

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут телекомунікаційних систем

Кафедра телекомунікацій

До захисту допущено:

Завідувач кафедри

_____ Сергій КРАВЧУК

« ____ » _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія та програмування
інфокомунікацій»**

спеціальності 172 Телекомунікації та радіотехніка

**на тему: «Розробка системи моніторингу для Blockchain мережі на базі
Cosmos SDK»**

Виконав:

студент IV курсу, групи ТЗ-02

Кириндась Нікіта Петрович _____

Керівник:

Професор кафедри ТК НН ІТС, д.т.н., професор,

Романов Олександр Іванович _____

Консультант:

Старший викладач кафедри ТК НН ІТС, к.т.н.,

Маньківський Володимир Броніславович _____

Рецензент:

Доцент кафедри ІТТ НН ІТС, к.т.н., доцент

Астраханцев Андрій Анатолійович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут телекомунікаційних систем
Кафедра телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інженерія та програмування інфокомунікацій»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій КРАВЧУК

«___» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу студенту
Кириндасю Нікіті Петровичу

1. Тема роботи «Розробка системи моніторингу для Blockchain мережі на базі CosmoSDK», керівник роботи Романов Олександр Іванович, професор, д.т.н., затверджені наказом по університету від «22» травня 2024 р. № 2064-с.

2. Термін подання студентом роботи 10 червня 2024 р.

3. Вихідні дані до роботи: Використання інструментів Terraform та Docker для автоматизації процесів моніторингу. Збір та обробка даних у реальному часі. Візуалізація даних за допомогою Grafana. Інтеграція з блокчейн мережею на базі Cosmos SDK.

4. Зміст роботи:

Дослідити: типи та загальні характеристики блокчейну; проаналізувати можливі типи створення блокчейн моніторингу; реалізувати систему моніторингу для блокчейн мережі за допомогою інструментів; проаналізувати ефективність системи та протестувати.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

Слайд №1 Назва, мета, задачі бакалаврської роботи

Слайд №2 Теоретичні основи

Слайд №3 Архітектура системи моніторингу блокчейн мережі на базі COSMOS SDK

Слайд №4 Аналіз та вибір інструментів для реалізації

Слайд №5 Ескіз та опис дашборду моніторингу на базі Grafana

Слайд №6 Практична реалізація системи

Слайд №7 Аналіз системи ефективності та тестування

Слайд №8 Висновки та рекомендації

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Проектування системи моніторингу	Маньківський В.Б., старший викладач		

7. Дата видачі завдання 14.04.2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд та аналіз літератури	20.02.2024 -01.04.2024	Виконано
2	Робота над першим розділом	01.04.2024-12.04.2024	Виконано
3	Робота над другим розділом	14.04.2024-15.05.2024	Виконано
4	Робота над третім розділом	20.05.2023-02.06.2023	Виконано
5	Оформлення дипломної роботи	02.06.2024-08.06.2024	Виконано

Студент

Нікіта КИРИНДАСЬ

Керівник

Олександр РОМАНОВ

РЕФЕРАТ

Пояснювальну записку обсягом 69 сторінки. Робота включає 28 ілюстрацій, 2 додатки та 24 бібліографічних найменувань.

Мета роботи: Розробити систему моніторингу, яка забезпечить високу прозорість, безпеку та ефективність управління ресурсами блокчейн-мережі.

Методи: Використовувалися методи збору та обробки даних у реальному часі, автоматизації процесів моніторингу за допомогою інструментів Terraform та Docker, а також візуалізації даних за допомогою Grafana.

Результати:

- Розроблена система моніторингу дозволяє операторам мережі своєчасно реагувати на аномалії та оптимізувати роботу мережі.
- Впровадження автоматизованих процесів моніторингу зменшує потребу в ручному контролі та економить людські ресурси.
- Система забезпечує коректне відображення даних та захист від несанкціонованого доступу.
- Результати тестування підтвердили високу адаптивність і надійність системи.

Рекомендації: Розроблена система може бути використана для створення та підтримки блокчейн-мереж на базі Cosmos SDK. Подальший розвиток системи дозволить підвищити її ефективність та масштабованість.

Ключові слова: блокчейн, система моніторингу, Cosmos SDK, автоматизація, Docker, Terraform, Grafana, безпека, оптимізація, візуалізація даних, валідатор, мережеві ресурси, адаптивність, надійність, дашборд, API.

Explanatory note of 69 pages. The work includes 28 illustrations, 2 appendices, and 24 bibliographic references.

Objective: To develop a monitoring system that ensures high transparency, security, and efficiency in managing blockchain network resources.

Methods: Real-time data collection and processing methods were used, along with automation of monitoring processes using Terraform and Docker tools, and data visualization using Grafana.

Results:

- The developed monitoring system allows network operators to timely respond to anomalies and optimize network performance.
- Implementation of automated monitoring processes reduces the need for manual control and saves human resources.
- The system ensures correct data display and protection against unauthorized access.
- Test results confirmed the high adaptability and reliability of the system.

Recommendations: The developed system can be used for creating and maintaining blockchain networks based on Cosmos SDK. Further development of the system will enhance its efficiency and scalability.

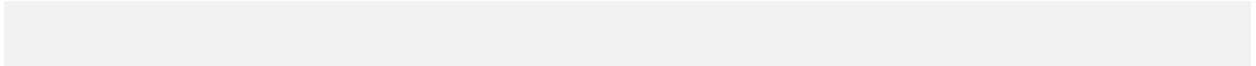
Keywords: blockchain, monitoring system, Cosmos SDK, automation, Docker, Terraform, Grafana, security, optimization, data visualization, validator, network resources, adaptability, reliability, dashboard, API.

	6
ВСТУП	9
РОЗДІЛ 1	10
АРХІТЕКТУРА БЛОКЧЕЙНУ	10
1.1 Загальні відомості Блокчейну	10
1.1.1 Історія створення блокчейну	11
1.1.2 Типи блокчейн мереж	13
1.1.3 Види консенсус алгоритмів	19
1.1.4 Введення в COSMOS SDK	23
1.1.5 Типи нод	24
1.1.6 Основні компоненти Cosmos SDK	28
Висновки	33
РОЗДІЛ 2	34
ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ	34
2.1 Визначення функціональних вимог	34
2.1.1 Архітектура системи моніторингу	36
2.2 Архітектура Node Exporter	37
2.2.1 Архітектура Grafana для візуалізації даних	39
2.2.2 Архітектура AWS для хостингу	40
Висновки	43
РОЗДІЛ 3	44
ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ	44
3.1 Написання Terraform файлів для розгортання на AWS системи	45
3.2 Створення Dockerfile та Docker-Compose для системи	52
3.3 Створення дашборда на базі Grafana	54

3.4 Ефективність та Тестування Дашборда для Blockchain Мережі на Базі Cosmos SDK	59
Висновки	60
ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
Додаток А	67
Додаток Б	69

ПЕРЕЛІК СКОРОЧЕНЬ

SDK	Software Development Kit
PoW	Proof of Work
PoS	Proof of Stake
PoA	Proof of Authority
AWS	Amazon Web Services
CLI	Command Line Interface
HCL	HashiCorp Configuration Language
VPC	Virtual Private Cloud
EC2	Elastic Compute Cloud



ВСТУП

У світі, де технології розвиваються надзвичайно швидко, поява та інтеграція блокчейн-технологій вимагає нових підходів до моніторингу та адміністрування мереж. BLOCKCHAIN вже став не лише новаторським способом транзакцій і зберігання даних, але й основою для створення нових форм фінансової взаємодії та діджиталізації активів. На тлі стрімкого зростання кількості транзакцій та учасників мережі, розробка надійної системи моніторингу для BLOCKCHAIN на базі COSMOS SDK стає нагальною потребою.

Ця робота присвячена створенню системи моніторингу, яка забезпечить високу прозорість та ефективність управління BLOCKCHAIN мережами. Розширення можливостей моніторингу відіграє ключову роль у підтримці безпеки, стабільності та швидкісної адаптивності блокчейн-мереж. Система, розроблена на основі COSMOS SDK, дозволить операторам мережі своєчасно реагувати на аномалії, оптимізувати роботу мережі та забезпечити безперебійну роботу в режимі реального часу.

Наразі, коли глобальний обсяг даних та вимоги до безпеки невпинно зростають, існує велика потреба в розвитку та адаптації нових технологій моніторингу, які можуть впоратися з викликами сучасного цифрового світу. У цьому контексті BLOCKCHAIN мережі на базі COSMOS SDK пропонують унікальні можливості завдяки своїй архітектурі, яка сприяє більш ефективному і гнучкому масштабуванню та взаємодії з різними блокчейн-протоколами.

Цей дослідження має на меті не тільки розробку системи моніторингу, але й глибше розуміння потенціалу COSMOS SDK як основи для створення масштабованих та надійних блокчейн-мереж, спроможних задовольнити сучасні вимоги до транзакційної прозорості та безпеки. Вивчення цих аспектів допоможе покращити інфраструктуру блокчейн-технологій і забезпечити їх стале впровадження у різноманітні галузі глобальної економіки.

РОЗДІЛ 1

АРХІТЕКТУРА БЛОКЧЕЙНУ

1.1 Загальні відомості Блокчейну

Історія блокчейну не така вже й давня. Перший крок блокчейну був започаткований у 1991 році Стюартом Хабером та В. Скоттом Сторнеттою, які працювали над криптографічно захищеним ланцюжком блоків, де ніхто не міг підмінити часові мітки документів. У 1992 році вони вдосконалили свою систему, додавши до неї дерева Меркла, що дозволило системі приймати більше документів в одному блоці. [1]



Рис.1.1. Стюартом Хабером та В. Скоттом [2]

Технологія блокчейн, яку зазвичай пов'язують з криптовалютами, такими як біткойн, функціонує як децентралізована база даних, що реєструє транзакції.

Ця база даних підтримується в глобальній мережі комп'ютерів, а не контролюється єдиним центральним органом, таким як банк. Така структура гарантує, що жодна особа чи організація не може змінити або видалити минулі транзакції, що підвищує безпеку і прозорість.

На відміну від традиційних централізованих баз даних, які розміщуються на одному сервері, блокчейн розподілений між користувачами програмного

забезпечення, що не дозволяє жодній стороні домінувати в мережі. Кожна транзакція в цій мережі перевіряється за допомогою комп'ютерних алгоритмів, які підтверджують її легітимність, перш ніж вона буде додана до існуючого ланцюжка транзакцій, таким чином формуючи так званий блокчейн.

Блокчейн працює на основі децентралізованої, однорангової мережі, що забезпечує відкритий доступ до записів транзакцій для всіх учасників мережі. Однією з відомих реалізацій технології блокчейн є біткойн, який підтримує цифрову книгу на своїй платформі. Цей реєстр дозволяє видобувати, зберігати і торгувати біткоїнами за допомогою складних алгоритмів у розподіленій мережі. Окрім обробки транзакцій, блокчейн також може слугувати реєстром та інвентаризацією різних активів.

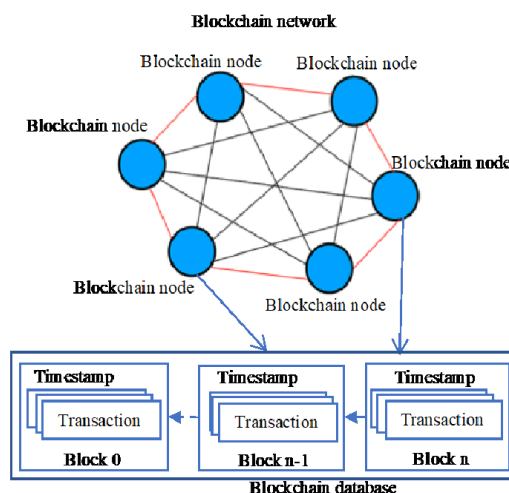


Рис.1.2. Типова архітектура,блокчейн мережі[3]

1.1.1 Історія створення блокчейну

Історія технології блокчейн відносно коротка, але значуща. Вперше концепція була розроблена в середині 1990-х років дослідниками Стюартом Хабером і В. Скоттом Сторнеттою, їхньою метою було надійне збереження історичних записів і захист цифрової інформації від підробки.

У їхній публікації 1991 року було представлено використання криптографічно захищеного ланцюжка блоків для збереження цілісності даних.

У 1993 році, у відповідь на зростання спаму та мережових зловживань, як контрзахід, було створено концепцію підтвердження виконання роботи (proof-of-work).

Переломний момент настав у 2008 році, коли загадкова постать Сатоші Накамото опублікував фундаментальний документ «Біткойн: пірингова система електронних грошей», який ознаменував початок ери блокчейну. У цьому документі детально описувалася нова форма валюти, біткойн, яка об'єднала криптографію, комп'ютерні науки та теорію ігор. (рис. 1.3)

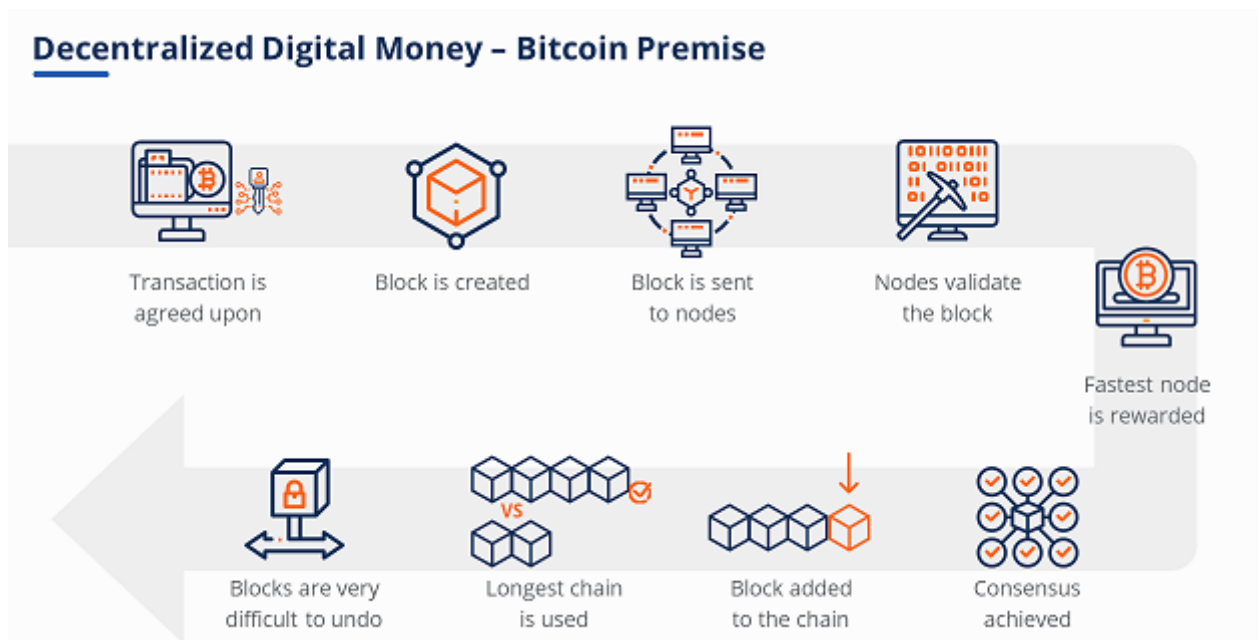


Рис.1.3. Концепція роботи Біткоїна [27]

У 2014 році з'явився Ethereum, який виводив інновації за рамки простої валюти чи реєстру транзакцій. Він містив віртуальну машину Ethereum, здатну виконувати обчислення і підтримувати смарт-контракти, що дозволило розробникам створювати децентралізовані додатки (dApps) безпосередньо на блокчейні.

