МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Бабенко Олександр Валерійович

Комп'ютерна мережа з VPN сервером / Computer network with VPN server

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ-41
Бабенко Олександр Валерійович

Науковий керівник
Дериш Б.Б.

ТЕРНОПІЛЬ - 2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Комп'ютерна мережа з VPN сервером» зі спеціальності 123 «Комп’ютерна інженерія» освітнього ступеня «бакалавр» містить 76 сторінок пояснюючої записки, 48 рисунків, 8 таблиць, 3 додатки.

Об'єктом кваліфікаційній роботи є дослідження та створення та налаштування VPN-рішення на базі Terraform та Pritunl в Google Cloud.

Предметом дослідження є механізми створення, конфігурації та управління віртуальними приватними мережами (VPN) в хмарному середовищі Google Cloud за допомогою Terraform та Pritunl.

Мета роботи полягає у розробці надійного та ефективного VPN-рішення, яке дозволить забезпечити безпечний доступ до ресурсів хмарного середовища Google Cloud для різних користувачів та груп користувачів, з використанням можливостей автоматизації Terraform і функціональності Pritunl.

Методами роботи будуть теоретичний аналіз та синтез, моделювання, програмування, тестування, порівняння, а також методи системного аналізу. Основою дослідження стане аналіз відкритої документації, створення прототипу VPN-рішення, його тестування та оптимізація на основі отриманих результатів.

Результатом роботи буде VPN-рішення, розгорнуте у хмарному середовищі Google Cloud за допомогою Terraform і Pritunl, яке дозволить забезпечити безпечний доступ до ресурсів хмари. Буде також розроблено документацію, яка містить рекомендації щодо налаштування та управління VPN, а також аналіз можливостей оптимізації і масштабування даного рішення.

Ключові слова: VPN, GPC, ІНТЕРНЕТ, ІНФРАСТРУКТУРА ЯК КОД, АВТОМАТИЗАЦІЯ

RESUME

Qualification thesis “Server VPN network” in the specialty 123 "Computer Engineering" of bachelor education degree contains 76 pages of explanatory notes, 48 figures, 8 tables, 3 appendixes.

The object of the qualification thesis is the exploration and creation of a VPN solution based on Terraform and Pritunl in Google Cloud.

The subject of the research is the mechanisms for creating, configuring, and managing Virtual Private Networks (VPN) in the Google Cloud environment using Terraform and Pritunl.

The goal of the work is to develop a reliable and effective VPN solution that will ensure secure access to resources of the Google Cloud environment for various users and user groups, using the automation capabilities of Terraform and the functionality of Pritunl.

The methods of work will be theoretical analysis and synthesis, modeling, programming, testing, comparison, as well as methods of system analysis. The basis of the study will be the analysis of open documentation, the creation of a prototype VPN solution, its testing and optimization based on the results obtained.

The result of the work will be a VPN solution deployed in the Google Cloud environment using Terraform and Pritunl, which will ensure secure access to cloud resources. Documentation will also be developed that contains recommendations for setting up and managing VPN, as well as an analysis of optimization and scaling possibilities for this solution.

Keywords: VPN, GPC, INTERNET, INFRASTRUCTURE AS CODE, AUTOMATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОС	–	Операційна система
VPN	–	Virtual Private Network
API	–	Application Programming Interface
GCP	–	Google Cloud Platform

ВСТУП

В наш час інтернет-простір зустрічається з різноманітними викликами. Однією з важливих проблем, які потребують уваги, є питання конфіденційності та безпеки онлайн. Існує багато способів, як можна захистити свої дані в інтернеті, але одним з найефективніших є використання VPN-сервісу. VPN, або Віртуальна приватна мережа, стала невід'ємною частиною життя багатьох користувачів інтернету, бо забезпечує захист їх приватності та даних.

Використання VPN стало важливішим, зокрема, через зростання кількості кіберзлочинності. Викрадення особистих даних, включаючи фінансову інформацію, персональні фотографії, паролі та інші відомості, є поширеною проблемою, яку можна запобігти, використовуючи VPN. Такий сервіс забезпечує шифрування даних, що перешкоджає доступ до них третім особам.

Також VPN допомагає обходити географічні обмеження, що встановлюються деякими веб-сайтами або онлайн-сервісами. Це може бути корисним для людей, які подорожують або живуть в країнах з суворими обмеженнями доступу до інтернету. За допомогою VPN вони можуть вільно доступатися до своїх улюблених сайтів або сервісів, незалежно від їх фізичного розташування.

У світлі цих викликів, VPN виступає важливим інструментом для підтримки приватності та безпеки в інтернеті. Отже, у світі, що швидко цифровізується, здатність забезпечувати свою приватність та безпеку в мережі стає все більш важливою.

На сьогоднішній день, коли ми все більше використовуємо інтернет для здійснення фінансових операцій, покупок, спілкування та зберігання особистої інформації, важливість VPN стає ще більшою. Багато людей не усвідомлюють, що при підключенні до відкритих мереж Wi-Fi, наприклад, в кав'ярнях, аеропортах або громадських місцях, їхні дані можуть бути під загрозою.

Ця кваліфікаційна робота стосується проектування та реалізації VPN-рішення, базованого на використанні Terraform та Pritunl в середовищі Google Cloud. Ці системи були вибрані через їхню гнучкість, надійність та ефективність, що робить їх ідеальною основою для нашого VPN-рішення. При цьому, реалізована безпечна комунікація між клієнтськими машинами, що вкрай важливо в сучасних умовах.

Мета роботи полягає у розробці надійного та ефективного VPN-рішення, яке дозволить забезпечити безпечний доступ до ресурсів хмарного середовища Google Cloud для різних користувачів та груп користувачів, з використанням можливостей автоматизації Terraform і функціональності Pritunl.

Для досягнення зазначеної мети було поставлено такі завдання:

– провести дослідження предметної області

– на основі отриманих в результаті аналізу даних зробити постановку завдання до майбутнього сервісу;

– здійснити конфігурацію програмного забезпечення, провести тестування розробленої системи.

Об'єктом дослідження є комп'ютерна мережа з VPN сервером

Предметом дослідження дослідження є мережеве програмування та технології Terraform, Pritunl та Google Cloud.

Публікація та апробація: VPN мережа на базі Google Cloud та з використанням Terraform та Pritunl / Дериш Б.Б., Бабенко О.В., VII НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «Інтелектуальні комп'ютерні системи та мережі»2023, 23 травня 2023р.: тези доп. - Тернопіль, 2023-

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ І ПОСТАНОВКА ЗАДАЧІ

1.1 Висвітлення проблеми мережевого захисту та потреби в віртуальному приватному мережевому (VPN) рішенні

Проблема мережевого захисту є актуальною в сучасному цифровому світі, оскільки зростає кількість кібератак, посягань на конфіденційність даних та віртуальну безпеку користувачів. Одним з рішень, яке може допомогти забезпечити високий рівень захисту мережі, є використання віртуальних приватних мережевих (VPN) рішень.

Одна з головних проблем мережевого захисту полягає в ризику перехоплення, зміни або витоку даних під час їх передачі через відкриті мережі, такі як Інтернет. Наприклад, зловмисники можуть використовувати техніки перехоплення даних, такі як "мен-ін-зе-мідл" (man-in-the-middle) атаки, щоб отримати доступ до конфіденційних даних, таких як паролі, фінансові відомості або корпоративні дані.

Крім того, багато користувачів стурбовані щодо своєї приватності в Інтернеті, оскільки провайдери Інтернету можуть відстежувати їхню активність в мережі, а також обмежувати доступ до певних ресурсів або вмісту. Багато користувачів також хочуть отримати доступ до різних регіональних веб-ресурсів, таких як відео- або аудіо-потоки, які можуть бути обмежені географічними обмеженнями.

Основні переваги використання VPN-рішень полягають у забезпеченні шифрування даних, конфіденційності, анонімності та захисту від перехоплення даних. VPN дозволяє захищати даних в мережі, забезпечуючи безпечну передачу інформації між користувачем та ресурсами в Інтернеті. Крім того, використання VPN дозволяє отримати доступ до різних регіональних ресурсів, таких як відео- або аудіо-потоки, які можуть бути обмежені географічними обмеженнями.

Додатково, використання VPN може допомогти захистити віртуальну приватність користувача, оскільки воно маскує його реальну IP-адресу та

місцезнаходження, роблячи важчим відстеження та ідентифікацію користувача в мережі. Це особливо важливо в ситуаціях, коли користувачі використовують непотрібні мережі, такі як громадські Wi-Fi мережі в кафе, аеропортах або готелях, де можуть існувати ризики безпеки.

Однак, слід зазначити, що не всі VPN-рішення є однаково захищеними, і обирання надійного та довіреного VPN-провайдера є важливим кроком у забезпеченні ефективного мережевого захисту. Деякі безкоштовні VPN-провайдери можуть збирати та використовувати користувацькі дані, що може ставити під загрозу конфіденційність та приватність користувача.

1.2 Визначення мети та завдань роботи

Метою моєї роботи є розробка надійної мережі VPN, яка має на меті забезпечити захищену та конфіденційну комунікацію між користувачами. Для досягнення цієї мети, я використовую різноманітні методи, включаючи ретельний аналіз документації технологій Terraform, Pritunl та Google Cloud, проведення налаштуванням мережі, та тестування рішення.

На протязі роботи над кваліфікаційною роботою, глибоко досліджую можливості технології Terraform, яка дозволяє автоматизувати процес розгортання та керування інфраструктурою в хмарних середовищах Google Cloud. Я також детально вивчаю можливості Pritunl - відкритого віртуального сервера, який дозволяє налаштовувати, керувати та моніторити VPN-сервери. За допомогою цих інструментів, я планую налаштувати мережу VPN, забезпечуючи високий рівень захисту та конфіденційності.

1.3 Огляд технології VPN

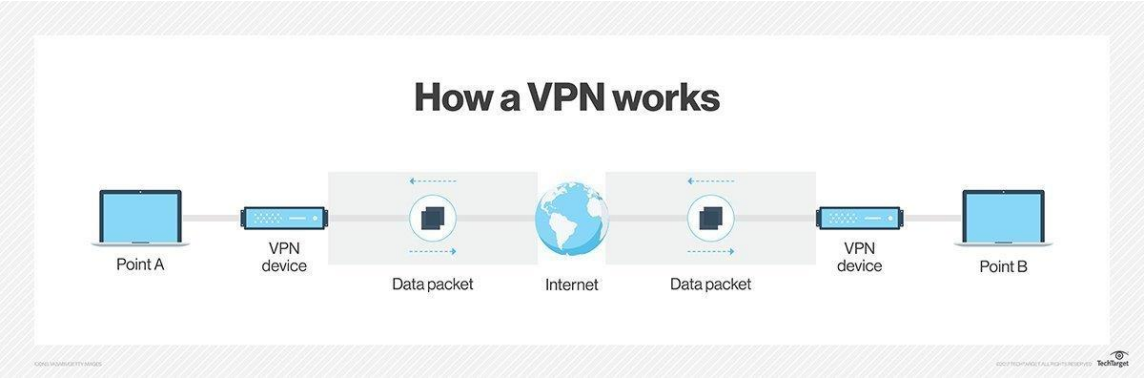

Рисунок 1.1 – Як працює VPN

Virtual Private Network (VPN) - це технологія, яка забезпечує безпечне з'єднання між пристроями через публічну мережу, таку як Інтернет. Вона дозволяє користувачам підключатися до приватної мережі з віддаленого місця і передавати дані в зашифрованому вигляді, забезпечуючи приватність, конфіденційність і безпеку віддаленого з'єднання.

1.4. Постановка задач кваліфікаційної роботи

Terraform - це інструмент відкритого коду, розроблений HashiCorp, призначений для автоматизації процесу розгортання, налаштування та керування інфраструктурою в хмарних середовищах та інших провайдерах. Він дозволяє описати інфраструктуру в коді, використовуючи декларативну мову конфігурації, та керувати станом ресурсів за допомогою коду. [5]

Основні можливості Terraform:

Декларативна мова конфігурації: Terraform використовує власну декларативну мову конфігурації, що дозволяє описати бажану структуру інфраструктури в коді. Це дозволяє легко керувати різними аспектами

інфраструктури, такими як віртуальні машини, мережі, сховища даних та інші ресурси. [2]

Підтримка різних провайдерів: Terraform підтримує багато різних провайдерів хмарних інфраструктур, таких як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), OpenStack, VMware, та багато інших. Це дозволяє користувачам використовувати Terraform для розгортання та керування різними хмарними середовищами. Як працює підтримка різних провайдерів показано на рисунку 1.3. та короткий перелік основних провайдерів з офіційного сайту Terraform зображений на рисунку 1.4.

Рисунок 1.3. – Як працює підтримка провайдерів

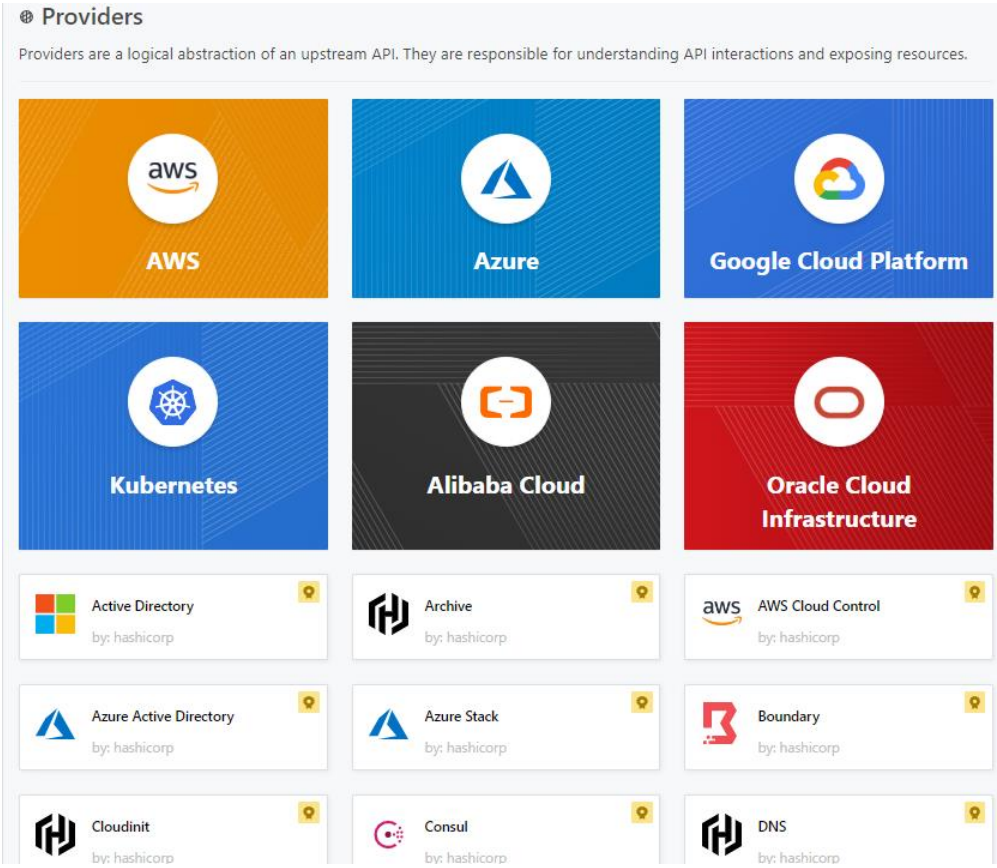

Рисунок 1.4 – Знімок екрану з коротким переліком провайдерів хмарних інфраструктур з офіційного сайту Terraform

Інфраструктура в коді: Одна з ключових особливостей Terraform - це можливість описувати інфраструктуру в коді, що дозволяє використовувати кращі практики розробки програмного забезпечення, такі як контроль версій, рецензування коду, та автоматичну перевірку на відповідність стандартам. Приклад інфраструктури як коду зображений на рисунку 1.5.

