klient z odblokowanym portfelem, a portfel musi przyznać uprawnienia do konta. Jeśli nie spełnisz tych warunków, transakcja się nie powiedzie. Teraz, gdy rozumiesz konta, jesteś gotowy do utworzenia własnego konta. Utworzyłeś już portfel i zaimportowałeś klucz nadrzędny. Aby utworzyć konto, uruchamiasz następującą składnię polecenia:

> cleos utwórz konto eosio [ACCOUNT_NAME] [OWNER_PUBLIC_KEY]

[ACTIVE_PUBLIC_KEY]

Wartość OWNER_KEY to klucz publiczny uprawnienia właściciela konta, a wartość ACTIVE_KEY to klucz publiczny uprawnienia aktywnego konta.

W tym przykładzie nazwijmy konto myaccount i użyjmy dwóch utworzonych przez Ciebie klucze. Polecenie będzie wyglądać tak (patrz Rysunek 6-5 dla oczekiwanego wyniku):

> cleos załóż konto eosio moje konto [PUBLIC_KEY_1]

[PUBLIC_KEY_2]

Wygenerowałeś dwa klucze, więc nie ma znaczenia, który klucz zdecydujesz się użyć jako aktywny, a który jako właściciel; po prostu pamiętaj, którego klucza użyłeś do którego. Aby zobaczyć listę konta, użyj tego:

> cleos get accounts [PUBLIC_KEY_1]

{

"account_names": [

"myaccount"

]

}

Uwaga: Możesz otrzymać komunikat o błędzie podczas próby utworzenia konta, jeśli pominąłeś którykolwiek z kroków opisanych w tym rozdziale. Błąd to „Błąd 3090003: dostarczone klucze, uprawnienia i opóźnienia nie spełniają zadeklarowanych autoryzacji”.

Portfele, klucze i konta: kompletne polecenia

Aby upewnić się, że w pełni rozumiesz ten proces, oto podsumowanie tego, jak utworzyć konto:

1. Upewnij się, że Nodeos działa w osobnym oknie Terminala.

> nodeos -e -p eosio --plugin eosio::chain_api_plugin

--plugin eosio::history_api_plugin --delete-all-blocks

--delete-state-history --hard-replay --contracts-console

2. Upewnij się, że Twój portfel jest odblokowany. Uruchom > lista portfeli cleos (sprawdź, czy obok nazwy portfela znajduje się gwiazdka).

3. Klucz nadrzędny konta specjalnego EOS.IO (5KQwrPbwdL6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3) został zaimportowany, aby załadować EOS.IO.

cleos wallet import --private-key 5KQwrPbwdL

6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3

4. Sprawdź listę kluczy za pomocą > kluczy portfela cleos. Powinien wypisać tablicę z zaimportowanymi kluczami. Aby podsumować, co zrobiłeś do tej pory lub powtórzyć cały proces tworzenia konta, oto pełne kroki:

> rm -rf ~/eosio-wallet

> cleos wallet create --to-console

> cleos wallet open

> cleos wallet unlock --password [DEFAULT_MASTER_KEY]

> cleos create key --to-console

> cleos create key --to-console

> cleos wallet import --private-key [PRIVATE_KEY_1]

> cleos wallet import --private-key [PRIVATE_KEY_2]

> cleos wallet import --private-key

5KQwrPbwdL6PhXujxW37FSSQZ1JiwsST4cqQzDeyXtP79zkvFD3

> cleos wallet keys

> cleos create account eosio myaccount [EOS∗ OWNER_KEY] [EOS∗

ACTIVE_KEY]

Niestandardowy, pojedynczy podpis (Single-Sig) i Multisignature (Multisig)

Domyślnie skonfigurowałeś swoje konto z pojedynczym podpisem (aka singlesig), ponieważ jest ono autoryzowane do działań z domyślnymi uprawnieniami (aktywne i właściciela). Możliwe jest jednak skonfigurowanie kont z multisygnaturą (aka multisig) lub z uprawnieniami niestandardowymi. Na przykład możesz skonfigurować swoje konto z wieloma kluczami, aby autoryzować określone działania właściciela i aktywne działania. Możesz użyć tej funkcji, na przykład, aby utworzyć uprawnienie o nazwie „publikowanie" i nadać to uprawnienie kontu, aby zezwalać tylko na publikowane inteligentne kontrakty bez możliwości wycofania tokenów.

Inteligentny kontrakt „HelloWorld"

Będziesz pisać inteligentną umowę z minimalnym kodem. Swój inteligentny kontrakt nazwiesz „HelloWorld".

Konta inteligentnych kontraktów „HelloWorld"

Aby rozpocząć, utworzysz dwa konta dla inteligentnej umowy, jedno do publikowania inteligentnej umowy i jedno do interakcji z użytkownikiem.

