Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра інформаційних систем та технологій
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента Гулько Іллі Ігоровича
(ПІБ)
академічної групи 123-16-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Комп'ютерна система управління персоналом підприємства «ModernDesign» з детальним опрацюванням побудови та налаштування трьох рівнів хмарової інфраструктури»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	ас. Бешта Л.В.			
розділів:				
апаратний розділ	доц. Ткаченко С.М.			
розрахунок мережі	ас. Панферова Я.В.			
економічний розділ	ст. викл. Яремчук І.О.			
охорона праці	доц. Іконніков М.Ю.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних систем
та технологій
_____ проф. Гнатушенко В. В.
(підпис) (прізвище,ініціали)

_____

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Гулько І. І. **академічної групи** 123-16-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»
за освітньо-професійною програмою 123 «Комп'ютерна інженерія»
(офіційна назва)

на тему «Комп'ютерна система управління персоналом підприємства «ModernDesign» з детальним опрацюванням побудови та налаштування трьох рівнів хмарової інфраструктури»

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № ____

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик інших науково-технічних джерел сформулювати завдання, конкретизувати предмет та мету роботи	18.05.2020
Технічні вимоги до комп'ютерної системи	На основі матеріалів виробничих практик інших науково-технічних джерел сформулювати технічні вимоги до розробки комп'ютерної системи	25.05.2020
Спеціальна частина	Розв'язати завдання з розробки комп'ютерної системи з опрацюванням побудови та налаштування інтерфейсу системи	01.06.2020
Економічна частина	Економічно обґрунтувати доцільність витрат на створення та дослідження системи	08.06.2020
Охорона праці	Розробити організаційно-технічні заходи, щодо реалізації правил безпеки при експлуатації системи	15.06.2020

Завдання видано _____ проф. Гнатушенко В.В
(підпис керівника) (прізвище, ініціали)

Дата видачі _27.01.2020_
Дата подання до екзаменаційної комісії _____
Прийнято до виконання _____ Гулько І. І.

РЕФЕРАТ

Пояснювальна записка: 91 с., 23 рис., 17 табл., 2 додатку, 10 джерел.

Об’єкт розробки: Комп’ютерна система управління персоналом з детальним опрацюванням побудови та налаштування трирівневої хмарної інфраструктури

Мета: створення та налаштування хмарної інфраструктури для розгортання трирівневого додатку.

Розробники створили додаток, який надає можливість малому і середньому бізнесу вести облік робочого часу і завдань співробітників. Для того, щоб цей додаток був безпечний і відмовостійкій для користувача, було вирішено використовувати трирівневий тип архітектури:

- Рівень front-end
- Рівень back-end
- Бази даних

Використовуючи цю архітектуру можна здійснити найкращі практики безпеки та відмовостійкості для всієї інфраструктури.

Результати перевірки у вигляді таблиць, графіків описані і наводяться у пояснювальній записці або додатках.

SaaS, PaaS, IaC, Terraform, CI/CD, AWS, EC2, VPC, S3, Secret Manager, Subnets, AWS Console, AWS Billing Dashboard, Three-tiered architecture, RDS, Availability Zones, AWS Region, on-premise

ЗМІСТ

ПЕРЕЛІК УМНОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Інфраструктура як послуга (IaaS) - це онлайн-сервіси, які надають високорівневі API для розмінування різних низькорівневих деталей базової мережевої інфраструктури, таких як фізичні обчислювальні ресурси, розташування, розподіл даних, масштабування, безпеку, резервне копіювання і т. д.

Платформа як послуга (PaaS) - модель хмарних обчислень, за якої споживач отримує доступ до використання платформ інформаційних технологій: операційних систем, систем управління базами даних, відповідного програмного забезпечення, інструментів для розробки та тестування розміщених у хмарних провайдерів.

Інфраструктура як код (IaC) - це спосіб забезпечення та управління обчислювальними та мережевими ресурсами, описуючи їх у вигляді програмного коду, на відміну від налаштування необхідного обладнання самостійно або використання інтерактивних інструментів. Прикладом IaC може бути Terraform.

Безперервна інтеграція (Continuous Integration) - практика розробки програмного забезпечення, яка полягає у впровадженні частих автоматизованих збірок проекту для швидкого виявлення та вирішення інтеграційних проблем.

Безперервна доставка (Безперервна доставка) - підхід у розробці програмного забезпечення, суть якого полягає в тому, що команди розробляють

програмне забезпечення за короткий проміжок часу, забезпечуючи надійний вихід версії в будь-який час.

Amazon Web Services (AWS) - дочірня компанією Amazon, яка надає платформу хмарних обчислень і API-інтерфейси на вимогу для приватних осіб, компаній і урядів на основі принципу «оплата за фактом».

Багаторівнева архітектура являє собою клієнт-серверну архітектуру, в якій функції уявлення, обробки додатків і управління даними фізично розділені. Найбільш поширеним використанням багаторівневої архітектури є трирівнева архітектура.

Угода про рівень обслуговування (SLA) - угода між постачальником послуг та користувачем про рівень послуги. Містить кількісні та якісні характеристики наданих послуг, такі як їх доступність, підтримка користувачів, час усунення несправностей тощо.

ВСТУП

Зростання IT сфери у всьому світі не можливо не помітити. В останні підприємство, яке не використовує можливості обчислювальної техніки не зможе досягти тих темпів зростання, які надають діджитал інструменти.

Паралельно з програмним рівнем IT сфери розвиваються і перетворюються обчислювальне обладнання. З даним фактом збільшується як потужності, так і витрати на їх утримання. Не кожній компанії вигідно мати свій дата центр, який потрібно підтримувати і забезпечувати достатній рівень безпеки.

Поступово, індустрія почала зсуватися від on-premise рішень в сторону централізованих обчислювальних платформ, в яких відповідальність за обладнання і безпеку несе власник дата центру. Така модель розподіленої відповідальності дозволяє знизити кількість людино-годин і фінансові витрати, які йдуть на утримання обчислювальної інфраструктури.

Також, разом з розвитком хмарних інструментів почали з'являтися окремі підходи до розробки інфраструктури: тепер інструменти автоматизації стали більш затребувані. Наприклад, компанія Hashicorp займається розробкою програмних рішень для спрощення завдань при розгортанні і оновленні великих інфраструктур (наприклад Terraform, Consul, Vault, Vagrant).

Ключовими гравцями на ринку провайдерів хмарних обчислювань є Amazon Web Services, Microsoft Azure, Google Cloud Platform.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Обґрунтування необхідності системи

Компанія «ModernDesign» має проблеми з надмірним навантаженням на працівників, хаосу серед поставлених задач та відсутній можливості слідкувати за прогресом їх виконання.

Майже кожен працівник використовує інструменти для організації якими вони вже користувались, тобто централізоване рішення відсутнє. Вони працюють над задачами несинхронно, та переважно стикаються з проблемою відсутності повного розуміння задачі. Керівництво може забувати про менш важливі задачі тим самим сповільнювати закінчення роботи над проектами. Наразі «ModernDesign» не мають чіткого, єдиного методу відстеження відпусток, задач і стану своїх робітників. Також самим робітникам часом важко налагодити ефективну взаємодію так як всі працюють віддалено, більшість робітників не мають контактних даних та не знають до кого можна звернутися з яким питанням.

Керівництвом компанії «ModernDesign» було прийнято рішення використання системи для контролю над працівниками. Над цим проектом працює команда розробників. Так як планується використовувати нову систему протягом багатьох років, замовник ставить акцент на якості інфраструктури. Також, є технічне завдання щодо доступності, відмовостійкості та безпеки хостової системи.

1.2 Аналіз ринку обчислюваних потужностей

При виборі нової системи планування загальнорганізаційних ресурсів одним з найбільш важливих факторів при прийнятті рішення буде вибір її розгортання на місці (on-premise) або в хмарі.

Хмарні системи зустрічаються частіше, ніж будь-коли раніше. Сьогодні майже кожен провайдер пропонує якусь форму розгортання в хмарі, а деякі взагалі відмовилися від своїх локальних пропозицій.

Найбільша різниця між цими двома системами полягає в тому, як вони розгорнуті. Хмарне програмне забезпечення розміщується на серверах постачальника і доступно через веб-браузер.

Локальне програмне забезпечення встановлюється локально, на власних комп'ютерах і серверах компанії.

Деякі постачальники також пропонують «гібридні» розгортання, в яких хмарне програмне забезпечення розміщується на приватних серверах організації.

Інша ключова відмінність між хмарними і локальними рішеннями полягає в тому, як вони оцінюються. Як правило, хмарне програмне забезпечення оплачується за передплатою або оплачується у міру надходження, з додатковими можливими зборами за підтримку, навчання і оновлення. Локальне програмне забезпечення зазвичай оцінюється по одноразової безстрокової ліцензійної плати. Є регулярні збори за підтримку, навчання і оновлення.

Таким чином, локальні системи зазвичай вважаються капітальними витратами (одна велика інвестиція). З іншого боку, хмарні системи зазвичай розглядаються як операційні витрати (додаткові накладні витрати, які організація буде продовжувати оплачувати).

1.2.1 Переваги хмарних рішень

1. Доступ в будь-якому місці і в будь-який час - Ви можете отримати доступ до своїх програм в будь-який час і в будь-якому місці через веб-браузер з будь-якого пристрою.
2. Фінансово доступно - хмара не вимагає авансових платежів, замість цього ви робите регулярні платежі, що робить його операційним. У той час як щомісячні витрати збільшуються з плином часу, послуги з обслуговування та підтримки включені, усуваючи необхідність в щорічних контрактах.
3. Передбачувані витрати - отримання вигоди від передбачуваних щомісячних платежів, які покривають ліцензії на програмне забезпечення, оновлення, підтримку і щоденне резервне копіювання (стосовно деяких сервісов).
4. Підтримка обладнання - оскільки хмарне програмне забезпечення розміщено для замовника, не потрібно турбуватися про підтримку програмного забезпечення мережевого обладнання або апаратного забезпечення, на якому воно знаходиться, про сумісність і оновлення піклується постачальник хмарних послуг.
5. Високий рівень безпеки - центри обробки даних застосовують заходи безпеки, які недоступні більшості компаній, тому дані часто більш безпечні в хмарі, ніж на сервері в офісах.
6. Швидке розгортання - програмне забезпечення на основі хмари розгортається через Інтернет за лічені години / дні, оскільки в порівнянні з локальними програмами, які необхідно встановити на фізичному сервері і на кожному ПК або ноутбуці.

7. Масштабованість - хмарні технології забезпечують більшу гнучкість, оскільки плата йде лише за те, що використовується, і легко масштабуються для задоволення попиту, наприклад, додаючи і зменшуючи ліцензії.
8. Зниження витрат на електроенергію - переходячи в хмару, вам більше не потрібно платити за обладнання локальних серверів або за підтримку їхнього середовища. Це значно зменшує суму за рахунками за електроенергію.

1.2.2 Переваги on-premise

1. Загальна вартість володіння - оскільки плата йде лише за свої власні ліцензії один раз, локальне рішення у деяких випадках може мати більш низьку сукупну вартість володіння, ніж хмарна система.
2. Повний контроль - дані, апаратні і програмні платформи - все це знаходится під безпосереднім контролем. Замовник вибираєте конфігурацію, поновлення і системні зміни.
3. Час безвідмовної роботи - з локальними системами замовник не покладаєтеся на підключення до Інтернету або зовнішні чинники для доступу до свого програмного забезпечення.

1.2.3 Загальне порівняння провайдерів

Для інфраструктури як послуги (IaaS) і платформи як послуги (PaaS), Amazon Web Services (AWS), Microsoft Azure і Google Cloud Platform (GCP) займають лідируючі позиції серед багатьох хмарних компаній.

AWS особливо домінує. Згідно зі звітом Synergy Research Group за 2020 рік, «зростання Amazon продовжує тісно відображати загальне зростання ринку, тому він зберіг свою 33-відсоткову частку на світовому ринку. Друге місце займає Microsoft, який знову виріс швидше за ринок, а її частка на ринку збільшилася майже на 3 процентні пункти за останні чотири квартали, досягнувши 18%.»

Тим часом, Microsoft особливо сильний в SaaS, в той час як Google Cloud, з його силою в штучному інтелекті, позиціонується для агресивного зростання у міру зростання ринку II - і відомий тим, що пропонує знижки.