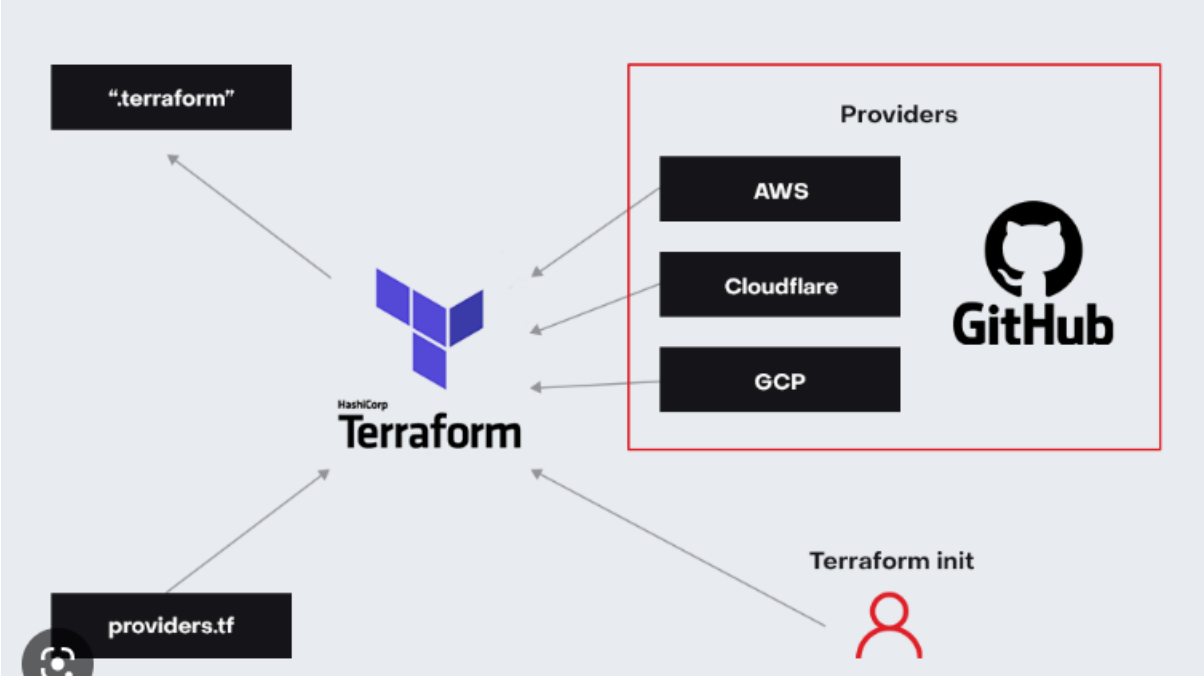

Рисунок 1.5 – Інфраструктура як код

Модульність: Terraform дозволяє використовувати модулі, які дозволяють перевикористовувати та організовувати конфігурацію інфраструктури в більших проектах. Модулі можуть бути написані в Terraform або імпортовані з репозиторіїв модулів, що дозволяє легко використовувати готові рішення та рефакторити код для більшої перевикористовуваності. Приклад модульності зображений на рисинку 1.6. [7]

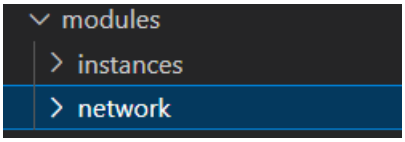

Рисунок 1.6 – Модульність

Управління станом: Terraform автоматично відстежує стан ресурсів, що були створені або змінені, в окремому файлі стану. Це дозволяє використовувати принцип "інфраструктура як код" та контролювати зміни в інфраструктурі, відстежувати їх, співпрацювати зі співробітниками та відновлювати попередні стани ресурсів при необхідності. Приклад управління станом зображений на рисунку 1.7. [10]

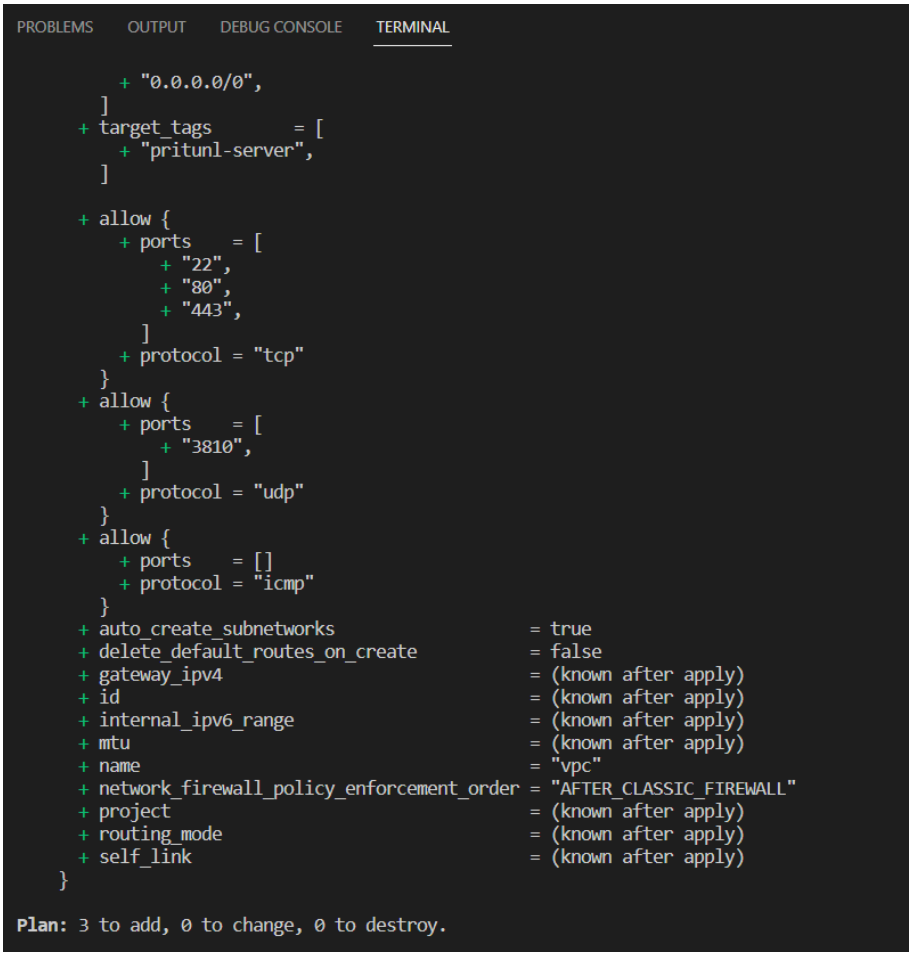

Рисунок 1.7 – Відстеження стану

Розширюваність: Terraform дозволяє розширювати свої можливості за допомогою власних провайдерів, які можуть бути розроблені та використані для взаємодії зі специфічними API провайдерів, що не підтримуються "із коробки". Це дає можливість розгортати та керувати різними типами інфраструктури, які не входять в стандартну колекцію провайдерів Terraform. Приклад, як виглядає провайдер GCP в коді зображений на рисунку 1.8.

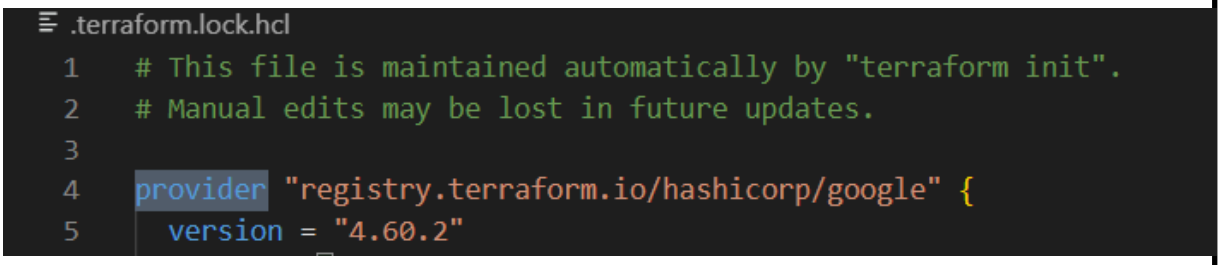

Рисунок 1.8 – офіційний провайдер для GCP

Спільнота та підтримка: Terraform має велику активну спільноту користувачів та розробників, яка забезпечує підтримку, внесення внесків, вирішення проблем та поширення знань. Terraform також має офіційну документацію, форуми, блоги та інші ресурси, які допомагають вивчити та використовувати його ефективно. Знімок екрану з офіційної спільноти Terraform зображений на рисунку 1.9. [13]

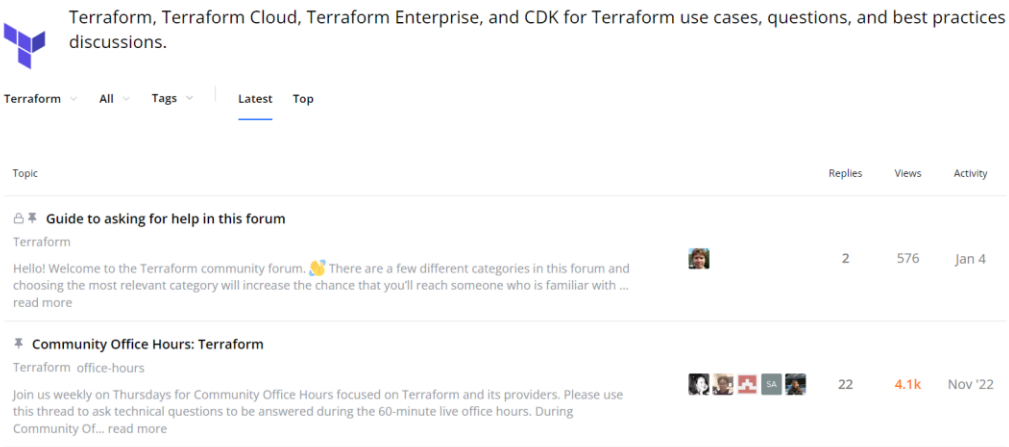

Рисунок 1.9 – офіційна спільнота Terraform

Google Cloud (також відомий як Google Cloud Platform або GCP) - це постачальник обчислювальних ресурсів для розробки, розгортання та експлуатації веб-додатків. [18]

Коли ви запускаєте веб-сайт, додаток або сервіс на GCP, Google відстежує всі ресурси, які він використовує - зокрема, скільки потужності обробки, об'єму даних, запитів до баз даних та мережевого з'єднання він використовує. Замість того, щоб орендувати сервер або DNS-адресу на місяць (як це робиться звичайним провайдером веб-сайтів), ви платите за кожен з цих ресурсів на основі оплати за кожну хвилину або навіть за кожну секунду, зі знижками, які застосовуються, коли ваші послуги активно використовуються. На рисунку 1.10. наведено зображення з головною сторінкою Google Cloud Platform. [16]

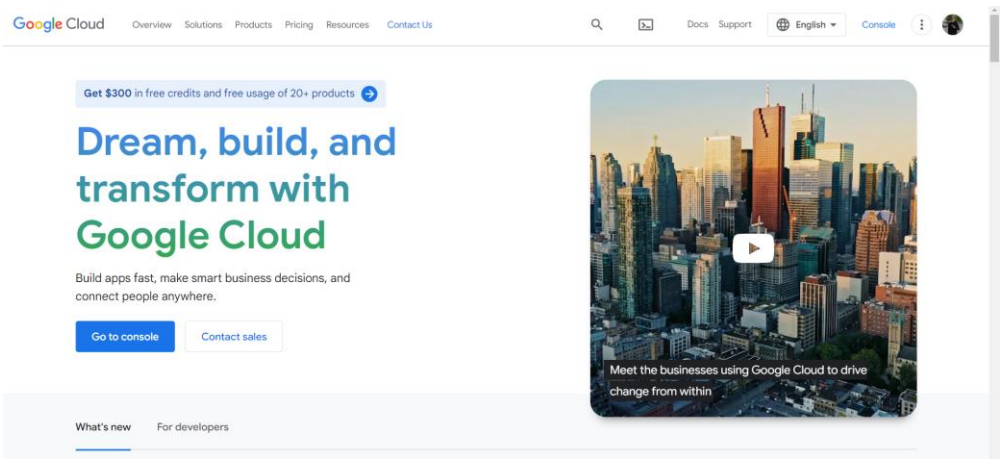

Рисунок 1.10 – головна сторінка GCP

З погляду материнської компанії Alphabet, GCP є окремим бізнес-підрозділом, який задовольняє бізнес-потреби підприємств та, в деяких випадках, фізичних осіб, щодо розгортання програмного забезпечення, яке можна використовувати через веб-браузери або веб-додатки. GCP орендує програмне забезпечення, разом з ресурсами, необхідними для підтримки цього програмного забезпечення, таких як обчислювальні ресурси (віртуальні машини, контейнери, сервери), мережеві ресурси (включаючи мережеві карти, балансувальники навантаження) та різноманітні послуги, такі як бази даних, аналітика даних, штучний інтелект, машинне навчання та інші. Крім того, GCP

також надає ряд інструментів та служб для розробки, тестування, розгортання та керування додатками, такі як Kubernetes Engine, App Engine, Cloud Storage, Cloud Functions та багато інших. [13]

З використанням GCP ми можемо використовувати різні можливості, такі як масштабування додатків вгору або вниз, автоматичне керування ресурсами, резервне копіювання та відновлення даних, захист даних, географічна реплікація даних та багато іншого. GCP також має глобальну мережу дата-центрів, що дозволяє розгортати свої додатки та послуги в різних регіонах світу з метою оптимізації продуктивності та доступності для користувачів. [16]

Ось опис деяких основних сервісів Google Cloud Platform:

Google Compute Engine (GCE): Це віртуальні машини, що дозволяють розробникам створювати та керувати віртуальними серверами в хмарному середовищі. Вони надають можливість вибрати операційну систему, налаштувати обладнання, мережу та зберігання, що дозволяє гнучко налаштовувати віртуальні сервери згідно з вимогами проекту. На рисунку 1.11 зображена головна сторінка GCE [20]

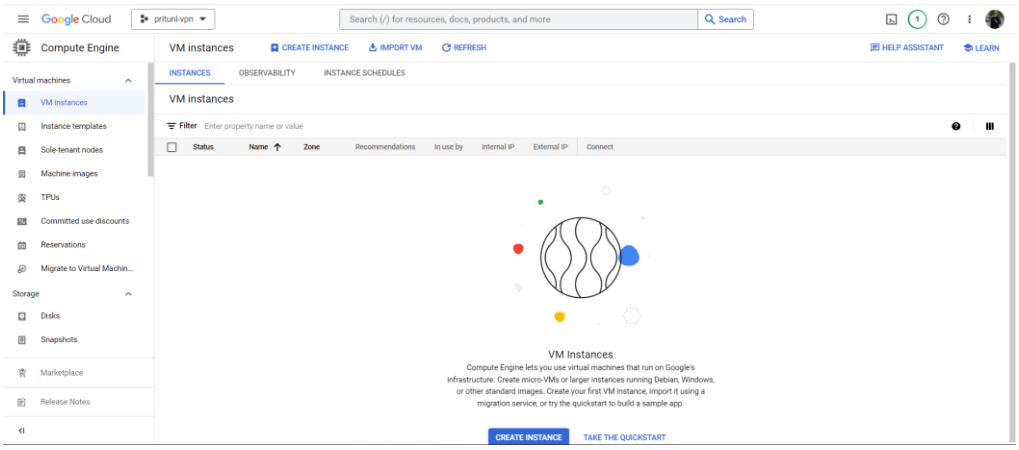

Рисунок 1.11 – головна сторінка GCE

Google Kubernetes Engine (GKE): Це керований сервіс контейнеризації, який дозволяє створювати, керувати та розгортати контейнери з використанням оркестрації Kubernetes. GKE дозволяє розробникам швидко створювати та

керувати масштабованими додатками, що базуються на контейнерах. На рисунку 1.12. зображена головна сторінка GKE [17]

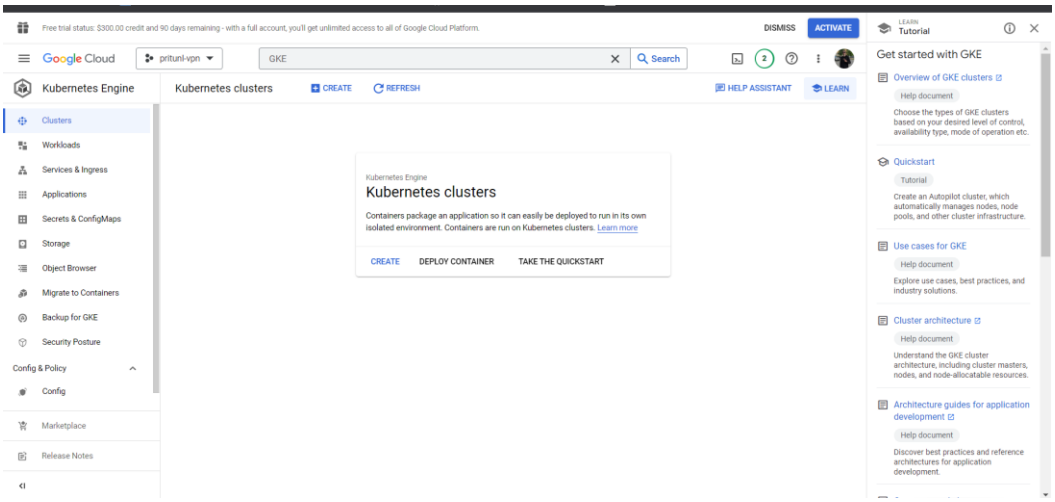

Рисунок 1.12 – головна сторінка GKE

Google Cloud Storage: Це служба зберігання об'єктів, яка надає необмежений доступ до збережених даних з будь-якого місця у світі. Вона дозволяє зберігати, керувати та передавати дані в хмарному середовищі, забезпечуючи високий рівень надійності та безпеки. На рисунку 1.13. зображена головна сторінка GCS [17]