> cleos create account eosio helloworld [PUBLIC_KEY]

> cleos create account eosio john [PUBLIC_KEY]

Kod C++ „HelloWorld”

EOS wybrał C++, co zaowocowało mieszanymi recenzjami społeczności programistów blockchain. C++ jest językiem niskiego poziomu i umożliwia lepsze zarządzanie zasobami, takimi jak wskaźniki pamięci i przeciążanie operatorów. Może to skutkować lepszą wydajnością; jednak wiąże się to z kosztem zwiększonego nakładu pracy nad kodem, zwłaszcza jeśli nie znasz C++. Infrastruktura EOS.IO jest napisana w C++, więc nie powinno dziwić, że C++ został wybrany przez zespół EOS.IO. Inteligentne kontrakty EOS.IO są napisane w C++ i zapisywane jako format pliku CPP; następnie kompilujesz kod C++ do WebAssembly, który jest następnie używany do wdrożenia. Uwaga Pliki źródłowe inteligentnej umowy EOS.IO można podzielić na trzy: CPP, HPP i Ricardian. Plik HPP definiuje klasę, akcje i tabele kontraktu inteligentnego. Plik CPP to kod C++, który implementuje logikę działania. Plik Ricarda jest dokumentem cyfrowym (więcej na ten temat w następnej sekcji). Zacznij od utworzenia katalogu kontraktów helloworld, przechodząc do katalogu.

> mkdir ~/Desktop/helloworld && cd $_

Zauważ, że używasz swojego pulpitu, ale możesz użyć dowolnego katalogu. Następnie wklej kod helloworld.cpp za pomocą vima lub ulubionego edytora tekstu.

> vim helloworld.cpp

```
#include < eosiolib/eosio.hpp >

using namespace eosio;

class helloworld : public contract {

public:

using contract::contract;

[[eosio::action]]

void hello( name user ) {

print( "World: User: ", user);

}

};

EOSIO_DISPATCH(helloworld, (hello))
```

Kod importuje biblioteki EOS.IO. Klasa HelloWorld jest typu kontrakt i tworzysz metodę o nazwie hello. Metodą jest twoje działanie; przekazujesz użytkownika i wypisujesz słowo world oraz nazwę użytkownika. Gdy użytkownik wejdzie w interakcję z Twoją umową i wywoła akcję powitania, otrzyma świat z nazwą użytkownika. Zauważ, że w tym przykładzie dołączono plik eosio.hpp. Aby debugować inteligentną umowę EOS.IO, musisz użyć staromodnego debugowania jaskiniowca.

Uwaga : Debugowanie Cavemana, czyli debugowanie printf(), to nic innego jak dodawanie instrukcji print wokół kodu. EOS.IO

Print API obsługuje tablicę znaków, 64-bitową i 128-bitową liczbę całkowitą bez znaku i inne. Drukowanie odbywa się poprzez zawinięcie kodu C++ printi, prints_l, printi128 i innych w print.hpp, który zawiera instrukcję biblioteki import eosio.hpp.

IDE inteligentnego kontraktu

Korzystanie z Terminala jest całkowicie akceptowalne, ale gdy kod staje się bardziej złożony, korzystanie z profesjonalnego środowiska IDE może być pomocne w uzupełnianiu kodu, wyróżnianiu i czytelności. Możesz użyć swojego IDE. Ponieważ korzystałeś już z WebStorm, możesz kontynuować i importować projekt do WebStorm. WebStorm zawiera już wtyczkę C++, więc nie ma potrzeby instalowania żadnej specjalnej wtyczki. Aby zaimportować projekt, wybierz Plik ➤ Otwórz i przejdź do lokalizacji projektu: ~/Desktop/helloworld.

Skompiluj umowę i wygeneruj ABI

Jak wspomniano, narzędzie eosio-cpp pobiera kod C++ i generuje WebAssembly i ABI. Odbywa się to poprzez uruchomienie następującego polecenia:

> eosio-cpp -o helloworld.wasm helloworld.cpp --abigen

Zauważ, że w poleceniu określasz nazwę pliku wyjściowego, czyli helloworld.wasm. Po uruchomieniu tego polecenia kompilator generuje następujące pliki: helloworld.wasm i helloworld.abi. Aby upewnić się, że kompilator działał zgodnie z oczekiwaniami, powinieneś być w stanie zobaczyć te dwa pliki

Kontrakty Ricardiańskie

Po wygenerowaniu plików WASM i ABI zauważ, że otrzymujesz ponad 20 ostrzeżeń. Wśród tych ostrzeżeń w danych wyjściowych powinny znaleźć się następujące ostrzeżenia:

Ostrzeżenie, pusty plik klauzuli ricardiańskiej

Ostrzeżenie, pusty plik klauzuli ricardiańskiej

Ostrzeżenie, akcja <hello> nie ma umowy ricardiańskiej

Uwaga: kontrakty Ricardiańskie zostały wymyślone przez Iana Grigga w 1996 roku, aby wypełnić lukę między aplikacją a sądem. Plik kontraktów Ricardiana w EOS jest dokumentem cyfrowym w formacie języka Markdown (.md, .markdown) i określa warunki interakcji między stronami. Jest ustawiony jako parametry, ale zapisany jako czytelny tekst. EOS używa kryptograficznie do podpisywania i weryfikacji umów Ricardiańskich.