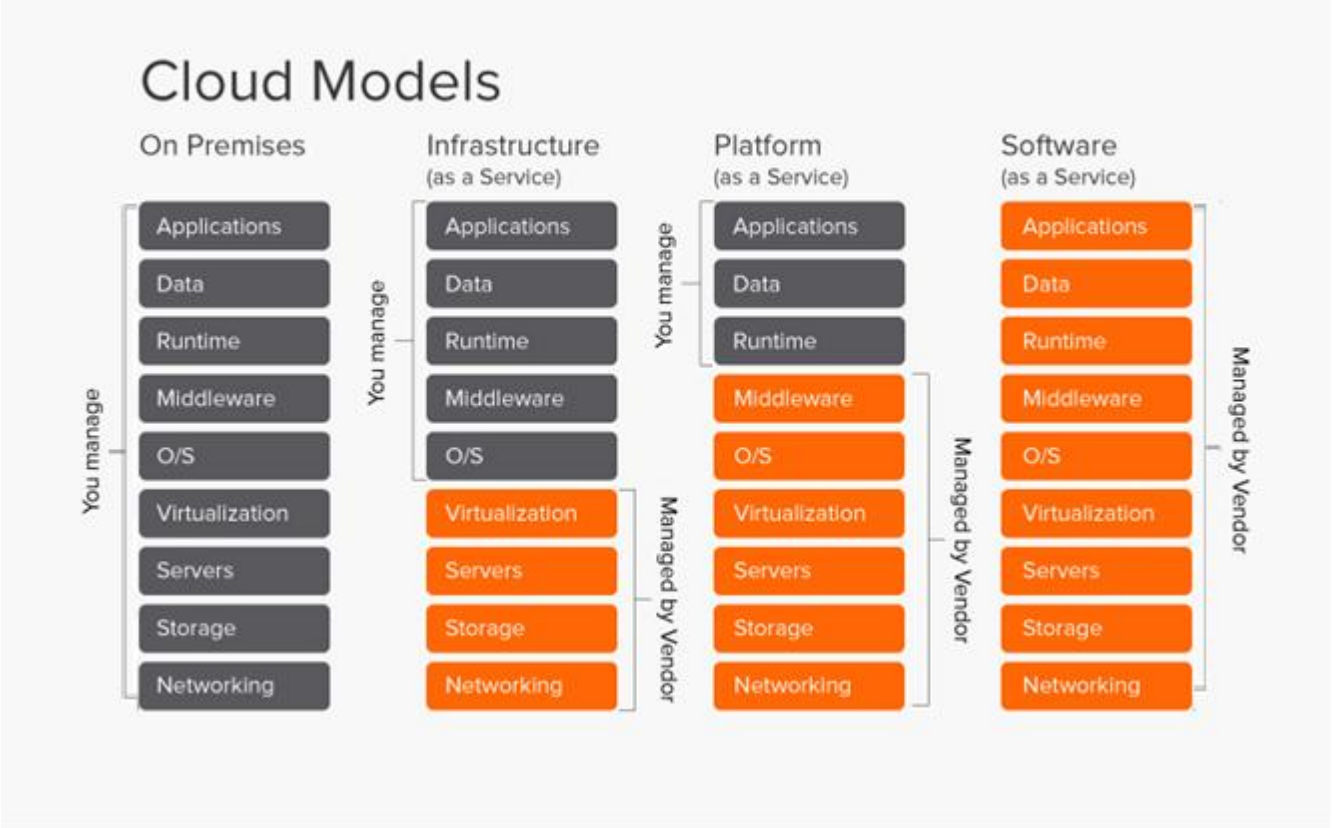

Рисунок 1.1 Порівняння моделі відповідальності у різних сервісах

Amazon Web Services - завдяки великому набору інструментів, який продовжує рости в геометричній прогресії, можливості Amazon не мають собі

рівних. Проте, його структура витрат може збивати з пантелику, а його особлива орієнтація на загальнодоступну а не на приватну хмару означає, що взаємодія з локальним центром обробки даних не є головним пріоритетом AWS.

Microsoft Azure - близький конкурент AWS з хмарної інфраструктурою. Для корпоративних клієнтів Azure має виняткові можливості - лише деякі компанії мають корпоративний досвід (і підтримку Windows) як Microsoft. Azure має фокус на локальні центри обробки даних, і платформа Azure гарно працює для взаємодії з центрами обробки даних; Гібридна хмара – це фокус Microsoft.

Google Cloud - добре фінансований конкурент, Google пізніше вийшов на хмарний ринок і не має корпоративного фокуса, який допомагає залучати корпоративних клієнтів. Але нові технічні знання і провідні в галузі інструменти для глибокого навчання і штучного інтелекту, машинного навчання та аналізу даних - це істотні переваги.

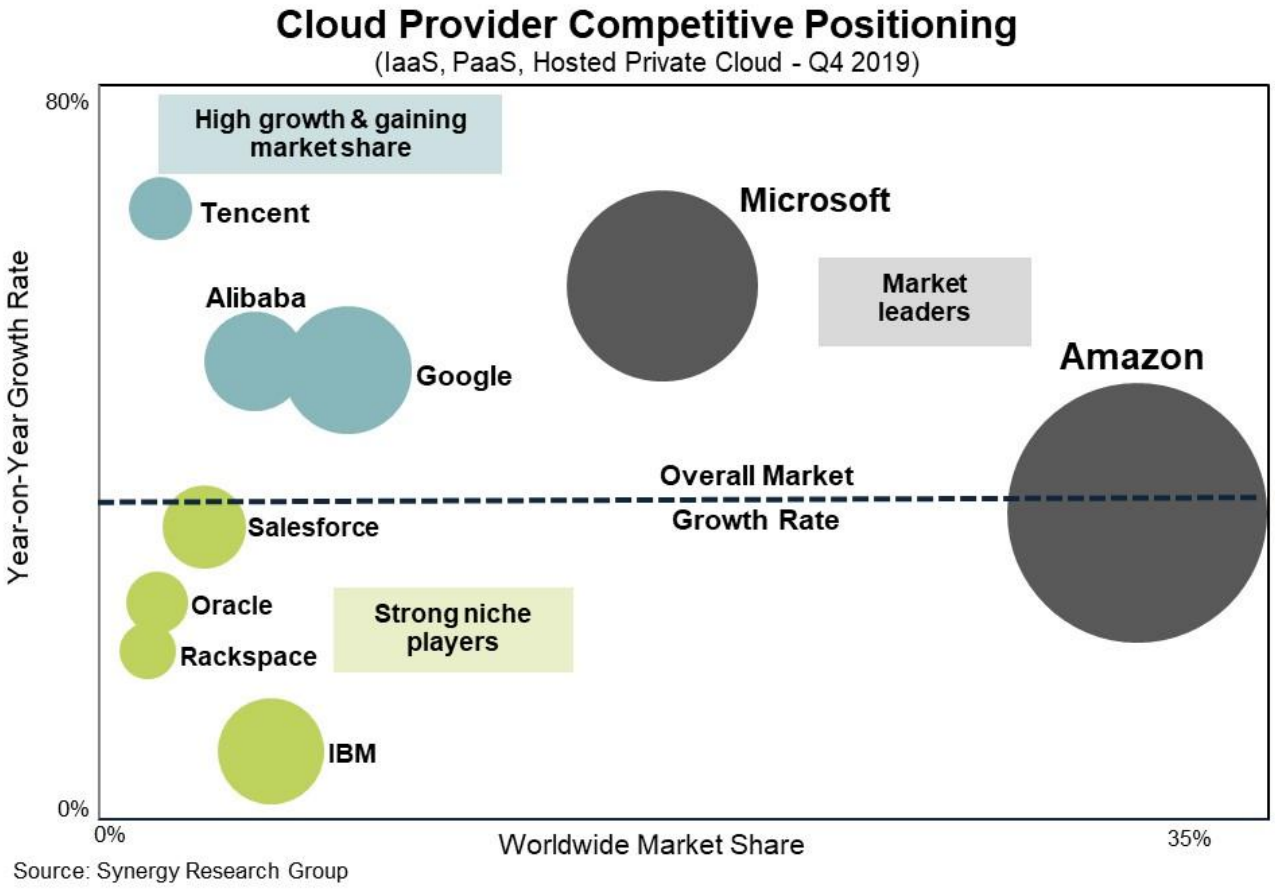

Рисунок 1.2 Порівняння хмарних провайдерів

Після обговорення з замовником було прийнято рішення, що найоптимальнішим вибором в даний момент є Amazon Web Services.

1.3 Забезпечення високої доступності додатку

Високодоступні системи надійні в тому сенсі, що вони продовжують працювати навіть в разі відмови критичних компонентів. Вони також є відмовостійкими, що означає, що вони можуть коректно обробляти збої без

переривання обслуговування або втрати даних і легко відновлюватися після такого збою. Висока доступність зазвичай вимірюється у відсотках від часу безвідмовної роботи. Кількість «дев'яток» зазвичай використовується для вказівки ступеня високої доступності. Наприклад, «чотири дев'ятки» вказують на те, що система працює до 99,99% часу, тобто вона не працює всього 52,6 хвилини протягом всього року.

Архітектура додатку полягає на 3 основних ресурса AWS: EC2, RDS, S3. За замовчуванням, Amazon Web Services відносно кожного сервісу має свій SLA та політику виплати кредитів на обслуговування (Service Credits):

- EC2 – 99.9% тривалість роботи на місяць

Таблиця 1.1 – політика AWS наразі відмови сервісу щодо EC2

Щомісячний відсоток тривалості роботи	**Відсоток кредитів на обслуговування**
Менше 99.99% але дорівнює або більше 99.0%	10%
Менше 99.0% але дорівнює або більше 95.0%	30%
Менше 95.0%	100%

- RDS – 99.95% тривалість роботи на місяць

Таблиця 1.2 – політика AWS наразі відмови сервісу щодо RDS

Щомісячний відсоток тривалості роботи	**Відсоток кредитів на обслуговування**
Менше 99.95% але дорівнює або більше 99.0%	10%
Менше 99.0% але дорівнює або більше 95.0%	25%
Менше 95.0%	100%

- S3 – 99.9% тривалість роботи на місяць

Таблиця 1.3 – політика AWS наразі відмови сервісу щодо S3

Щомісячний відсоток тривалості роботи	**Відсоток кредитів на обслуговування**
Менше 99.9% але дорівнює або більше 99.0%	10%
Менше 99.0% але дорівнює або більше 95.0%	25%
Less than 95.0%	100%

Наступні елементи допомагають вам впроваджувати системи високої доступності:

Резервування - забезпечення того, щоб критичні компоненти системи мали інший ідентичний компонент з тими ж даними, які можуть бути замінені у разі збою.

Моніторинг - виявлення проблем у виробничих системах, які можуть порушити або погіршити обслуговування.

Аварійне перемикання - можливість перемикання з активного системного компонента на резервний компонент в разі збою, неминучого збою, зниження продуктивності або функціональності.

Failback - можливість перемикання з надлишкового компонента на основний активний компонент після його відновлення після збою.

Для досягнення данних параметрів ми будемо використовувати такі технології і сервіси Amazon Web Services:

Обчислення - Amazon EC2, дозволяє надавати обчислювальні ресурси, надає функції високої доступності, такі як балансування навантаження, автоматичне масштабування і ініціалізація в зонах доступності Amazon (AZ), що представляють ізольовані частини баз даних дата центрів Amazon.

База даних - Amazon RDS та інші керовані бази даних SQL надають опції для автоматичного розгортання баз даних з резервної реплікою в різних сервісах AZ.

Сховище даних - сервіси зберігання Amazon, такі як S3, EFS і EBS, забезпечують вбудовані опції високий ой доступності. S3 і EFS автоматично зберігають дані в різних AZ, а EBS дозволяє розгортати моментальні знімки в різних AZ.

Пружне балансування навантаження - запуск кількох примірників EC2 і розподіляти трафік між ними завдяки

Автоматичне масштабування – при збільшення навантаження чи збію на існуючому сервері динамічно додати більше примірників EC2 Instances.

2 РОЗРОБКА ЗАГАЛЬНОЇ АРХІТЕКТУРИ

2.1 Сервіси AWS, використовувані для побудови інфраструктури

Для основних обчислювальних потужностей використовується сервіс EC2 (Elastic Compute Cloud) - це веб-сервіс, який забезпечує безпечну обчислювальну ємність із змінним розміром в хмарі. Він призначений для спрощення хмарних обчислень в масштабах мережі для розробників.

Розробники визначили основною базою даних для додатка PostgreSQL. Для того, щоб розмістити цю базу даних в AWS можна використовувати RDS (Relational Database Service). Це спрощує настройку, експлуатацію і масштабування реляційної бази даних в хмарі. RDS забезпечує економічну і змінну ємність при автоматизації трудомістких адміністративних завдань, таких як підготовка обладнання, настройка бази даних, виправлення і резервне копіювання.

Розробники запросили окремий сервіс для зберігання даних різного типу (документи, фото і т. д.). В AWS є сервіс зберігання об'єктів - S3 (Simple Storage Service). Він пропонує масштабованість, доступність даних, безпеку і продуктивність зі старту.

Також, для логування всіх рівнів додатки буде використовуватися Amazon CloudWatch. Він надає дані і корисну інформацію для моніторингу додатків, реагування на загальносистемні зміни продуктивності, оптимізації використання ресурсів та отримання уявлення про експлуатаційну працездатність.

2.2 Налаштування мережі та роутінгу

Всі віртуальні машини будуть запущені в ізольованій VPC (Virtual Private Cloud). В VPC інстанси будуть розподілені по підмережам, що дасть можливість налаштувати роутинг між усіма компонентами. Це дозволить ізолювати різні типи компонентів один від одного, що в свою чергу збільшить безпеку чутливих даних та знизить ризик кібератак типу SSRF (підробка запиту на стороні сервера).

Для додаткової безпеки в мережі будуть використовуватися Security Groups і Access Control List.

Security Group діє як віртуальний брандмауер для контролю вхідного і вихідного трафіку віртуальної машини. При запуску інстанси можна призначити до п'яти груп безпеки. Варто відзначити, що групи безпеки діють на рівні віртуальної машини, а не на рівні підмережі.

Список контролю доступу до мережі (ACL) - це додатковий рівень безпеки для VPC, який виступає в якості брандмауера для управління трафіком в одній або декількох підмережах. Можна налаштовувати ACL з правилами, аналогічними правилам в Security Groups.

З міркувань безпеки, за замовчуванням з Virtual Private Cloud немає доступу в Інтернет. Для того, щоб додаток був доступний глобально потрібно додати Internet Gateway до VPC. Він служить як шлях за замовчуванням для

маршруту через Інтернет, але і ще виконує перетворення мережевих адрес (NAT) для інстансів, яким були призначені публічні адреси.

Щоб трафік був гнучким і хости могли вийти в Інтернет, потрібно налаштувати кілька додаткових таблиць маршрутів. Вони містять набір правил, які використовуються для визначення напрямку мережевого трафіку.

2.3 Безпека кореневого аккаунту

Забезпечити безпеку AWS аккаунту можна через сервіс Управління ідентифікацією і доступом (Identity and Access Management).

В першу чергу потрібно убезпечити кореневої аккаунт, так як через його ключі доступу можна здійснювати будь-які операції. Також варто налаштувати багатофакторну аутентифікацію (MFA) з фізичного ключа або мобільного пристрою. Це проста рекомендація, яка додає додатковий рівень захисту поверх імені користувача та пароля.