До 2015 року біткойн набув значної популярності. У 2016 році технологія блокчейн почала привертати значну увагу, особливо у фінансовому секторі. У 2017 році Harvard Business Review запропонував блокчейн як фундаментальну

технологію. Сьогодні екосистема блокчейн динамічно розвивається і стрімко розширюється, з наростаючим імпульсом до децентралізованого майбутнього.

1.1.2 Типи блокчейн мереж

У світі цифрових транзакцій і децентралізованих систем, що стрімко розвивається, технологія блокчейн виділяється як фундаментальна інновація, що зробила революцію в різних галузях.

Цю технологію можна розділити на три основні типи, кожен з яких відповідає певним вимогам безпеки, конфіденційності та масштабованості. А саме публічні, приватні, консорціум та гібридні блокчейни.

Кожна форма слугує окремим цілям і підходить для різних бізнес- та операційних середовищ, що відображає різноманітність застосувань і широкий потенціал технології блокчейн.

Заглиблюючись у специфіку кожного типу, важливо розуміти унікальні особливості та переваги, які роблять ці блокчейни невід'ємною частиною сучасної цифрової інфраструктури.

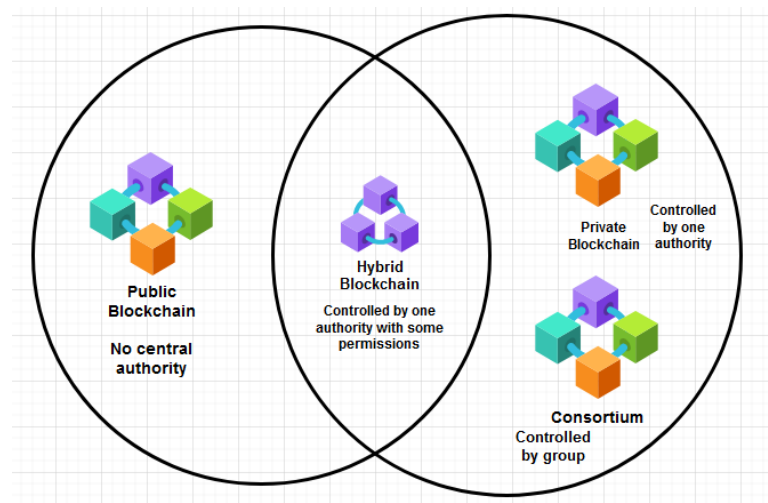


Рис. 1.10 Діаграма Ейлера типів блокчейн мереж

Публічний блокчейн - це один з різновидів технології блокчейн. Це розподілений реєстр з відкритим вихідним кодом, до якого може приєднатися

будь-хто і використовувати його для здійснення транзакцій. Кожен учасник мережі має копію цього реєстру. Крім того, будь-хто, хто має доступ до Інтернету, може переглянути публічний блокчейн.

Біткоїн був одним з перших публічних блокчейнів, доступних для широкого загалу, що дозволило здійснювати децентралізовані транзакції будь-кому, хто має доступ до інтернету. Для перевірки транзакцій публічні блокчейни використовують механізми консенсусу, такі як Proof-of-Work (PoW), Proof-of-Stake (PoS) та інші. Щоб публічний блокчейн працював ефективно, вузли-учасники повинні обробляти перевірку транзакцій. Без достатньої кількості однорангових вузлів, які обробляють транзакції, публічний блокчейн стане неефективним.

Різні блокчейн-платформи використовують ці типи блокчейнів як основу для своїх проектів, кожна з яких пропонує унікальні функції на додаток до стандартних.

Публічні блокчейни можуть служити різним цілям. Ось кілька прикладів:

- **Голосування:** Уряди можуть використовувати публічні блокчейни для забезпечення прозорих і надійних процесів голосування.
- **Збір коштів:** Компанії та проекти можуть використовувати публічні блокчейни для підвищення прозорості та довіри до своєї діяльності зі збору коштів.

Публічні блокчейни мають кілька переваг:

1. **Доступність:** Будь-хто може приєднатися до публічного блокчейну.
2. **Довіра:** Це сприяє зміцненню довіри між усією спільнотою користувачів.
3. **Мотивація:** Заохочує кожного робити свій внесок у розвиток мережі.
4. **Усунення посередників:** Усуває потребу в посередниках.
5. **Безпека:** Безпека публічних блокчейнів залежить від кількості вузлів-учасників.
6. **Прозорість:** Забезпечує прозорість всієї мережі, роблячи дані доступними для перевірки.

Публічні блокчейни також мають деякі недоліки:

1. **Швидкість транзакцій:** час виконання транзакцій може становити від кількох хвилин до кількох годин. Наприклад, Bitcoin може обробляти лише сім транзакцій на секунду, тоді як VISA може обробляти 24 000 транзакцій на секунду. Ця затримка пов'язана з часом, необхідним для завершення транзакції та розв'язання математичних задач.
2. **Масштабованість:** Публічні блокчейни мають проблеми з масштабуванням. Чим більше вузлів приєднується, тим повільнішою і громіздкішою стає мережа. Для вирішення цієї проблеми докладаються зусилля, наприклад, Біткоїн зосередився на мережевому освітленні, яке включає позамережеві транзакції для прискорення і масштабування основної мережі Біткоїна.
3. **Механізм консенсусу:** Вибір механізму консенсусу може бути недоліком. Наприклад, Біткоїн використовує Proof-of-Work (PoW), який є енергоємним. Однак, більш ефективні методи, такі як Proof-of-Stake (PoS), були введені, щоб частково вирішити цю проблему.

Приватний блокчейн - це тип технології блокчейн, який використовується в обмеженому середовищі, наприклад, в закритій мережі. Зазвичай цим дозволенням блокчейном керує одна організація.

Приватні блокчейни ідеально підходять для внутрішнього використання всередині приватної фірми або організації. Вони дозволяють контролювати доступ до мережі блокчейн, обмежуючи його кількома обраними користувачами. Відповідальна організація може встановлювати параметри мережі, такі як доступ і авторизація.

Приватні блокчейни відрізняються від публічних насамперед моделлю доступу. Незважаючи на цю відмінність, приватні блокчейни пропонують подібні функції до публічних блокчейнів, такі як прозорість, довіра та безпека для обраних учасників.

Однією з характерних рис приватних блокчейнів є високий рівень централізації, коли єдиний орган контролює моніторинг мережі. Як наслідок,

вони не є децентралізованими в традиційному розумінні. На основі приватних блокчейнів побудовано кілька блокчейн-платформ, кожна з яких має свої унікальні особливості.

Приватні блокчейни також називають дозволеними блокчейнами. Однак термін «дозволений блокчейн» ширший і може включати в себе як публічні, так і дозволені блокчейни.

Застосування приватного блокчейну

Приватні блокчейни мають безліч застосувань, зокрема:

1. **Управління ланцюгами поставок:** Управління ланцюжком поставок компанії.
2. **Власність на активи:** Моніторинг та перевірка права власності на активи.
3. **Внутрішнє голосування:** Сприяння внутрішнім процесам голосування.

Приватні блокчейни мають кілька переваг:

1. **Швидкість:** Приватні блокчейни працюють на високих швидкостях завдяки меншій кількості учасників, що дозволяє швидше досягати консенсусу та здійснювати транзакції.
2. **Масштабованість:** Вони більш масштабовані, ніж публічні блокчейни, оскільки лише обмежена кількість вузлів підтверджує транзакції. Це гарантує, що приватний блокчейн зберігає свою швидкість і ефективність у міру зростання мережі.
3. **Централізоване прийняття рішень:** Централізований контроль може впорядкувати процеси та прийняття рішень.

Незважаючи на свої переваги, приватні блокчейни також мають суттєві недоліки:

1. **Децентралізація:** Приватні блокчейни не є повністю децентралізованими, що суперечить фундаментальній концепції технології розподіленого реєстру та блокчейну.
2. **Довіра:** Побудова довіри в приватному блокчейні є складним завданням, оскільки рішення приймаються централізованими вузлами.

3. **Безпека:** Безпека може бути скомпрометована через обмежену кількість вузлів. Якщо певні вузли стають зловмисними, вони можуть підірвати механізм консенсусу в мережі.

Консорціумний блокчейн

Консорціумний блокчейн, також відомий як федеративний блокчейн, - це тип блокчейн-технології, який поєднує в собі особливості як публічних, так і приватних блокчейнів. Він пропонує унікальне рішення для підприємств, які потребують поєднання прозорості та конфіденційності.

У консорціумному блокчейні певні аспекти діяльності організацій-учасниць стають публічними, тоді як інші залишаються приватними. Консенсусна діяльність управляється заздалегідь налаштованими вузлами, зберігаючи децентралізовану природу, незважаючи на те, що вона недоступна широкому загалу. Таке децентралізоване управління досягається завдяки тому, що блокчейн контролюється кількома організаціями, а не одним центральним органом.

Консорціум працює з вузлами-валідаторами, які можуть підтверджувати, ініціювати або приймати транзакції, забезпечуючи безперебійну роботу. Вузли-учасники можуть як ініціювати, так і приймати транзакції, забезпечуючи гнучкість та ефективність.

Таким чином, консорціумний блокчейн забезпечує переваги приватного блокчейну, такі як прозорість, конфіденційність і ефективність, але при цьому не контролюється однією установою.

Переваги консорціумного блокчейну:

1. **Кастомізація та управління ресурсами:** Дозволяє створювати індивідуальні рішення та ефективно розподіляти ресурси.
2. **Безпека та масштабованість:** Більш безпечні та масштабовані, ніж публічні блокчейн-мережі, з підвищеною ефективністю.
3. **Обмеження доступу та управління:** Працює з визначеними рамками управління та контролем доступу.

Недоліки консорціумного блокчейну:

1. Недоброчесність учасників: Недоброчесність учасників може скомпрометувати всю мережу.
2. Прозорість: Менш прозорі порівняно з іншими типами блокчейнів.
3. Регулювання та цензура: Сприйнятливий до регулювання та цензури, що впливає на продуктивність мережі.
4. Анонімність: Пропонує менше анонімності, ніж інші типи блокчейнів.

Використання консорціумного блокчейну:

1. Банківська справа та платежі: Банки можуть створювати консорціум для контролю перевірки транзакцій.
2. Дослідження: Сприяє обміну даними та результатами досліджень.
3. Відстеження продуктів харчування: Корисно для відстеження ланцюгів постачання продуктів харчування.

Гібридний блокчейн - це тип блокчейн-технології, який поєднує в собі риси як приватних, так і публічних блокчейнів. Хоча він може здатися схожим на консорціумний блокчейн, він має чіткі відмінності. Гібридні блокчейни пропонують переваги як приватних, так і публічних блокчейнів, що робить їх придатними для організацій, які шукають збалансоване рішення, не віддаючи перевагу жодному з типів.

Використання гібридного блокчейну:

1. Нерухомість: Компанії, що займаються нерухомістю, можуть використовувати гібридні мережі для управління своїми системами, в той час як широка громадськість може отримати доступ до інформації.
2. Роздрібна торгівля: Рітейлери можуть використовувати гібридні мережі для оптимізації бізнес-процесів.
3. Регульовані сектори: Ідеально підходить для галузей з високим рівнем регулювання, таких як фінансові ринки.

Переваги гібридного блокчейну:

1. Закрите середовище: Працює в закритому середовищі, що дозволяє змінювати правила за потреби, не оприлюднюючи їх.

2. Безпека: Стійкий до 51% атак і забезпечує конфіденційність, залишаючись підключеним до публічної мережі.
3. Масштабованість: Забезпечує кращу масштабованість, ніж суто публічні мережі.

Недоліки гібридного блокчейну:

1. Прозорість: Не такий прозорий, як інші типи блокчейнів.
2. Проблеми модернізації: Перехід на гібридний блокчейн може бути складним через відсутність фінансових стимулів для участі та внеску в мережу.

1.1.3 Види консенсус алгоритмів

Технологія блокчейн докорінно змінила наше уявлення про безпеку, прозорість та децентралізацію даних. В основі кожного блокчейну лежить алгоритм консенсусу - ключовий компонент, який гарантує, що всі учасники мережі погоджуються з дійсністю транзакцій і станом реєстру. У цьому дипломі ми розглянемо чотири найпопулярніші алгоритми консенсусу, які сформували ландшафт технології блокчейн: Proof of Work (PoW), Proof of Stake (PoS) та Proof of Authority. Кожен з цих алгоритмів пропонує унікальні підходи до досягнення консенсусу, балансуючи такі фактори, як безпека, масштабованість, ефективність і децентралізація. Розуміння цих механізмів консенсусу є важливим для тих, хто хоче заглибитися у світ блокчейну та його застосувань.[6]

Proof of Work (PoW) був вперше запропонований Dwork і Naor в 1993 році для боротьби зі спамом і управління доступом до спільних ресурсів. У 2008 році Біткоїн включив цю концепцію в свій протокол як алгоритм консенсусу для перевірки транзакцій і запобігання подвійним витратам. PoW - це оригінальний і перший алгоритм консенсусу, який використовується в технології блокчейн.