Aby pomóc w generowaniu kontraktów Ricardiańskich, możesz skopiować skrypt Pythona i szablon od kontrybutora, który automatycznie generuje pliki:

https://github.com/EOS-Mainnet/governance.

Ponieważ to tylko trzy pliki, zamiast klonowania możesz użyć wget, aby pobrać te pliki. Sprawdź, czy masz zainstalowany wget na swoim komputerze.

> wget --help

Jeśli nie jest zainstalowany, zainstaluj wget na macOS przez Ruby and Brew.

> ruby -e "$(curl -fsSL https://raw.githubusercontent.com/

Homebrew/install/master/install)" < /dev/null 2> /dev/null

> brew install wget

> brew upgrade wget

Następnie w swoim projekcie helloworld utwórz katalog i pobierz potrzebne pliki.

> cd ~/desktop/helloworld

> mkdir rc && cd $_

> wget https://raw.githubusercontent.com/EOS-Mainnet/

governance/master/scripts/abi_to_rc/abi_to_rc.py

> wget https://raw.githubusercontent.com/EOS-Mainnet/

governance/master/scripts/abi_to_rc/rc-action-template.md

> wget https://raw.githubusercontent.com/EOS-Mainnet/

governance/master/scripts/abi_to_rc/rc-overview-template.md

Next, run the Python script.

> cd ../

> python rc/abi_to_rc.py helloworld.abi

Skrypt generuje automatycznie helloworld-rc.md i helloworld-hello-rc.md już sformatowane w języku Markdown. Jeśli przejrzysz te pliki, zauważysz, że skrypt Pythona wykorzystał pobrane szablony do wygenerowania plików i możesz wypełnić warunki dotyczące inteligentnej umowy. Możesz określić wytyczne dotyczące tego, co dokładnie kupują/wymieniają Twoi użytkownicy i zapewnić większe zaufanie między stronami; może zawierać warunki, takie jak zamiar, gwarancja, środki zaradcze, siła wyższa, rozstrzyganie sporów, obowiązujące prawo i wiele innych. Zwróć szczególną uwagę na ustalone przez siebie warunki, ponieważ mogą one zostać wyegzekwowane w sądzie. Warunki te pozwalają pominąć pośredników, takich jak prawnicy, aby inteligentna umowa ustalała warunki, na które zgadzają się obie strony.

Wdróż kontrakt

Aby wdrożyć inteligentną umowę w lokalnej sieci testowej, polecenie kontraktu służy do załadowania kontraktu. Zobacz Rysunek 6-8 dla oczekiwanego wyniku.

> cleos set contract helloworld ~/Desktop/helloworld -p

helloworld@active

Wejdź w interakcję z działaniem Smart Contract

Teraz, gdy masz już wdrożoną inteligentną umowę w lokalnym łańcuchu bloków, możesz wchodzić w interakcję z utworzoną akcją powitania. Wywołasz akcję hello i przekażesz swoją nazwę użytkownika do aktywnego klucza użytkownika.

> cleos push action helloworld witam '["john"]' -p john@active

Inteligentne tokeny kontaktowe

Projekt EOS.IO GitHub posiada bibliotekę inteligentnych kontraktów jako przykładów, które można wykorzystać. Jedną z tych bibliotek jest inteligentna umowa o nazwie eosio. znak. Ta umowa umożliwia programistom tworzenie innych tokenów, a także przenoszenie tokena. Będziesz używać tych bibliotek do tworzenia własnych tokenów. Aby rozpocząć, utworzysz nowy projekt inteligentnej umowy i nazwiesz go eosio.token.

> mkdir ~/Desktop/eosio.token && cd $_

Utwórz konta

Token jest wydawany przez konto „wydawcy". Zaczniesz od utworzenia konta „wydawcy" i konta o nazwie jane, którego możesz użyć do przeniesienia niektórych tokenów.

> cleos create account eosio eosio.token [public key]

> cleos create account eosio jane [public key]

Skompiluj wasm z najnowszym kodem eosio.token

Aby wydać eosio.token, użyjesz pliku eosio.token.hpp, który definiuje klasę, akcje i tabele kontraktu, a także eosio.token.cpp, który przechowuje logikę i kodowanie. Możesz znaleźć te pliki i cały projekt SmartContract tutaj: https://github.com/Apress/theblockchain-developer/chapter6/eosio.token/. Następnie upewnij się, że zmieniłeś instrukcję include w kodzie CPP, aby wskazywała na plik HPP pobrany z GitHub za pomocą vima lub ulubiony edytor tekstu.