Створюючи окремих користувачів IAM для людей, які отримують доступ до окремої облікового запису, можна надати кожному користувачу IAM унікальний набір доступу. При необхідності в будь-який час можна змінити дозволу користувача IAM, що досить важко зробити з кореневим аккаунтом.

Якщо користувачів відносно багато, зручніше створювати групи доступу, яким потрібно надати різні робочі функції. Всі користувачі IAM призначені в певну групу і успадковують її дозволи. Таким чином зручніше вносити зміни всієї групі юзерів, а не індивідуальним користувачам.

Для делегування повноважень визначених повноважень можна використовувати окремі ролі. Роль - це сутність, яка має власний набір дозволів, але не є користувачем або групою. У ролей також немає власного постійного набору облікових даних, як у користувачів IAM, тому для присвоювання рівня доступу додатку, якій працює на інстансі Amazon EC2 використовуються ролі. У разі Amazon EC2 IAM динамічно надає тимчасові облікові дані для EC2, і ці облікові дані автоматично змінюються для додатку.

2.4 Ведення звітності по фінансових витратах

Згодом використання ресурсів AWS стає досить складно відстежити і передбачити самому, які витрати чекають компанію через якийсь час. Для ведення фінансового обліку в Amazon існує кілька рішень: AWS Cost Explorer (Рисунок 2.1) і AWS Budgets (Рисунок 2.2).

AWS Cost Explorer дозволяє візуалізувати і управляти своїми витратами і використанням AWS з плином часу.

Рисунок 2.1 Зразок інтерфейсу Cost Explorer

AWS Budgets дає можливість встановлювати власні ліміти на бюджетну суму. Budgets може прогнозувати витрати. На прогнозовані величини також можна ставити ліміти.

Ліміти можна виставляти не тільки на фінансові витрати, але також і на показники використання ресурсів, наприклад, кількість годин роботи EC2

Instance.

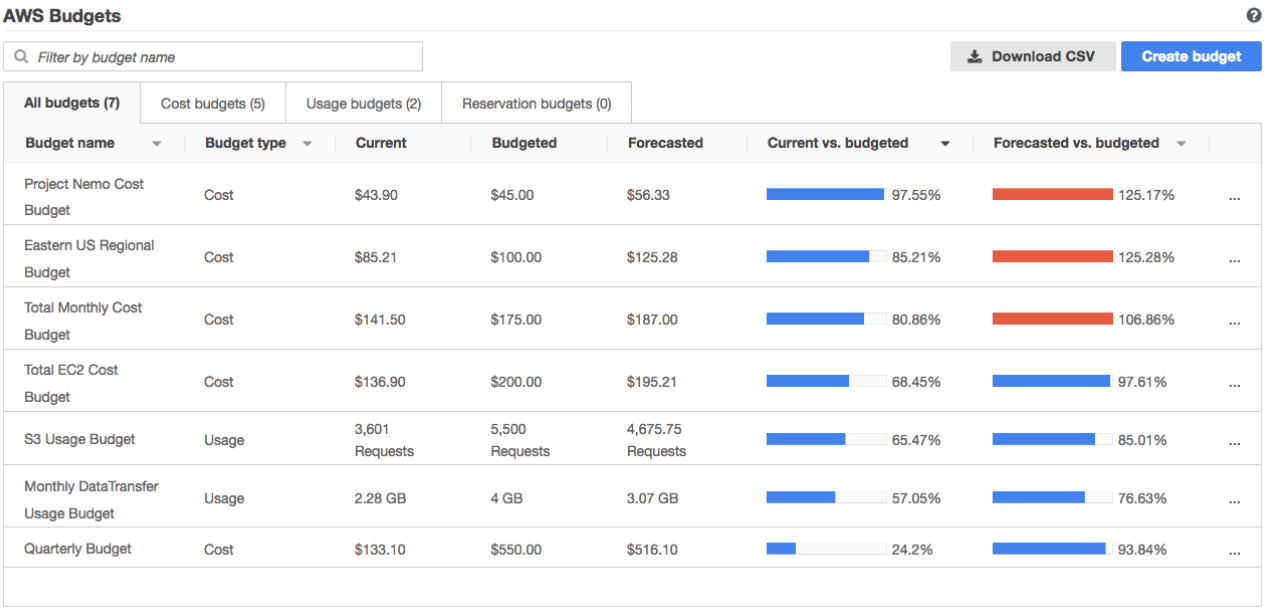

Рисунок 2.2 Зразок інтерфейсу Budgets

3 АВТОМАТИЗАЦІЯ ВИРОБНИЦТВА АРТЕФАКТІВ

Для автоматизації і прискорення запуску додатка на віртуальних машинах в подальшому, потрібно визначити артефакти цих додатків. Артефакт - це один з багатьох видів відчутних побічних продуктів, що створюються при розробці програмного забезпечення. Деякі артефакти допомагають описати функції, архітектуру і дизайн програмного забезпечення. Інші артефакти пов'язані з самим процесом розробки, такі як плани проектів, бізнес-кейси і оцінки ризиків. Термін артефакт в зв'язку з розробкою програмного забезпечення в значній мірі пов'язаний з конкретними методами або процесами розробки.

Для автоматизації процесу створення артефактів буде використовуватися Jenkins. Jenkins - це безкоштовний сервер автоматизації з відкритим вихідним кодом. Це допомагає автоматизувати частини розробки програмного забезпечення, пов'язані зі складанням, тестуванням і розгортанням, полегшуючи безперервну інтеграцію і безперервне постачання.

3.1 Налаштування Jenkins

Мінімальні апаратні вимоги запуска Jenkins:

256 МБ ОЗУ

1 ГБ дискового простору

Рекомендована конфігурація обладнання:

1 ГБ+ ОЗУ

50 ГБ+ дискового простору

Сучасні версії Jenkins мають наступні вимоги Java:

Середовища виконання Java 8, підтримуються як 32-бітові, так і 64-бітові версії.

Обсяг пам'яті, в якому потребує Jenkins, багато в чому залежить від багатьох факторів, тому виділена для нього оперативна пам'ять може варіюватися від 200 МБ для невеликої установки до 70+ ГБ для одного і великого майстра Jenkins.

Кожне з'єднання вузла збірки займе 2-3 потоку, що становить близько 2 МБ або більше пам'яті. Також потрібно буде враховувати навантаження на процесор для Jenkins, якщо є багато користувачів, які будуть звертатися до призначеного для користувача інтерфейсу Jenkins.

У нашій ситуації до сервера Master Jenkins будуть підключатися три розробника, і запускатися буде близько двох Freestyle Job. Також, всі великі дані (вихідний код і артифакти) будуть зберігатися на Simple Storage Service, тому обсяг диска для нас не настільки важливий.

Як основний Amazon Machine Image обраний загальнодоступний AMI на основі операційної системи LTS Ubuntu 18.04.

Було вирішено використовувати EC2 інстанс t2.micro, який має 1 обчислюване ядро та 1 ГБ ОЗУ за замовчуванням. Як постійна пам’ять використовується EBS Storage об’ємом 8 ГБ. (Детальний опис типів EC2 у 4.3)

Майстер Jenkins обслуговуватиме HTTP-запити і зберігати всю важливу інформацію для додатка Jenkins в своїй папці $JENKINS_HOME (конфігурації, історії складання та плагіни).

Мережа для інфраструктури розробки досить нескладна, так як ми не очікуємо великого навантаження. Як основну Virtual Private Cloud вибираємо мережу 172.16.0.0/16.

Віртуальну машину з Jenkins Master піднімаємо в підмережі 172.16.100.0/24. Також, для можливих майбутніх цілей тестування програмного забезпечення створюємо другу підмережу 172.16.100.0/24.

Також використовуємо Internet Gateway для доступу в інтернет зсередини VPC. В Security Groups налаштовуємо рівень доступу тільки для розробників. (Детальніше про налаштування VPC у 4.2)

В результаті маємо VPC в AWS Console (Рисунок 3.1, 3.2)

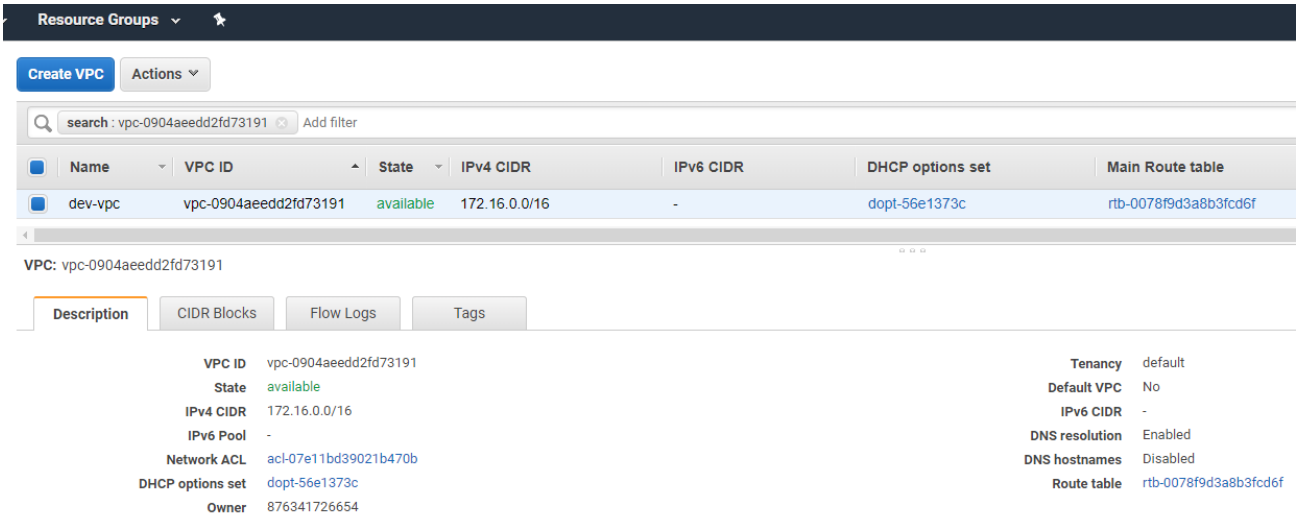

Рисунок 3.1 Virtual Private Cloud

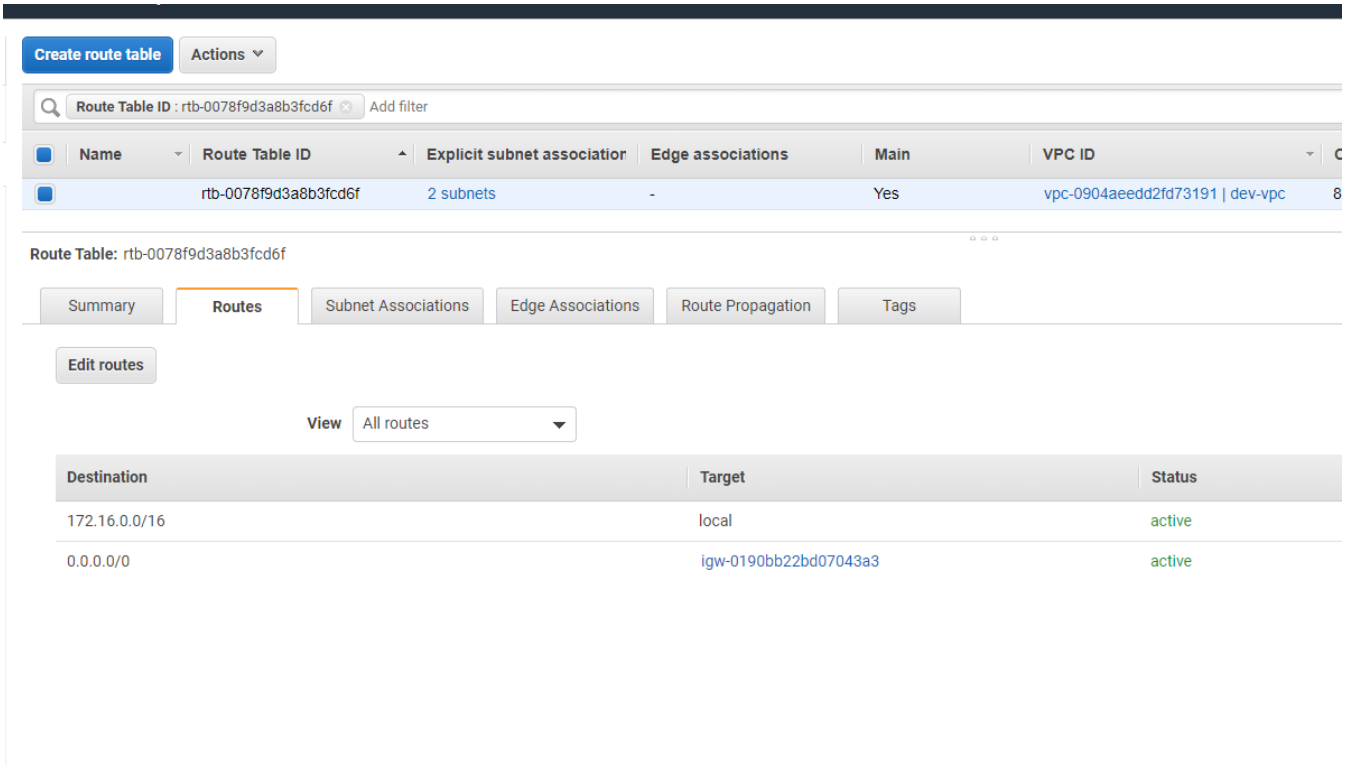

Рисунок 3.2 Route Table

Після мануального запуску t2.micro інстанси через AWS Console, маємо EC2 віртуальну машину з публічним IP адресою, через який ми можемо підключитися по SSH і провести первинну настройку Jenkins Server.