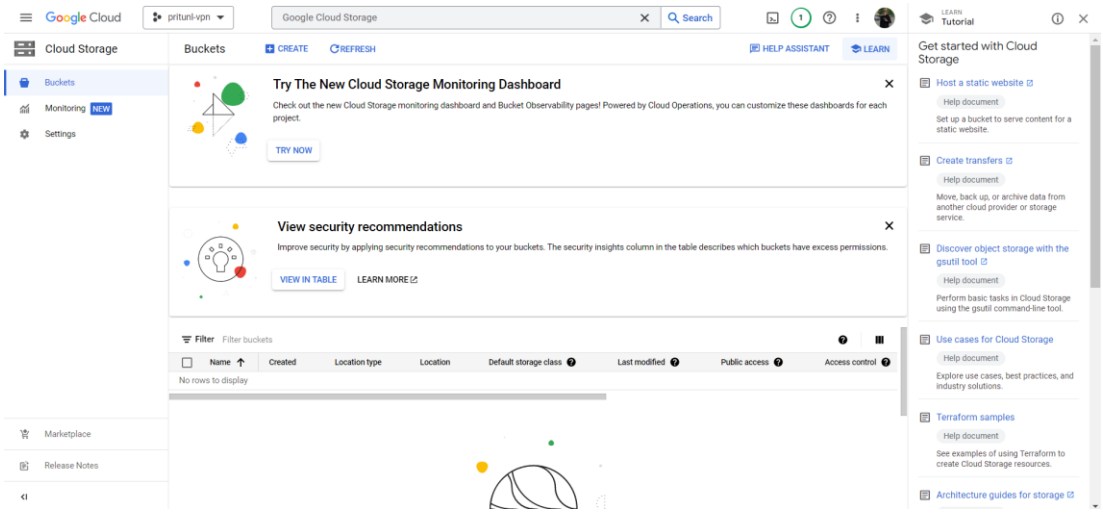

Рисунок 1.13 – головна сторінка GCS

Google Cloud BigQuery: Це служба аналітики даних, яка дозволяє виконувати швидкі та потужні аналітичні запити на великих обсягах даних. BigQuery використовує потужності масивних масштабів Google для аналізу даних в режимі реального часу, дозволяє виконувати складні запити та забезпечує інтеграцію з іншими інструментами аналітики даних. На рисунку 1.14. зображена головна сторінка Google Cloud BigQuery [17]

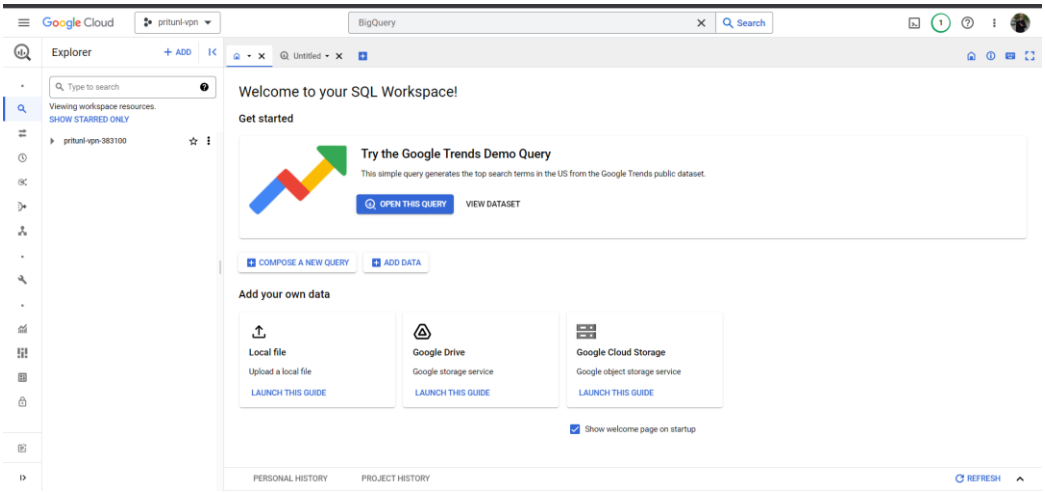

Рисунок 1.14 – головна сторінка Google Cloud BigQuery

Google Cloud AI Platform: Це платформа штучного інтелекту, яка дозволяє розробникам розробляти, навчати та розгортати моделі машинного навчання. Вона надає інструменти для розробки моделей, навчання на великих обсягах даних та оцінки їх ефективності. AI Platform також має інтеграцію з іншими сервісами Google Cloud, що дозволяє легко використовувати моделі машинного навчання в різних застосунках. [13]

Google Cloud Firestore: Це гнучка та масштабована база даних документів, яка дозволяє зберігати та синхронізувати дані в реальному часі між додатками та пристроями. Firestore забезпечує швидкий доступ до даних та автоматичну масштабованість, що робить його відмінним вибором для розробки додатків в реальному часі. На рисунку 1.15. зображена головна сторінка Google Cloud Firestore

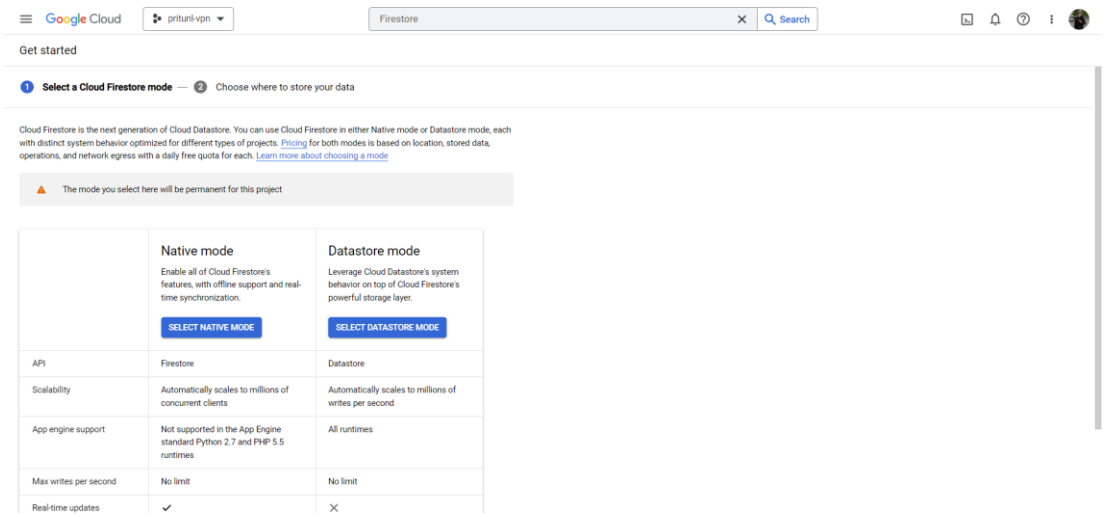

Рисунок 1.15– головна сторінка Google Cloud Firestore

Google Cloud Pub/Sub: Це послуга асинхронної передачі повідомлень, яка дозволяє створювати, надсилати та отримувати повідомлення між незалежними компонентами додатків. Pub/Sub забезпечує гнучкий та масштабований спосіб обміну даними між компонентами додатків в режимі реального часу.

Google Cloud Identity and Access Management (IAM): Це сервіс керування доступом, який дозволяє керувати правами доступу користувачів до різних ресурсів на Google Cloud Platform. IAM дозволяє встановлювати деталізовані рівні доступу, контролювати права користувачів та груп користувачів, створювати власні ролі та налаштовувати політики безпеки. На рисунку 1.16. зображена головна сторінка IAM

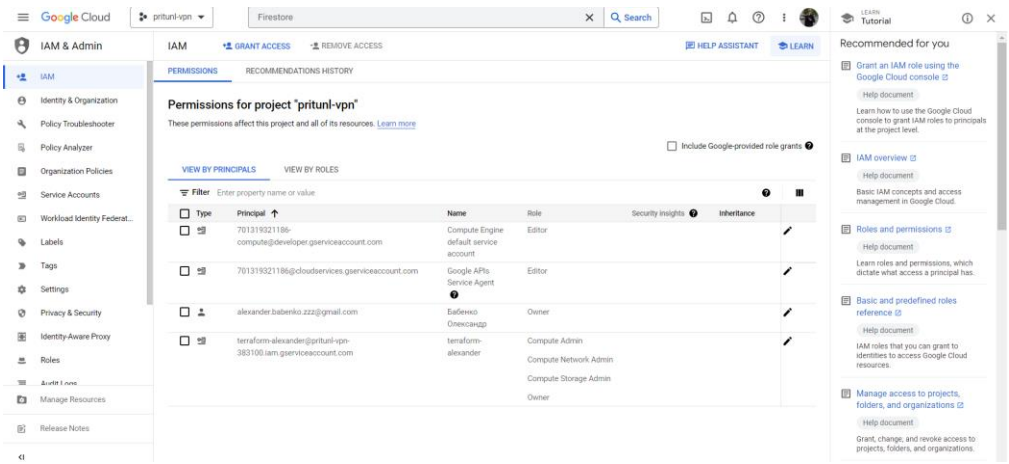

Рисунок 1.16 – головна сторінка IAM

Google Cloud CDN: Це мережева служба доставки контенту, яка забезпечує швидку та надійну доставку веб-контенту користувачам зі світових розподілених серверів. Cloud CDN дозволяє прискорювати завантаження веб-сторінок, зменшувати латентність та оптимізувати використання мережевих ресурсів. На рисунку 1.17. зображена головна сторінка Google Cloud CDN

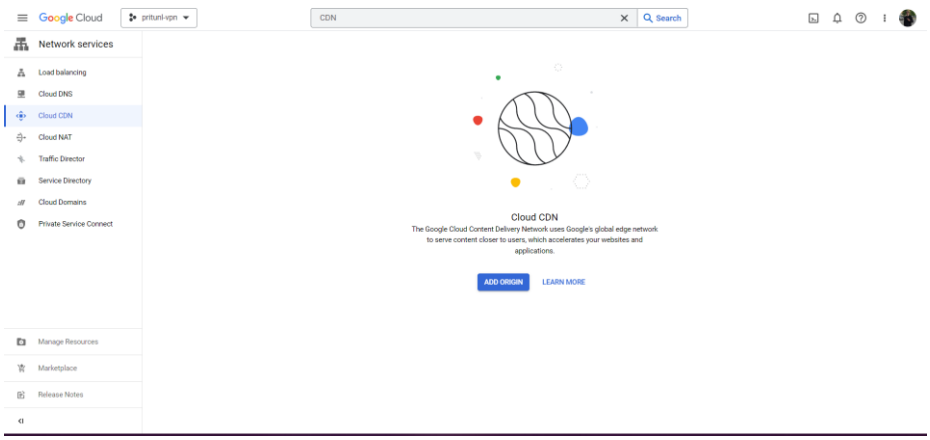

Рисунок 1.17 – головна сторінка Google Cloud CDN

Google Cloud DNS: Це служба доменних імен, яка дозволяє реєструвати, керувати та масштабувати доменні імена для веб-додатків на Google Cloud Platform. Cloud DNS забезпечує швидке та надійне керування доменними іменами, можливість налаштування зон відповідальності та інтеграцію з іншими сервісами Google Cloud. На рисунку 1.18. зображена можливість активувати Cloud DNS API

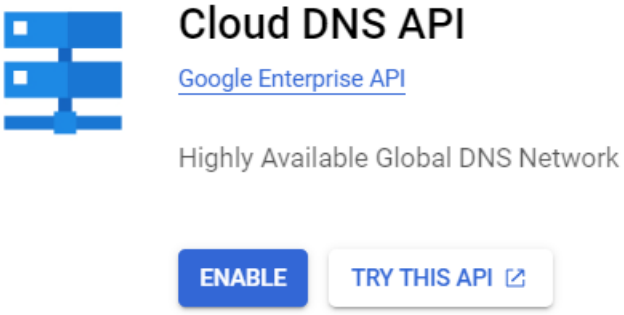

Рисунок 1.18 – Cloud DNS API

Google Cloud Security Command Center: Це служба безпеки, яка надає засоби для моніторингу, аналізу та виявлення потенційних загроз безпеці в середовищі Google Cloud Platform. Security Command Center дозволяє виявляти вразливості, аналізувати журнали подій, контролювати доступ та налаштовувати правила безпеки для різних ресурсів на платформі.

Загалом, GCP є високопродуктивною хмарною платформою, яка дозволяє компаніям розробляти, розгортати та керувати своїми додатками та послугами в хмарі, забезпечуючи масштабованість, надійність, безпеку та гнучкість.

Pritunl — це програмне рішення з відкритим кодом для налаштування та керування віртуальною приватною мережею (VPN). Він забезпечує простий у використанні веб-інтерфейс і підтримує широкий спектр протоколів VPN, включаючи OpenVPN і WireGuard. На рисунку 1.19. зображений офіційний сайт Pritunl VPN

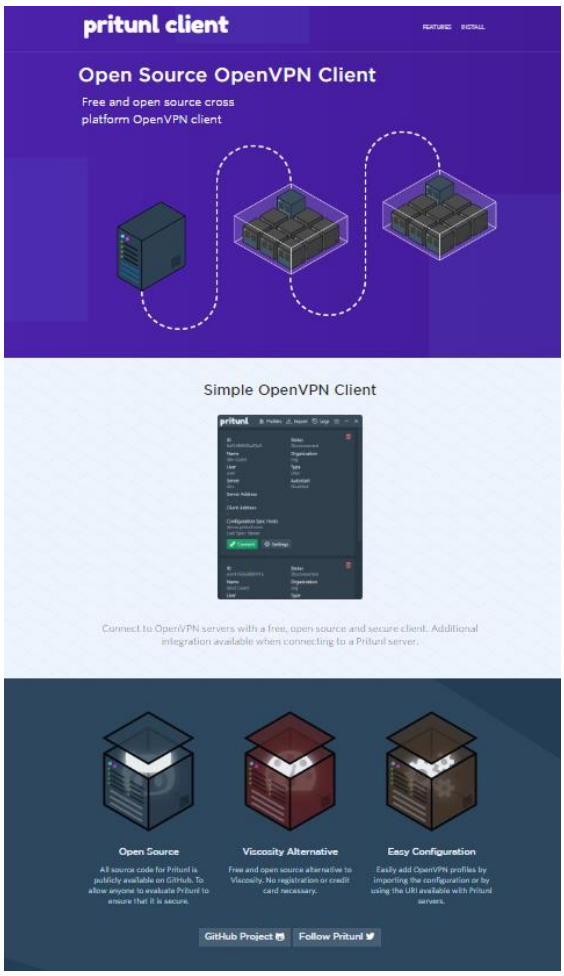

Рисунок 1.19 – офіційний сайт Pritunl VPN

Деякі з ключових функцій Pritunl включають:

Керування користувачами: Pritunl забезпечує централізований спосіб керування обліковими записами користувачів і підтримує багатофакторну автентифікацію (MFA) для додаткової безпеки.

Сегментація мережі: Pritunl дозволяє адміністраторам сегментувати свою мережу на різні організаційні підрозділи (OU) і призначати різні привілеї різним користувачам. Це допомагає забезпечити дотримання політики безпеки та запобігання несанкціонованому доступу.

Балансування навантаження та відновлення після збоїв: Pritunl підтримує балансування навантаження та відновлення після збою, щоб забезпечити високу доступність та стійкість VPN-з'єднань. Це означає, що якщо один сервер VPN виходить з ладу, навантаження автоматично перенаправляється на інший сервер.

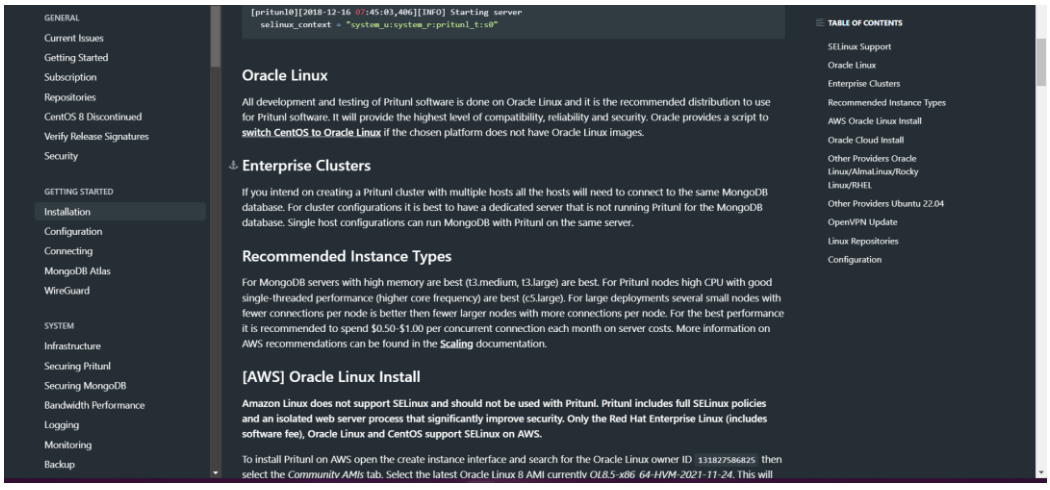

Рисунок 1.20 – офіційна документація по розгортанню Pritunl VPN

Просте розгортання: Pritunl можна розгортати на широкому спектрі платформ, включаючи хмарні служби, такі як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP), а також на локальних серверах. На рисунку 1.20. зображена офіційна документація по розгортанню Pritunl VPN на різних серверах

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ПІДБІР АЛГОРИТМІВ

2.1 Опис і розробка інфраструктурної діаграми

На рисунку 2.1 зображено діаграму взаємодії користувача з інфраструктурою.