Основна ідея PoW полягає в тому, щоб стимулювати майнерів використовувати свої обчислювальні потужності для вирішення складної математичної головоломки. Як тільки майнер вирішує головоломку, він транслює блок в мережу для перевірки іншими майнерами. Якщо рішення правильне, блок додається до блокчейну. Технічно, завдання майнера полягає в тому, щоб знайти випадкове число, яке називається *nonce*, в заголовку блоку. Майнер збільшує *nonce* до тих пір, поки не знайде хеш блоку з певною кількістю початкових нульових бітів, яка визначається параметром складності.

Хеш блоку створюється шляхом конкатенації даних блоку зі знайденим *nonce* і пропускання через хеш-функцію. Складність змінюється відповідно до загальної обчислювальної потужності в мережі, забезпечуючи постійну швидкість генерації нових блоків. У Біткоїна ця швидкість становить в середньому 10 хвилин на блок.

Багато інших криптовалют, таких як Ethereum і Litecoin, також використовують PoW. Рис. 1.11

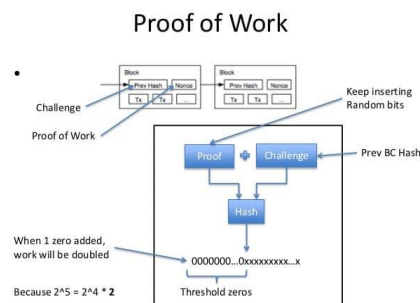


Рис. 1.11 Принцип роботи консенсусу PoW [7]

Proof of Stake (PoS) - це альтернативний алгоритм консенсусу, вперше реалізований у криптовалюті Peercoin, запропонований Кінгом і Надалом у 2012 році. На відміну від Proof of Work (PoW), який покладається на конкурентний, енергоємний процес, PoS обирає валідаторів на основі їхньої частки в мережі. У системі PoS валідатори, подібно до майнерів у системі PoW, обираються для перевірки нових блоків на основі їх права власності на монети, що вимірюється метрикою «вік монети». Вік монети обчислюється шляхом множення кількості валюти на час, протягом якого вона перебуває у власності. Після вибору

валідатор перевіряє блок, роблячи на нього ставку, і отримує винагороду, пропорційну ставці.

Хоча Peercoin була першою криптовалютою, яка реалізувала PoS, вона не покладається виключно на PoS; вона також включає PoW для початкового створення валюти, що робить її гібридною системою. На противагу цьому, Nxt є прикладом криптовалюти, яка використовує виключно PoS, не поєднуючи його з PoW. Крім того, Ethereum планує перейти на PoS у своїй версії 2.0, відомій як протокол Casper . Рис. 1.12

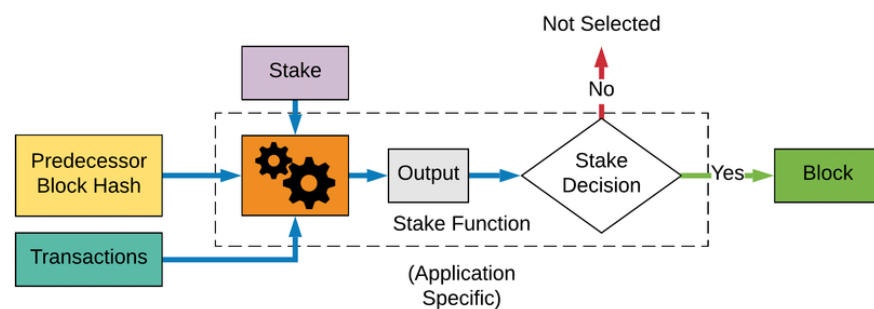


Рис. 1.12 Принцип роботи PoS[8]

Proof of Authority (PoA) - це механізм консенсусу, в якому лише заздалегідь визначені авторитети мають право підтверджувати транзакції та додавати блоки до блокчейну. Цей алгоритм визначає певний набір вузлів як валідаторів, надаючи їм повноваження оновлювати реєстр і захищати блокчейн. В результаті, згода централізована в руках кількох відомих учасників. PoA зазвичай використовується в приватних і консорціумних блокчейнах, де ці авторитети вважаються довіреними вузлами. На відміну від інших механізмів, що використовують монети, PoA покладається на особистість і репутацію валідаторів. Оскільки особи валідаторів відомі, вони зацікавлені діяти чесно, щоб зберегти свою репутацію.

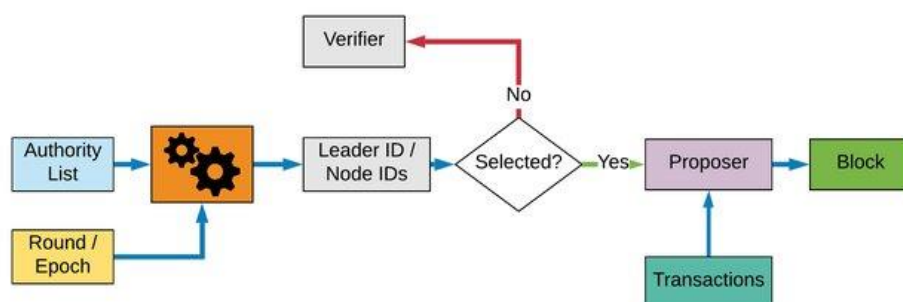


Рис. 1.12 Принцип роботи РоА[8]

1.1.4 Введення в COSMOS SDK

Cosmos SDK - це фреймворк з відкритим вихідним кодом, призначений для створення як публічних блокчейнів Proof-of-Stake (PoS), таких як Cosmos Hub, так і дозволених блокчейнів Proof-of-Authority (PoA). Блокчейни, розроблені за допомогою Cosmos SDK, зазвичай відомі як блокчейни для конкретних додатків.[9]

Основна мета Cosmos SDK - дозволити розробникам легко створювати власні блокчейни з нуля, які можуть безперешкодно взаємодіяти з іншими блокчейнами. Cosmos SDK задуманий як фреймворк, подібний до npm, що полегшує створення безпечних блокчейн-додатків на основі CometBFT.

Блокчейни, побудовані за допомогою SDK, складаються з модульних компонентів, більшість з яких мають відкритий вихідний код і легко доступні для розробників. Будь-хто може створити модуль для Cosmos SDK, а інтеграція готових модулів у ваш блокчейн-додаток є простою. Крім того, Cosmos SDK використовує систему, засновану на можливостях, яка підвищує безпеку, дозволяючи розробникам краще керувати взаємодією між модулями. Для більш детального розуміння зверніться до об'єктно-орієнтованої моделі можливостей.

Поширеним підходом до розробки в блокчейн-просторі є модель блокчейну з віртуальною машиною, наприклад, Ethereum, де розробка зосереджена на створенні децентралізованих додатків (dApps) на існуючому блокчейні за допомогою смарт-контрактів. Хоча смарт-контракти ефективні для певних додатків, наприклад, для одноразового використання, таких як ICO, вони часто не підходять для створення складних децентралізованих платформ через обмеження в гнучкості, суверенітеті та продуктивності.

Блокчейни, орієнтовані на конкретні додатки, представляють принципово іншу парадигму розробки. Ці блокчейни налаштовані на підтримку одного додатку, що дає розробникам повний контроль над вибором дизайну, необхідного для оптимальної продуктивності. Такий підхід забезпечує підвищену суверенність, безпеку та продуктивність.

Переваги Cosmos SDK

Cosmos SDK виділяється як провідний фреймворк для розробки кастомних блокчейнів для конкретних додатків. Ось кілька причин розглянути можливість використання Cosmos SDK для вашого децентралізованого додатку:

- 1. Механізм консенсусу CometBFT:** Механізм консенсусу за замовчуванням в Cosmos SDK, CometBFT, є найбільш зрілим механізмом консенсусу BFT. Він широко визнаний як галузевий стандарт для побудови систем Proof-of-Stake.
- 2. Відкритий вихідний код і модульний дизайн:** Cosmos SDK має відкритий вихідний код і призначений для спрощення побудови блокчейнів за допомогою модульних компонентів. З розширенням екосистеми модулів Cosmos SDK з відкритим вихідним кодом будувати складні децентралізовані платформи буде все простіше.
- 3. Безпека:** Cosmos SDK натхненний безпекою, заснованою на можливостях, і включає в себе уроки, отримані за роки розробки блокчейн-станів, забезпечуючи безпечне середовище для побудови блокчейнів.
- 4. Перевірений список:** Cosmos SDK вже використовувався для розробки численних прикладних блокчейнів, які зараз знаходяться у виробництві, включаючи Cosmos Hub, IRIS Hub, Binance Chain, Terra і Kava, а також багатьох інших, що знаходяться в стадії розробки.

1.1.5 Типи нод

Ноди в мережах блокчейн

Ноди - це пристрої, які зберігають копії даних блокчейну і додають нові блоки транзакцій до ланцюжка на основі механізму консенсусу.[10]

Технічно вузли складаються з:

1. *Апаратного забезпечення:* Це може бути віртуальний або фізичний сервер, а також обладнання для майнінгу.
2. *Програмне забезпечення:* Це дозволяє вузлу взаємодіяти з іншими

вузлами мережі і правильно виконувати свої функції.

Залежно від вимог блокчейну, вузли можуть бути розгорнуті на чому завгодно - від домашнього ПК до виділеного сервера з потужною конфігурацією.

Важливо зазначити, що у будь-якому блокчейні нода є фундаментальною структурною одиницею. Існування децентралізованої мережі неможливе без незалежних вузлів, саме тому розробники блокчейн-проектів часто заохочують встановлення вузлів на ранніх стадіях розробки.

Ключові функції нод

Блокчейн включає в себе різні типи вузлів для вирішення різних завдань, але ядро мережі складається з повноцінних вузлів, які виконують наступні функції.

1. **Верифікація блоків:** Майнери або валідатори обробляють транзакції та формують блоки, але кожен повноправний вузол мережі повинен перевірити сформований блок і або додати його до своєї копії історії транзакцій, або відхилити. Саме тому для реалізації хардфорків, які впливають на базові параметри блокчейну, необхідна підтримка всіх вузлів, інакше вони не зможуть перевіряти транзакції за новими правилами.
2. **Зберігання історії транзакцій:** Ноди записують і зберігають повну або часткову інформацію про транзакції на своїх дисках. Залежно від розміру та децентралізації блокчейну, таких копій можуть бути десятки, сотні або навіть тисячі, що практично виключає ризик незворотної втрати даних.
3. **Ретрансляція даних:** Після того, як вузол перевірів і додав блок до своєї копії бази даних, він може передати цю інформацію іншим вузлам, щоб забезпечити синхронізацію стану мережі та цілісність бази даних. Функція ретрансляції є важливою частиною безпеки блокчейну, оскільки дозволяє новим вузлам отримувати доступ до даних без дозволу.
4. **Підключення до блокчейну:** Всі дії в мережі, чи то ініціювання

транзакції, чи то виклик функції смарт-контракту, здійснюються через вузли. На ранніх стадіях розвитку Біткоїна взаємодіяти з мережею могли лише повноправні власники вузлів. Пізніше з'явилися провайдери інфраструктури, які запропонували свої вузли як канали доступу до блокчейну для легких гаманців та інших користувачів.

Ці завдання виконують всі основні вузли, але щоб краще зрозуміти, як працює блокчейн, давайте розглянемо всі існуючі типи вузлів і їх функції в складній структурі децентралізованої мережі.

Категорії вузлів на основі завдань та конфігурації

Вузли в мережах блокчейн поділяються на категорії залежно від їхніх завдань і конфігурації:

1. Легкі вузли:

Функція: Зберігають лише часткові дані про транзакції, тому не можуть самостійно перевіряти блоки і покладаються на повні вузли для синхронізації бази даних. Вони в першу чергу забезпечують доступ до блокчейну і часто використовуються приватними особами, які вважають за краще не залежати від сторонньої інфраструктури.

Переваги: Низькі технічні вимоги і простота налаштування.

Підтримка: Майже всі блокчейни підтримують легкі ноди.

2. Повні ноди:

Функція: Зберігають всю історію транзакцій, розширюючи можливості взаємодії з блокчейном. Повні вузли поділяються на підкатегорії:

- **Урізані повні вузли:** Зберігають лише частину історії транзакцій (наприклад, останні 20 ГБ). Вони слугують компромісом для користувачів, яким потрібно більше, ніж лайт-вузол, але підтримувати повну базу даних занадто дорого. Обрізані повні вузли підтримуються Bitcoin, Ethereum та багатьма іншими мережами.
- **Архівні повні вузли:** Зберігають повну історію транзакцій за весь час існування мережі. Вони дозволяють використовувати додаткові функції, такі як:

- **Майнінг-вузли:** Обробляють транзакції і формують блоки в мережах на основі PoW. Вони вимагають додаткової обчислювальної потужності (наприклад, ASIC) для обробки транзакцій і хешування. Популярними мережами для майнінг-вузлів є Bitcoin (високий поріг входу), Dogecoin і Litecoin (нижчий поріг входу).
- **Ноди стейкінгу:** Виконують аналогічні функції, як і майнінг-вузли, але в мережах на основі PoS. Вони мають більш високі технічні вимоги і вимагають зберігання певної кількості власних токенів (наприклад, 32 ETH для Ethereum).
- **Авторизаційні вузли:** Подібні до валідаторів в мережах PoA (VeChain) або PoSA (Binance Smart Chain). Ці вузли обробляють транзакції на основі репутації та схвалення спільноти, а не тільки обладнання або частки.
- **Майстер-ноди:** Не формують блоки, але виконують інші специфічні завдання. Їхні права та конфігурація залежать від конкретного блокчейну. Наприклад, в Dash тільки мастер-ноди можуть проводити миттєві та приватні транзакції. Майстер-ноди, як правило, мають вищі вимоги до розгортання, в тому числі, певну суму стейкінгу.

Додаткові категорії вузлів

1. Суперноди:

- **Функція:** Класифікуються або як просунуті повноцінні вузли, або як майстер-вузли з додатковою функціональністю. Вони можуть мати вищу пропускну здатність або спеціальні функції, але є менш поширеними через меншу децентралізацію мережі.
- **Варіант використання:** В основному використовуються в менш відомих блокчейнах, таких як Graft.