Secure Shell (SSH) - це криптографічний мережевий протокол для безпечної роботи мережевих служб в незахищеній мережі. Типові програми включають віддалену командний рядок, вхід в систему і віддалене виконання команд, але будь-який мережевий сервіс може бути захищений за допомогою SSH.

SSH забезпечує безпечний канал через незахищену мережу, використовуючи архітектуру клієнт-сервер, поєднуючи клієнтську програму SSH з сервером SSH. Специфікація протоколу розрізняє дві основні версії, звані SSH-1 і SSH-2. Стандартний порт TCP для SSH - 22.

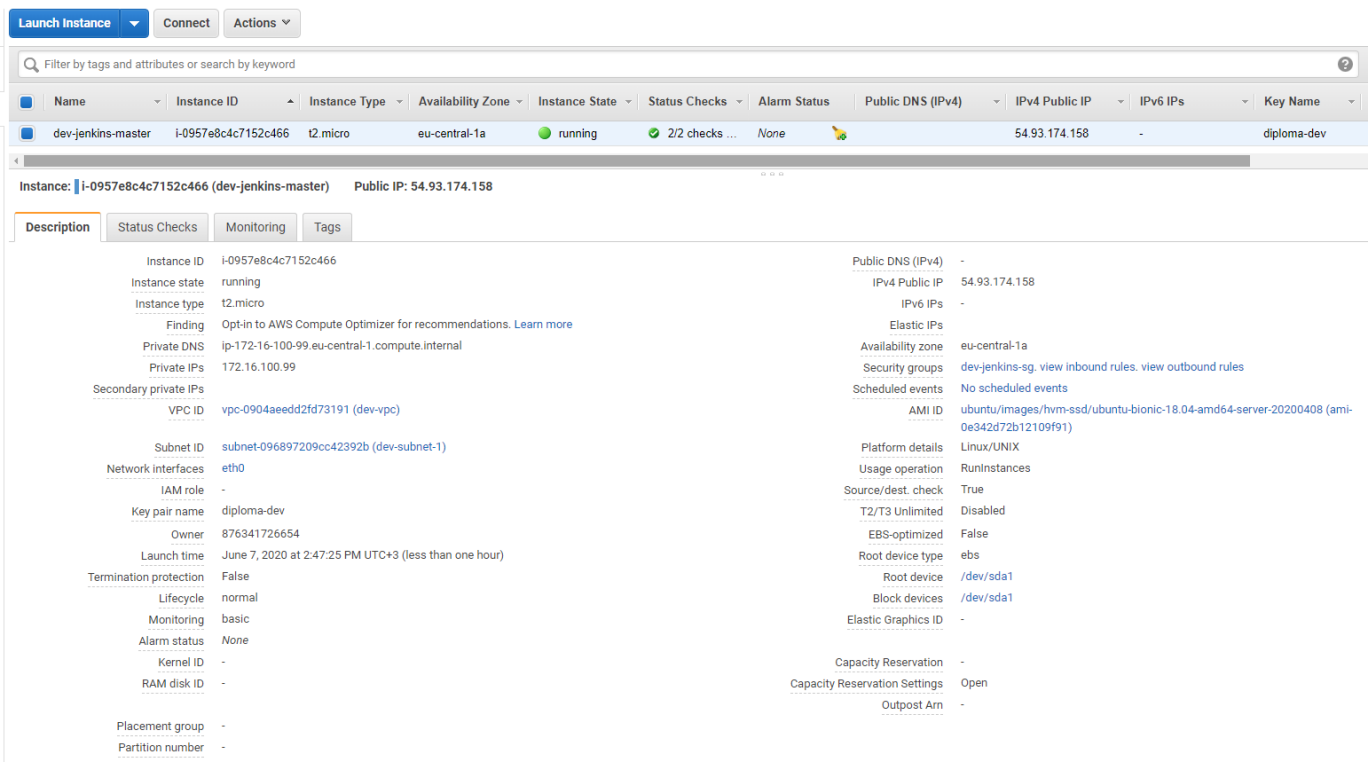

Рисунок 3.3 EC2 Jenkins Master

Спочатку треба встановити потрібну версію Java 8 для запуску Jenkins. Це можна зробити використовуючи APT. Далі ми будемо використовувати його для автоматичного налаштування серверів.

Advanced Package Tool є вільним програмним інтерфейсом користувача, який працює з основними бібліотеками для управління установкою і видаленням програмного забезпечення в Debian, Ubuntu і пов'язаних дистрибутивах Linux. APT спрощує процес управління програмним забезпеченням в Unix-подібних комп'ютерних системах за рахунок автоматизації пошуку, настройки і установки пакетів програмного забезпечення або з попередньо скомпільовані файлів, або шляхом компіляції вихідного коду.

Встановуємо openjdk-8-jdk:

$ sudo apt-get update

```
$ sudo apt-get install openjdk-8-jre-headless
```

Перевіряємо встановлену версію Java:

```
$ java -version
openjdk version "1.8.0_125"
OpenJDK Runtime Environment (build 1.8.0_125-8u125-b14-2ubuntu0.16.04.2-b14)
OpenJDK 64-Bit Server VM (build 25.125-b14, mixed mode)
```

Встановлюємо посилання на LTS версію репозиторію:

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add –
$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
/etc/apt/sources.list.d/jenkins.list'
```

Оновлюємо список посилань та встановлюємо LTS Jenkins:

```
$ sudo apt-get update
$ sudo apt-get install Jenkins
```

Після цього маємо доступ до веб-інтерфейсу Jenkins Server на порту 8080. (Рисунок 3.4)

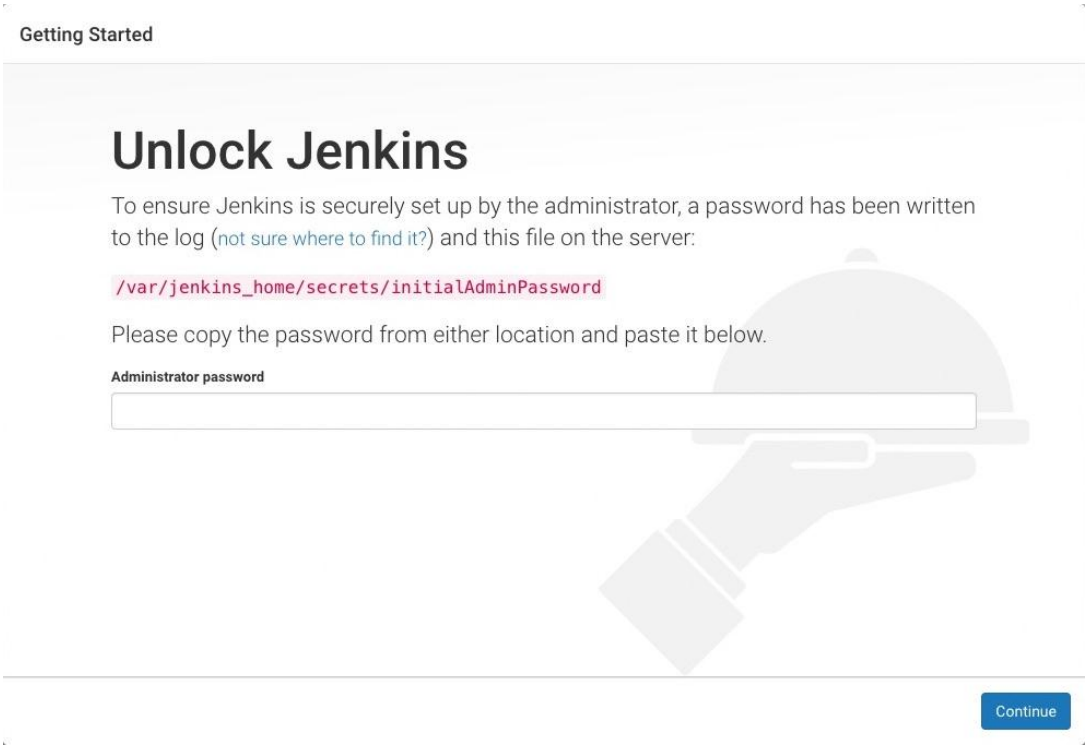

Рисунок 3.4 Welcoming Jenkins Screen

3.2 Розробка Freestyle Jobs

Після налаштування і завантаження додатків ми можемо приступити до розробки автоматизації створення артефактів. Це можна зробити за допомогою Pipeline або Freestyle Job. Pipeline Job дозволяє розбивати роботу на різні етапи і може мати будь-які етапи, які необхідно представляти. Але якщо щось піде не так, на якомусь етапі виникне проблема, весь Pipeline припинить роботу. Freestyle Job - це спрощена форма завдання одного типу, але з різними змінними і підзадачами. У нашому випадку достатньо створити дві Freestyle Job, за допомогою яких ми запустимо процес генерації артефактів і в разі успіху помістимо на S3.

Спочатку, налаштовуємо Amazon S3 Profile, завдяки якому ми зможемо помістити артефакти до S3. Створення Amazon IAM юзера з відповідним рівнем доступу (Рисунок 3.5)

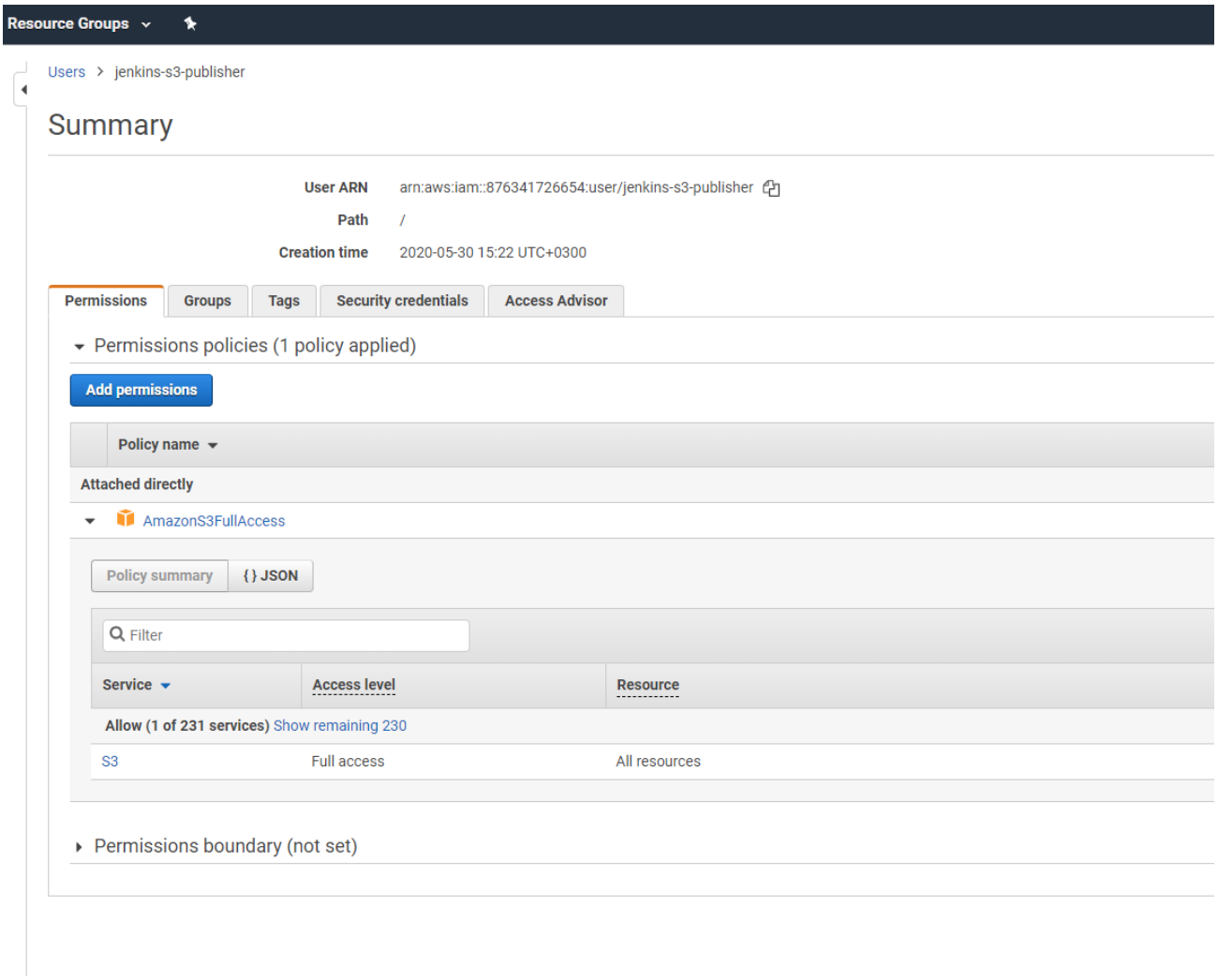

Рисунок 3.5 Профіль тільки для доступу до S3

Після створення профілю на AWS потрібно додати ключи цього профіля до Jenkins Configuration (Рисунок 3.6)

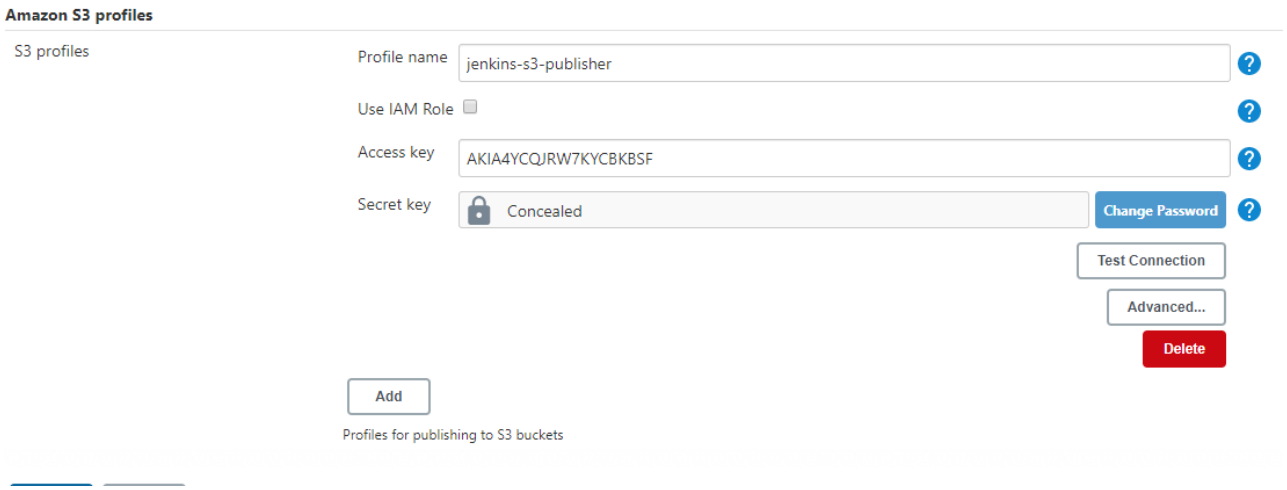

Рисунок 3.6 Амазон профіль у Jenkins S3 publisher

Вихідний код Front End і Back End знаходиться на Github. Щоб автоматизувати синхронізацію з Jenkins потрібно використовувати Git Jenkins Plugin. Для кожної BackEnd and FrontEnd Freestyle Job потрібно налаштувати константи та змінні:

URL Repository - URL-адресу віддаленого сховища. Плагін git передає URL віддаленого сховища в git (командний рядок або JGit). Допустимі URL-адреси сховища включають в себе https, ssh, scp, git, локальний файл і інші форми. Дійсні форми URL сховища описані в документації git.

Credentials - Облікові дані визначаються за допомогою плагіна облікових даних Jenkins. Вони вибираються із списку, їх ідентифікатор зберігається у визначенні завдання.

Name - git використовує коротку назву, щоб спростити посилання користувачів на URL-адресу віддаленого сховища. Коротке ім'я за замовчуванням - джерело. Інші значення можуть бути призначені і потім використані у визначенні завдання для посилання на віддалений репозиторій.

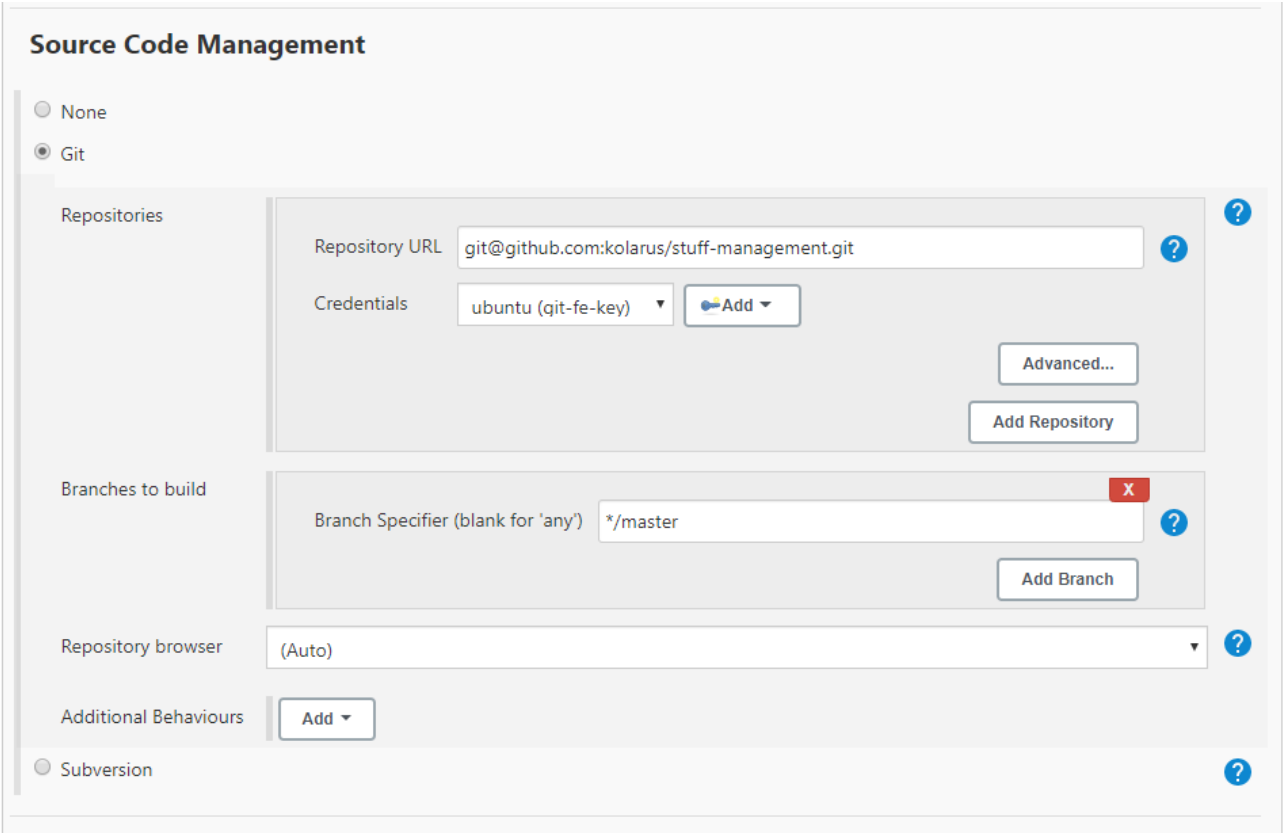

Рисунок 3.7 Зразок налаштування Git Repository

Головна частина автоматизації це Build стадії. Ми будемо використовувати дві - shell execute (Рисунок 3.8) та post-build action: publish artifacts to S3 Bucket (Рисунок 3.9).

Рисунок 3.9 Shell Build

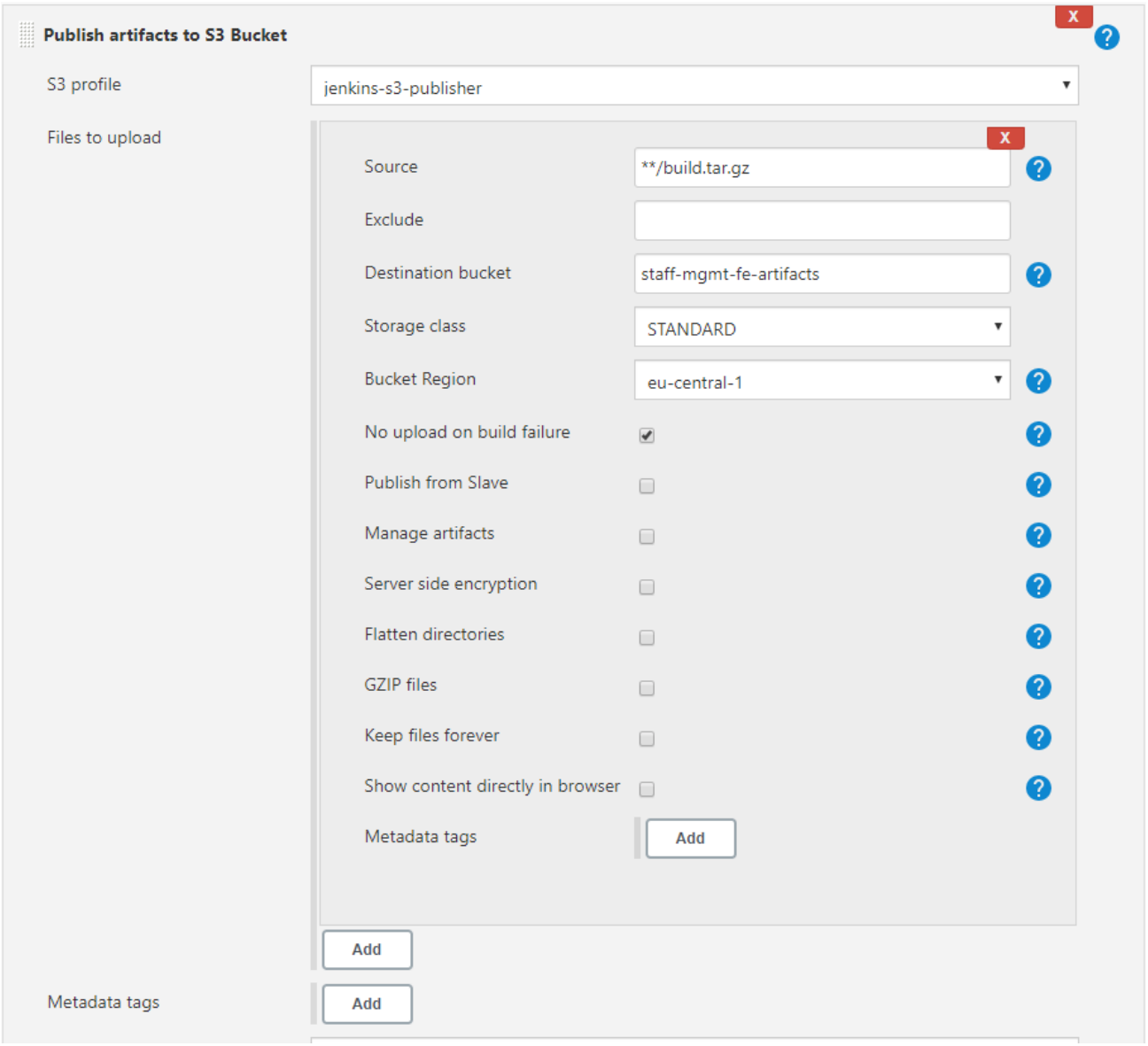

Рисунок 3.10 Publish to S3 Bucket

Після налаштування обох Freestyle Job головна сторінка Jenkins Master виглядає так (Рисунок 3.11)

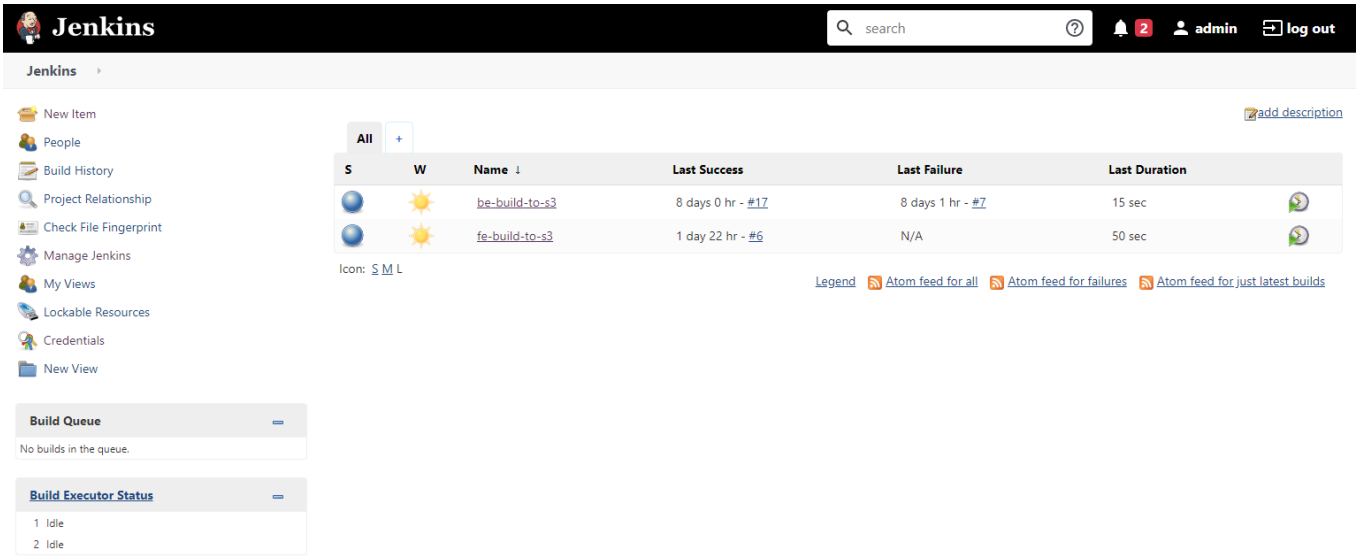

Рисунок 3.11 Jenkins Main Page

Завдяки тому, що Jenkins зберігає весь прогрес на жорсткий диск, можна переводити інстанси в стан Stopped без втрати даних, що дозволяє економити ресурси аккаунту.

4 РОЗРОБКА АРХІТЕКТУРИ ЗАСТОСУНКУ

4.1 Налаштування та використання Terraform IaC

Інструментом Infrastructure as Code буде використаний Terraform - інструмент з відкритим вихідним кодом, так звана інфраструктура як програмний код, створений HashiCorp. Він дозволяє користувачам визначати і надавати інфраструктуру центру обробки даних, використовуючи мову конфігурації високого рівня, відомий як Hashicorp Configuration Language (HCL), або JSON. Terraform підтримує ряд постачальників хмарної інфраструктури, таких як Amazon Web Services, Google Cloud Platform, DigitalOcean і інші.

Код конфігурації описує Terraform компоненти, необхідні для запуску окремого додатка. Terraform створює план виконання, що описує, що він буде робити для досягнення бажаного стану, а потім виконує його для побудови описаної інфраструктури. У міру зміни конфігурації Terraform може визначити, що змінилося, і створити додаткові плани виконання, які можна застосовувати.

Щоб встановити Terraform CLI достатньо завантажити виконавчий файл з офіційного сайту да додати його до одної з $PATH директорії. В цьому проекті буде використано останню версію Terraform Linux CLI - 0.12.25.

По-перше, потрібно налаштувати головний root.tf файл. До нього входять основні елементи, наприклад *provider, terraform, module.*

Provider - відповідає за розуміння взаємодії API і розкриття ресурсів. У нашому випадку provider “aws”.

Terraform {backend} - визначає, як завантажується стан і як виконується така операція, як apply. Ця абстракція дозволяє зберігати стан файлу віддалено. У нашому випадку використовується S3 Bucket “s3backend-for-

terraform”

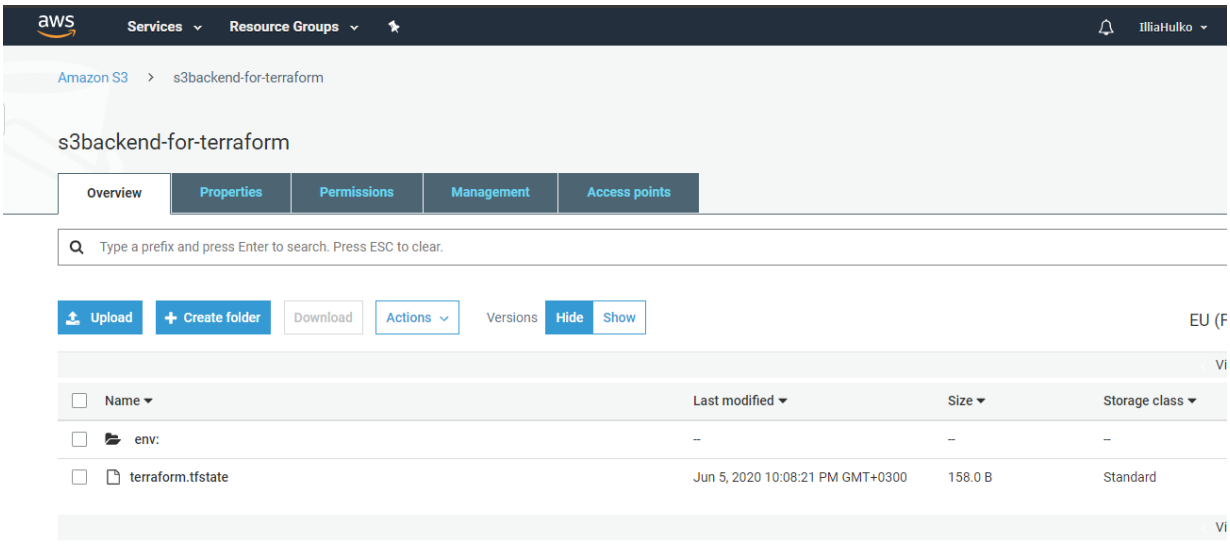

Рисунок 4.1 Terraform Backend Bucket

Module - це контейнер для декількох ресурсів, які використовуються разом. Кожна конфігурація Terraform має принаймні один модуль, відомий як його кореневої модуль, який складається з ресурсів, визначених у файлах .tf в основному робочому каталозі.

Після базового налаштування потрібно використати команду *terraform init* - використовується для ініціалізації робочого каталогу, що містить файли конфігурації Terraform. Це перша команда, яка повинна бути виконана після запису нової конфігурації Terraform або клонування існуючої з системи управління версіями.

4.2 Розрахунок мережі

4.2.1 Subnets та VPC

Грунтом для будь-якої мережевого налаштування в Amazon Web Services є Virtual Private Cloud.

VPC - це набір автономних підмереж із загальним блоком безкласової маршрутизації (CIDR) (до маски /16), які працюють в одній географічній області (Region) в декількох центрах обробки даних (Availability Zones). VPC схожий на віртуальний центр обробки даних, за винятком того, що він фізично розподілений по зонам доступності. VPC мають підключення до мережі всередині регіону, в якому вони створені. Для підключення VPC до інших мереж можна використовувати підключення до Інтернету, підключення до віртуальної приватної мережі (VPN) і VPC Peering. Рекомендується використання RFC 1918.

Зона доступності (Availability Zone) - це набір будівель, інтернет-каналів і джерел живлення. Можна розглядати це як центр обробки даних, але деякі зони доступності містять більше одного фізичного центру обробки даних. В даний час в світі існує 76 зони доступності.

AWS Region - це колекція зон доступності. Можна уявити компоненти регіону як кілька центрів обробки даних в одній географічній області. Наприклад, регіон з двома зонами доступності був би придатний для аварійного відновлення з синхронною реплікацією. AWS має 24 регіонів по всьому світу.

Рисунок 4.2 Мапа регіонів та зон доступності AWS

Налаштування підмереж можливо завдяки Subnets. Підмережі займають змінну частину блоку CIDR, призначеного VPC. Застосовуються звичайні правила і позначення підмереж, тому вони повинні не перетинатися з локальними центрами обробки даних або іншими VPC. Підмережа залежить від зони доступності, і в кожній зоні доступності може бути декілька підмереж. Кожна підмережа має одну прив'язку таблиці маршрутів, а кожна підмережа пов'язана зі списком управління доступом до мережі (ACL мережі). AWS використовує п'ять адрес в кожній підмережі: перші чотири і остання адреса. Мінімальний розмір підмережі - /28 маска мережі з 11 адресами. Підмережі можуть бути дуже великими, оскільки широкомовний трафік виключається. Крім того, такі пристрої, як Elastic Load Balancer (ELB), використовують IP-адреси, і багато робітників навантаження є еластичними і використовують більше адрес у міру їх масштабування, тому це повинно враховуватися при визначенні розміру підмереж.