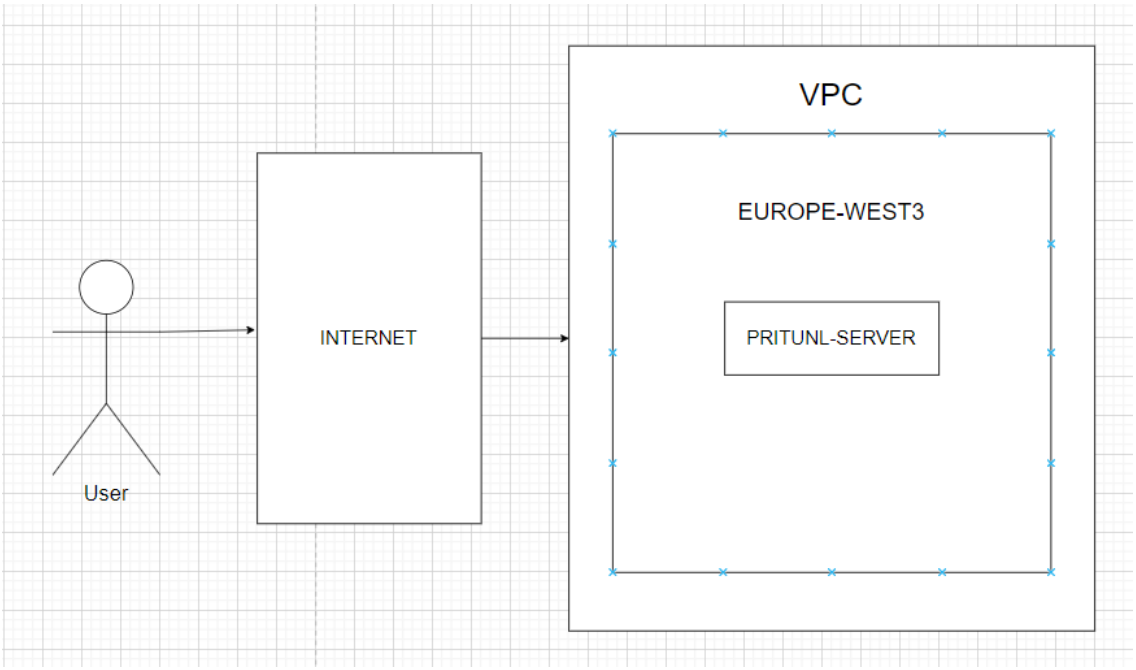

Рисунок 2.1 – діаграма взаємодії користувача з інфраструктурою

В цій діаграмі представляю такі компоненти:

Інтернет - зовнішня мережа, через яку здійснюється з'єднання з VPC.

VPC Network (Virtual Private Cloud) - це віртуальна приватна мережа, у якій налаштований регіон europe-west3. VPC використовується для групування ресурсів, таких як сервери, бази даних та інші, щоб вони могли взаємодіяти між собою в межах цього віртуального простору. В рамках цієї VPC створено pritunl-server.

Pritunl-server - це сервер, на якому встановлено налаштоване VPN-рішення. Воно забезпечує можливість з'єднання з приватною мережею VPC з

будь-якого місця через Інтернет. А також, Pritunl-VPN надає безпечний тунель для доступу до ресурсів, які розташовані у VPC Network.

2.2 Алгоритм роботи VPN серверу

На рисунку 2.2 зображено діаграму алгоритму роботи VPN серверу

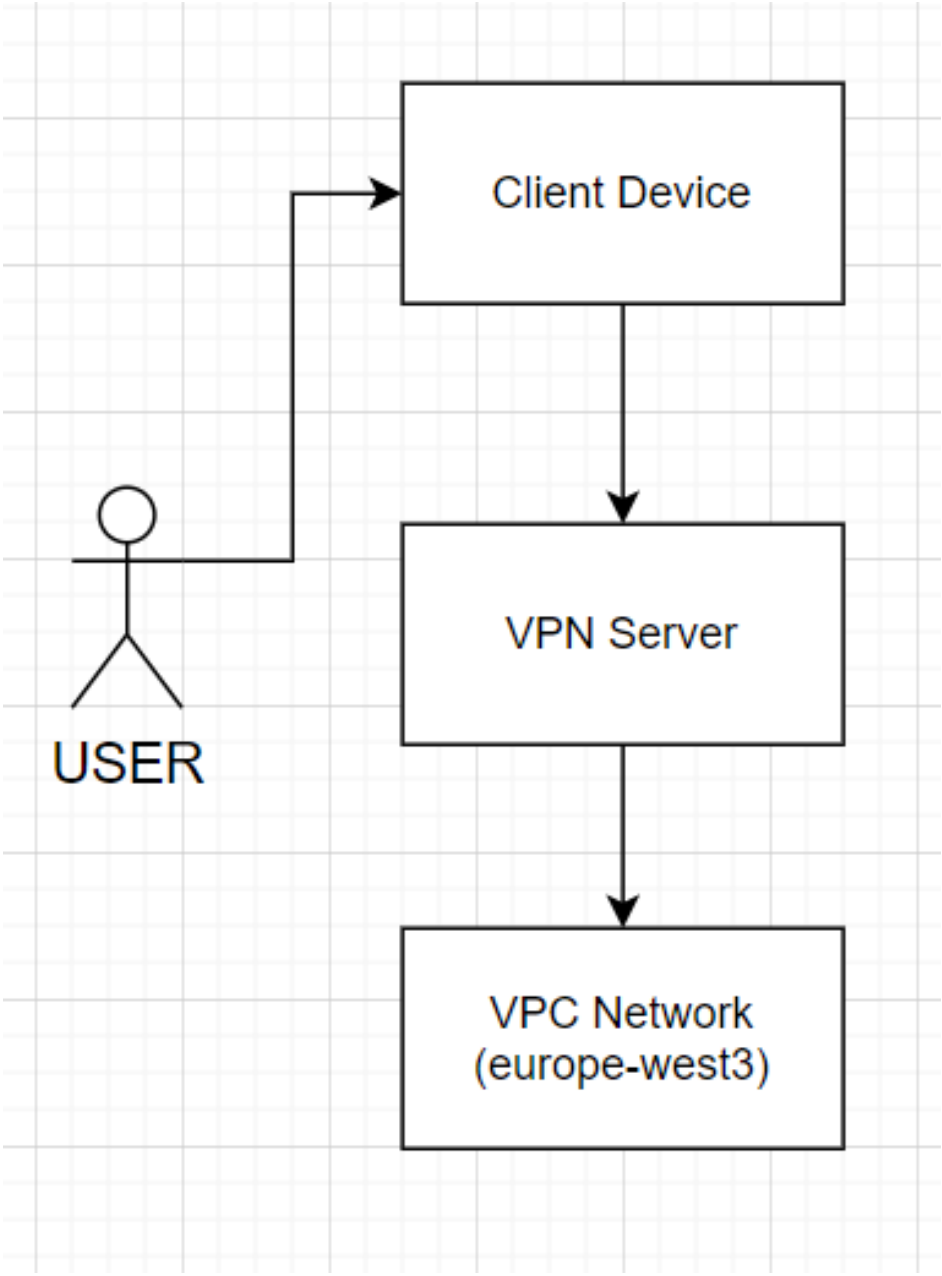

Рисунок 2.2 – діаграма алгоритму роботи VPN серверу

Користувач (USER) на своєму пристрої (Client Device) запускає VPN-клієнтську програму та встановлює з'єднання з VPN сервером (VPN Server).

Клієнтська програма передає запит на з'єднання до VPN сервера.

VPN сервер отримує запит від клієнта і розпочинає процес аутентифікації користувача. Він може перевірити логін/пароль або використати сертифікати безпеки для перевірки ідентифікації користувача.

Після успішної аутентифікації, VPN сервер встановлює шифроване з'єднання з клієнтом. Це забезпечує конфіденційність і цілісність даних, які передаються між клієнтом та сервером.

Після встановлення VPN-тунелю, клієнтський пристрій та VPN сервер обмінюються зашифрованим трафіком. Клієнтський пристрій шифрує свій вихідний трафік, а VPN сервер дешифрує його та пересилає до призначеного місця.

VPN сервер, як посередник, перенаправляє трафік між клієнтом та ресурсами у VPC Network (europe-west3). Це означає, що клієнтський пристрій може отримувати доступ до ресурсів, які знаходяться у внутрішній мережі VPC Network, таких як сервери, бази даних тощо та “виходити” в інтернет з IP-адресою VPN-серверу.

Всі дані, які передаються між клієнтом та VPN сервером, залишаються зашифрованими і захищеними від перехоплення та перегляду.

2.3 Компоненти системи та взаємодія з ними

Після створення створення та налаштування VPN-рішення, користувач зможе підключитись до моєї VPN мережі і виконувати різні операції в інтернеті зі збереженням приватних даних користувача в безпеці та захищеними від доступу третіх осіб. При розробці VPN-рішення було передбачено наступні

можливості: захист даних, конфіденційність, доступ до обмеженого контенту всередині нашої країни, безпеки зв'язку та захист від атак, таких як DDoS атаки, MITM атаки, фішинг тощо.

Для використання VPN-рішення на базі Terraform та Pritunl в Google Cloud, достатньо завантажити json-файл, який містить інформацію, необхідну для аутентифікації та авторизації користувача в хмарному середовищі та пройти короткий етап налаштування Pritunl VPN. Після цього користувач зможе отримати доступ до власного VPN серверу.

Потенційні користувачі VPN-рішення на базі Terraform та Pritunl в Google Cloud можуть бути різними організаціями та окремими користувачами, які мають потребу у безпечному та захищеному з'єднанні до Інтернету. Основні категорії користувачів можуть включати:

Бізнес-клієнти - компанії будь-якого розміру, які мають потребу у безпечному з'єднанні для підключення віддалених робітників, доступу до обмеженого контенту та сервісів, які можуть бути заблоковані у їхній країні або регіоні, а також захисту від різних видів атак.

Організації з великою кількістю віддалених робітників, такі організації можуть бути компаніями, установами освіти або державними організаціями, які мають потребу у безпечному та захищеному з'єднанні для підключення віддалених співробітників до центральної мережі.

Люди, які працюють з дому, студенти або ті, хто просто хоче зберігати своє з'єднання з Інтернетом в безпеці.

Компанії, які пропонують послуги в Інтернеті, такі компанії можуть мати потребу у використанні VPN для захисту від різних видів атак, а також для надання можливості своїм клієнтам підключатись до їхніх сервісів з будь-якої точки світу.

WSL (Windows Subsystem for Linux) - це функціональність, яка дозволяє користувачам Windows запускати та використовувати програми Linux у середовищі Windows.

WSL включає дві версії - WSL 1 та WSL 2. WSL 1 є майже повністю сумісним з Linux API, використовуючи перекладачів системних викликів, щоб забезпечити доступ до файлової системи Linux та мережі. WSL 2 замість цього використовує віртуалізацію на рівні ядра, що дозволяє користувачам використовувати більш широкий діапазон програм Linux та підвищує продуктивність за рахунок зменшення накладних витрат на переклад системних викликів.

Windows Subsystem for Linux (WSL) - це компонент, що інтегрує середовище Linux з операційною системою Windows. Він розроблений Microsoft для полегшення розробки програмного забезпечення, забезпечення сумісності та полегшення використання інструментів, які зазвичай використовуються у середовищі Linux.

WSL використовує технологію віртуалізації, щоб створити окреме середовище для Linux, використовуючи ядро Linux. При цьому віртуалізація відбувається на рівні системи, що означає, що вона працює ближче до апаратного забезпечення та забезпечує кращу продуктивність порівняно з традиційними віртуальними машинами.

WSL дозволяє виконувати бінарні файли Linux, включаючи утиліти командного рядка та програми, безпосередньо в середовищі Windows. Це означає, що розробники та системні адміністратори можуть використовувати звичайні інструменти Linux без необхідності переходити на повноцінну Linux-систему, надає можливість встановлювати і використовувати різні дистрибутиви Linux з офіційного магазину Windows або за допомогою ручної установки. Наприклад, можна встановити Ubuntu, Fedora, Debian та багато інших дистрибутивів.

WSL 2 внесло значні покращення у порівнянні зі своїм попередником WSL 1. У WSL 2 використовується віртуалізація на рівні ядра, що дозволяє запускати повноцінне ядро Linux в окремій віртуальній машині. Це забезпечує кращу сумісність з Linux та поліпшену продуктивність. WSL 2 має підтримку системного виклику, що означає, що додатки, що використовують системні

виклики, будуть працювати краще в WSL 2. Також він використовує нову файлову систему - 9P, яка забезпечує кращу продуктивність при доступі до файлів між Windows та Linux.

Також WSL дозволяє використовувати файли і папки Windows з середовища Linux та навпаки. Це дозволяє зручно редагувати файли Linux з використанням улюблених текстових редакторів у Windows та використовувати розроблені в Linux інструменти на Windows-платформі. Користувачам дозволяється запускати графічні додатки Linux на Windows. Для цього можна використовувати сервер віконних систем, такий як X-сервер, у поєднанні з додатком для Windows, який може підключатися до X-сервера.

WSL також інтегрується з інструментами розробки, такими як Visual Studio Code, що дозволяє розробникам працювати зі своїми проектами на платформах Windows та Linux без переключення середовищ. WSL 2 підтримує автоматичне оновлення, що означає, що користувачі отримуватимуть оновлення та поліпшення безпосередньо від Microsoft через Windows Update.

Він підходить для розробки програмного забезпечення, системного адміністрування, тестування та навчання Linux. З його допомогою можна виконувати команди та скрипти командного рядка Linux, розробляти та запускати веб-сервери, бази даних, мережеві сервіси та багато іншого, спрощувати спільну роботу між розробниками, які працюють на різних платформах. Він дозволяє розробникам Windows та Linux працювати над спільними проектами, ділитися кодом та ресурсами без проблем сумісності.

Для встановлення WSL користувачі повинні виконати кілька кроків. По-перше, потрібно ввімкнути функцію "Підтримка підсистем Linux" у налаштуваннях Windows. Це зображено на рисунку 2.3. Далі, користувачі можуть завантажити та встановити дистрибутив Linux з магазину Windows або вручну з терміналу. Список доступних дистрибутивів у якості підсистем зображено на рисунку 2.4.

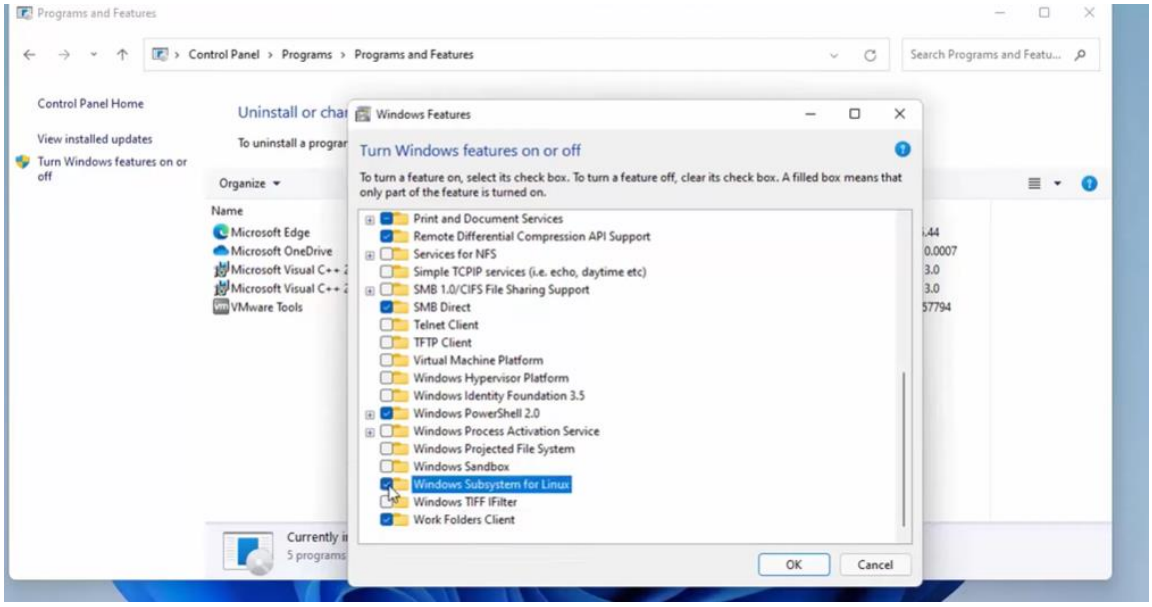

Рисунок 2.3 – увімкнення функції "Підтримка підсистем Linux"

Після встановлення користувачі можуть запускати програми Linux у вікні терміналу Windows за допомогою командного рядка. WSL також дозволяє користувачам звертатися до файлів та папок від Windows, що дозволяє зручно обмінюватися даними між обома середовищами.

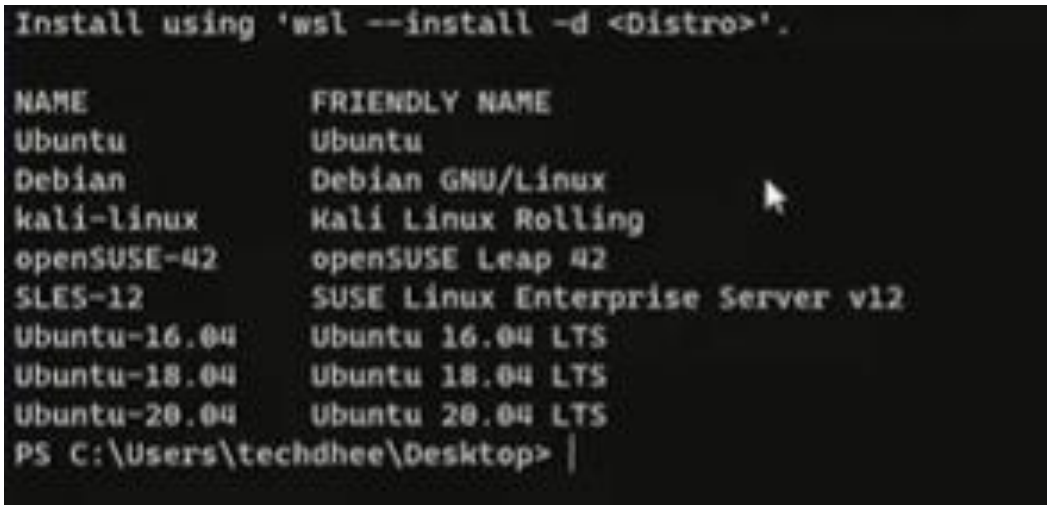

Рисунок 2.4 – список доступних дистрибутивів у якості підсистеми

Загалом, WSL є потужним інструментом, який дозволяє інтегрувати середовище Linux з операційною системою Windows. Він спрощує розробку програмного забезпечення, полегшує сумісність між платформами та надає доступ до багатьох корисних інструментів Linux на Windows-платформі.