2. Lightning Nodes:

- **Функція:** Спеціальні вузли, що використовуються в мережі Lightning для біткоїна. Вони встановлюють і підтримують прямі платіжні канали між адресами на вершині основної мережі Біткоїн.

- Ресурси: Ви можете дослідити активні Lightning-вузли і дізнатися більше про них тут.

Ці категорії ілюструють різноманітність вузлів в мережі блокчейн, кожен з яких виконує певні функції для забезпечення функціональності та безпеки мережі Рис 1.13

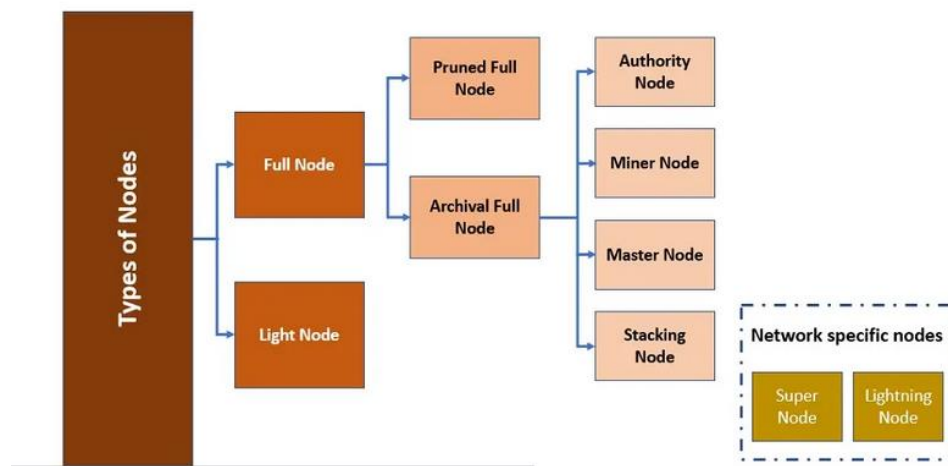


Рис. 1.13 Діаграма типів нод[8]

1.1.6 Основні компоненти Cosmos SDK

Само собою зрозуміло, що фреймворк складається з декількох дуже важливих компонентів, без яких блокчейн мережа не є мережою.

Тому, розглянемо декілька з цього списку разом з ілюстраціями основних компонентів.

Основні компоненти Cosmos SDK

Tendermint Core є серцем Cosmos SDK і забезпечує надійну і безпечну платформу для роботи блокчейнів. Це консенсусний механізм і мережевий рівень, який відповідає за синхронізацію блоків, збереження консенсусу і високу пропускну здатність мережі. Tendermint Core (рис.1.22) забезпечує швидку фіналізацію транзакцій і високу продуктивність, що є критично важливим для ефективної роботи сучасних блокчейнів.

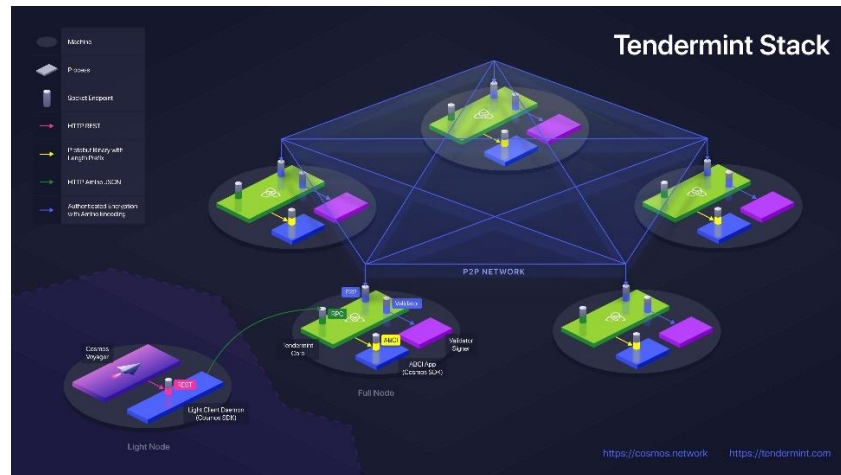
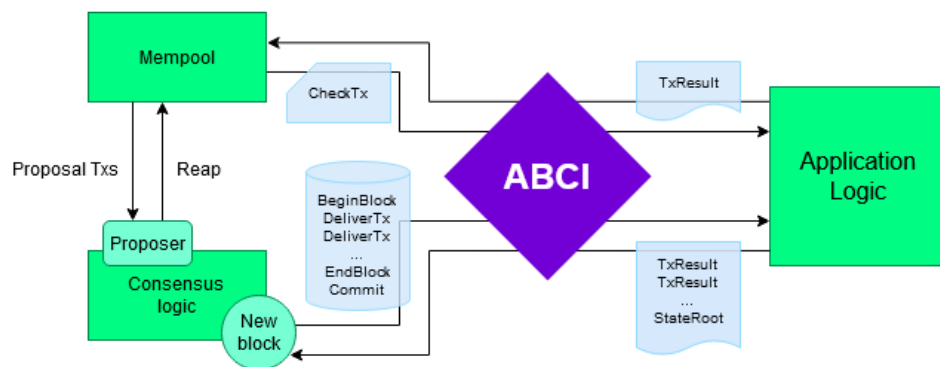


Рис.1.22. Архітектура Tendermint Core[12]

ABCI (Application Blockchain Interface) (рис.1.23) – це інтерфейс, який дозволяє розробникам використовувати різні мови програмування для створення децентралізованих додатків. ABCI є універсальним інструментом, що забезпечує гнучкість і зручність при розробці, дозволяючи інтегрувати будь-які мови програмування і технології в блокчейн.



Overview of the flow of messages via the ABCI protocol

Рис.1.23. Архітектура ABCI [13]

Cosmos SDK має модульну архітектуру, що дозволяє розробникам використовувати базові модулі для забезпечення стандартних функцій блокчейнів, таких як управління обліковими записами, транзакціями, стейкінгом і голосуванням.

Це значно спрощує процес розробки і дозволяє зосередитися на створенні унікальних функцій для конкретного проекту. Крім того, розробники можуть

створювати власні модулі для додавання специфічних функцій і можливостей у свої блокчейни, що дозволяє адаптувати їх під конкретні вимоги проекту.

Модель об'єктних можливостей (**Object-Capability Model**) рис.1.24 є ще одним важливим компонентом Cosmos SDK. Ця система дозволяє розробникам контролювати доступ до різних функцій і ресурсів блокчейну, забезпечуючи підвищену безпеку та ізоляцію модулів. Це дозволяє уникати конфліктів між модулями і забезпечує високу надійність і безпеку системи.

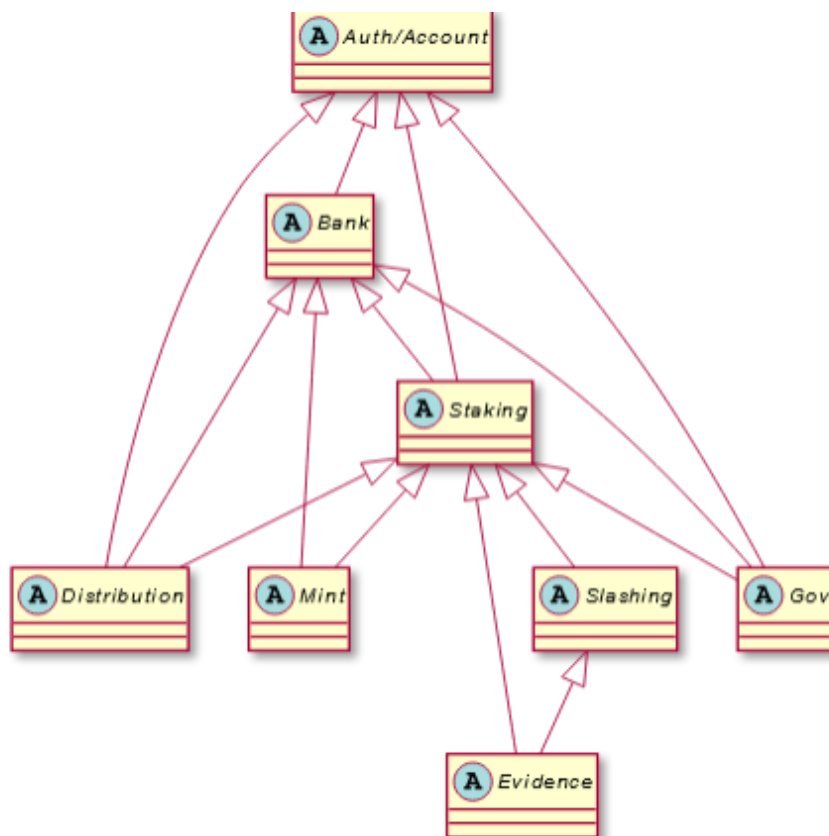


Рис.1.24 Приклад **Object-Capability Model** залежностей між зберігачами
[14]

Інструменти розробки, такі як **Cosmos SDK CLI (Command Line Interface)** і **Starport**, значно спрощують процес створення, тестування і управління блокчейнами. CLI надає набір команд для виконання основних операцій, а Starport автоматизує більшість рутинних завдань, що дозволяє розробникам зосередитися на логіці свого додатку. Це робить процес розробки

більш ефективним і зручним, знижуючи бар'єр для входу і скорочуючи час розробки.

Cosmos SDK також пропонує мережеві інструменти, такі як **IBC (Inter-Blockchain Communication)**, рис.1.25 що забезпечує взаємодію між різними блокчейнами. IBC дозволяє безпечно передавати дані і активи між блокчейнами в екосистемі Cosmos, забезпечуючи високу інтероперабельність і розширюючи можливості для створення децентралізованих додатків і сервісів.

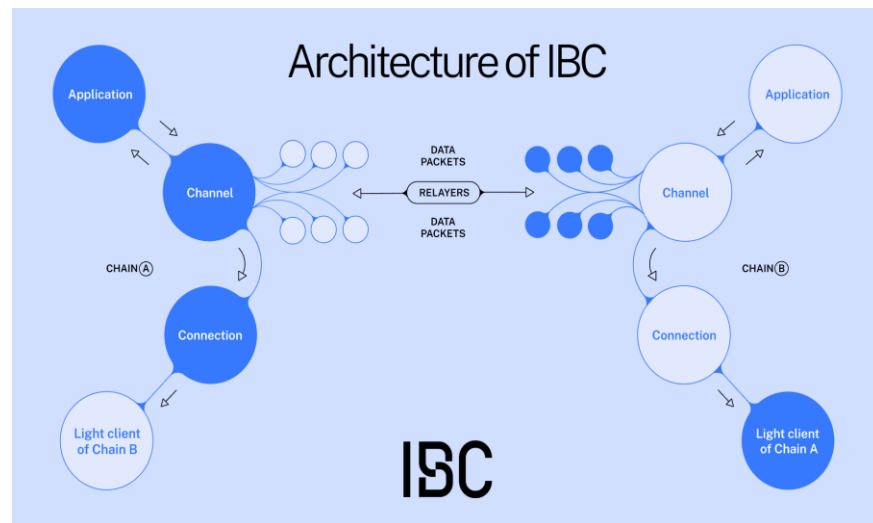


Рис.1.25 Архітектура IBC[15]

Безпека та відповідність є ключовими аспектами Cosmos SDK. Інструменти для аудиту та тестування дозволяють перевірити безпеку блокчейнів і тестувати нові функції перед їх впровадженням у продуктивну мережу. Відстеження та моніторинг стану мережі, продуктивності і безпеки блокчейнів забезпечують високу надійність і стабільність системи.

Валідатори рис.1.26 відіграють критичну роль у забезпеченні безпеки та стабільності мережі Cosmos. Вони підтверджують транзакції та додають нові блоки у блокчейн, виконуючи ключові функції для підтримки консенсусу.

У PoS-системах валідатори обираються на основі їхньої частки (stake) у мережі. Чим більше токенів застейкано, тим більша ймовірність бути обраним валідатором.

Валідатори отримують винагороди за свою роботу у вигляді нових токенів або транзакційних комісій. Підпис блоків валідаторами забезпечує їхню

дійсність та додає їх у блокчейн, гарантує правильність усіх транзакцій і підтримує консенсус у мережі.

Popular

Active

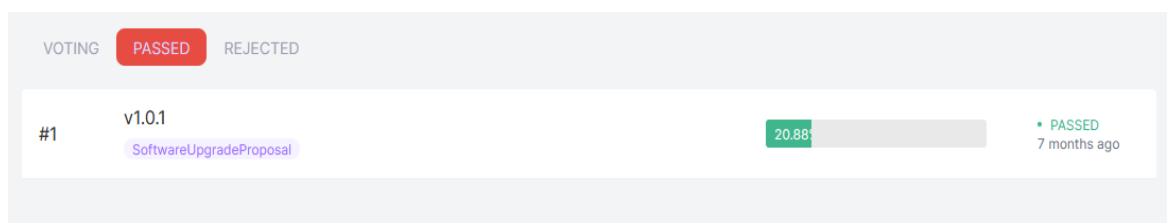
Inactive

81/100

RANK	VALIDATOR	VOTING POWER	24H CHANGES	COMMISSION	ACTIONS
1	 validator3 -	90,005 SWTR 10.43%		10%	
2	 validator2 -	90,005 SWTR 10.43%		10%	
3	 validator -	90,001 SWTR 10.43%		10%	
4	 ahmkah https://github.com/ahmkah	10,331 SWTR 1.2%		20%	
5	 Nodes Hub https://nodeshub.online	10,290 SWTR 1.19%		10%	
6	 GentlemenValidators https://t.me/gentlemenvalidators	10,186 SWTR 1.18%		5%	Delegate
7	 NodeWarrior https://nodewarrior.xyz	10,167 SWTR 1.18%		5%	Delegate
8	 StakingCabin https://stakingcabin.com	10,121 SWTR 1.17%		5%	Delegate

Рис.1.26 Приклади валідаторів в мережі

Пропозиції та голосування рис.1.27 є важливими компонентами системи управління в Cosmos SDK. Користувачі можуть вносити пропозиції щодо змін у мережі, таких як оновлення протоколу або зміни в економічних параметрах. Пропозиції мають бути формалізовані і представлені на розгляд спільноти. Учасники мережі голосують за або проти пропозицій, використовуючи свої застейкані токени. Результати голосування визначають, чи буде пропозиція прийнята, що забезпечує децентралізоване управління мережею.