> vim eosio.token.cpp

Zmień plik eosio.token.cpp w wierszu 6, aby wskazywał lokalizację pliku eosio.token.hpp; w tym przypadku jest tutaj:

include "~/Desktop/eosio.token/eosio.token.hpp"

Wdróż eosio.token

Wyposażony w eosio.token.hpp i eosio.token.cpp, masz wszystkie potrzebne pliki. Możesz skompilować najnowsze pliki HPP i CPP, aby wygenerować kod .was m za pomocą polecenia eosio-cpp, tak jak w przykładzie z inteligentnym kontraktem HelloWorld.

> eosio-cpp -o eosio.token.wasm eosio.token.cpp --abigen

Następnie wdróż kontrakt eosio.token za pomocą polecenia set contract.

> cleos wallet unlock --password [DEFAULT_MASTER_KEY]

> cleos set contract eosio.token ~/Desktop/eosio.token --abi

eosio.token.abi -p eosio.token@

Utwórz token EOS.IO

Aby utworzyć nowy token, użyj akcji tworzenia. Przekażesz typ symbol_name, który zawiera dwa parametry.

- Maksymalny przepływ podaży: W tym przykładzie ustawisz to na 20 milionów jako swoje maksymalne tokeny: 20000000.0000.

- Symbol: W przypadku symbol_name musisz wybrać nazwę.

Nazwa musi składać się wyłącznie z wielkich liter; w tym przykładzie wybierz nazwę TOKEN. Konto „wydawcy" ma uprawnienia do wykonania akcji związanej z wywołaniem lub innych działań, takich jak wycofywanie, zamrażanie i umieszczanie właścicieli na białej liście. Aby utworzyć nową akcję tokena, uruchom następujące polecenie.

> cleos wallet unlock --password [DEFAULT_MASTER_KEY]

> cleos push action eosio.token create '["eosio",

"20000000.0000 TOKEN"]' -p eosio.token@active

Możesz potwierdzić, że tokeny zostały wydane, wywołując polecenie statystyki waluty.

> cleos get currency stats eosio.token TOKEN

{

"TOKEN": {

"supply": "0.0000 TOKEN",

"max_supply": "20000000.0000 TOKEN",

"issuer": "eosio"

}

}

Wydanie tokenów

Stwórzmy kolejne konto, za pomocą którego możesz wysłać część wydanych przez Ciebie tokenów. Nazwiemy to konto Jane.

> cleos create account eosio jane [public key]

Następnie wywołaj akcję „wydaj", aby wydać tokeny. W tym przykładzie wydasz Jane 500 tokenów.

> cleos push action eosio.token issue '["jane", "500.0000

TOKEN", "move tokens to Jane"]' -p eosio@active

Aby zobaczyć saldo TOKENÓW na koncie Jane, możesz użyć polecenia pobierz walutę.

> cleos get currency balance eosio.token jane TOKEN

500.0000 TOKEN

Przenieś tokeny

Aby przenieść tokeny, uruchamiasz akcję transferu. Jako przykład przenieśmy tokeny z konta Jane na konto Jana.

> cleos push action eosio.token transfer '["jane", "john",

"100.0000 TOKEN", "transfer tokens"]' -p jane@active

Możesz potwierdzić, że konto Jana otrzymało tokeny, uruchamiając polecenie salda walutowego na obu kontach, aby upewnić się, że matematyka się sumuje.

> cleos get currency balance eosio.token jane TOKEN

400.0000 TOKEN

> cleos get currency balance eosio.token john TOKEN

100.0000 TOKEN

Łączenie się z producentem publicznego bloku testnetowego

W chwili pisania tego tekstu EOS.IO udostępnia dwie publiczne sieci testowe, dzięki czemu można testować w bardziej realistycznym środowisku przed zatwierdzeniem kodu w sieci mainnet.

– Jungle2.0: https://jungletestnet.io/

– Kylin: https://www.cryptokylin.io/

W tym przykładzie wybrałem Jungle 2.0 jako publiczną sieć testową, ale zachęcam do przetestowania obu; to ten sam proces, tylko z różnymi punktami końcowymi. Sieć testowa EOS Jungle jest prawie identyczna z twoją lokalną siecią testową. Ty tylko musisz skonfigurować punkt końcowy Jungle API i wygenerować tokeny kranu EOS, aby zapłacić za utworzenie konta i wykorzystanie pamięci RAM.

Uwaga: Zawsze testuj w testnet przed opublikowaniem kodu w sieci mainnet. Tylko we wrześniu 2018 r. tokeny EOS o wartości 240 000 USD zostały skradzione z kont inteligentnych kontraktów EOSBet, a to z powodu błędu programowania inteligentnych kontraktów, który został wykorzystany przez hakerów, a nie błędów w samej platformie EOS.IO.