Microsoft Visual Studio є одним з продуктів компанії Майкрософт і представляє собою інтегровану середу розробки програмного забезпечення разом з інструментальними засобами. Цей продукт надає можливість створювати консольні додатки, додатки з графічним інтерфейсом, включаючи підтримку технології Windows Forms, а також веб-сайти, веб-додатки та веб-служби для різних платформ, які підтримуються Microsoft, такі як Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework і Microsoft Silverlight.

Visual Studio містить редактор вихідного коду з підтримкою технології IntelliSense та можливість простого рефакторингу коду. Вбудований відладчик дозволяє працювати як з вихідним кодом, так і з машинним рівнем. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу додатків, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

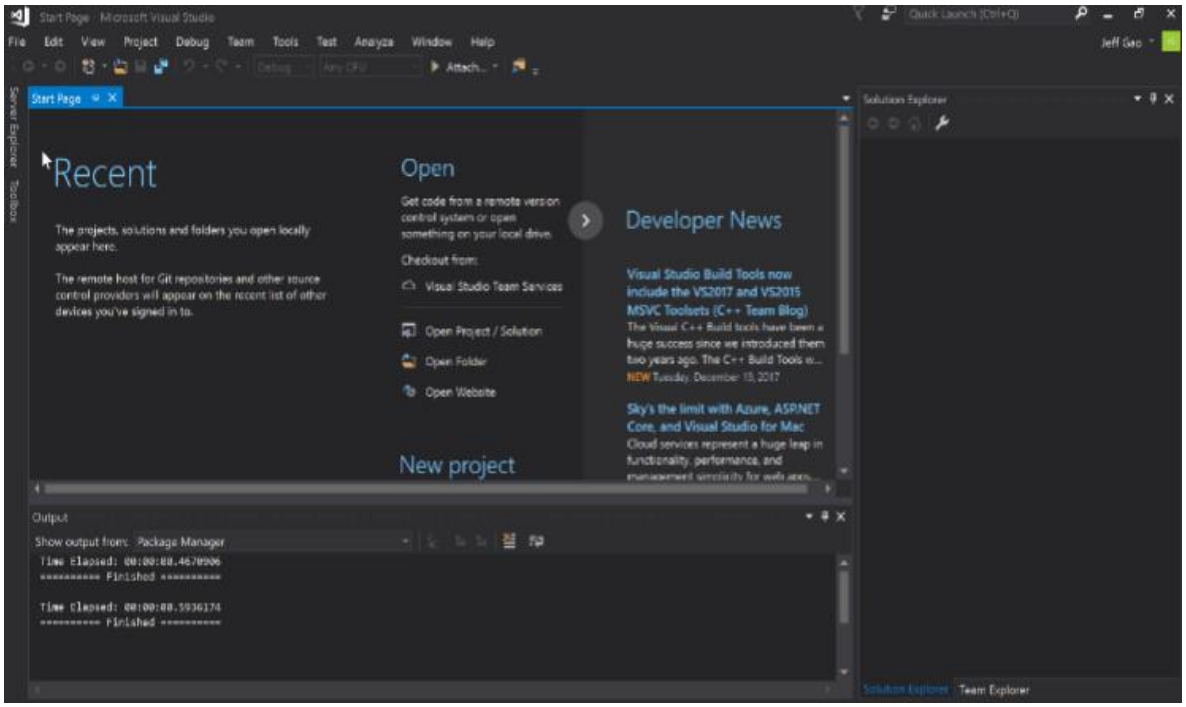

Рисунок 2.5 – початок роботи у Visual Studio Code

Visual Studio також дозволяє розширювати свої можливості шляхом створення та підключення плагінів для підтримки систем контролю версій

вихідного коду, додавання нових наборів інструментів, таких як редагування та візуальне проектування коду на мовах програмування, орієнтованих на об'єкти, або для інших аспектів розробки програмного забезпечення. Загальний вигляд вікна Visual Studio під час його завантаження, початку роботи та створення нового проекту зображений на рисунку 2.5.

У інтерфейсі Visual Studio є такі вікна інструментів, які зображено на рисунку 2.6. та 2.7.

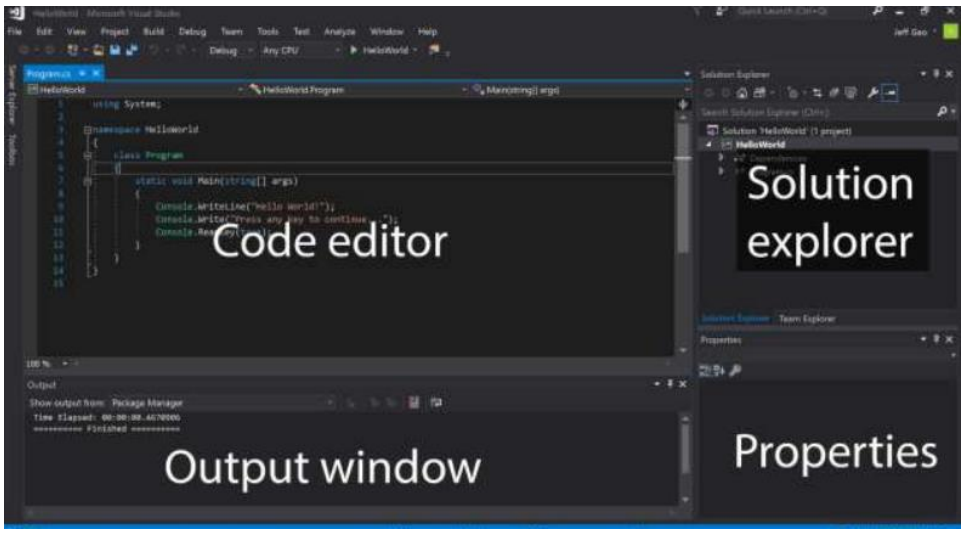

Рисунок 2.6 – Вікна інструментів

Вихідне вікно - показує повідомлення про налагодження, помилки, попередження компілятора, стан та інші результати.

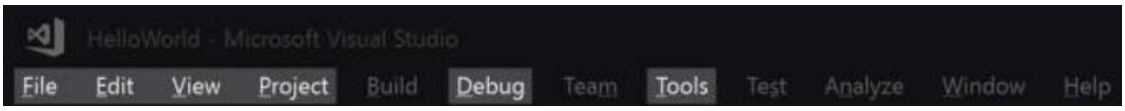

Рисунок 2.7 – Верхня панель

Редактор коду - інструмент для написання програмного коду. Провідник рішень - відображає файли, з якими ви працюєте. Властивості - надають додаткову інформацію та контекст щодо вибраних частин проекту.

3 РЕАЛІЗАЦІЯ КОМПЮТЕРНОЇ МЕРЕЖІ З VPN СЕРВЕРОМ

3.1 Реалізація інфраструктури з використанням Terraform

Terraform є інструментом для реалізації інфраструктури у вигляді коду, що дозволяє визначати структуру та параметри інфраструктури. У нашому випадку, ми використовуємо Terraform для створення інфраструктури в Google Cloud. [22]

Код складається з двох модулів, які використовуються для створення наступних компонентів інфраструктури: network і instances. Кожен з цих модулів має блоки коду, які дозволяють налаштувати і створити відповідну компоненту. Список модулів показаний на рисунку 3.1.

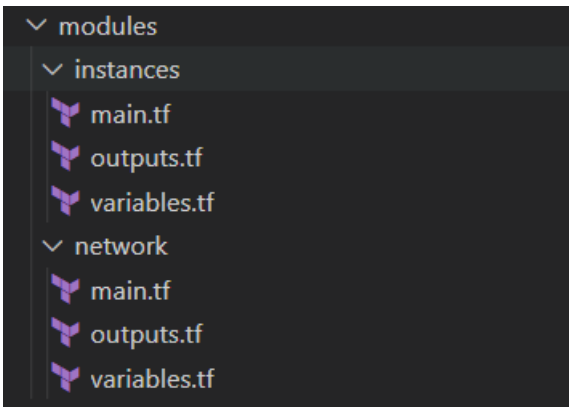

Рисунок 3.1 – список модулів

Розглянемо модуль instances.

name: ім'я екземпляра в Google Cloud.

machine_type: тип машини

can_ip_forward: відправка та отримання пакетів з різних ресурсів на різні IP адреси.

metadata_startup_scrip: шлях до файлу скрипта запуску, який буде виконаний при запуску екземпляра.

tags: мітки (теги) для екземпляра, які можуть використовуватися для організації та фільтрації трафіку.

boot_disk: параметри ініціалізації завантажувального диска екземпляра.

initialize_params: образ операційної системи

network_interface: параметри мережевого інтерфейсу екземпляра.

network: ім'я мережі, до якої буде підключений екземпляр

access_config: доступ до мережі.

metadata: метадані екземпляру.

ssh-keys: публічний ключ SSH, який буде використовуватися для підключення до екземпляру.

Вміст файлу main.tf модулю instances зображений на рисунку 3.2

modules > instances > main.tf

```
resource "google_compute_instance" "Instance" {
  name                    = "pritunl-server"
  machine_type            = "e2-micro"
  can_ip_forward          = true
  metadata_startup_script = file("./startup_script.sh")

  tags = ["pritunl-server", "http-server", "https-server"]

  boot_disk {
    initialize_params {
      image = "ubuntu-2004-focal-v20230104"
    }
  }

  network_interface {
    network = var.network_name
    access_config {}
  }

  metadata = {
    ssh-keys = format("%s%s%s", var.ssh_user, ":", file(var.ssh_pub_key_path))
  }
}

```

Рисунок 3.2 – main.tf файл модулю instances

Всі параметри інстансу, такі як назва, тип машини, можливість пересилати IP-пакети, теги, завантажувальний диск та мережевий інтерфейс, визначаються у цьому блоку. Крім того, використовується скрипт запуску, який вказується шляхом до файлу startup_script.sh, та SSH-ключі для авторизації, які форматуються змінними var.ssh_user та var.ssh_pub_key_path. [19]

Вміст файлу outputs.tf зображено на рисунку 3.3.

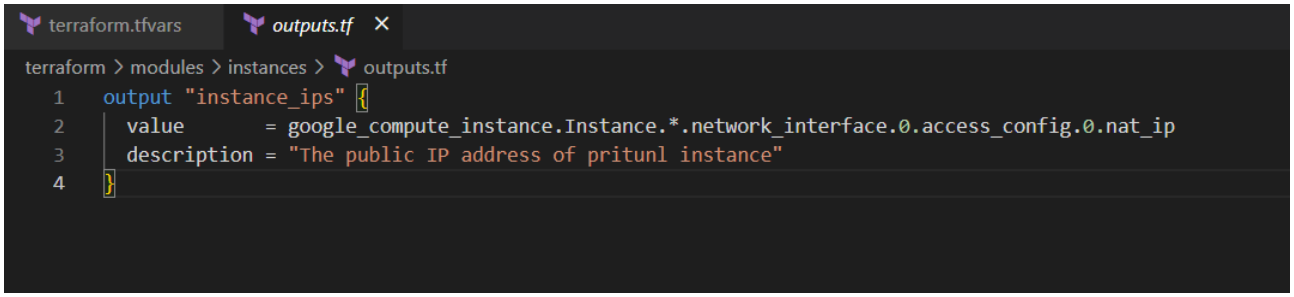

Рисунок 3.3 – outputs.tf файл модулю instances

Файл outputs.tf в Terraform використовується для визначення виведених значень (output values) з вашого інфраструктурного коду. Цей файл дозволяє вам оголосити значення, які Terraform може вивести після успішного розгортання вашої інфраструктури. [21]

Файл variables.tf включає три оголошені змінні:

network_name: змінна для визначення назви мережі або інфраструктури в середовищі, в якому працює інфраструктура. [17]

ssh_user: змінна для визначення імені користувача SSH, якого слід використовувати для з'єднання з VPN інстансом. [14]

ssh_pub_key_path: змінна для вказівки шляху до публічного ключа SSH, який буде використовуватися для автентифікації з'єднань SSH з VPN інстансом.

Вміст файлу variables.tf модулю network зображений на рисунку 3.4

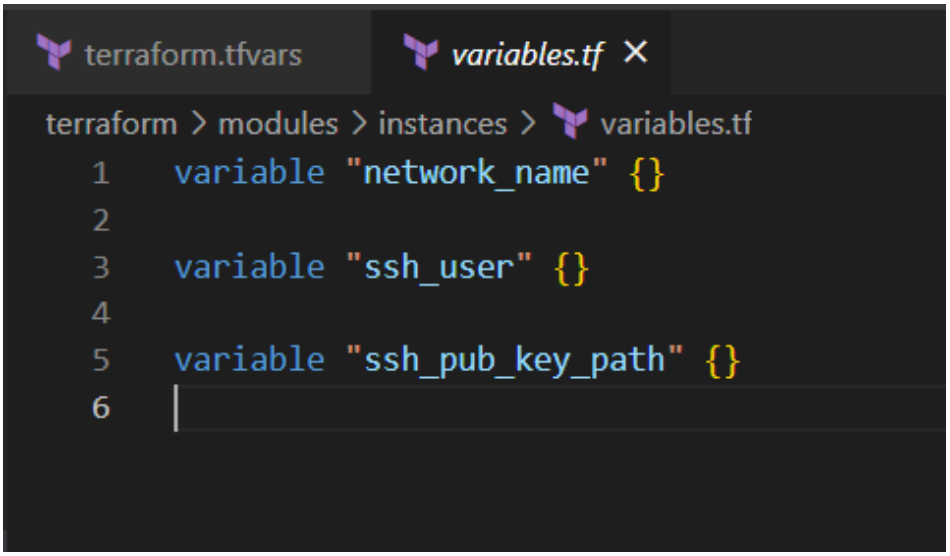

Рисунок 3.4 – variables.tf файл модулю instances

Розглянемо модуль network. [22]

name: ім'я правила брандмауера в Google Cloud.

network: Вказує ім'я мережі (VPC), до якої буде застосовуватися це правило.

description: опис правила брандмауера.

allow: список дозволених з'єднань (allow rules) для правила.

protocol: протокол, який дозволений для з'єднання.

ports: список портів, які дозволені для з'єднання. У даному випадку дозволені порти SSH (22), HTTP (80), HTTPS (443) та порт для Pritunl server, зазначений у змінній var.server_port.

target_tags: мітки (теги) цього правила, які повинні бути надані екземплярам для застосування правила.

source_ranges: список діапазонів IP-адрес, які мають доступ до цього правила. У даному випадку дозволений доступ з будь-якої IP-адреси (0.0.0.0/0).

Вміст файлу main.tf модулю network зображений на рисунку 3.5

Рисунок 3.5 – main.tf файл модулю network

Цей модуль Terraform створює мережу з заданими параметрами за допомогою ресурсу google_compute_network і правило брандмауера з заданими дозволами за допомогою ресурсу google_compute_firewall. Після застосування конфігурації Terraform ці ресурси будуть створені в Google Cloud Platform з вказаними властивостями. [17]

3.2 Ініційний скрипт у Terraform та Ansible Playbook

Ініційний скрипт у Terraform - це файл, який містить команди та конфігурацію для налаштування зовнішніх ресурсів та початкового стану інфраструктури. Цей скрипт виконується при запуску Terraform для побудови та керування інфраструктурою.

Ansible є інструментом для автоматизації конфігурації та управління системами. Він дозволяє змінювати стан системи через звичайні файли конфігурації, а не скрипти або програми. Базуючись на декларативній моделі програмування, Ansible дає можливість користувачам описувати бажаний стан

системи, і самостійно забезпечує зміну системи відповідно до цієї моделі. Ansible надає зручний та ефективний спосіб управління системами, де користувачі можуть визначати задачі в термінах бажаного стану системи. [3]

Для реалізації автоматизації використовуючи Ansible я написав даний вміст configuration.yml, який зображений на рисунку 3.10

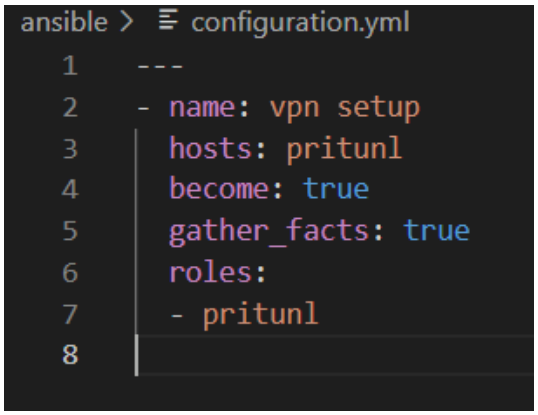

Рисунок 3.6 – вміст configuration.yml

"configuration.yml" в визначає параметри конфігурації для встановлення та налаштування VPN. [15]

name: vpn setup - Вказує назву цього блоку встановлення VPN. Це просто описовий рядок і не впливає на сам процес. [18]

hosts: pritunl - Визначає групу хостів, до якої будуть застосовані налаштування. У цьому випадку використовується група "pritunl". Можна використовувати інші назви груп або конкретні хост-машини.

become: true - Запитує Ansible на виконання команд з привілеями суперкористувача. Це дозволяє виконувати привілейовані дії під час встановлення VPN. [22]

gather_facts: true - Запитує Ansible зібрати факти про хост-машини перед виконанням дій. Це включає інформацію про операційну систему, мережеві налаштування та інші деталі. [17]

roles: - Вказує список ролей, які будуть використовуватися для встановлення VPN. У цьому випадку використовується роль "pritunl". Ролі

дозволяють організувати код Ansible у більш структуровану і повторно використовувану форму.