VOTING		PASSED	REJECTED
#1	v1.0.1 SoftwareUpgradeProposal	20.88%	PASSED 7 months ago

Рис.1.27 Приклади голосування в мережі

Блоки є основними структурними одиницями блокчейна, що містять інформацію про транзакції. Кожен блок включає хеш попереднього блоку, забезпечуючи цілісність і незмінність ланцюга. Висота блоку вказує на його

позицію в ланцюзі блоків, починаючи з генезисного блоку (висота 0). Висота блоку використовується для відстеження прогресу ланцюга і перевірки послідовності блоків, що є критично важливим для підтримки цілісності та правильності блокчейна.

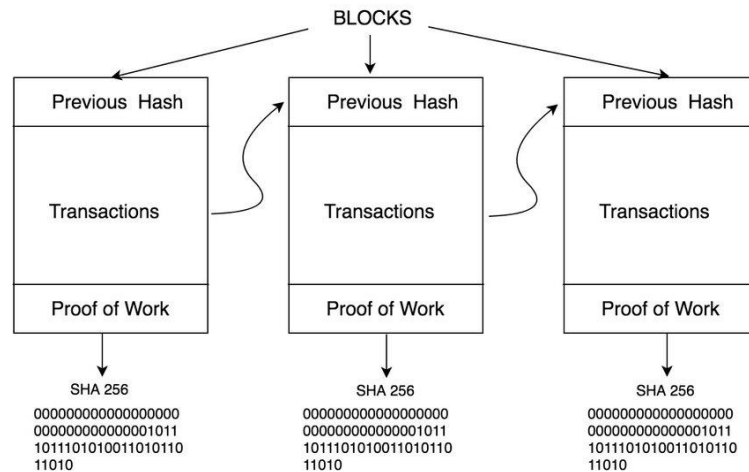


Рис.1.28 Вміст блоку та структура[16].

Ці компоненти складають основу Cosmos SDK, надаючи розробникам усі необхідні інструменти для створення потужних, безпечних і гнучких блокчейнів, що можуть взаємодіяти один з одним у масштабованій децентралізованій екосистемі.

Висновки

В даному розділі було розглянуто архітектуру Інтернету речей та її складові; технології, які використовує IoT за для забезпечення ефективної роботи та виконання поставлених задач. Було розглянуто два основних стільникових стандарти, які було створено для IoT та найпопулярніші протоколи, які використовуються в IoT. Було також докладно розглянуто застосування IoT та як це вплинуло на сучасне життя.

РОЗДІЛ 2

ПРОЕКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ

Проектування системи моніторингу є невід'ємною складовою сучасних інформаційних технологій, що спрямовані на забезпечення безперервного нагляду, контролю та аналізу діяльності систем і процесів. З розвитком цифрових технологій та зростанням обсягів даних, ефективні інструменти моніторингу стають ключовими елементами для підтримки надійності, безпеки та продуктивності будь-якої організації чи інфраструктури.

Моніторингова система дозволяє своєчасно виявляти відхилення від нормальної роботи, прогнозувати можливі проблеми, а також оперативно реагувати на інциденти. Вона забезпечує збір, обробку та аналіз даних у реальному часі, що дозволяє приймати обґрунтовані рішення на основі актуальної інформації. Таким чином, проектування ефективної системи моніторингу є надзвичайно важливим для забезпечення стабільного функціонування інформаційних систем. Блокчейн-технології на сьогоднішній день отримали широке визнання завдяки своїй децентралізованій природі, високій безпеці та прозорості. Однак, зважаючи на складність та різноманітність блокчейн-систем, моніторингова система повинна бути спроектована таким чином, щоб забезпечити повне охоплення усіх критичних параметрів та процесів.

2.1 Визначення функціональних вимог

Загальні вимоги

Система моніторингу повинна бути розроблена для блокчейну на базі COSMOS SDK з консенсусом Po, що забезпечить її інтеграцію з іншими блокчейн мережами, побудованими на цій платформі. Веб-додаток, який буде інтерфейсом для користувачів, має бути зручним та інтуїтивно зрозумілим, що дозволить користувачам легко взаємодіяти із системою. Крім того, система

повинна бути розширюваною, щоб у майбутньому можна було додавати нові функціональності без значних зусиль.

Вимоги до користувацького інтерфейсу

Інтерфейс системи повинен забезпечувати авторизацію та аутентифікацію користувачів, що гарантує безпеку доступу до конфіденційної інформації. Користувацький інтерфейс має бути розроблений з урахуванням принципів зручності використання, з метою забезпечення легкості у навігації та взаємодії з основними функціями системи. Дашборд, який буде основним елементом інтерфейсу, повинен відображати ключові показники стану блокчейн мережі.

Вимоги до функціональності моніторингу

Основна функціональність системи моніторингу полягає у зборі та відображенні метрик, які характеризують стан блокчейн мережі. Система повинна збирати та відображати такі показники, як кількість блоків, історію пропущених блоків валідатора, загальну інформацію про валідатора. Крім того, користувачі повинні мати можливість налаштовувати сповіщення про критичні події в мережі. Це забезпечить оперативне реагування на потенційні проблеми.

Вимоги до безпеки

Безпека є одним з ключових аспектів будь-якої системи моніторингу, особливо коли йдеться про блокчейн мережі. Система повинна забезпечувати захист даних користувачів та метрик блокчейн мережі. Реалізація механізмів захисту від несанкціонованого доступу є обов'язковою. Крім того, система повинна мати можливість резервного копіювання даних, щоб запобігти їх втраті у випадку збоїв.

Вимоги до продуктивності

Система повинна бути здатна обробляти великий обсяг даних в режимі реального часу, що є важливим для ефективного моніторингу блокчейн мережі. Висока доступність та надійність системи є критичними вимогами, оскільки користувачі повинні мати змогу отримувати актуальну інформацію про стан мережі в будь-який момент часу.

Вимоги до інтеграції

Система повинна підтримувати інтеграцію з іншими блокчейн мережами на базі COSMOS SDK, що дозволить розширити її функціональність та застосування. Крім того, повинна бути можливість інтеграції з іншими системами моніторингу та аналітики, що забезпечить комплексний підхід до управління блокчейн мережею.

2.1.1 Архітектура системи моніторингу

На схемі Рис. 2.1 представлена архітектура системи моніторингу для блокчейн мережі. Ця архітектура включає кілька ключових компонентів, які взаємодіють між собою для забезпечення збору, обробки, аналізу та візуалізації даних з блокчейн мережі.

Компоненти системи:

Github

1. Використовується для зберігання вихідного коду.

Terraform

1. Інструмент для управління інфраструктурою як кодом (IaC).
2. Використовується для автоматичного розгортання інфраструктури системи моніторингу.

Docker

1. Платформа для контейнеризації додатків.
2. Використовується для упаковки компонентів системи моніторингу у контейнери, що забезпечує їх портативність та ізолюваність.

Prometheus

1. Система моніторингу та збору метрик.
2. Використовується для збору метрик з вузлів блокчейн мережі та інших компонентів системи.
3. Забезпечує можливості для запитів та аналізу зібраних даних.

Grafana

1. Платформа для візуалізації даних.
2. Використовується для створення дашбордів та графіків на основі даних, зібраних Prometheus.
3. Забезпечує зручний інтерфейс для аналізу метрик та моніторингу стану системи.

Blockchain REST API

1. Інтерфейс для взаємодії з блокчейн мережею.
2. Використовується для отримання даних з блокчейн мережі, таких як стан транзакцій, кількість блоків тощо.

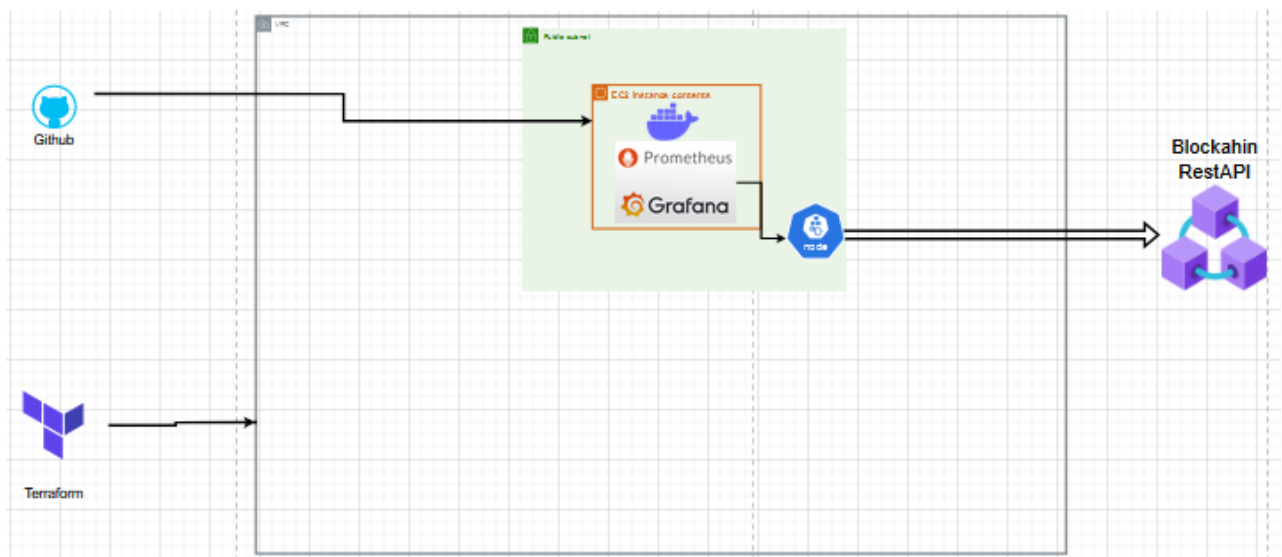


Рис. 2.1 .Загальна архітектура Моніторингу

2.2 Архітектура Node Exporter

Prometheus Node Exporter рис. 2.2. є невеликим агентом, який встановлюється на кожному хості, що потребує моніторингу. Його головне завдання – збір детальних метрик про стан операційної системи та апаратного забезпечення. Node Exporter збирає інформацію про використання процесора, пам'яті, дискових підсистем, мережевих інтерфейсів та інші критичні показники.

Процес збору та експорту метрик[17]:

1. Збір метрик:

Node Exporter безперервно збирає метрики з хоста. Це включає в себе збір даних про використання центрального процесора (ЦП), завантаженість оперативної пам'яті, стан дискових підсистем, активність мережевих інтерфейсів та інші показники. Зібрані дані є надзвичайно важливими для розуміння поточного стану системи та виявлення потенційних проблем.

В нашому випадку, метрики збираються з API ендпоінта, в якому зберігається вся інформація про блокчейн мережу.

Приклад запиту

1. Експортування метрик:

Зібрані метрики зберігаються у вигляді часових рядів. Node Exporter відкриває HTTP-сервіс на певному порту (зазвичай порт 9100), де ці метрики стають доступними для збору. Кожна метрика має свій унікальний URL, що дозволяє легко отримати доступ до конкретної інформації.

2. Запити з Prometheus:

Система моніторингу Prometheus періодично надсилає HTTP-запити до Node Exporter, щоб отримати актуальні метрики. Ці запити надсилаються до визначеного API Endpoint, що вказує на адресу Node Exporter на кожному хості. Prometheus збирає ці метрики та зберігає їх у своїй базі даних для подальшого аналізу.

3. Обробка та аналіз метрик:

Отримані метрики зберігаються у Prometheus і можуть бути використані для створення різноманітних дашбордів та графіків. Це дозволяє адміністраторам системи швидко оцінювати поточний стан інфраструктури, виявляти аномалії та реагувати на потенційні проблеми. Крім того, Prometheus підтримує налаштування сповіщень, що дозволяє автоматично повідомляти про критичні події.

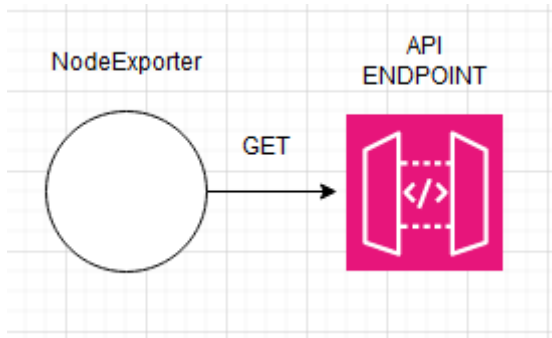


Рис. 2.2. Архітектура Cosmos NodeExporter

2.2.1 Архітектура Grafana для візуалізації даних

Цей дашборд Grafana показує метрики валідатора. Ось ключові елементи архітектури рис 2.3:

Uptime	Validator Name	Validator Status	Total Stake
Missed Blocks History		Uptime History	

Рис. 2.3. Ескіз дашбоарда

Верхній лівий кут:

Validator Uptime: Колесоподібний індикатор, що показує відсоток часу, коли валідатор був активний.

Центральна частина:

- **Validator Name:** Ім'я валідатора
- **Jail Status:** Стан валідатора, чи він у в'язниці
- **Active Status:** Чи активний валідатор
- **Validator Commission:** Відсоток комісії валідатора
- **Rank:** Рейтинг валідатора
- **Validators Count:** Загальна кількість валідаторів

Верхній правий кут:

- **Tokens Staked:** Кількість токенів, поставлених на валідатора

Нижня частина:

- **Missed Blocks:** Графік, що показує кількість пропущених блоків за певний період.
- **Uptime History:** Графік історії часу роботи валідатора, що відображає зміни uptime з часом.

2.2.2 Архітектура AWS для хостингу

У сучасному світі управління IT-інфраструктурою все більше уваги приділяється автоматизації та оркестрації. Один з найпопулярніших інструментів для цього - Terraform від HashiCorp, який дозволяє керувати інфраструктурою як кодом (IaC). У даній дипломній роботі розглядається архітектура хостингу веб-додатків на базі AWS з використанням Amazon EC2, VPC, Security Groups де Terraform використовується для автоматизації процесів розгортання та управління інфраструктурою.

Використання Terraform

Terraform є потужним інструментом для створення, оновлення та управління інфраструктурою через конфігураційні файли. Основні етапи використання Terraform в даній архітектурі включають:

Написання конфігураційних файлів:

Всі ресурси AWS, такі як VPC, EC2, Security Groups, описуються в конфігураційних файлах Terraform з використанням HashiCorp Configuration Language (HCL).