В консолі AWS можна побачити посилання на публічні та приватні підмережі. Це швидкий спосіб описати, чи має підмережа прямий маршрут до Інтернету, чи ні. Як правило, загальнодоступна підмережа має прямий доступ до Інтернету. Приватна підмережа не має зовнішнього доступу - вона підключається через віртуальний приватний шлюз (VGW) до локального центру обробки даних або використовує перетворення мережевих адрес (NAT) для трафіку, пов'язаного з Інтернетом. В підмережі також є параметр «Автоматичне призначення загальнодоступних IP-адрес», який за замовчуванням може давати віртуальним машинам загальнодоступні IP-адреси (і може бути перевизначений для кожної VM).

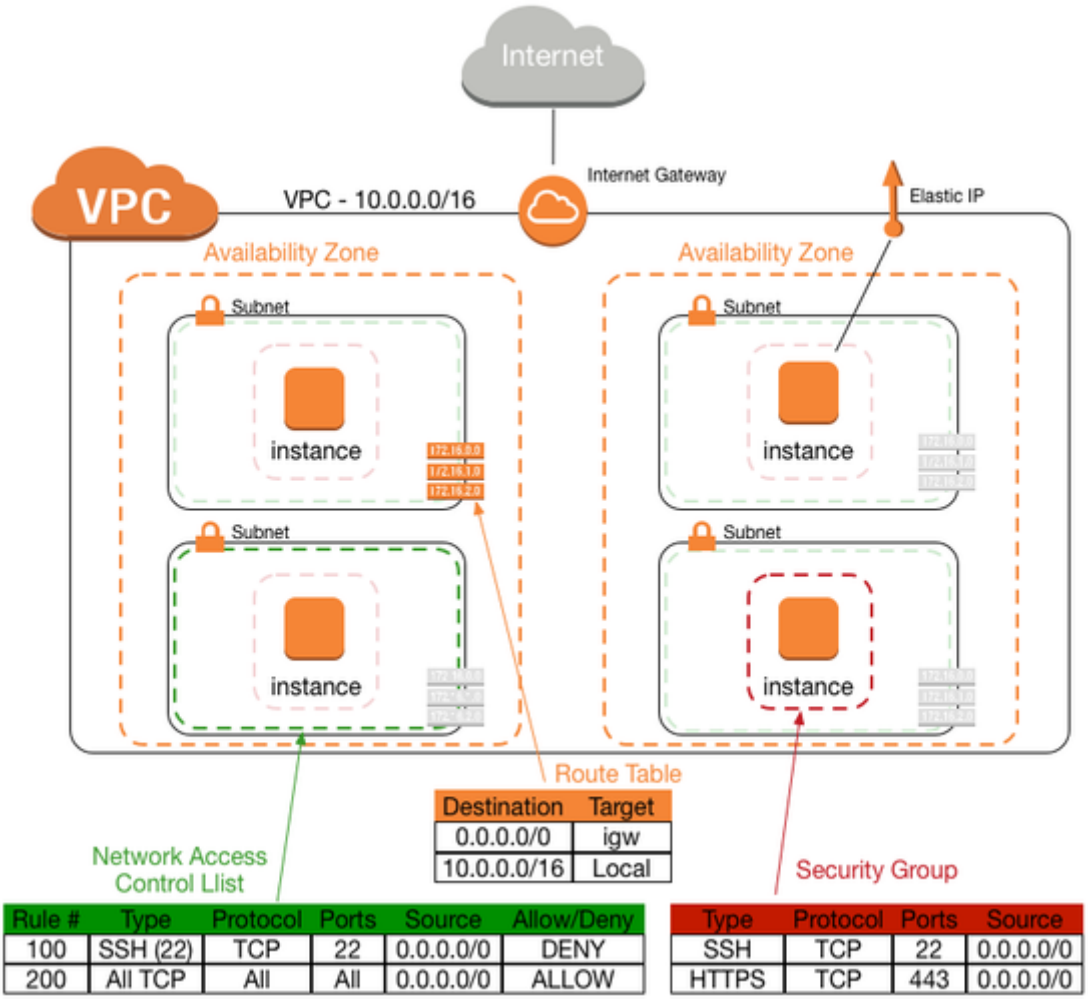

Рисунок 4.3 Приклад налаштування VPC

Для розгортання додатка був обраний регіон Франкфурт - eu-central-1, як найближчий датацентр з трьома зонами доступності. Для підвищення надійності будуть використовуватися дві зони доступності: eu-central-1a та eu-cetnral-1b.

Таблиця 4.1 – Налаштування мережі VPC

Name	Region	Net/Mask bits	Range of IPs	Hosts
VPCProduction	eu-central-1	10.10.0.0/24	10.10.0.0 – 10.10.0.255	256

Для кожного типу ресурса (Front End VMs, Back End VMs, Database RDS Instance) буде зареєстровано дві підмережі у різних зонах доступу.

Таблиця 4.2 – Мережеве налаштування підмереж у VPC

Name	Region	Net/Mask bits	Range of IPs	Hosts	Public IP
FESubnet1	eu-central-1a	10.10.0.0/28	10.10.0.1 - 10.10.0.14	14	No
FESubnet2	eu-central-1b	10.10.0.16/28	10.10.0.17 - 10.10.0.30	14	No
BESubnet1	eu-central-1a	10.10.0.192/28	10.10.0.193 - 10.10.0.206	14	No
BESubnet2	eu-central-1b	10.10.0.208/28	10.10.0.209 - 10.10.0.222	14	No
DBSubnet1	eu-central-1a	10.10.0.224/28	10.10.0.225 -	14	No

			10.10.0.238		
DBSubnet2	eu-central-1b	10.10.0.240 /28	10.10.0.241 - 10.10.0.254	14	No

4.2.2 Security Groups

Група безпеки (Security Group) схожа на список контролю доступу (ACL), зазвичай застосовується до маршрутизаторів і міжмережевих екранів, за винятком того, що він застосовується безпосередньо до інстансу. Групи безпеки обробляють більшу частину безпеки в AWS для захисту інстансів і забезпечують детальну безпеку для кожного інстансу. Групи безпеки мають стан і відстежують стан з'єднання TCP, UDP і ICMP. Групи безпеки містять правила груп безпеки, які схожі на записи ACL. Правила групи безпеки працюють за білим списком і містять неявне правило DENY ANY. Групи безпеки можуть посилатися на інші групи безпеки, діючи аналогічно групі мережі/об'єкта в брандмауерах. Ця можливість дозволяє трафік від певних груп інстансів, не вимагаючи використання IP-адрес. В цілому групи безпеки схожі на розподілений міжмережевий екран рівня 4.

Групи безпеки існують зі станом: якщо запит відправляється з інстансу, у відповідь трафіку на цей запит дозволяється надходити незалежно від правил входу до групи безпеки. Відповіді на дозволений вхідний трафік можуть передаватися незалежно від правил вихідних повідомлень.

Групи безпеки пов'язані з мережевими інтерфейсами. Після запуску інстансу можна змінити групи безпеки, пов'язані з екземпляром, які змінюють групи, пов'язані з основним мережевим інтерфейсом (за замовчуванням це eth0). Також можна вказати або змінити групи безпеки, пов'язані з будь-яким

іншим мережевим інтерфейсом. За замовчуванням при створенні мережевого інтерфейсу він зв'язується з групою безпеки за для VPC, якщо не вказана інша група.

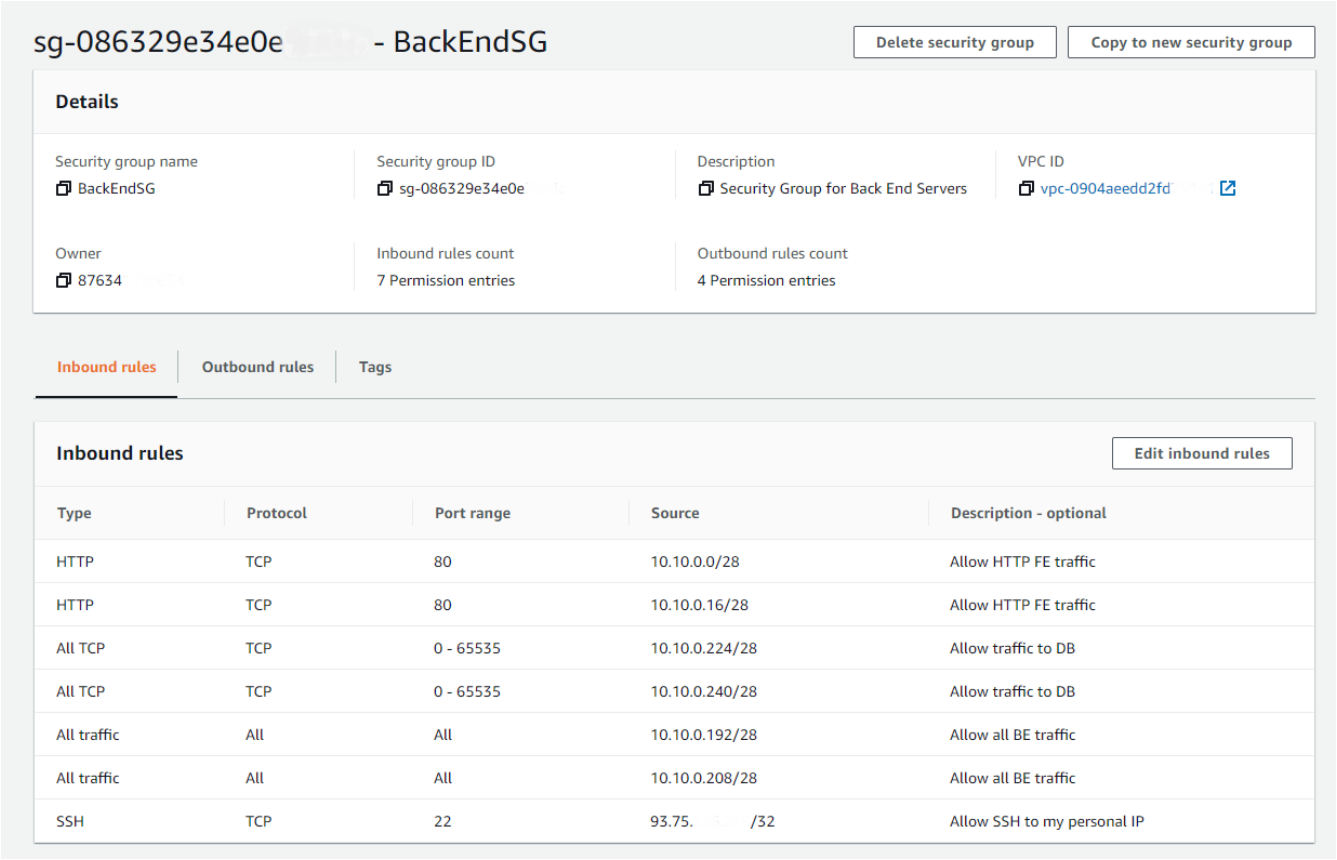

Рисунок 4.4 Зразок налаштування Security Group

Мережеві ACL застосовуються до кожної підмережі і забезпечують більш широке охоплення, ніж групи безпеки. Вони будуть використовуватися як ACL VLAN (VACL) або ACL рівня 2 (хоча в VPC немає рівня 2). Мережеві ACL-списки впорядковані і мають дозволи і відмови. Мережеві ACL, на відміну від груп безпеки, не мають стану. Крім того, мережеві списки ACL працюють тільки в діапазонах CIDR і не можуть посилатися на інші об'єкти, такі як інстанси. Мережевий ACL за замовчуванням дозволяє весь трафік.

Таблиця 4.3 – Налаштування груп безпеки на всіх рівнях

Тип ресурсу	Протокол/тип	Порти	Джерело	Опис
Front End Application Load Balancer	INGRESS HTTP/TCP	All	10.10.0.0/28 10.10.0.16/28	Accept all from FE Subnets
	INGRESS HTTP(S)/TCP	80 – 443	0.0.0.0/0	Accept all HTTP or HTTPS traffic
Front End EC2	INGRESS HTTP/TCP	80	10.10.0.0/28 10.10.0.16/28	Accept HTTP from ALB or FE
	INGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow HTTP from BE
Back End EC2	INGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow traffic in between BE
ElasticSearch Domain	INGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow traffic from BE
	EGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow traffic to BE
RDS Instances	INGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow traffic from BE
	EGRESS HTTP/TCP	All	10.10.0.192/28 10.10.0.208/28	Allow traffic to BE

4.2.3 Route Tables and Internet Gateway

Кожна підмережа має таблицю маршрутів, і одна таблиця маршрутів може бути пов'язана з декількома підмережами. Таблиці маршрутів діють як правило маршрутизації на основі політик (policy-based routing). Тобто можна вибрати, в якому напрямку повинні відправлятися пакети, виходячи з підмережі, в якій знаходиться інстанси. Наприклад, приватні підмережі можуть мати таблиці маршрутів з маршрутом за замовчуванням до шлюзу NAT, а публічні підмережі можуть мати таблиці маршрутів в Інтернет з використанням інтернет-шлюзу. Приватні підмережі також можуть мати маршрути назад в локальні центри обробки даних з використанням віртуального приватного шлюзу.

Таблиці маршрутів також можна описати як таблиці віртуальної маршрутизації і пересилання (VRF). Однак підмережі в одному і тому ж VPC можуть безпосередньо зв'язуватися один з одним, тому поділ VRF зазвичай не застосовується. Функціонально все в одному VPC має досяжність рівня 2. Групи безпеки і мережеві списки використовуються для управління доступом, щоб забезпечити більш детальні настройки доступу.

Таблиці маршрутів містять маршрути, які вказують блок CIDR на інстанси, інтернет-шлюз, віртуальний приватний шлюз, шлюз NAT, спеціальний робочий вузол VPC або кінцеву точку VPC. Таблиці маршрутів також можуть отримувати динамічні маршрути BGP від віртуального приватного шлюзу.

Таблиця маршрутів за замовчуванням автоматично приєднується до нових підмереж, якщо вони не пов'язані вручну з іншою таблицею маршрутів. Для більш високого рівня контролю та гнучкості потрібно створити нову таблицю маршрутів для підмереж замість використання таблиці маршрутів за замовчуванням. Якщо служба є локальною для зони доступності (наприклад,

шлюзу NAT), рекомендується визначати таблиці маршрутів для кожної зони доступності.

Інтернет-шлюз не є окремим фізичним пристроєм. Він аналогічний багатоколійні маршруту з рівними витратами (Equal-Cost Multi-Path routing) на набір інтернет-маршрутизаторів. Один інтернет-шлюз вважається високодоступних в межах регіону без будь-яких додаткових налаштувань, як якщо вже є кілька маршрутів однакової вартості до одного і того ж пункту призначення.

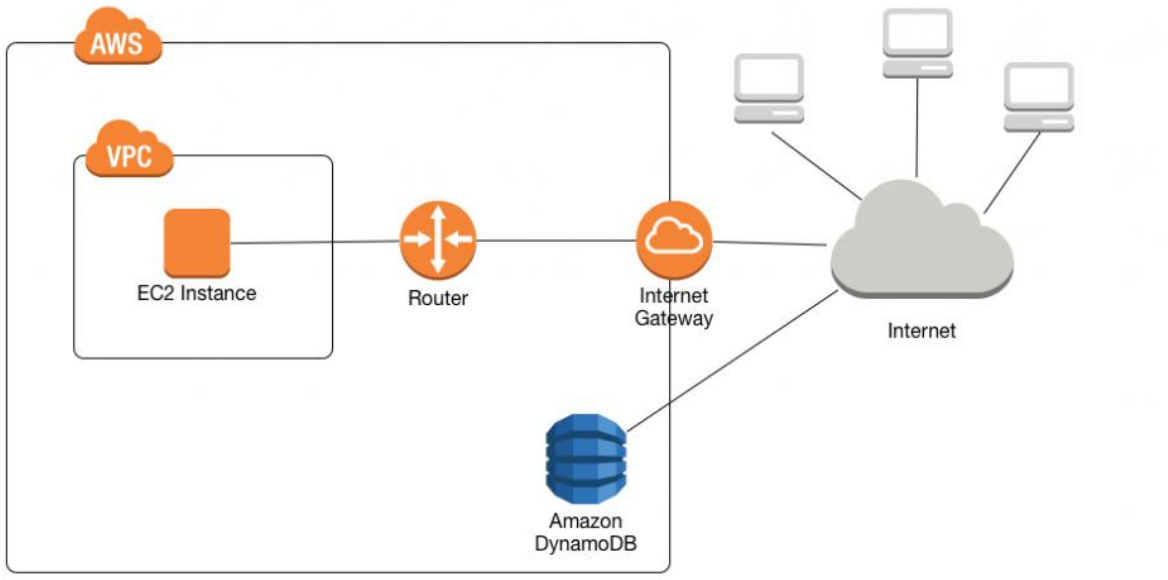

Рисунок 4.5 Зразок налаштування Internet Gateway

4.3 Налаштування обчислюваних потужностей

Amazon EC2 є віртуальним обчислювальним середовищем, що дозволяє використовувати інтерфейси веб-служб для запуску інстансів з різними операційними системами, завантаження їх в призначену для користувача

середу додатків, управління дозволами доступу вашої мережі і запуску образу з використанням якомога більшої або меншої кількості систем.

Щоб використовувати Amazon EC2 потрібно вибирати попередньо налаштований шаблонний образ машини Amazon (AMI) для негайного запуску віртуальної машини, або створити AMI самостійно, щоб він містив програми, бібліотеки, дані та відповідні параметри конфігурації; налаштувати безпеку і доступ до мережі на інстансі Amazon EC2; вибрати потрібний тип інастансів.

4.3.1 Типи EC2 та інших інстансів