Розглянемо ansible роль. Моя роль поділена на директорії, такі як: defaults, handlers, tasks, vars.

Директорія "defaults": У цій директорії розміщений файл із значеннями за замовчуванням для змінних, які використовуються в ролі. Ці значення можуть бути перезаписані користувачем, якщо вони визначені в інших частинах ролі. Наприклад, тут встановлена версія mongodb для конфігурації Pritunl. Вміст файлу main.yml з директорії defaults, зображений на рисунку 3.7. [21]

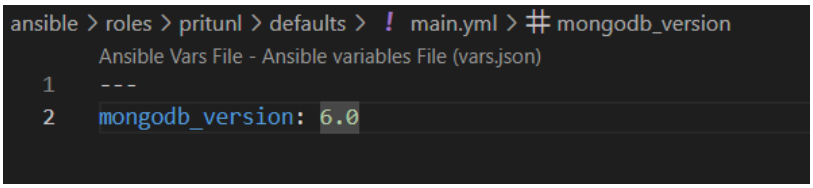

Рисунок 3.7 – вміст main.yml з директорії defaults

Директорія "handlers": У цій директорії розміщений файл із обробниками подій (handlers). Обробники подій - це дії, які виконуються в ролі відповідно до подій, спричинених іншими завданнями в плейбуці. В моєму прикладі обробник подій для старту служби VPN, увімкнення MongoDB та запуску MongoDB. Вміст файлу main.yml з директорії handlers, зображений на рисунку 3.8. [20]

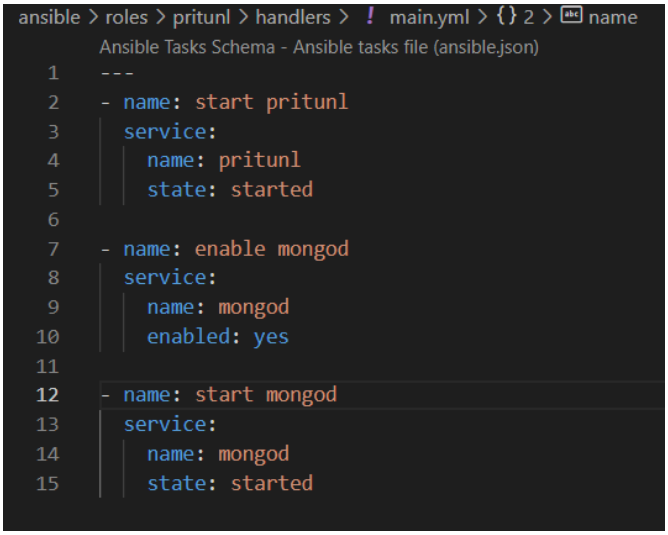

Рисунок 3.8 – вміст main.yml з директорії handlers

Директорія "tasks": У цій директорії розміщений файл із набором завдань, які виконуються в ролі. [22] Завдання - це окремі кроки або команди, які Ansible виконує на хост-машині. Вміст файлу main.yml з директорії tasks, зображений на рисунку 3.9. [21]

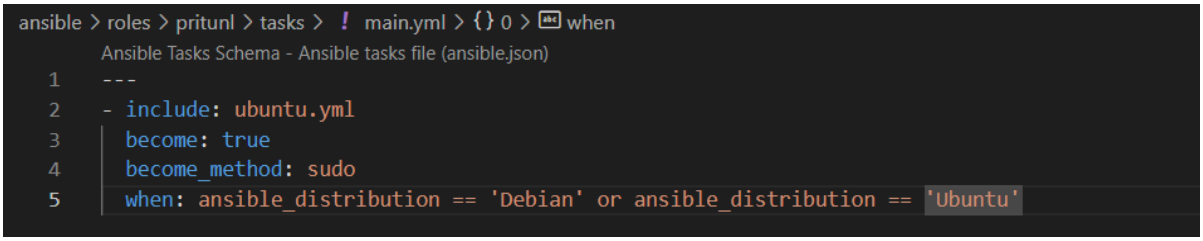

Рисунок 3.9 – вміст main.yml з директорії tasks

when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu' - Ця умовна конструкція вказує, що набір завдань виконуватиметься тільки в разі, якщо дистрибутив хост-машини є Debian або Ubuntu. [23] Це дозволяє встановлювати робити налаштування тільки для цих дистрибутивів, а інші дистрибутиви будуть пропускатися. [19]

Вміст файлу ubuntu.yml з директорії tasks, зображений на рисунку 3.10. та рисунку 3.11 [18]

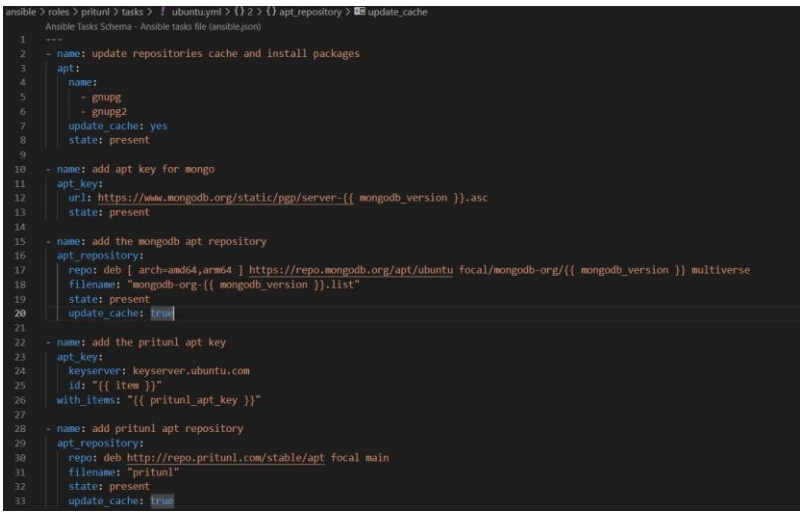

Рисунок 3.10 – вміст ubuntu.yml з директорії tasks

```
- name: Update and upgrade apt packages
  become: true
  apt:
    upgrade: yes
    update_cache: yes

- name: install pritunl and mongodb
  raw: apt-get --assume-yes install pritunl mongodb-org
  notify:
    - start pritunl
    - enable mongod
    - start mongod
```

Рисунок 3.11 – вміст ubuntu.yml з директорії tasks

Також Ansible playbook можна запустити у будь-який момент, та відлагодити, на відміну від init скрипта у Terraform. [24]

3.3 Налаштування Pritunl

Після встановлення Pritunl VPN на сервері я зробив наступні налаштування. У браузері я пишу публічну IP адресу серверу та потрапляє на 443 порт де «біжить» веб.інтерфейс Pritunl VPN.

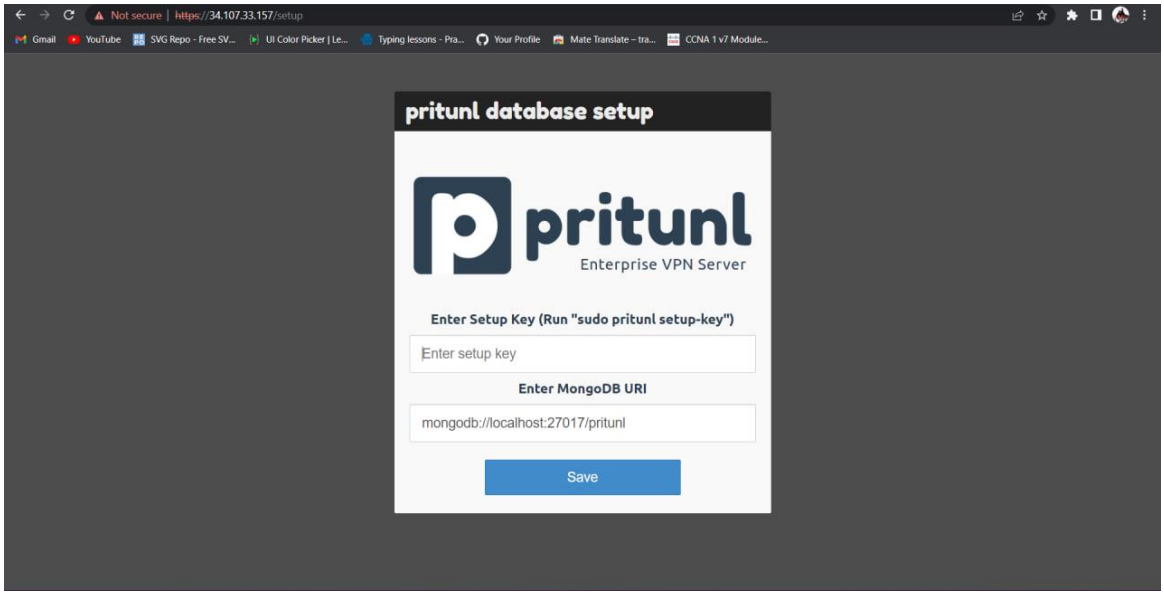

Рисунок 3.12 – веб.інтерфейс Pritunl VPN

Далі я підключаюсь до серверу по 22 порту та створюю setup-key за допомогою команди «sudo pritunl setup-key» та задаю налаштування по-замовчуванню для входу в адмін-панель. Це зображено на рисунку 3.13.

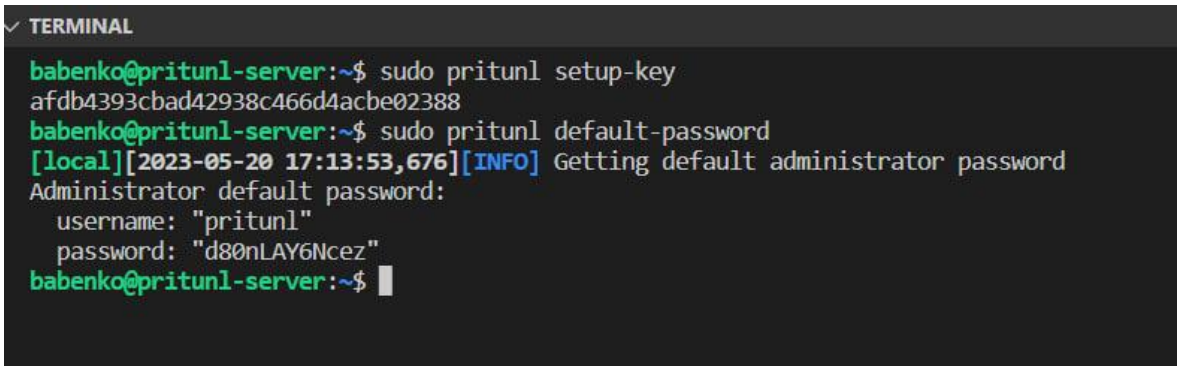

Рисунок 3.13 – створення setup-key та налаштування по-замовчуванню

Після входу в систему я задаю налаштування для серверу, вказавши порт який я «відкрив» у Terraform скрипті та налаштовую користувачів й організацію. Що зображено на рисунках 3.14, 3.15 та 3.16.

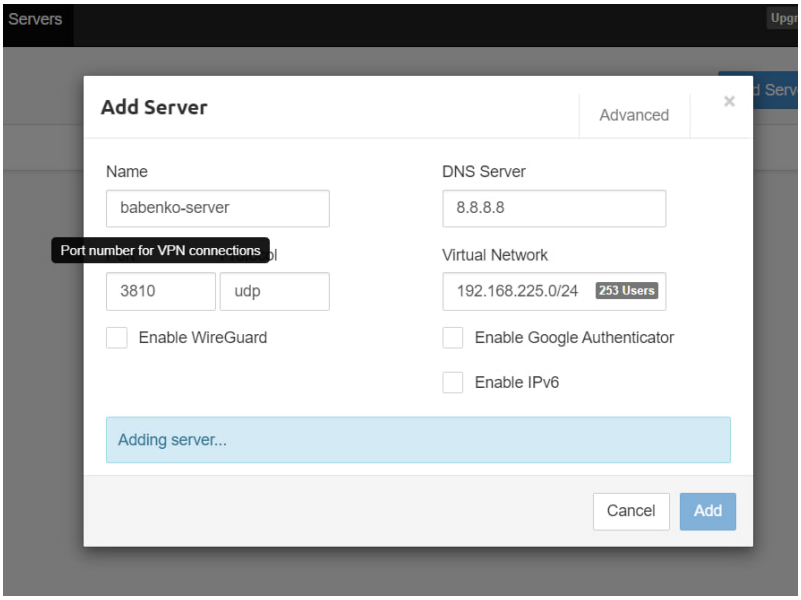

Рисунок 3.14 – додавання налаштувань серверу

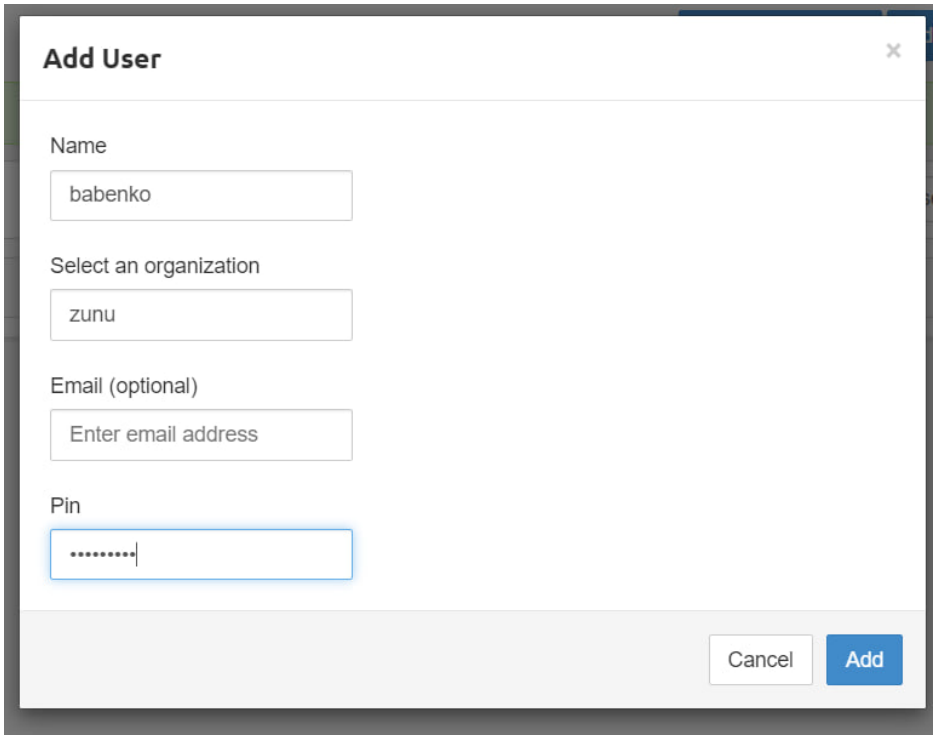

Рисунок 3.15 – додавання користувача

Кожен користувач отримує індивідуальні облікові дані, такі як ім'я користувача та пароль.

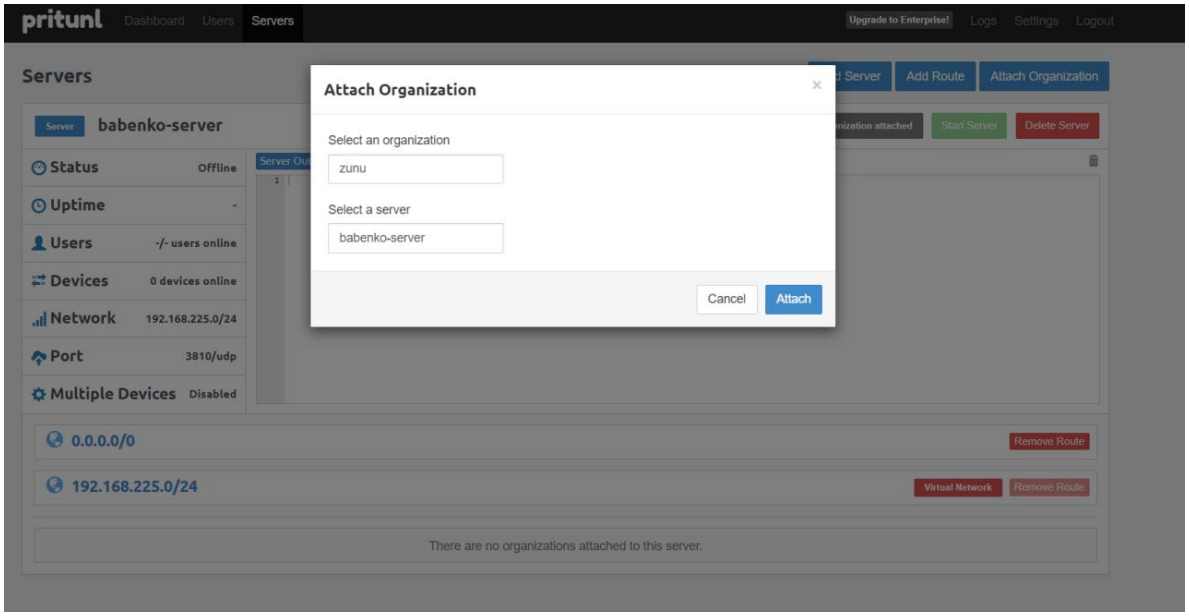

Рисунок 3.16 – додавання організації до серверу

Організація - це контейнер для VPN-підключень. Створіть організацію та призначте адміністраторів, які будуть керувати підключеннями до VPN.