Переваги використання Terraform

Автоматизація та консистентність:

Використання Terraform дозволяє автоматизувати процеси створення та управління інфраструктурою, забезпечуючи консистентність середовища. Це зменшує ризик людських помилок і спрощує розгортання нових середовищ.

Інфраструктура як код (IaC):

Інфраструктура описується як код, що дозволяє зберігати конфігурації в системах контролю версій (наприклад, Git). Це забезпечує прозорість змін та можливість відстеження історії модифікацій.

Масштабованість:

Terraform спрощує процес масштабування інфраструктури, дозволяючи легко додавати або видаляти ресурси відповідно до потреб додатка.

Взаємодія з різними провайдерами:

Terraform підтримує безліч хмарних провайдерів і сервісів, що дозволяє використовувати його для управління мультихмарними середовищами.

Планування змін:

Команда terraform plan дозволяє попередньо переглянути плановані зміни до інфраструктури, що допомагає уникнути непередбачених наслідків.

Повторюваність:

Конфігураційні файли можуть бути повторно використані для створення однакових середовищ у різних регіонах або для різних етапів розробки (наприклад, staging і production), яка включає ключові компоненти AWS: Amazon EC2, Amazon VPC, Security Groups,.

Amazon VPC (Virtual Private Cloud)

Опис: Віртуальна приватна хмара, що забезпечує ізольоване мережеве середовище для ресурсів AWS.

Функції: Налаштування підмереж, маршрутів, інтернет-шлюзів та NAT-шлюзів для управління трафіком.

Amazon EC2 (Elastic Compute Cloud)

Опис: Віртуальні сервери, на яких розгортається веб-додаток.

Функції: Обчислювальні ресурси для запуску додатків, масштабування залежно від навантаження.

Security Groups

Опис: Віртуальні брандмауери для контролю вхідного та вихідного трафіку інстансів EC2.

Функції: Забезпечують контроль доступу до інстансів EC2, обмежуючи трафік на основі IP-адрес та портів.

Архітектура та взаємодія компонентів

Amazon VPC

Налаштування: Архітектура починається з налаштування VPC, де створюються публічні та приватні підмережі.

Підмережі: Публічні підмережі містять інстанси EC2, доступні через Інтернет. Приватні підмережі використовуються для ресурсів, які не потребують прямого доступу з Інтернету (наприклад, DynamoDB).

Amazon EC2

Розгортання: Веб-сервери запускаються на інстансах EC2 у публічних підмережах.

Авто-масштабування: Використання груп авто-масштабування для автоматичного додавання або видалення інстансів залежно від навантаження.

Security Groups

Налаштування: Security Groups налаштовуються для контролю вхідного та вихідного трафіку до інстансів EC2.

Політики: Вхідні правила можуть обмежувати доступ за IP-адресами та портами (наприклад, дозволяти HTTP/HTTPS трафік).

Висновки

Система моніторингу є важливою складовою будь-якої блокчейн мережі, що забезпечує її стабільну та безпечну роботу. Розроблена в даній роботі система моніторингу для блокчейн мережі на базі COSMOS SDK з консенсусом PoS відповідає сучасним вимогам та має наступні характеристики:

- **Інтеграція з іншими блокчейн мережами:** Система може бути інтегрована з іншими блокчейн мережами на базі COSMOS SDK, що робить її універсальною та масштабованою.
- **Зручний веб-інтерфейс:** Користувачкий інтерфейс системи простий у використанні та інтуїтивно зрозумілий, що дозволяє користувачам легко отримувати доступ до інформації про стан мережі.
- **Комплексний моніторинг:** Система збирає широкий спектр метрик, що характеризують стан блокчейн мережі, включаючи кількість блоків, історію пропущених блоків валідатора, загальну інформацію про валідатора, тощо.
- **Налаштовувані сповіщення:** Користувачі можуть налаштовувати сповіщення про критичні події в мережі, що забезпечує оперативне реагування на potential problems.
- **Високий рівень безпеки:** Система використовує сучасні методи захисту даних та доступу, щоб гарантувати безпеку інформації.
- **Масштабованість та продуктивність:** Система здатна обробляти великий обсяг даних в режимі реального часу, що робить її надійним інструментом для моніторингу блокчейн мережі з великою кількістю користувачів.
- **Інтеграція з іншими системами:** Система може бути інтегрована з іншими системами моніторингу та аналітики, що забезпечує комплексний підхід до управління блокчейн мережею.

Впровадження даної системи моніторингу дозволить покращити надійність, безпеку та ефективність роботи блокчейн мережі, що сприятиме її ширшому впровадженню та використанню.

РОЗДІЛ 3

ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

В сучасному IT-середовищі автоматизація та ефективне управління інфраструктурою є критично важливими для успішного функціонування веб-додатків та сервісів. Використання хмарних платформ, таких як AWS, надає безліч переваг, включаючи гнучкість, масштабованість та високу надійність. Одним із потужних інструментів для автоматизації управління інфраструктурою є Terraform, який дозволяє описувати інфраструктуру як код (IaC).

У цій роботі розглядається впровадження та експлуатація системи хостингу веб-додатків на базі AWS з використанням Amazon EC2, VPC, Security Groups. Використання Terraform для автоматизації процесів розгортання та управління інфраструктурою забезпечує значні переваги, такі як консистентність середовища, зменшення ризику людських помилок та спрощення масштабування.

Крім того, в роботі розглядається використання Docker та Docker Compose для контейнеризації додатків, що забезпечує портативність та ізоляцію середовищ. Інструменти Docker дозволяють легко розгортати та керувати контейнеризованими додатками, спрощуючи процеси розробки та розгортання.

Для моніторингу та візуалізації метрик використовуються Prometheus та Grafana. Prometheus відповідає за збір та зберігання метрик, а Grafana – за їх візуалізацію у вигляді дашбордів. Це забезпечує прозорість та контроль над станом інфраструктури, дозволяючи оперативно реагувати на можливі проблеми.

У цьому розділі розглядається практична точка зору на впровадження та експлуатацію цієї системи. Будуть детально описані кроки налаштування та інтеграції всіх компонентів: від початкового налаштування інфраструктури за допомогою Terraform, до розгортання контейнеризованих додатків за допомогою Docker та Docker Compose, а також налаштування системи моніторингу за допомогою Prometheus та Grafana. Основна мета цього розділу –

показати практичну реалізацію та переваги комплексного підходу до управління інфраструктурою AWS, використовуючи сучасні інструменти автоматизації та моніторингу.

3.1 Написання Terraform файлів для розгортання на AWS системи

Перше, що ми створимо це EC2 та головний файл **main.tf**

Зокрема, цей код створює новий екземпляр EC2 (Elastic Compute Cloud) з попередньо визначеними параметрами, налаштовує ключі SSH для доступу та використовує скрипти для початкового налаштування сервера. Такий підхід забезпечує автоматизацію процесів та зменшує ризик людських помилок, підвищуючи ефективність управління інфраструктурою.

provider "aws": Цей блок визначає конфігурацію провайдера AWS, який буде використовуватися для взаємодії з сервісами AWS. У даному випадку вказується регіон us-east-1 та профіль terraform. Рис.3.1

aws_key_pair "id_rsa": Цей ресурс створює нову пару ключів SSH у AWS. Публічний ключ завантажується з локального файлу id_rsa.pub, що дозволяє отримати доступ до створених інстансів EC2 за допомогою SSH. Рис.3.31

data "template_file" "userdata": Цей блок завантажує скрипт install.sh з локального файлу та передає йому змінні для шаблонізації. У даному випадку буде використано скрипт для початкового налаштування інстанса. Рис.3.31

```

provider "aws" {
  region = "us-east-1"
  profile = "terraform"
}

resource "aws_key_pair" "id_rsa" {
  key_name    = "id_rsa"
  public_key = file("E:\\keypair\\id_rsa.pub")
}

|

data "template_file" "userdata" {
  template = file("./scripts/install.sh")
  vars = {
    vpc_id = module.vpc.vpc_id
  }
}

```

Рис.3.1 Файл main.tf[20]

Ресурс **aws_instance "backendonee"** створює новий інстанс EC2 з назвою backendonee, забезпечуючи повну автоматизацію процесу розгортання. Цей інстанс розміщується в обраній зоні доступності та підмережі, що належать до віртуальної приватної хмари (VPC). Визначення таких параметрів, як тип інстанса та Amazon Machine Image (AMI), дозволяє чітко встановити апаратні та програмні характеристики створюваного інстанса.Рис 3.2

```

# -----
# BACKEND EC2
# -----
|

resource "aws_instance" "backendonee" {
  instance_type      = var.instance_type
  ami                = var.ami
  vpc_security_group_ids = [aws_security_group.web_sg.id]
  subnet_id          = element(module.vpc.private_subnets, 0)
  availability_zone    = element(module.vpc.azs, 0)
  associate_public_ip_address = true
  user_data            = data.template_file.userdata.rendered

  root_block_device {
    volume_size = 20 # Set the desired root volume size in GB
  }

  key_name = aws_key_pair.id_rsa.key_name
  tags = {
    Name        = "Backend Main"
    Environment = "Dev"
  }
}

```

Рис.3.2 Файл main.tf[20]

Після цього, потрібно створити наш файл з мережею , а саме **network.tf**.

Цей етап створює та налаштовує віртуальну приватну хмару (VPC) у AWS за допомогою модуля Terraform. Модулі Terraform дозволяють повторно використовувати та організовувати конфігурації, що спрощує управління інфраструктурою.

Створення VPC забезпечує ізольоване мережеве середовище для ресурсів, таких як EC2 інстанси, бази даних та інші служби. Використання модуля для створення VPC спрощує процес налаштування та забезпечує стандартний підхід до створення мережевої інфраструктури.

Модулі Terraform дозволяють легко повторно використовувати конфігурації, підвищуючи ефективність та зменшуючи кількість коду. Використання готового модуля terraform-aws-modules/vpc/aws забезпечує надійне та перевірене рішення для створення VPC, що зменшує ймовірність помилок і спрощує процес налаштування.

Опис параметрів Рис.3.3

- **source:** Вказує на модуль VPC, який знаходиться у Terraform Registry.
- **name:** Ім'я VPC, визначене змінною `vpc_name`.
- **cidr:** CIDR-блок для VPC, визначений змінною `vpc_cidr`.
- **azs:** Список зон доступності, де буде розміщена інфраструктура.
- **public_subnets:** CIDR-блоки для публічних підмереж.
- **private_subnets:** CIDR-блоки для приватних підмереж.
- **database_subnets:** CIDR-блоки для підмереж баз даних.
- **create_database_subnet_group:** Визначає, чи потрібно створювати групу підмереж для баз даних.
- **create_database_subnet_route_table:** Визначає, чи потрібно створювати таблицю маршрутів для підмереж баз даних.
- **enable_nat_gateway:** Вмикає використання NAT-шлюзу для приватних підмереж.
- **single_nat_gateway:** Визначає, чи використовувати один NAT-шлюз для

всіх приватних підмереж.

- **tags:** Теги для ресурсів VPC, що спрощують управління та ідентифікацію.

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = var.vpc_name
  cidr = var.vpc_cidr

  azs          = var.vpc_azs
  public_subnets = var.vpc_public_subnets
  private_subnets = var.vpc_private_subnets
  database_subnets = var.vpc_database_subnets

  |

  create_database_subnet_group      = var.vpc_create_database_subnet_group
  create_database_subnet_route_table = var.vpc_create_database_subnet_route_table
  enable_nat_gateway                = true
  single_nat_gateway                 = true

  tags = var.vpc_tags
}
```

Рис.3.3 Файл network.tf[21]

Наступним етапом буде створення, фаєрволу для нашої мережі та серверу. Створення групи безпеки є важливим кроком у процесі забезпечення захисту ресурсів у хмарній інфраструктурі AWS. Група безпеки, створена у даній конфігурації, служить для захисту веб-сервера, визначаючи правила доступу до нього. Вона містить ім'я, яке легко ідентифікувати, та пов'язується з віртуальною приватною хмарою (VPC), що забезпечує додатковий рівень ізоляції та безпеки.

Правила групи безпеки Рис3.4

Для забезпечення належного рівня захисту та функціональності веб-сервера, до групи безпеки додаються різні правила. Ці правила визначають, який трафік дозволено входити та виходити з ресурсів, захищених цією групою безпеки.

4. **SSH доступ:** Одним з найважливіших правил є дозвіл на вхідні з'єднання по SSH. Це правило забезпечує можливість адміністраторам

підключатися до серверів для управління та налаштування через захищене з'єднання, використовуючи порт 22. Дозвіл для всіх IP-адрес (0.0.0.0/0) забезпечує гнучкість, але може бути налаштований для обмеження доступу з певних IP-адрес з міркувань безпеки.

5. **Доступ до бекенд сервісів:** Для забезпечення функціональності веб-додатків необхідно дозволити доступ до різних бекенд сервісів. Правила дозволяють вхідні з'єднання на порти, призначені для основного бекенду, аутентифікації та зберігання даних. Це забезпечує доступність критично важливих сервісів для веб-додатка.
6. **Вихідні з'єднання:** Дозвіл на всі вихідні з'єднання забезпечує, що сервери можуть ініціювати з'єднання з будь-яким зовнішнім ресурсом. Це важливо для завантаження оновлень, спілкування з іншими сервісами та виконання інших завдань, що потребують вихідного трафіку.
7. **HTTP та HTTPS доступ:** Для того щоб веб-сервер міг обслуговувати запити користувачів, необхідно дозволити вхідні з'єднання по HTTP (порт 80) та HTTPS (порт 443). Ці правила забезпечують доступність веб-додатка для користувачів, дозволяючи обробляти як незашифрований, так і зашифрований веб-трафік.