Aby utworzyć konto, wygenerujesz dwa domyślne uprawnienia: właściciel i aktywny. Możesz to zrobić na https://nadejde.github.i/eostoken-sale/ lub uruchamiając ten sam wiersz poleceń, którego użyłeś wcześniej dwukrotnie.

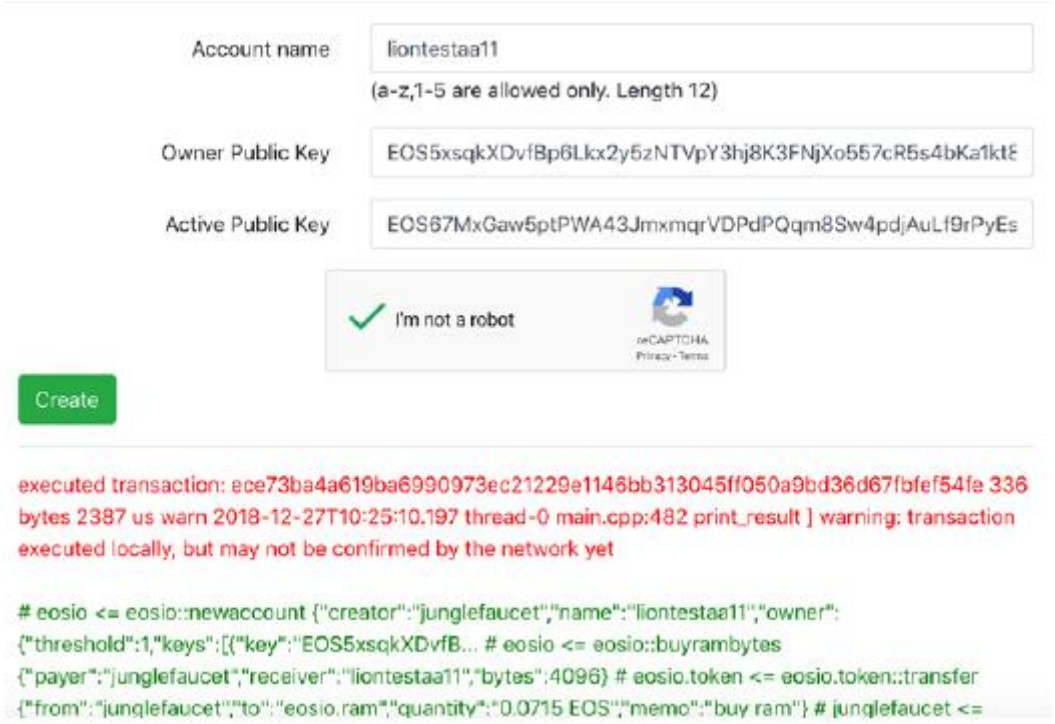
> cleos create key --to-console

Private key: [key]

Public key: [key]

Następnie musisz utworzyć konto. Możesz założyć konto odwiedzając stronę Jungle i korzystając z wygenerowanych kluczy publicznych: https://monitor.jungletestnet.io/#account.

Wybrałem losową nazwę liontestaa11, ale możesz użyć dowolnej nazwy imię chcesz. Tylko uważaj na ścisłe ograniczenia dotyczące nazw (a–z i 1–5 są dozwolone tylko z długością 12). Jeśli nie zastosujesz się do tego ścisłego ograniczenia nazwy, Twoje konto nie zostanie utworzone.



Zauważ, że otrzymujesz to samo ostrzeżenie, które otrzymałeś w lokalnej sieci testowej, dotyczące wykonywanej transakcji, ale nie jest to potwierdzone. Aby uzyskać informacje o sieci testowej, możesz uruchomić to samo polecenie get info, które uruchomiłeś dla lokalnej sieci testowej. Wystarczy dodać argument URL punktu końcowego Jungle.

> cleos --url https://jungle.eosio.cr:443 get info

Wszystkie polecenia cleos wymagają argumentu punktu końcowego adresu URL; możesz edytować swój plik bash, aby wskazywał cleos na żądany adres URL. Edytuj profil bash i wskaż publiczną sieć testową producenta bloku, jednocześnie wskazując na swój lokalny komputer dla portfela.

> vim ~/.bash_profile

Dodaj następujący wiersz:

alias cleos-testnet='cleos --url https://jungle.eosio.cr:443

-wallet-url http://localhost:8888'

Nie ustawiłeś tutaj pliku config.ini z niestandardowym portem, ale zmieniasz port na 8888. Pamiętaj, aby uruchomić profil bash w celu zatwierdzenia zmian, uruchom to:

> . ~/.bash_profil

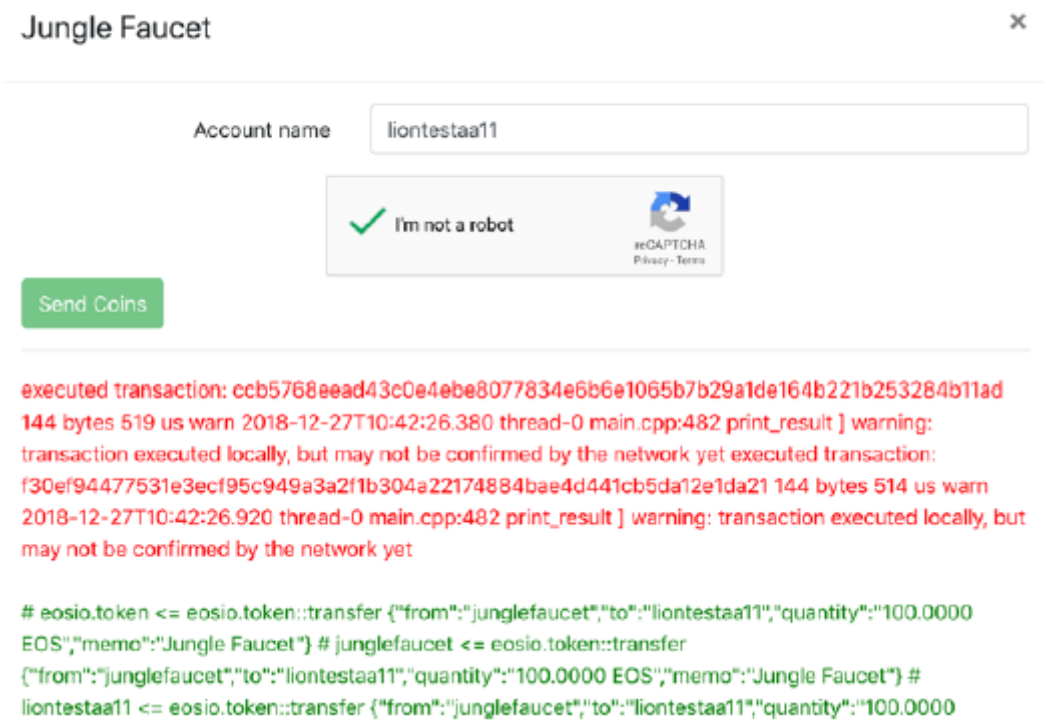
Teraz możesz uruchomić wszystkie polecenia za pomocą cleos-testnet.

> cleos-testnet get info

Kup alokację zasobów u producenta publicznego bloku testowego

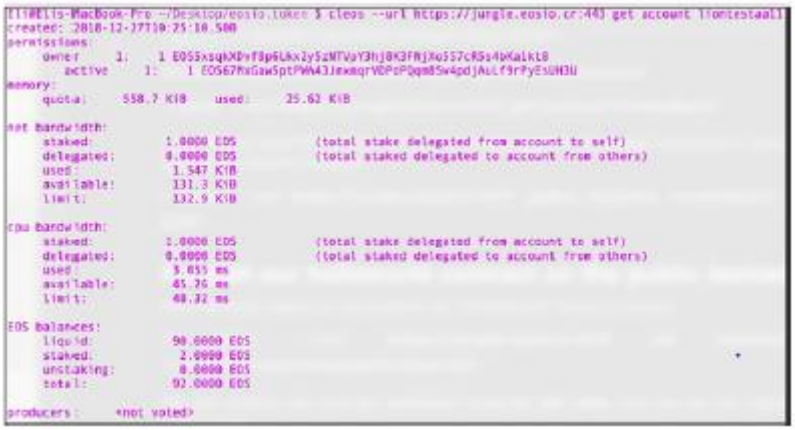
Teraz będziesz publikować inteligentną umowę „HelloWorld”, którą utworzyłeś w poprzedniej sekcji. Jeśli publikujesz swoją umowę w sieci mainnet, musisz kupić pamięć RAM i zapłacić, aby utworzyć konto, aby móc opublikować inteligentną umowę. Tokeny EOS służą do zakupu zasobów. W publicznej

sieci testowej nie musisz wydawać rzeczywistych pieniędzy na swoje zasoby. Otrzymujesz fałszywe żetony do kranu, które mogą być wykorzystane przez producenta klocków Jungle do zakupu twoich zasobów. Aby uzyskać te tokeny, potrzebujesz tylko nazwy konta. Wpisz nazwę swojego konta, aby otrzymać tokeny z kranu Jungle: http://monitor. jungletestnet.io/#faucet. Zobacz rysunek



Alokację zasobów wyjaśnię bardziej szczegółowo w następnej sekcji, gdy będziesz gotowy do opublikowania w sieci mainnet.

Saldo konta możesz sprawdzić za pomocą polecenia get account; zobacz dane wyjściowe na rysunku



> cleos --url https://jungle.eosio.cr:443 get account liontestaa11

Teraz, gdy masz tokeny EOS, możesz uruchomić polecenie cleos system buyram, aby kupić pamięć RAM, aby opublikować inteligentną umowę.