Користувач може завантажити офіційни клієнт на свій Desktop під такі популярні ОС, як Windows, MacOS та такі дистрибутиви Linux, як Arch Linux, Ubuntu, Debian, Oracle Linux, Fedora. На рисунку 3.17. зображений офіційний сайт Pritunl VPN з завантаженням клієнту.

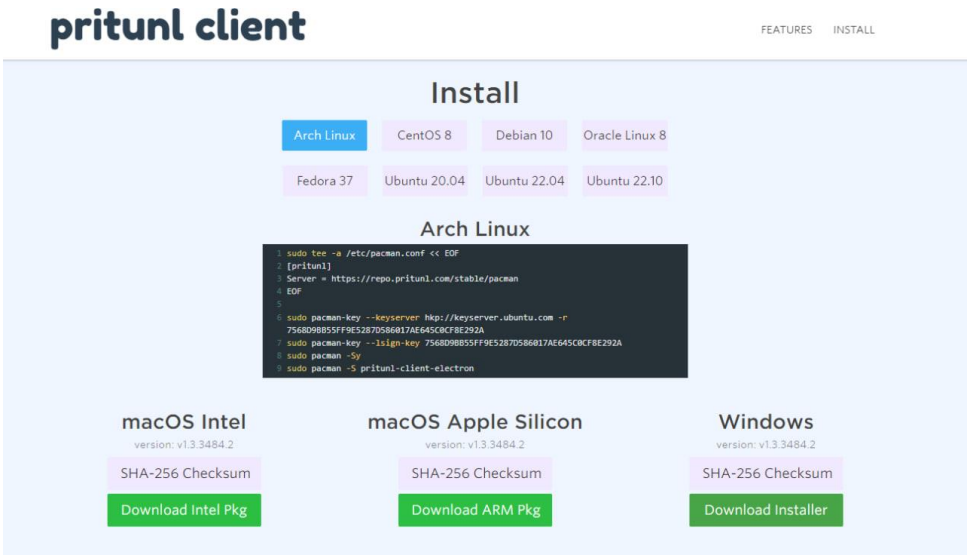

Рисунок 3.17 – офіційний сайт Pritunl VPN з завантаженням клієнту

Після завантаження користувачу достатньо імпортувати профіль користувача, який був створений на етапі створення користувача та використовувати VPN. Інтерфейс клієнту зображений на рисунку 3.18.

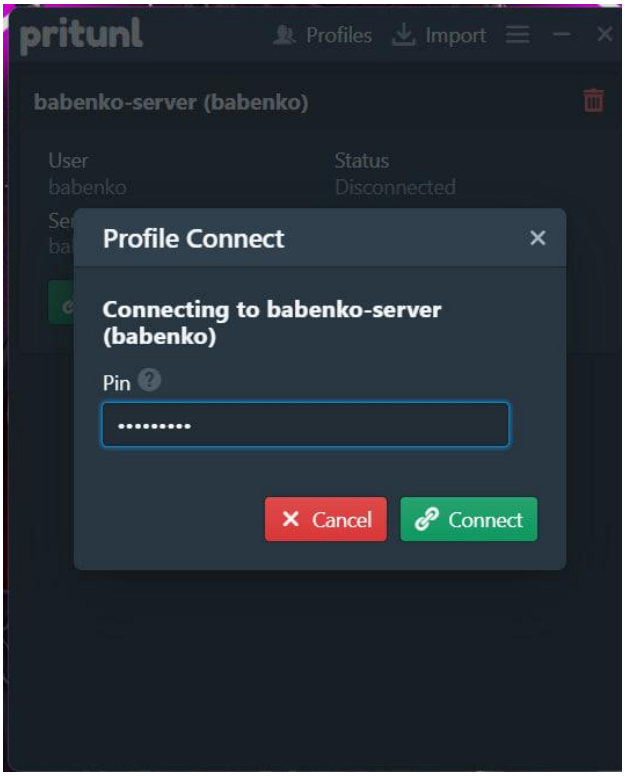

Рисунок 3.18 – інтерфейс клієнту

Демонструю роботу VPN рішення на рисунку 3.19

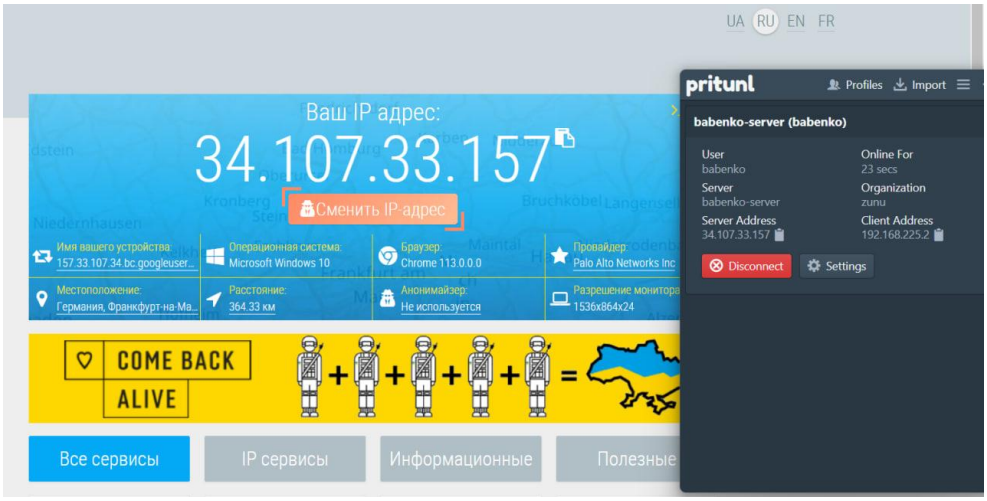

Рисунок 3.19 – робота VPN рішення

Моє VPN-рішення універсальне і його можна використовувати і без офіційного клієнту. Демонстрація роботи VPN на ОС Android без офіційного клієнту зображена на рисунку 3.20. та 3.21.

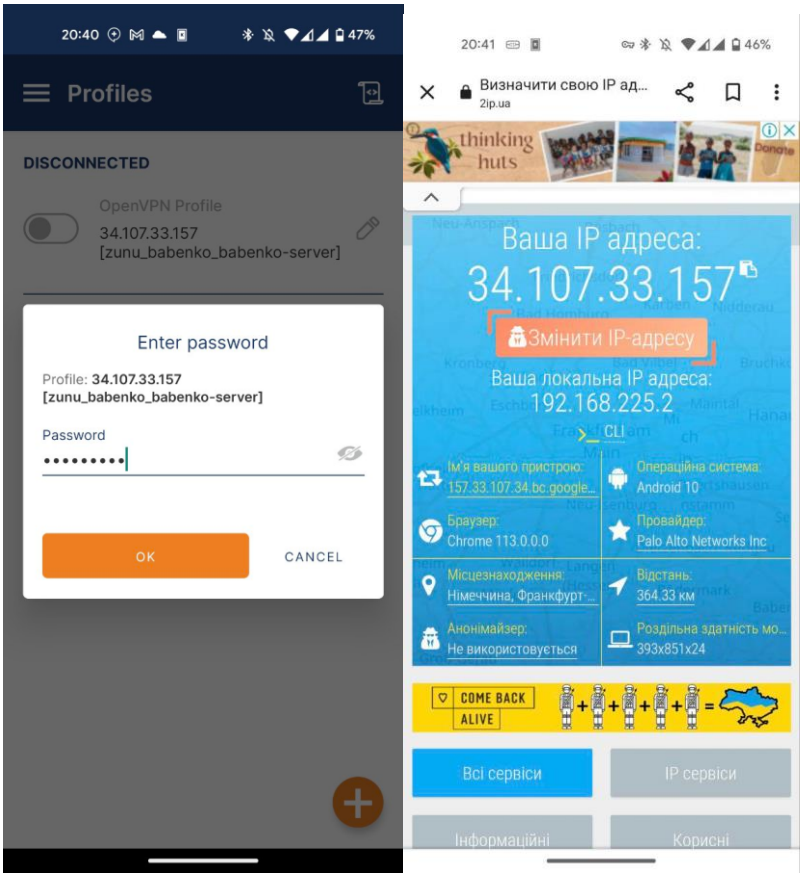

Рисунок 3.20 – робота VPN-рішення без офіційного клієнту на ОС Android

Перевірка IP-адреси - це процес визначення різних характеристик та властивостей, пов'язаних з певною IP-адресою. В даному прикладі, IP-адреса співпадає з зовнішньою IP-адресою сервера.

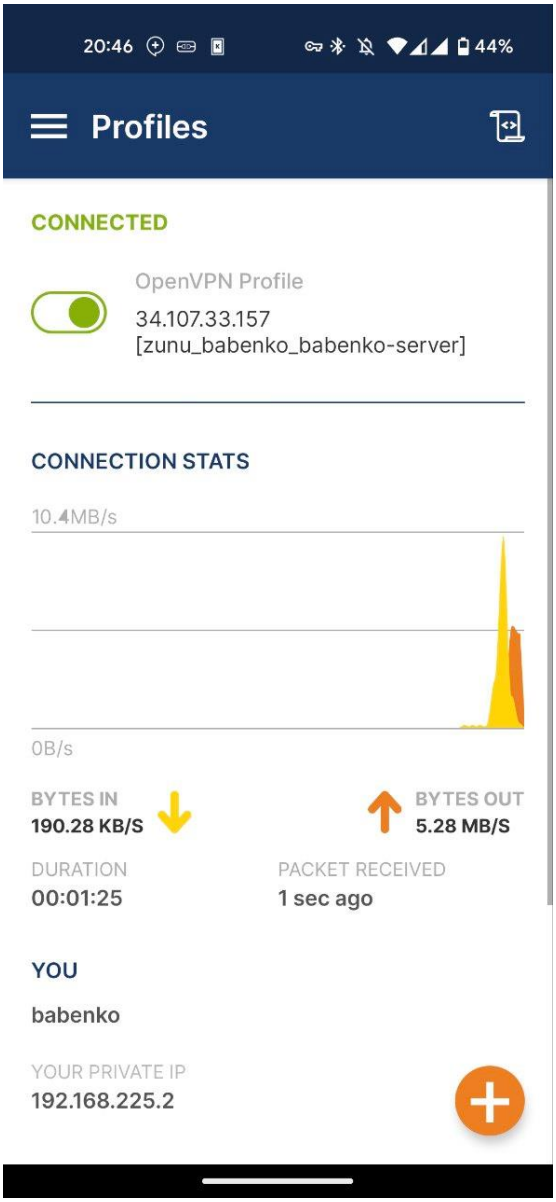

Рисунок 3.21 – робота VPN-рішення без офіційного клієнту на ОС Android

Оскільки Pritunl базується на стандартних протоколах OpenVPN та WireGuard, ви можете використовувати будь-який клієнтський програмний забезпечення, що підтримує ці протоколи. Такий підхід забезпечує гнучкість у виборі оптимального для ваших потреб VPN-клієнта.

4 ТЕХНІКО – ЕКОНОМІЧН**ИЙ РОЗДІЛ**

4.1 Визначення витрат на оплату праці та відрахувань у соціальні фонди

У даному розділі кваліфікаційної роботи проводиться економічне обґрунтування доцільності розробки програмного забезпечення мережевого обладнання. Зокрема, здійснюється розрахунок витрат на розробку даного програмного продукту, експлуатаційних витрат, ціни на споживання проектного рішення, визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення мережевого обладнання призначено для правильного налаштування обладнання.

Витрати на розробку і впровадження програмних засобів (К) включають [5]:

$$K = K_1 + K_2, \tag{4.1}$$

де K_1 – витрати на розробку програмних засобів, грн.;

K_2 – витрати на відлагодження і дослідну експлуатацію програми вирішення задачі на комп’ютері, грн.

Витрати на розробку програмних засобів включають:

- витрати на оплату праці розробників;
- витрати на відрахування у спеціальні державні фонди;
- витрати на покупні вироби;
- витрати на придбання спецобладнання для проведення експериментальних робіт;
- накладні витрати;
- інші витрати.

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування програмного засобу. Перелік необхідної програмної документації визначено відповідно до ДСТУ 3008-95 та включає:

- текст програми;
- керівництво користувача, яке включає інструкцію користувача;
- опис програми – відомості про логічну і фізичну модель, відомості щодо функціонування програми;
- пояснювальна записка – схема алгоритму, загальний опис алгоритму або функціонування програми, а також обґрунтовування прийнятих технічних і технічно-економічних рішень.

У таблиці 4.1 відображено інформацію щодо етапів технологічного процесу розробки проекту.

Таблиця 4.1 – Стадії розробки програмного засобу

№ п/п	Назва операції (стадії)	Виконавець, посада	Середній час виконання операції, год.
1	Підготовка, складання ТЗ	Менеджер проекту	3
2	Створення та налаштування VPN-рішення	Team lead, SRE (2)	2
3	Створення алгоритму системи	Архітектор (1)	3
4	Розробка системи	DevOps інженер(6), SRE (2)	140
5	Тестування продукту	Тестувальник (3)	10
6	Попереднє представлення для замовника	Менеджер проекту	4
7	Представлення реалізованої системи	Менеджер проекту	1
Всього		14	163

Витрати на оплату праці розробників проекту - це сума грошей, яка витрачається на компенсацію розробникам за їх час і зусилля, які вони

вкладають у проект. Ці витрати враховують різні аспекти, включаючи кількість виконавців, затрачений час та ставки оплати праці.

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij}\,, \tag{4.2}$$

де n_{ij} – чисельність розробників і-ої спеціальності j-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником і-ої спеціальності j-го тарифного розряду, год.;

C_{ij} – годинна ставка працівника і-ої спеціальності j-го тарифного розряду, грн.

За умов, якщо середньогодинну ставку розробника не відомо, її можна розрахувати за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}\,, \tag{4.3}$$

де C_{ij}^{0} – основна місячна заробітна плата розробника і-ої спеціальності j-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати;

$РЧ_{i}$ – місячний фонд робочого часу працівника і-ої спеціальності j-го розряду, год.

Таким чином, витрати на оплату праці розробників враховують не тільки час, який розробники витратили на проект, але й їх спеціальність та тарифний розряд, які впливають на їх годинну ставку.

Результати розрахунків записуємо у таблицю 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці при розробці проекту розглянутого типу

№ п/п	Посада виконавця	Час розробки, год.	Погодинна заробітна плата, грн	Витрати на оплату праці, грн
1	DevOps інженер (2)	25	365	18 250
2	SecOps інженер (2)	25	320	16 000
4	Team lead (1)	15	440	6 600
5	CloudOps інженер (1)	35	370	12 950
6	Проектний менеджер(1)	10	480	4 800
7	Тестувальник (3)	10	270	2 700
8	SRE (2)	20	320	12 800
9	Архітектор (1)	3	240	720
Всього		143		74 820

Оскільки виконавцем кваліфікаційної роботи є студент, то він є і розробником і тестувальником і дизайнером. Таким чином, оплата його праці – це стипендія без додаткових нарахувань, надбавок та премій. Стипендія студента становить 1960 грн. Зважачи на це, вартість проекту включає стипендію студента, а також витрати керівника на керівництво розробкою проекту та консультанта із написання техніко-економічного розділу. У таблицю 4.3 записуємо витрати на розробку програмного засобу у вигляді написання кваліфікаційної роботи.

Таблиця 4.3 – Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год (за весь рік)	Витрати на розробку, грн.
1	Керівник КР, викладач	8	450	3 600
2	Консультант з техніко-економічного розділу, старший викладач	1	450	450
3	Студент	150	31	4 650
Разом				8 700

У таблиці 4.4 наведено витрати на матеріали та комплектуючі вироби при виконанні кваліфікаційної роботи.

Загальна сума витрат на матеріальні ресурси B_M визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \quad (4.4)$$

де K_i - витрата і-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ - ціна за одиницю і-го типу матеріалу, грн.;

і - тип матеріального ресурсу;

n - кількість типів матеріальних ресурсів.