```
resource "aws_security_group" "web_sg" {
  name     = var.web_sg_name
  vpc_id   = module.vpc.vpc_id
  tags = {
    "Name" = var.web_sg_name
  }
}

resource "aws_security_group_rule" "allow_ssh" {
  type            = "ingress"
  from_port       = var.ssh_port
  to_port         = var.ssh_port
  protocol        = "tcp"
  cidr_blocks     = ["0.0.0.0/0"]
  security_group_id = aws_security_group.web_sg.id
}

resource "aws_security_group_rule" "allow_backendmain" {
  type            = "ingress"
  from_port       = var.backendmain_port
  to_port         = var.backendmain_port
  protocol        = "tcp"
  cidr_blocks     = ["0.0.0.0/0"]
  security_group_id = aws_security_group.web_sg.id
}
```

Рис.3.4 Приклад файла securitygroups.tf[22]

Останнім пунктом, буде винесення змінних в окремий файл[23], а саме:

Змінні для мережі (Network Variables)

Мережеві змінні визначають конфігурацію віртуальної приватної хмари (VPC) та її підмереж, що є основою для побудови надійної та ізольованої мережевої інфраструктури.

- `vpc_name`: Ця змінна визначає ім'я VPC, що дозволяє легко ідентифікувати мережу серед інших ресурсів. Значення за замовчуванням встановлене як "VPC_SocialNetwork", що вказує на призначення мережі для соціальної мережі.
- `vpc_cidr`: Визначає CIDR-блок для VPC, задаючи діапазон IP-адрес, які будуть використовуватися в цій мережі. Значення за замовчуванням "10.99.0.0/18" забезпечує достатню кількість IP-адрес для внутрішнього використання.
- `vpc_azs`: Ця змінна містить список зон доступності, у яких будуть розміщені ресурси VPC. Використання кількох зон доступності ("us-east-1a", "us-east-1b", "us-east-1c") забезпечує високу доступність і відмовостійкість інфраструктури.
- `vpc_public_subnets`: Визначає CIDR-блоки для публічних підмереж, де розміщуються ресурси, доступні з Інтернету. Значення за замовчуванням включають дві публічні підмережі: "10.99.0.0/24" і "10.99.1.0/24".
- `vpc_private_subnets`: Визначає CIDR-блоки для приватних підмереж, де розміщуються ресурси, що не потребують прямого доступу з Інтернету. Набір приватних підмереж забезпечує сегментацію та безпеку мережі.
- `vpc_database_subnets`: Визначає CIDR-блоки для підмереж, призначених для баз даних. Використання окремих підмереж для баз даних підвищує безпеку та ізоляцію даних.
- `vpc_tags`: Змінна для додаткових тегів, що додаються до ресурсів VPC.

Теги включають ім'я ("Name"), власника ("Owner") та середовище ("Environment"), що спрощує управління ресурсами та їх ідентифікацію.

Змінні для груп безпеки (Security Group Variables)

Змінні для груп безпеки визначають налаштування доступу до ресурсів, що забезпечує захист мережі.

- `web_sg_name`: Назва групи безпеки для веб-сервера. Значення за замовчуванням `"web_sg"` використовується для ідентифікації групи безпеки.
- `ssh_port`: Порт для SSH доступу, встановлений на значення 22. Це дозволяє адміністраторам підключатися до серверів через захищений канал.
- `backendmain_port`: Порт для основного бекенду, встановлений на значення 3000, що використовується для доступу до основних сервісів бекенду дашборду.
- `backendauth_port`: Порт для бекенд аутентифікації, встановлений на значення 9090. Цей порт забезпечує доступ до сервісів аутентифікації моніторингу.
- `backendstorage_port`: Порт для експортера сховища, встановлений на значення 9560. Використовується для доступу до сервісів зберігання даних.
- `http_port`: Порт для HTTP доступу, встановлений на значення 80. Використовується для нешифрованого веб-трафіку.
- `https_port`: Порт для HTTPS доступу, встановлений на значення 443. Використовується для зашифрованого веб-трафіку.

Змінні для EC2 (EC2 Variables)

Змінні для EC2 визначають параметри інстансів EC2, що створюються в рамках інфраструктури.

- `instance_count_backend`: Визначає кількість інстансів EC2 для

розгортання бекенду. Значення за замовчуванням встановлено на 1, що означає розгортання одного інстанса.

- **ami:** Ідентифікатор Amazon Machine Image (AMI), що використовується для створення інстансів EC2. Значення за замовчуванням вказує на конкретний AMI, який містить необхідне програмне забезпечення.
- **instance_type:** Тип інстансів EC2, що створюються. Значення за замовчуванням "t2.medium" забезпечує баланс між продуктивністю та вартістю.

3.2 Створення Dockerfile та Docker-Compose для системи

Створення Dockerfile

Dockerfile — це текстовий файл, який містить інструкції для створення Docker-образу. Кожен рядок у Dockerfile є командою, яка виконується для побудови образу. Процес створення Dockerfile включає кілька ключових кроків:

1. **Вибір базового образу:** На першому кроці обирається базовий образ, який буде використовуватися для побудови нового образу. Базовий образ повинен містити необхідне середовище для запуску додатку.
2. **Копіювання файлів:** Файли додатку та конфігураційні файли копіюються у контейнер. Це може включати бінарні файли, скрипти, конфігураційні файли тощо.
3. **Встановлення залежностей:** Якщо додатку потрібні додаткові залежності або бібліотеки, вони встановлюються на цьому етапі.
4. **Налаштування робочого каталогу:** Вказується робочий каталог, у якому будуть виконуватися команди під час запуску контейнера.
5. **Команда запуску:** Визначається команда, яка буде виконуватися при запуску контейнера.

Приклад Dockerfile рис3.5 для запуску бінарного файлу `cosmos_exporter` з конфігураційним файлом:

```
FROM alpine:latest

WORKDIR /app

COPY cosmos-validators-exporter /app/
COPY config.toml /app/config.toml

EXPOSE 9565

CMD ["/cosmos-validators-exporter", "--config", "/app/config.toml"]
```

Рис.3.5 Dockerfile metrics exporter [24]

Короткий опис команд у нашому файлі, наступний:

Базовий образ: FROM alpine:latest

Використовується офіційний образ Alpine Linux, який є легким і оптимізованим для контейнерів.

Робочий каталог: WORKDIR /app

Встановлює робочий каталог всередині контейнера на /app, де будуть виконуватися всі наступні команди.

Копіювання файлів:

COPY cosmos-validators-exporter /app/: Копіює бінарний файл cosmos-validators-exporter у робочий каталог контейнера.

COPY config.toml /app/config.toml: Копіює конфігураційний файл config.toml у робочий каталог контейнера.

Відкриття порту: EXPOSE 9565

Вказує, що контейнер прослуховує порт 9565 для зовнішніх з'єднань.

Команда запуску: CMD ["/cosmos-validators-exporter", "--config", "/app/config.toml"]

Визначає команду, яка буде виконуватися при запуску контейнера. У цьому випадку це запуск бінарного файлу cosmos-validators-exporter який ми отримали з репозиторію[17] з вказаним шляхом до конфігураційного файлу.

Наступнім етапом, буде створення загального Docker Compose файлу, для нашого застосунку що збиратиме метрики, та самої системи моніторингу, яка складається з двох частин Prometheus та Grafana.

Файл `docker-compose.yml` ми маємо першочергову компіляцію, для нашого експортеру метрик, та двох окремих сервісів в контейнері які ми отримуємо з офіційного репозиторію Docker Hub.

Запускаємо проект Рис 3.6 перед цим вказавши усі необхідні конфігураційні файли **Б1. Конфігураційний файл**

```
root@development-bot:~/monitor# docker compose up -d
WARN[0000] /root/monitor/docker-compose.yaml: 'version' is obsolete
[+] Running 3/0
 ✓ Container monitor-cosmos_exporter-1 Running 0.0s
 ✓ Container prometheus Running 0.0s
 ✓ Container grafana Running 0.0s
```

Рис 3.6 Запуск контейнерів

Та перевіряємо результат Рис 3.7 за допомогою команди `docker ps -a` та запитом `curl localhost:9565/metrics` Рис 3.8

```
root@development-bot:~/monitor# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
a4be5d5a6450   grafana/grafana:latest             "/run.sh"               5 hours ago   Up 5 hours   0.0.0.0:3000→3000/tcp, :::3000→3000/tcp
grafana
e25d1b722b65   prom/prometheus:latest             "/bin/prometheus --c..." 5 hours ago   Up 5 hours   0.0.0.0:9090→9090/tcp, :::9090→9090/tcp
prometheus
9895b5d4354d   monitor-cosmos_exporter             "./cosmos-validators..." 5 hours ago   Up 5 hours   0.0.0.0:9565→9565/tcp, :::9565→9565/tcp
monitor-cosmos_exporter-1
```

Рис 3.7 Перевірка контейнерів