Amazon EC2 надає широкий вибір типів інстансів, оптимізованих під різні варіанти використання. Типи інстансів включають в себе різні комбінації ресурсів процесора, оперативної пам'яті, постійного сховища і мережевих ресурсів, які надають гнучкість у виборі відповідного поєднання ресурсів для різних додатків. Кожен тип інастансу включає один або кілька розмірів примірників, що дозволяє масштабувати ресурси відповідно до вимог цільового робочого навантаження.

Інстанси Amazon EC2 A1 засновані на ARM архітектурі, тобто розраховані на навантаження цього типу, які підтримуються екосистемою ARM. Інстанси A1 працюють на процесорах AWS Graviton з 64-бітними ядрами Arm Neoverse.

Інстанси T3 - це універсальний тип, який забезпечує базовий рівень продуктивності ЦП з можливістю збільшення використання ЦП в будь-який час протягом необхідного періоду. Інстанси T3 пропонують баланс обчислювальних, оперативних і мережевих ресурсів і призначені для додатків з помірним використанням ЦП, які відчувають тимчасові сплески використання.

Інстанси T3 накопичують кредити ЦП, коли робоче навантаження нижче базового порогу. Кожен зароблений кредит ЦП дає примірнику T3 можливість при необхідності збільшити продуктивність повного ядра ЦП на одну хвилину. Примірники T3 можуть використати кредити в будь-який час стільки, скільки потрібно в режимі.

Інстанси T2 забезпечують базовий рівень продуктивності ЦП з можливістю перевищення базового рівня. Інстанси T2 за налаштуванням T2 Unlimited можуть підтримувати високу продуктивність процесора, поки це потрібно робочому навантаженню. Для більшості робочих навантажень загального призначення екземпляри T2 Unlimited забезпечують достатню продуктивність без будь-яких додаткових витрат. Якщо інстанс повинен працювати з більш високою завантаженням ЦП протягом тривалого періоду, він також може робити це з фіксованою доплатою за vCPU-годину.

Інстанси M5 це останнє покоління загального призначення на базі процесорів Intel Xeon® Platinum 8175M з частотою до 3,1 ГГц з новим набором команд Intel Advanced Vector Extension (AVX-512). Ці CPU забезпечують баланс обчислювальних, оперативних і мережевих ресурсів і є хорошим вибором для багатьох додатків.

Для нашого додатку найбільш оптимізованим вибором буде використання інстансів класу T3, тому що ми не очікуємо загрузки на диск.

Для FrontEnd інстансів буде використано t3.medium з двома ядрами CPU та 4 ГіБ оперативної пам’яті. Для BackEnd – t3.large, з однаковим числом ядер але вдвічі більше ОЗУ, тому що ми очікуємо деяке кешування з боку додатку.

Таблиця 4.3 – Характеристики обраних EC2 інстансів

Name	vCPU, Cores	Memory, GIB	Instance Storage	Linux/UNIX Usage, $ per Hour
t3.medium	2	4	EBS Only	$0.048
t3.large	2	8	EBS Only	$0.096

4.3.2 Операційні системи

Образ машини Amazon (AMI) надають інформацію, необхідну для запуску інстансу. AMI повинне бути вказано при запуску інстансу. Можна запустити кілька екземплярів з одного AMI, коли потрібно кілька інстансів з однаковою конфігурацією.

Ми будемо використовувати образ дистрибутиву на базі Ubuntu, як одна з найпоширеніших дистрибутивів Server Linux.

Ubuntu - це дистрибутив Linux, заснований на Debian, в основному складається з безкоштовного програмного забезпечення з відкритим кодом. Ubuntu офіційно випущений в трьох випусках: Desktop, Server і Core для Інтернету речей, пристроїв і роботів. Всі версії можуть запускатися на комп'ютері або на віртуальній машині. Ubuntu - популярна операційна система для хмарних обчислень з підтримкою OpenStack.

Ubuntu випускається кожні шість місяців, а версія з довгостроковою підтримкою (LTS) - кожні два роки. Станом на 23 квітня 2020 року самий останній випуск версії з довгостроковою підтримкою - 20.04 («Focal Fossa»), який підтримується до 2025 року в рамках загальної підтримки і до 2030 року в якості платного варіанту.

Ubuntu розроблена Canonical у співтоваристві з іншими розробниками. Canonical надає оновлення безпеки та підтримку для кожного випуску Ubuntu, починаючи з дати випуску і поки випуск не досягне призначеної дати закінчення терміну служби (End Of Life). Canonical генерує дохід за рахунок продажу преміальних послуг, пов'язаних з Ubuntu.

Образи Ubuntu завантажуються і реєструються в хмарі Amazon EC2. Кожен AMI має свій унікальний ідентифікатор. Щоб запустити екземпляр в хмарі EC2, спочатку потрібно знайти його ідентифікатор.

Ми будемо використовувати минулу версію Long Term Support, яка добре зарекомендувала себе - Ubuntu 18.04

Таблиця 4.4 – Загальний опис AMI

Зона (Регіон)	eu-central-1
Ім’я	bionic
Версія	18.04
Архітектура	amd64
Тип інстансу	hvm-ssd
Реліз	20200408
Ідентифікатор	ami-0e342d72b12109f91

4.3.3 Диски

Amazon Elastic Block Store (Amazon EBS) надає томи сховищ на рівні блоків для використання з інстансами EC2. Тома EBS поводяться як необроблені, неформатовані блокові пристрої. Можна змонтувати ці томи як пристрої в інстансах. Можна підключити декілька томів до одного інастансу і

одночасно підключити кілька томів. Також, можна створити файлову систему поверх цих томів або використовувати їх будь-яким способом, яким використовується блоковий пристрій (наприклад, жорсткий диск). Можно динамічно змінювати конфігурацію тому, підключеного до інстансу.

Томи EBS - це високодоступні та надійні томи зберігання, які можна підключити до будь-якого запущеного інстансу в тій же зоні доступності. Томи EBS, підключені до інстансу EC2, відображаються як томи зберігання, які працюють незалежно від терміну служби інстансу.

Таблиця 4.5 – Ціни за використання різних типів EBS

General Purpose SSD (gp2) Volumes	$0.119 per GB-month of provisioned storage
Provisioned IOPS SSD (io1) Volumes	$0.149 per GB-month of provisioned storage AND $0.078 per provisioned IOPS-month
Throughput Optimized HDD (st1) Volumes	$0.054 per GB-month of provisioned storage
Cold HDD (sc1) Volumes	$0.03 per GB-month of provisioned storage

У нашому випадку ми не очікуємо завантаження великої кількості даних на диски, але в той же час бажано щоб будь-яке звернення до диска займало якомога менше часу. Тому, у відповідності до вимог додатку, ми будемо використовувати 10 GiB на FrontEnd та 30 GiB на BackEnd серверах.

4.3.4 Налаштування Auto Scaling Groups

Група автоматичного масштабування містить колекцію інстансів Amazon EC2, які розглядаються як логічна група для автоматичного масштабування і управління. Auto Scaling Group також дозволяє використовувати функції автоматичного масштабування Amazon EC2, такі як заміни перевірки працездатності і політики масштабування. Підтримка кількості інстансів в Auto Scaling Group і автоматичне масштабування є основними функціями сервісу.

Розмір групи автоматичного масштабування залежить від кількості інастансів, які вказані в якості бажаної ємності. Його розмір можна налаштувати або відповідно до попиту, або вручну, або за допомогою автоматичного масштабування.

Група автоматичного масштабування починається з запуску достатньої кількості інстансів для досягнення бажаної ємності. Auto Scaling Group підтримує кількість інстансів, виконуючи періодичні перевірки працездатності інстансів в групі. Група автоматичного масштабування продовжує підтримувати фіксовану кількість інстансів, навіть якщо він стає нездоровим. Якщо інстанс стає нездоровим, група завершує роботу нездорового інстанса і запускає другий для його заміни.

Можна використовувати політики масштабування для динамічного збільшення або зменшення кількості інстансів у групі відповідно до умов, що змінюються. Коли діє політика масштабування, група «Автоматичне масштабування» регулює бажану ємність групи між зазначеними мінімальним і максимальним значеннями ємності і запускає або завершує інстанси в міру необхідності. Також можена масштабувати Auto Scaling Group за розкладом.