> cleos --url https://jungle.eosio.cr:443 system buyram

liontestaa11 liontestaa11 "10 EOS"

Opublikuj swój kontrakt HelloWorld w publicznej sieci testowej

Teraz, gdy masz już tokeny, możesz opublikować inteligentną umowę HelloWorld w publicznej sieci testowej. Uruchom polecenie set contract.

> cleos --url https://jungle.eosio.cr:443 set contract

liontestaa11 ~/Desktop/helloworld

Możesz potwierdzić, że kod został opublikowany za pomocą polecenia pobierz kod. Cały oczekiwany wynik można zobaczyć na rysunku

> cleos --url https://jungle.eosio.cr:443 pobierz kod liontestaa1

Łączenie z siecią Mainnet

Sieć główna EOS.IO jest prawie taka sama jak sieć testowa; wystarczy użyć innego punktu końcowego API i faktycznie zapłacić za konta i pamięć RAM za pomocą prawdziwych tokenów EOS. Istnieją trzy główne sposoby zdobycia tokenów EOS:

• Kopalnia: Tworzy producenta bloku i kopie EOS.

• Kup tokeny EOS: można je kupić na giełdach kryptowalut.

• Prezent: To otrzymuje EOS w prezencie od kogoś.

Jak widzieliśmy w poprzednich rozdziałach, tworzenie producenta bloków i bycie wybranym przez sieć EOS.IO nie jest procesem łatwym ani gwarantowanym, a ponieważ potrzebujesz tylko monet, aby kupić pamięć RAM do otwarcia konta i uzyskania zasobów, nie potrzebujesz za dużo monet. W tym momencie łatwo jest po prostu kupić te tokeny. Musisz najpierw kupić bitcoin, Ethereum lub inne monety na giełdzie fiducjarnej, takiej jak Coinbase, CEX.io lub Coinmama. Następnie użyj giełd, takich jak Binance lub Changelly, aby zamienić swoje monety na tokeny EOS. Powodem jest to, że w chwili pisania tego tekstu nie ma znanej wymiany, która mogłaby bezpośrednio zmienić twój fiat na EOS. Następnie potrzebujesz punktu końcowego. 21 wybranych producentów bloków jest w stanie zapewnić Ci punkt końcowy. Wszystkich dostępnych producentów bloków i inne dane dotyczące wydobywanych bloków można znaleźć tutaj:

– http://eosnetworkmonitor.io/

– https://eostracker.io/producers

Po znalezieniu producenta bloków, którego chcesz użyć, dołączasz

/bp.json na końcu adresu URL, aby znaleźć punkt końcowy. Oto przykład: https://api.eosnewyork.io/bp.json.

Wyjście JASON daje informacje o producencie bloku i

zapewnia gotowość do użycia. Aby ustawić adres URL, po prostu dostosuj flagę --url do producenta bloku, z którym chcesz się połączyć; pozostałe polecenia są takie same jak w publicznej sieci testowej.

> cleos --url https://api.eosnewyork.io:443 uzyskaj informacje

Tak jak poprzednio, możesz edytować plik profilu bash, tak jak w przypadku publicznej sieci testowej.

alias cleos-mainnet='cleos --url https://api.eosnewyork.io:443

--wallet-url http://localhost:8888'

Twój profil bash powinien wyglądać tak:

PATH="/usr/local/eosio/bin:${PATH}"

alias cleos-testnet='cleos --url https://jungle.eosio.cr:443

--wallet-url http://localhost:8888'

alias cleos-mainnet='cleos --url https://api.eosnewyork.io:443

--wallet-url http://localhost:8888'

Potwierdź, że działa, uruchamiając polecenie get info.

> cleos-mainnet get info

Oszczędzę Ci powtarzania tych samych kroków, co w sekcji publicznej sieci testowej i wydawania rzeczywistych tokenów na przykładowy inteligentny kontrakt „HelloWorld”. Jednak omówię alokację zasobów, ponieważ potrzebujesz dobrego zrozumienia tego, aby publikować inteligentne kontrakty w sieci mainnet.

Objaśnienie alokacji zasobów

Mówiłem trochę o alokacji zasobów, kiedy omawiałem sieci testowe, ponieważ trzeba było zdobyć tokeny EOS, aby opublikować inteligentną umowę w publicznej sieci testowej. W przypadku sieci mainnet potrzebujesz rzeczywistych tokenów EOS, aby kupić pamięć RAM i utworzyć konto. Istnieją trzy rodzaje zasobów zużywanych przez konta EOS.IO.

– Dysk: przepustowość i przechowywanie dziennika (dysk)

– CPU: Obliczanie tyczenia i backlog obliczeniowy (CPU)

– Ram: Przechowywanie stanu tyczenia

Kup pamięć RAM w Mainnet

Aby zwolnić pamięć RAM, musisz usunąć dane ze stanu konta

mechanizm, a następnie RAM można sprzedać na rynku RAM po aktualnej cenie RAM.