```
root@development-bot:~/monitor# curl localhost:9565/metrics
# HELP cosmos_validators_exporter_active Whether a validator is active (1 if yes, 0 if no)
# TYPE cosmos_validators_exporter_active gauge
cosmos_validators_exporter_active{address="swtrvaloper12uvappnj3e6xnrsjsre9ktd4tw95j5dtjxfp5y",chain="swisstronik_1291-1"} 1
# HELP cosmos_validators_exporter_active_set_tokens Tokens needed to get into active set (last validators' stake, or 0 if not enough va
idators)
# TYPE cosmos_validators_exporter_active_set_tokens gauge
cosmos_validators_exporter_active_set_tokens{chain="swisstronik_1291-1"} 0
# HELP cosmos_validators_exporter_base_denom Base denom info
# TYPE cosmos_validators_exporter_base_denom gauge
cosmos_validators_exporter_base_denom{chain="swisstronik_1291-1",denom="swtr"} 1
# HELP cosmos_validators_exporter_commission Validator current commission
# TYPE cosmos_validators_exporter_commission gauge
```

Рис 3.8 Перевірка застосунку

3.3 Створення дашборда на базі Grafana

Дашборди є важливим інструментом для візуалізації даних, які дозволяють швидко отримати уявлення про стан системи або процесу. Grafana є одним з найпопулярніших інструментів для створення дашбордів, що надає можливість підключатися до різних джерел даних і створювати візуально привабливі графіки та звіти. В цьому есе буде описано процес створення дашборду в Grafana на прикладі JSON-файлу, що містить конфігурацію дашборду для моніторингу валідаторів мережі.

Перед початком , нам потрібно вибрати джерело даних Grafana підтримує багато джерел даних, таких як Prometheus, InfluxDB, MySQL та інші. В нашому випадку ми будемо використовувати Prometheus. Після встановлення Grafana необхідно додати джерело даних Prometheus Рис 3.9:

1. Увійдіть в Grafana.
2. Перейдіть до розділу "Configuration" та виберіть "Data Sources".
3. Натисніть кнопку "Add data source" та виберіть Prometheus.
4. Введіть URL сервера Prometheus та натисніть "Save & Test".

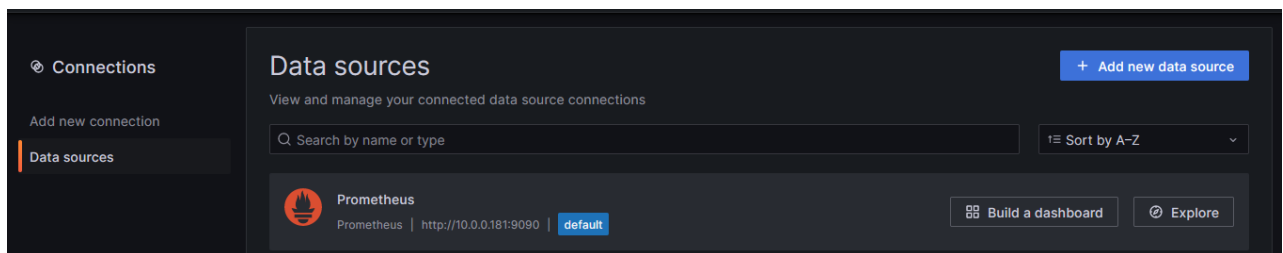


Рис 3.9 Приклад конфігурації Grafana Data Source

Наступним етапом буде імпортування готового дашборду у вигляді json-моделі[25]

Нам потрібно виконати наступні кроки:

На головній панелі інструментів Grafana натисніть кнопку "Create" (Створити) в лівому верхньому куті.

1. Виберіть "Import" (Імпорт) з випадаючого меню.
2. У вікні імпорту виберіть опцію "Import via panel json" (Імпортувати через JSON панелі).
3. Вставте скопійований JSON-код у текстове поле.
4. Переконайтеся, що весь код вставлено правильно, і натисніть "Load" (Завантажити).

Мережа Cosmos SDK Swisstronik є децентралізованою блокчейн-системою, яка використовує валідаторів для забезпечення безпеки та підтримки функціонування мережі. Для ефективного моніторингу стану валідаторів і своєчасного виявлення проблем використовуються дашборди в Grafana Рис 3.39. У цьому есе ми розглянемо дашборд, який складається з кількох панелей, кожна з яких відображає важливі метрики валідаторів.

Опис Панелей рис3.21

Validator Uptime (Час роботи валідатора):

Опис: Ця панель відображає відсоток часу, протягом якого валідатор був активний і доступний.

Навіщо це потрібно: Високий uptime є критично важливим для забезпечення надійності мережі. Він показує, наскільки стабільним і надійним є валідатор. Валідатори з низьким uptime можуть бути оштрафовані або виключені з активного сету.

Як використовується: Цю метрику використовують для оцінки надійності валідаторів. Якщо uptime знижується, оператори валідаторів можуть вжити заходів для покращення доступності.

Validator Name (Ім'я валідатора):

Опис: Панель відображає назву валідатора.

Навіщо це потрібно: Ідентифікація валідаторів є важливою для їх відрізнєння в мережі, особливо коли в мережі працює багато валідаторів.

Як використовується: Використовується для швидкої ідентифікації валідатора у візуалізаціях та звітах.

Jail (Статус ув'язнення)

Опис: Панель показує, чи був валідатор ув'язнений (Jailed) через порушення правил мережі.

Навіщо це потрібно: Ув'язнення є механізмом покарання валідаторів за неналежну поведінку, таку як подання неправильних блоків або тривала недоступність.

Як використовується: Оператори валідаторів можуть використовувати цю метрику для перевірки, чи були їхні валідатори ув'язнені і вжити необхідних заходів для виправлення ситуації.

Active (Активний статус):

Опис: Панель показує, чи активний валідатор у мережі.

Навіщо це потрібно: Активний статус вказує на те, що валідатор бере участь у процесі валідування блоків.

Як використовується: Використовується для моніторингу участі валідатора у мережі та для вжиття заходів у разі відсутності активності.

Tokens Staked (Заставлені токени):

Опис: Панель відображає кількість токенів, заставлених за валідатора.

Навіщо це потрібно: Більша кількість заставлених токенів підвищує безпеку мережі та довіру до валідатора.

Як використовується: Використовується для моніторингу довіри делегаторів до валідатора. Оператори можуть використовувати цю інформацію для залучення більше делегаторів.

Validator Commission (Комісія валідатора):

Опис: Панель відображає відсоток комісії, який валідатор бере з нагороди делегаторів.

Навіщо це потрібно: Інформація про комісію важлива для делегаторів, які хочуть знати, скільки вони заплатять валідатору за його послуги.

Як використовується: Використовується делегаторами для вибору валідатора з оптимальною комісією.

Rank (Ранг):

Опис: Панель показує ранг валідатора у мережі.

Навіщо це потрібно: Ранг визначається на основі кількості заставлених токенів та продуктивності валідатора.

Як використовується: Використовується для оцінки позиції валідатора у мережі.

Validators Count (Кількість валідаторів):

Опис: Панель показує загальну кількість валідаторів у мережі.

Навіщо це потрібно: Важливо знати загальну кількість валідаторів для оцінки децентралізації мережі.

Як використовується: Використовується для аналізу масштабу та децентралізації мережі.

Missed Blocks (Пропущені блоки):

Опис: Панель відображає кількість блоків, які валідатор пропустив.

Навіщо це потрібно: Пропущені блоки можуть свідчити про проблеми з валідатором, такі як технічні несправності або недоступність.

Як використовується: Використовується для моніторингу стабільності валідатора та виявлення проблем.

Uptime History (Історія часу роботи):

Опис: Панель показує історію uptime валідатора за певний період.

Навіщо це потрібно: Аналіз історії uptime дозволяє оцінити довгострокову надійність валідатора.

Як використовується: Використовується для виявлення патернів та трендів у доступності валідатора.



Рис 3.21 Дашбоард валідатора для мережі Swisstronik[19]

3.4 Ефективність та Тестування Дашборда для Blockchain Мережі на Базі Cosmos SDK

Ми розглянемо ефективність використання дашбордів для моніторингу та методи тестування їх функціональності, а також кількісні вигоди від їх впровадження.

Ефективність Дашборда

1. Швидке виявлення проблем:

Завдяки відображенню метрик у режимі реального часу, дашборди дозволяють операторам швидко виявляти проблеми, такі як низький uptime, високий рівень пропущених блоків або статус "Jailed". Це значно скорочує час простою валідаторів та зменшує ризик фінансових втрат.

Наприклад, зниження часу реагування з 30 хвилин до 5 хвилин може запобігти втратам, пов'язаним з оштрафуванням валідаторів, що складають до 1000 доларів США за кожну годину простою.

2. Підвищення надійності мережі:

Використання дашбордів дозволяє підтримувати високу надійність валідаторів, що підвищує загальну надійність мережі. Постійний моніторинг та оперативні дії при виявленні проблем сприяють підвищенню довіри делегаторів. **Збільшення uptime на 1%** може призвести до додаткових доходів для валідатора у розмірі до 5000 доларів США на рік, враховуючи збільшення кількості делегованих токенів.

3. Економія ресурсів:

Автоматизація процесів моніторингу за допомогою дашбордів зменшує потребу в ручному контролі та аналізі даних, що економить людські ресурси.

За оцінками, автоматизація моніторингу дозволяє зекономити до 20 годин на місяць на одного оператора, що у грошовому еквіваленті складає приблизно 600 доларів США на місяць.

Тестування Дашборда

Функціональне тестування:

Перевірка коректного відображення всіх панелей: uptime, статусів валідаторів, кількості заставлених токенів, комісій тощо. Тестування проводиться шляхом порівняння даних з дашборда з фактичними даними з джерел (Prometheus). Відхилення повинні бути мінімальними або відсутні.

Тестування продуктивності:

Оцінка часу завантаження дашборда та окремих панелей. Оптимальний час завантаження не повинен перевищувати 5 секунд. Навантажувальне тестування для перевірки стійкості дашборда при великій кількості одночасних запитів або підключень користувачів.

Провівши тестування, помилок в роботі не було виявлено, система функціонує повністю.

Висновки

Моніторинг валідаторів у блокчейн-мережі на базі Cosmos SDK є критично важливим для забезпечення її стабільності, безпеки та надійності. Впровадження дашбордів у Grafana значно підвищує ефективність управління валідаторами, дозволяючи операторам швидко виявляти та усувати проблеми, оптимізувати продуктивність і забезпечувати високу якість обслуговування мережі.

Ефективність дашбордів проявляється у швидкому виявленні проблем, підвищенні надійності мережі та економії ресурсів. Використання реальних даних у режимі реального часу допомагає операторам підтримувати стабільну роботу валідаторів, знижувати ризики фінансових втрат та залучати більше делегаторів. Зокрема, зниження часу реагування на проблеми та автоматизація процесів моніторингу дозволяють економити значні кошти та підвищувати доходи валідаторів.

Тестування дашбордів на функціональність, продуктивність та безпеку забезпечує їх надійність і стабільну роботу під навантаженням. Завдяки цьому оператори можуть бути впевнені у коректності відображення даних та захисті від несанкціонованого доступу.

У підсумку, впровадження дашбордів у Grafana для моніторингу валідаторів у блокчейн-мережі на базі Cosmos SDK є важливим кроком до підвищення ефективності управління, оптимізації ресурсів та забезпечення високої надійності мережі. Ці інструменти сприяють кращому розумінню стану валідаторів і швидкому прийняттю рішень, що робить їх незамінними для сучасних блокчейн-систем.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

У цій дипломній роботі була розроблена та протестована система моніторингу для блокчейн мережі на базі Cosmos SDK. Основною метою роботи було забезпечення високого рівня прозорості, безпеки та ефективності в управлінні мережевими ресурсами.

1. Ефективність системи:

Розроблена система моніторингу дозволяє операторам мережі своєчасно реагувати на аномалії, оптимізувати роботу мережі та забезпечувати безперебійну роботу в режимі реального часу. Використання дашбордів Grafana дозволяє візуалізувати ключові показники роботи валідаторів, що підвищує зручність управління та аналізу даних.

2. Автоматизація та економія ресурсів:

Впровадження автоматизованих процесів моніторингу значно зменшує потребу в ручному контролі та аналізі даних, що дозволяє економити людські ресурси та знижувати операційні витрати. Автоматизація моніторингу дозволяє зекономити до 20 годин на місяць на одного оператора, що у грошовому еквіваленті складає приблизно 600 доларів США на місяць.

3. Надійність та безпека:

Проведені тестування системи на функціональність, продуктивність та безпеку підтвердили її високу адаптивність та надійність. Система забезпечує коректне відображення даних та захист від несанкціонованого доступу, що є критично важливим для забезпечення стабільної роботи блокчейн мережі.

4. Тестування та результати:

Результати тестування продемонстрували ефективність пропонованої системи моніторингу у виявленні несправностей, оптимізації роботи мережі та забезпеченні високого рівня безпеки. Система була протестована в різних умовах, що підтвердило її високу адаптивність і надійність.

5. Перспективи розвитку:

Розроблена система моніторингу може бути використана при створенні і підтримці будь-яких блокчейн мереж, особливо на базі Cosmos SDK. Вона може

бути корисною для розробників, операторів мережі та організацій, які використовують блокчейн-технології для своєї діяльності. Подальший розвиток та вдосконалення системи дозволить ще більше підвищити її ефективність та масштабованість.

Отже, розробка та впровадження системи моніторингу для блокчейн мережі на базі Cosmos SDK є важливим кроком до підвищення ефективності управління, оптимізації ресурсів та забезпечення високої надійності мережі. Ці інструменти сприяють кращому розумінню стану валідаторів і швидкому прийняттю рішень, що робить їх незамінними для сучасних блокчейн-систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Toshendra Sharma. Introduction to Blockchain [Електронний ресурс] / Toshendra Sharma // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/343601688_Introduction_to_Blockchain.
2. Samantha Radocchia. Blockchain Technology: Understanding its Meaning, Architecture, and Diverse Applications [Електронний ресурс] / Samantha Radocchia // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/373236964_Blockchain_Technology_Understanding_its_Meaning_Architecture_and_Diverse_Applications.
3. Muthu Ramachandran, et al. Probabilistic Blockchains: A Blockchain Paradigm for Collaborative Decision-Making [Електронний ресурс] / Muthu Ramachandran, et al. // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/333486562_Probabilistic_Blockchains_A_Blockchain_Paradigm_for_Collaborative_Decision-Making/figures?lo=1.
4. Rohit Khandelwal, et al. An Overview of Blockchain Consensus Algorithms: Comparison, Challenges and Future Directions [Електронний ресурс] / Rohit Khandelwal, et al. // IRJET. – Режим доступу: <https://www.irjet.net/archives/V7/i10/IRJET-V7I10109.pdf>.
5. Bitcoin Explained [Електронний ресурс] // Corporate Finance Institute. – Режим доступу: <https://corporatefinanceinstitute.com/course/bitcoin-explained/>.
6. S. N. Singhal. Probabilistic Blockchains: A Paradigm for Collaborative Decision-Making [Електронний ресурс] / S. N. Singhal // IJCPUB. – Режим доступу: <https://rjpn.org/ijcspub/papers/IJCSP22A1066.pdf>.
7. Rohit Khandelwal, et al. An Overview of Blockchain Consensus Algorithms: Comparison, Challenges, and Future Directions [Електронний ресурс] / Rohit Khandelwal, et al. // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/344865960_An_Overview_of_Blockchain_Consensus_Algorithms_Comparison_Challenges_and_Future

8. Proof of Work vs. Proof of Stake in Cryptocurrency Exchange Development [Электронный ресурс] // Antier Solutions. – Режим доступа: <https://www.antiersolutions.com/proof-of-work-proof-of-stake-in-cryptocurrency-exchange-development/>.
9. S. N. Singhal, et al. Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership, and Structure [Электронный ресурс] / S. N. Singhal, et al. // ResearchGate. – Режим доступа: https://www.researchgate.net/publication/335337656_Deconstructing_Blockchains_A_Comprehensive_Survey_on_Consensus_Membership_and_Structure/figures?lo=1.
10. Cosmos Network Overview [Электронный ресурс] // Cosmos Documentation. – Режим доступа: <https://docs.cosmos.network/main/learn/intro/overview>.
11. The Ultimate Guide to Nodes [Электронный ресурс] // Incrypted. – Режим доступа: <https://incrypted.com/ua/ultimate-guide-po-nodah/>.
12. Tendermint Core: App Architecture [Электронный ресурс] // Tendermint Documentation. – Режим доступа: <https://docs.tendermint.com/v0.33/app-dev/app-architecture.html>.
13. ABCI: Proactive Calls to the Blockchain [Электронный ресурс] // Autonolas Network. – Режим доступа: https://docs.autonolas.network/open-autonomy/key_concepts/abci/#proactive-calls-to-the-blockchain.
14. Cosmos SDK: OCAP [Электронный ресурс] // Cosmos Documentation. – Режим доступа: <https://docs.cosmos.network/v0.46/core/ocap.html>.
15. How IBC Works [Электронный ресурс] // IBC Protocol. – Режим доступа: <https://www.ibcprotocol.dev/how-ibc-works>.
16. Chain of Blocks in Blockchain [Электронный ресурс] // ResearchGate. – Режим доступа: https://www.researchgate.net/figure/Chain-of-blocks-in-blockchain_fig4_331040157.
17. Cosmos Validators Exporter [Электронный ресурс] // GitHub. – Режим доступа: <https://github.com/QuokkaStake/cosmos-validators-exporter/tree/main>.

- 18.Swisstronik [Электронный ресурс] // Swisstronik. – Режим доступа: <https://www.swisstronik.com/>.
- 19.MojitoTea. main.tf [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/main.tf>.
- 20.MojitoTea. network.tf [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/network.tf>.
- 21.MojitoTea. securitygroups.tf [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/securitygroups.tf>.
- 22.MojitoTea. variables.tf [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/variables.tf>.
- 23.MojitoTea. Dockerfile [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/Dockerfile>.
- 24.MojitoTea. dashboard.json [Электронный ресурс] / MojitoTea // GitHub. – Режим доступа: <https://github.com/MojitoTea/sdk-deploy/blob/main/dashboard.json>.

Додаток А

Файл `docker-compose.yml`

```
version: '3.8'

services:
  cosmos_exporter:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "9565:9565"
    volumes:
      - ./config.toml:/app/config.toml
    restart: unless-stopped

  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    ports:
      - 9090:9090
    restart: unless-stopped
    volumes:
      - ./prometheus:/etc/prometheus
      - prom_data:/prometheus
    depends_on:
      - cosmos_exporter

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - 3000:3000
    restart: unless-stopped
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=grafana
    volumes:
      - ./grafana:/etc/grafana/provisioning/datasources
    depends_on:
```

- prometheus

volumes:

- prom_data:

Додаток Б

Приклади конфігураційних файлів

Б1. Конфігураційний файл

```

# Global timeout for RPC queries, in seconds. Defaults to 5.
timeout = 5

# The address the exporter will listen on .Defaults to ":9560".
listen-address = ":9565"


[log]

level = "debug"


json = false


# Per-chain config.
[[chains]]

name = "swisstronik_1291-1"

lcd-endpoint = "https://swisstronik-t-api.noders.services"

base-denom = "swtr"
denoms = [
    { denom = "uswtr", display-denom = "swtr" },
]

bech-wallet-prefix = "swtr"

validators = [
    { address = "swtrvaloper12uvappnj3e6xnrsjsre9ktd4tw95j5dtjxfp5y", consensus-
address = "swtrvalcons1sl2x0frssruqqwkagfm60mdrl847m3dsxjk8af" }

]

# List of queries to enable/disable.
# If the list is not provided, or the value for query is not specified,
# then this query will be enabled. Useful if some queries on some chains are
broken or

```

```
# do not return any meaningful value (like signing info on e-Money) or are too
heavy and
# the node can't handle such requests (like delegators count on Cosmos Hub).
[chains.queries]
# Query for validator info
validator = true
# Query for delegators count
# Query for self-delegated amount
# Query for all delegators count/ranking. Also used in total bonded tokens
calculation.
validators = true
# Query for validator unclaimed commission
commission = true
# Query for validator wallet balance
balance = true
# Query for validator signing info
signing-info = true
# Query for chain slashing params/missed blocks window
slashing-params = true
# Query for chain staking params/max validators count
staking-params = false
```