Таблиця 4.4 – Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1	Папір (формат А4)	уп	192,00	2	384,00	38,4	422,4
2	Ручка кулькова	шт	15,00	1	15,00	1,5	16,5
3	Олівець простий	шт	12,00	2	24,00	2,4	26,4
4	Диски CD-R	шт	25	2	50,00	5,0	55,00
5	Зошит, 24 арк.	шт	12	1	12,00	1,2	13,2
6	Тонер для принтера	уп	80	1	80,00	8,0	88,0
Разом							621,5

Накладні витрати включають три групи витрат: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за

встановленими відсотками від витрат на оплату праці. При цьому накладні витрати складають 70% від заробітної плати:

$$H = 74820 \cdot 0{,}70 = 52374 \text{ грн}$$

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$\text{І} = 8700 * 0{,}1 = 870 \text{ грн}$$

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_{Ф} + B_{КВ} + I \,.$$

(4.4)

$$K_1 = 8700 + 754{,}6 + 621{,}5 + 870 = 10946{,}1 \text{ грн}$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від},$$

(4.5)

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн./год (приймаємо 6);

$t_{від}$ – час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 25 днів, середній щоденний час роботи на комп'ютері – 1 год., вартість години роботи комп'ютера дорівнює 6 грн. Звідси витрати на відлагодження та експлуатацію розраховуємо:

$$K_2 = 6 \cdot 25 = 150 \text{ грн.}$$

Оскільки розробка проекту включає застосування засобів обчислювальної техніки, то розрахуємо витрати на електроенергію, а результати розрахунків занесемо у таблицю 4.5.

Загальну суму витрат на електроенергію розраховуємо за формулою:

$$B_E = \sum_{i=1}^{\mathbf{n}} P_i \cdot k_i \cdot T_i \cdot Ц \quad , \qquad (4.6)$$

де Pi – паспортна потужність і-го електрообладнання, кВт;

ki – коефіцієнт використання потужності і-го електрообладнання (приймається 0.7…0.9);

Ti – час роботи і-го устаткування за весь період розробки, год;

Ц – ціна електроенергії, грн / кВт*год;

і – тип електрообладнання;

n – кількість електрообладнання.

Таблиця 4.5 – Витрати на електроенергію

Назва устаткування	Паспортна потужність, кВт	Коефіцієнт використання потужності	Час роботи обладнання, год	Ціна Електроенергії грн, кВт/год	Сума, грн
Комп’ютер	0.5	0.7	163	1,68	95,85

До амортизації основних фондів включається сума амортизаційних відрахувань від вартості обладнання і приладів, що використовуються при розробці програмного продукту. Амортизаційні відрахування розраховуємо за формулою:

$$B_{AM} = \sum_{i=1}^{n} \frac{B_i \cdot H_i \cdot T_i}{100 \cdot T_{EФi}}, \qquad (4.7)$$

де Bi – вартість і-го устаткування, грн.;

Hi – річна норма амортизації і-го устаткування, %;

Ti – час роботи і-го устаткування за весь період розробки, год.;

ТЕФi – ефективний фонд часу роботи і-го устаткування за рік, год / рік;

і – тип устаткування;

n – кількість устаткування.

Усі розрахунки заносимо у таблицю 4.6.

Таблиця 4.6 – Амортизація основних фондів

Найменування устаткування	Вартість устатку-вання, грн	Річна норма амортизації,%	Ефективний фонд часу роботи обладнання, год / рік	Час роботи обладнання для розробки системи, год	Сума, грн.
Комп’ютер	20000	60	1700	163	150,6
Разом амортизаційні відрахування					1150,6

Обчислення дають нам змогу зрозуміти, скільки ресурсів нам потрібно для успішного виконання кожної з задач. На основі отриманих даних в результаті обчислень складаємо кошторис витрат на розробку програмного забезпечення і заносимо їх у таблицю 4.7.

Таблиця 4.7 – Кошторис витрат на розробку програмного забезпечення

№ п/п	Найменування витрат	Сума витрат, грн.
1	Витрати на оплату праці	8700
2	Витрати на електроенергію	95,85
3	Витрати на куповані вироби	621,5
4	Амортизаційні відрахування	1150,6
5	Інші витрати	870
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	150
Разом		11587,95

4.2 Розрахунок ціни проекту

Для оцінки економічної ефективності розробленого програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення. Для цього визначимо експлуатаційні витрати на робробку проекту.

Експлуатаційні одноразові витрати по розробці програмного забезпечення і його аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_П = E_{1П} + E_{2П}, \quad (4.8)$$

де $E_П$ – одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ –вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2П}$ – вартість роботи комп'ютера для розробки програмного продукту (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою:

$$B_{ЕП} = E_П * N_П, \quad (4.9)$$

де $N_П$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1П} = \sum_{l=1}^{n} n_i t_i c_i, \quad (4.8)$$

де і – категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i – кількість працівників і-ої категорії, осіб;

t_i – трудомісткість роботи співробітників і-ої категорії по підготовці даних, год.;

c_i – середньогогодинна ставка працівника і-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення (4.3).

Трудомісткість підготовки даних для проектного рішення складає 4 год., для аналога 2 год.

Таблиця 4.7 – Розрахунок витрат на реалізацію програмного забезпечення

№	Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати, грн.
	Проектне рішення		
1	4	8	32
	Аналог		
1	2	32	64

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t * S_{МГ},$$

де t – витрати машинного часу для реалізації проектного рішення (аналогу), год.;

$S_{МГ}$ – вартість однієї години роботи комп'ютера, грн./год.

Зважаючи на вищенаписане, проверемо розрахунки:

$$E_{2П} = 4 \cdot 6 = 24 \text{грн.},\ E_{2П_a} = 2 \cdot 6 = 12 \text{грн.};$$

$$E_{П} = 32 + 24 = 56 \text{ грн.},\ E_{П_a} = 64 + 12 = 78 \text{ грн.};$$

$$B_{ЕП} = 56 \cdot 260 = 14560 \text{ грн},$$

$$B_{ЕП_a} = 78 \cdot 260 = 20280 \text{ грн}$$

Ціна програмного продукту – це витрати на придбання і експлуатацію програмного засобу за весь період його служби:

$$Ц_П = К \cdot (1 + \frac{П_Р}{100}) + К_0 + К_к, \quad (4.9)$$

де К – кошторисна вартість;

$П_р$ – рентабельність;

$К_0$ – витрати на встановлення та освоєння програмного засобу на конкретному об'єкті, грн.;

$К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн. Зважаючи на вищеописане, розрахуємо ціну програмного засобу

$$Ц_{П_а} = 150950,35 \cdot (1 + 0,3) = 196235,4 \text{ грн}$$

$$Ц_П = 11587,95 \cdot (1 + 0,3) = 15064,3 \text{ грн}$$

У наступному підрозділі проведемо аналіз економічної ефективності розробки програмного продукту.

4.3 Визначення економічної ефективності розробки проекту

Для того, щоб побудувати таблицю показників економічної ефективності розробки програмного продукту, проведемо розрахунки необхідних показників.

Економічний ефект в сфері проектування проектного рішення розраховуємо за формулою:

$$Е_{ПР} = Ц_П - Ц_А, \quad (4.10)$$

$$Е_{ПР} = 196235,4 - 15064,3 = 181171,1 \text{ грн}$$

Річний економічний ефект від експлуатації програмного продукту:

$$E_{KC} = B_{EA} - B_{EП}, \tag{4.11}$$

$$E_{KC} = 20280 - 14560 = 5700 \text{ грн}$$

Дохід від розробки ПЗ у і-му періоді розраховуємо за формулою:

$$D_i = J_i(B_i - C_i), \tag{4.12}$$

де B_i – ціна продажу програмного продукту в і-му періоді;

C_i – собівартість програмного продукту (фактично дорівнює сумі витрат на розробку ПЗ);

J_i – кількість ПЗ.

$$D_i = 1 \cdot (15064{,}3 - 11587{,}95) = 3476{,}35 \text{ грн}$$

Економічний ефект - це вимір зміни в економічному стані або діяльності в результаті певного рішення, заходу або події. Економічний ефект полягає у відношенні результату від розробленого програмного продукту до затрачених ресурсів та розраховується за формулою:

$$E = \frac{D_i}{B_{заг}}. \tag{4.13}$$

$$E = \frac{3476{,}35}{11587{,}95} = 0{,}3.$$

Тоді термін окупності обчислюємо за такою формулою:

$$T = \frac{1}{E} \tag{4.14}$$

$$T = \frac{1}{0,3} = 3,3 \text{ р.}$$

Зважаючи на проведені розрахунки ефективності розробки програмного продукту, обчислимо сумарний ефект від розробки програмного продукту за формулою:

$$E = E_{ПР} + E_{КС}$$

$$E = 181171,1 + 5700 = 186871,1 \text{ грн}$$

Оцінюючи ефективність розробки програмного продукту, важливо розглянути не лише витрати на розробку, але й користь, яку він принесе. Показники економічної ефективності проектного рішення зображено на таблиці 4.8.

Таблиця 4.8 – Показники економічної ефективності проектного рішення

№	Найменування	Значення показників	
		Аналог	Новий варіант
1	Капітальні вкладення	74820	11585,95
2	Ціна придбання	196235,4	15064,3
3	Економічний ефект в сфері проектування	-	181171,1
4	Економічний ефект в сфері експлуатації	-	5700
5	Дохід від розробки		3476,35
6	Сумарний ефект	186871,1	
7	Термін окупності проекту	3,3	

Отже, у цьому розділі проведено розрахунок витрат на розробку програмного забезпечення. Показники, що характеризують витрати на розробку програмного продукту порівняно із показниками, які характеризують програмний продукт із аналогічним функціональним призначенням.

Розроблене програмне забезпечення має суттєві переваги у порівнянні із аналогами, зокрема простота використання, швидкість проведення розрахунків, стійкість до неоднорідних даних, зручність.

Згідно із проведеними розрахунками, що обґрунтовують економічну ефективність розробки програмного продукту, можна зробити висновок, що розроблене програмне забезпечення є суттєво дешевшим, оскільки у ролі розробника виступає студент. Отримано економічний ефект у розмірі 186871,1 грн., що свідчить про економічну доцільність розробки і впровадження комп'ютерної мережі з VPN сервером.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано практичну задачу розробка і впровадження комп'ютерної мережі з VPN сервером. При цьому отримано такі практичні результати:

1. Проведено дослідження та аналіз VPN-рішень, що ґрунтуються на використанні Terraform та Pritunl, і показано, що налаштування таких рішень включає створення віртуальної машини, встановлення потрібних компонентів, налаштування Terraform та доступу до Google Cloud API.

2. Описано основні етапи створення та налаштування VPN, включаючи створення Terraform-конфігураційних файлів для визначення архітектури проекту, запуск Terraform для створення VPN та налаштування Pritunl та створення користувачів з доступом до VPN.

3. Розглянуто альтернативне рішення для автоматизації налаштування Pritunl.

4. Розроблено VPN-рішення, що забезпечує безпечну комунікацію між клієнтськими машинами з використанням всіх переваг VPN. Результати дослідження демонструють ефективність використання Terraform та Pritunl для розгортання та налаштування VPN-інфраструктури в хмарному середовищі Google Cloud.

5. Результати цієї роботи можуть слугувати основою для подальшого вдосконалення та розширення функціональності VPN-інфраструктури з урахуванням специфічних вимог та потреб організацій.

6. Обгрунтовано техніко-економічні показники ефективності розробки проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Google Cloud Platform URL: https://cloud.google.com/docs/

2. Комп’ютерні мережі Частина 1 Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем»/ Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2020. 336 с. URL: https://ela.kpi.ua/handle/123456789/36615

3. Голь В.Д., Ірха М.С. Телекомунікаційні та інформаційні мережі: навчальний посібник. Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2021. 250 с. URL: https://ela.kpi.ua/bitstream/123456789/45409/1/TIM_navch_posib.pdf

4. The Cisco Learning Network URL: https://learningnetwork.cisco.com

5. Захист інформації в комп’ютерних системах : підручник для студ. спец. 123 «комп’ютерна інженерія» / уклад. О. М. Гапак, С. І. Балога; рец. : М. І. Глебена. – Ужгород: ПП "АУТДОР-ШАРК, 2021. – 184 с. URL: https://dspace.uzhnu.edu.ua/jspui/handle/lib/36506

6. Bonaventure Olivier. Computer Networking : Principles, Protocols and Practice. Saylor, 2022. 278 p.

7. Camisso Jamon. Making Servers Work: A Practical Guide to Linux System Administration. DigitalOcean, 2020. 281 p. URL: https://www.digitalocean.com/community/books/sysadmin-ebook-making-servers-work

8. ДСТУ 3008:2015 Національний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Введ. 01.07.2017. К.: ДП "УкрНДНЦ, 2016. 25 с

9. ДСТУ 8302:2015 Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Введ. 01.07.2016. К.: ДП «УкрНДНЦ», 2017. 16 с.

10. Методичні вказівки до випускних кваліфікаційних робіт освітнього рівня “Бакалавр” спеціальності “Комп’ютерна інженерія”/ О.М. Березький, Г.М. Мельник, Л.О.Дубчак, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2021. – 52 с.

11. Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп’ютерних систем”/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

12. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

13. Офіційна документація Google Cloud Platform https://cloud.google.com/docs/

14. Офіційна документація Terraform URL: https://www.terraform.io/docs/

15. Офіційна документація Ansible URL: https://docs.ansible.com/

16. Блог Google Cloud Platform URL:https://cloud.google.com/blog/

17. Блог HashiCorp, розробників Terraform URL: https://www.hashicorp.com/blog

18. Блог Red Hat, розробників Ansible URL: https://www.redhat.com/en/blog/products/ansible

19. Книга "Terraform: Up & Running: Writing Infrastructure as Code" автора Євгена Фрейзера URL: https://www.terraformupandrunning.com/, 2022. 12 р.

20. Книга «Ansible: Up & Running: Automating Configuration Management and Deployment the Easy Way» автора Лорі Маквейд URL: https://www.oreilly.com/library/view/ansible-up/9781491979792/, 2019. 3 р.

21. Курси на платформі Udemy з Terraform та Ansible: https://www.udemy.com/topic/terraform/

22. Книга «Terraform for Google Cloud Essential Guide: Learn How to Provision Infrastructure in Google Cloud Securely and Efficiently» автора Бернд Нордхаузен, 2021. 36 р.

23. Книга «Запускаємо Ansible. Простий спосіб автоматизації керування конфігураціями та розгортанням програми» авторів Хохштейн Л., Мозер Р. 2021. 42 р.

24. Shah, Gourav. Основи Ansible Playbook, 2021. 57 р.

25. Shah, Gourav. Основи Ansible Playbook: створення автоматичних шаблонів за допомогою Ansible Playbook для управління та оркестрування інфраструктурою з різними рівнями, 2022. 88 р.

26. Hall, Daniel. Управління конфігураціями Ansible, 2-ге видання: Використання потужності Ansible для ефективного управління вашою інфраструктурою, 2020. 13 р.

27. Hochstein, Lorin. Ansible: Початок роботи: Автоматизація управління конфігурацією та розгортання простим способом, 2018. 44 р.

28. Heap, Michael. Ansible: від новачка до професіонала, 2019. 33 р.

29. Bentley, Walter. Адміністрування OpenStack за допомогою Ansible 2, 2019. 23 р.

30. Mohaan, Madhurranjan, Raithatha, Ramesh. Packt Publishing, 2019. 39 р.

31. Geerling, Jeff. Ansible для DevOps, 2018. 12 р.

32. Deshpande, Anand, Kumar, Manish, Chaudhari, Vikram. Практичне використання штучного інтелекту на платформі Google Cloud: створення інтелектуальних додатків за допомогою TensorFlow, Cloud AutoML, BigQuery та Dialogflow, 2022. 2 р.

33. Deshpande, Anand, Kumar, Manish, Chaudhari, Vikram. Практичне використання штучного інтелекту на платформі Google Cloud: створення інтелектуальних додатків за допомогою TensorFlow, Cloud AutoML, BigQuery та Dialogflow, 2021. 11 р.

34. Hunter, Ted, Porter, Steven, Rajan, Legorie PS. Створення рішень для платформи Google Cloud: розробка масштабованих додатків з нуля та їх глобальне поширення майже будь-якою мовою, 2020. 9 р.

35. Rajan, Legorie PS. Кухарська книга платформи Google Cloud, 2021. 43 p.

36. Arif, Tariq M. Вступ до глибокого навчання для інженерів: використання Python і платформи Google Cloud, 2019. 11 p.

37. Cłapa, Konrad, Gerrard, Brian. Посібник з сертифікації професійного хмарного архітектора Google Cloud: створення міцних засад на платформі Google Cloud для отримання найбільш прибуткового сертифікату IT, 2-ге видання, 2020. 7 p.

38. Medina, Oscar; Schumann, Ethan. DevOps для SharePoint (з використанням Packer, Terraform, Ansible, і Vagrant), 2017. 1 p.

39. Brikman, Yevgeniy. Terraform: Початок роботи: написання інфраструктури кодом, 2018.

40. Turnbull, James. Книга про Terraform, 2016. 34 p.

41. Davis, Ashley. Створення мікросервісів за допомогою Docker, Kubernetes та Terraform: практичний посібник, 2020. 18 p.

42. Krief, Mikael. Кухарська книга по Terraform: ефективне визначення, запуск та управління інфраструктурою у вигляді коду на різних хмарних платформах, 2019. 4 p.

43. Winkler, Scott. Terraform в дії, 2021. 6 p.

44. Patil, Ankita; Soni, Mitesh. Автоматизація інфраструктури з Terraform, 2017. 7 p.

45. Brikman, Yevgeniy. Terraform: Початок роботи: написання інфраструктури кодом, 2016. 8 p.

46. Nordhausen, Bernd. Основний посібник по Terraform для Google Cloud: навчіться забезпечувати інфраструктуру в Google Cloud безпечно та ефективно, 2022. 31 p.

47. Beermann, Tim; Kastl, Johannes; Rost, Christian; Schifferdecker, Thorsten; Waldt, Eike. Terraform. 2015. 22 p.