Utwórz konto EOS.IO w Mainnet

Konta EOS.IO są niezbędne, ponieważ są potrzebne do interakcji z siecią EOS.IO i utworzenia konta. Jak wyjaśniłem wcześniej, ktoś, kto już ma konto, musi poręczyć za utworzenie nowych kont. Jeśli nie masz osoby z kontem EOS, która może utworzyć Twoje konto, możesz uzyskać konto utworzone u dostawców zewnętrznych. Dostawcy zewnętrzni zwykle pobierają opłatę. Na przykład możesz pobrać EOS Lynx na swój telefon i zapłacić 2 USD, aby utworzyć konto EOS.IO.

Zmień klucze publiczne i prywatne swojego konta

Po uzyskaniu konta mainnet nie skończysz. Musisz upewnić się, że zmieniłeś swój klucz prywatny przed zasileniem konta, ponieważ usługa, która tworzy Twoje konto, może po prostu przechowywać Twoje klucze prywatne i zabrać Twoje środki. Znasz już wszystkie te kroki; jedyne nowe polecenie to remove_key, które usuwa stary klucz z portfela. Tworzysz nowy klucz, odblokowujesz portfel, resetujesz uprawnienia nowym kluczem i usuwasz stary klucz publiczny, a także importujesz nowy klucz prywatny. Wykonaj następujące kroki:

> cleos create key

> cleos wallet unlock

> cleos set account permission [ACCOUNT NAME] active [PUBLIC

KEY] owner -p [ACCOUNT NAME]@owner

> cleos set account permission MYACCOUNT owner [PUBLIC KEY] -p

[ACCOUNT NAME]@owner

> cleos wallet remove_key [OLD PUBLIC KEY]

> cleos wallet import [PRIVATE KEY]

Przydział procesora i przepustowości

Aby uzyskać przepustowość i procesor, musisz przydzielić tokeny EOS, a zasób będzie dla Ciebie dostępny automatycznie proporcjonalnie do kwoty posiadanej w okresie obowiązywania umowy tyczenia. Na przykład, w oknie stakowania powiedzmy, że chciałbyś zużyć 1 jednostkę procesora. Aby to zrobić, musiałbyś konkurować z innymi kontami, aby mieć na swoim koncie 0,1 procent wszystkich tokenów obciążających procesor lub aby ktoś inny przekazał te tokeny na twoje konto. Po okresie obstawiania zużyte zasoby zwalniają się i możesz ponownie wykorzystać te same postawione tokeny, więc nie ma potrzeby kupowania kolejnych tokenów EOS za każdym razem. Tokeny EOS można cofnąć po zakończeniu.

Dokąd się udać?

EOS.IO oferuje zasoby online z linkami; zobacz https://developers.eos.io. Zasoby dla programistów zawierają cenną dokumentację, a także informacje o innych narzędziach, których nie omówiłem, takich jak:

– Obsługa stanu: demux-js

– biblioteka JavaScript: eosjs

Polecam również zapoznanie się z przykładami inteligentnych kontraktów EOS GitHub, które pomogą Ci poznać wszystkie funkcjonalności i możliwości EOS.IO.

Streszczenie

Tu bardziej szczegółowo omówiłem blockchain EOS.IO. Skonfigurowałeś lokalne środowisko testnetowe, instalując biblioteki EOS.IO i EOSIO.CDT i nauczyłeś się konfigurować keosd i nodeos. Dowiedziałeś się już o portfelach EOS.IO, w tym o tym, jak tworzyć, usuwać i tworzyć kopie zapasowe portfeli, a także jak tworzyć portfel z niestandardowymi nazwami i wykonywać operacje, takie jak otwieranie, blokowanie i odblokowywanie portfela. Następnie omówiłem pary kluczy portfela oraz sposób obracania i ponownego obracania węzła lokalnego (nodeos), aby uruchomić lokalnego producenta bloków. Dowiedziałeś się o uprawnieniach aktywnych i właścicielskich, a także o uprawnieniach typu single-signature (single-sig) i multisignature (multisig). Aby zrozumieć inteligentne kontrakty EOS.IO, utworzyłeś inteligentny kontrakt i tokeny „HelloWorld", najpierw tworząc konta, a następnie pisząc kod C++. Następnie skompilowałeś i wygenerowałeś pliki WebAssembly i ABI, a także kontrakty Ricardian. Następnie nauczyłeś się, jak wdrażać utworzone kontrakty i wchodzić z nimi w interakcje. Po wygenerowaniu tokenów można było wydawać i przenosić tokeny między kontami. Kontynuowałeś, łącząc się z publicznym producentem bloków testnetowych, aby przetestować swoje inteligentne kontrakty w bardziej realistycznym środowisku, a na koniec dowiedziałeś się, jak łączyć się i publikować w sieci mainnet i poznałeś alokację zasobów w sieci EOS.IO. W następnym rozdziale omówię portfele blockchain NEO i inteligentne kontrakty NEO.