Після того, як інстанс повністю налаштован і пройшов початкові перевірки працездатності, Amazon EC2 Auto Scaling вважає його працездатним.. Коли ASG визначає, що інстанс є нездоровим, він закриває

його і запускає новий. Це допомагає підтримувати кількість запущених інстансів на мінімальній кількості (або необхідній кількості, якщо вказано), яке визначено у конфігурації.

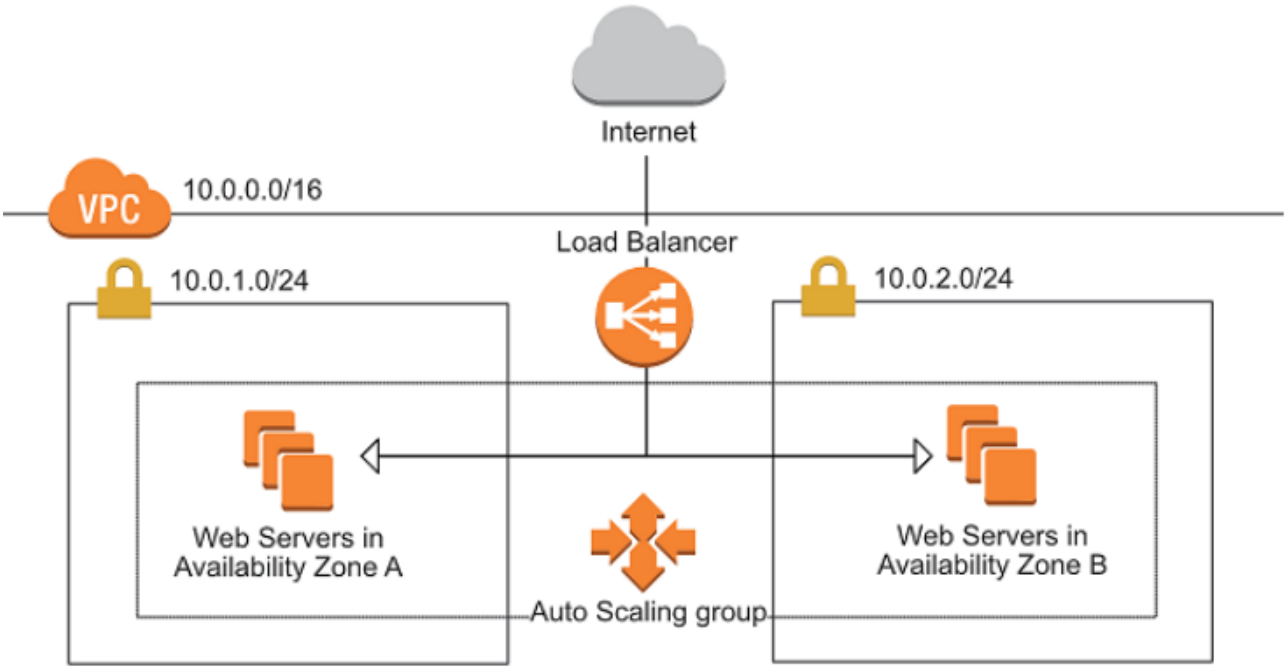

Рисунок 4.6 Зразок налаштування Auto Scaling Group

У нашої архітектурі для FrontEnd будуть налаштовані перевірки працездатності за портом 80, для BackEnd за портом 1234. Для BE & FE тип підключення TCP / HTTP.

Бажана кількість FrontEnd інстансів - 4, BackEnd інстансів - 3.

4.4 Налаштування Load Balancing

Application Load Balancer служить єдиною точкою контакту для клієнтів. Балансувальник навантаження розподіляє вхідний трафік додатків по декількох цілях, таким як інстанси EC2, в декількох зонах доступності. Це збільшує доступність застосування.

Слухач перевіряє запити на підключення від клієнтів, використовуючи протокол і порт, який було налаштовано. Правила, які були визначені для слухача, визначають, як балансувальник навантаження направляє запити до зареєстрованих цілях. Кожне правило складається з пріоритету, одного або декількох дій і одного або декількох умов. При виконанні умов для правила виконуються його дії.

Кожна цільова група направляє запити до однієї або декількох зареєстрованих цілей, таким як інстансі EC2, використовуючи зазначений протокол і номер порту. Можна налаштувати перевірки працездатності для кожної цільової групи. Перевірка працездатності виконується для будь-яких цілей, зареєстрованих в цільовій групі, зазначеної в правилі прослуховувач для балансувальника навантаження.

Балансувальник мережевого навантаження функціонує на четвертому рівні моделі взаємодії відкритих систем (OSI). Він може обробляти мільйони запитів в секунду. Після того, як балансувальник навантаження отримує запит на підключення, він вибирає ціль з цільової групи для правила за замовчуванням. Він намагається відкрити TCP-з'єднання з обраною ціллю на порту, зазначеному в конфігурації прослуховувача.

При включенні зони доступності для балансувальника навантаження Elastic Load Balancing створює вузол балансування навантаження в зоні доступності. За замовчуванням кожен вузол балансування навантаження розподіляє трафік за зареєстрованими цілям тільки в своїй зоні доступності. Якщо балансування навантаження між зонами включене, кожен вузол балансування навантаження розподіляє трафік за зареєстрованими цілям у всіх включених зонах доступності.

Для трафіку TCP балансувальник навантаження вибирає ціль з використанням алгоритму хешування потоку на основі протоколу, IP-адреси

джерела, порту джерела, IP-адреси призначення, порту призначення і порядкового номера TCP. З'єднання TCP від клієнта мають різні вихідні порти і порядкові номери і можуть бути спрямовані на різні цілі. Кожне окреме TCP-з'єднання направляється на одну ціль протягом всього терміну життя з'єднання.

Для UDP-трафіку балансувальник навантаження вибирає мету з використанням алгоритму хешування потоку на основі протоколу, IP-адреси джерела, порту джерела, IP-адреси призначення і порту призначення. Потік UDP має один пункт джерела і пункт призначення, тому він послідовно прямує до однієї цілі протягом всього терміну служби. Різні потоки UDP мають різні вихідні IP-адреси і порти, тому вони можуть бути спрямовані на різні цілі.

Elastic Load Balancing створює мережевий інтерфейс для кожної включеної зони доступності. Кожен вузол балансування навантаження в зоні доступності використовує цей мережевий інтерфейс для отримання статичної IP-адреси.

Ми будемо використовувати один Application Load Balancer для роутінгу трафіка на Front End інстансі та один Public Network Load Balancer для Back End інстансів.

4.5 Налаштування баз даних та data storage

Амазон має 15 спеціалізованих механізмів баз даних, включаючи реляційні бази даних, бази даних ключі-значення, сховище документів, Time series database, графи, часові ряди і регістри. Набір спеціалізованих баз даних AWS підтримує різні моделі даних і дозволяє створювати розподілені додатки з високим ступенем масштабованості, засновані на сценаріях використання**.**

4.5.1 AWS ES Domain

Amazon Elasticsearch Service (Amazon ES) - це керований сервіс, який спрощує розгортання, експлуатацію і масштабування Elasticsearch, популярного механізму пошуку і аналізу з відкритим вихідним кодом. Amazon ES також пропонує параметри безпеки, високу доступність, надійність даних і прямий доступ до API Elasticsearch.

Amazon ES надає всі ресурси для кластера Elasticsearch і запускає його. Він також автоматично виявляє і замінює нездорові вузли Elasticsearch, зменшуючи витрати.

Щоб почати використовувати Amazon ES, потрібно створити домен. Домен Amazon ES є синонімом кластера Elasticsearch. Домени - це кластери з заданими параметрами, типами інстансів, їх кількістю і ресурсами зберігання. Кожен інстанс діє як один вузол Elasticsearch.

Ми будемо використовувати версію Elasticsearch 7.4 на двох збалансованих інстансах m5.large.elasticsearch

Таблиця 4.6 –Характеристики t2.medium.elasticsearch

Name	vCPU, Cores	Memory, GIB	Instance Storage	Linux/UNIX Usage, $ per Hour
m5.large.elasticsearch	2	8	EBS Only	$0.17

4.5.2 RDS PostgreSQL

Amazon Relational Database Service (Amazon RDS) - це веб-сервіс, який спрощує настройку, роботу і масштабування реляційної бази даних в хмарі AWS. Він забезпечує економічно ефективну ємність із змінними розмірами

для стандартної реляційної бази даних і управляє загальними завданнями адміністрування бази даних.

Основним будівельним блоком Amazon RDS є інстанс БД - це ізольована середа бази даних в хмарі AWS. Інстанс БД може містити кілька користувальницьких баз даних. Можливо створювання і зміна інстансу БД за допомогою інтерфейсу командного рядка AWS, API Amazon RDS або Консолі управління AWS.

Кожен інстанс БД запускає механізм Бази Даних. Amazon RDS в даний час підтримує MySQL, MariaDB, PostgreSQL, Oracle і Microsoft SQL Server DB. Кожен механізм БД має свої власні підтримувані функції, і кожна версія механізму БД може включати в себе певні функції. Крім того, кожен механізм БД має набір параметрів в групі параметрів БД, які керують поведінкою баз даних, якими він управляє.

В нашому проекті розробники вирішили використовувати движок PostgreSQL – безкоштовна база даних з відкритим кодом і система управління реляційними базами даних (RDBMS), що підкреслює розширюваність і відповідність SQL. Вона буде розташована на двох інстансах у двох зонах доступності: db.t3.medium який коштує $0.168 за годину.

4.5.3 Simple Storage Service

Amazon Simple Storage Service (Amazon S3) - це сховище для Інтернету. Amazon S3 може бути використано для зберігання та вилучення будь-якого обсягу даних в будь-який час з будь-якої точки мережі.

Amazon S3 зберігає дані у вигляді об'єктів відро (Bucket). Об'єкт - це файл і будь-які додаткові метадані, які описують файл. Щоб зберегти файл в Amazon S3, він завантажується в відро. Коли файл завантажується як об'єкт, можна встановити дозволу для об'єкта і будь-яких метаданих.

Відро - це контейнер для файлів. Можна контролювати доступ для кожного сегмента, вирішуючи, хто може створювати, видаляти і перераховувати в ньому об'єкти. Також можливо вибрати географічний регіон, в якому Amazon S3 буде зберігати кошик і її вміст, а також переглядати журнали доступу для відра і його об'єктів.

Таблиця 4.7 – Ціни за використання S3 Standart

PUT, COPY, POST, LIST requests (per 1,000 requests)	GET, SELECT, and all other requests (per 1,000 requests)	Storate of First 50 TB / Month
$0.0054	$0.00043	$0.0245 per GB

Ми будемо використовувати 4 Bucket'а:

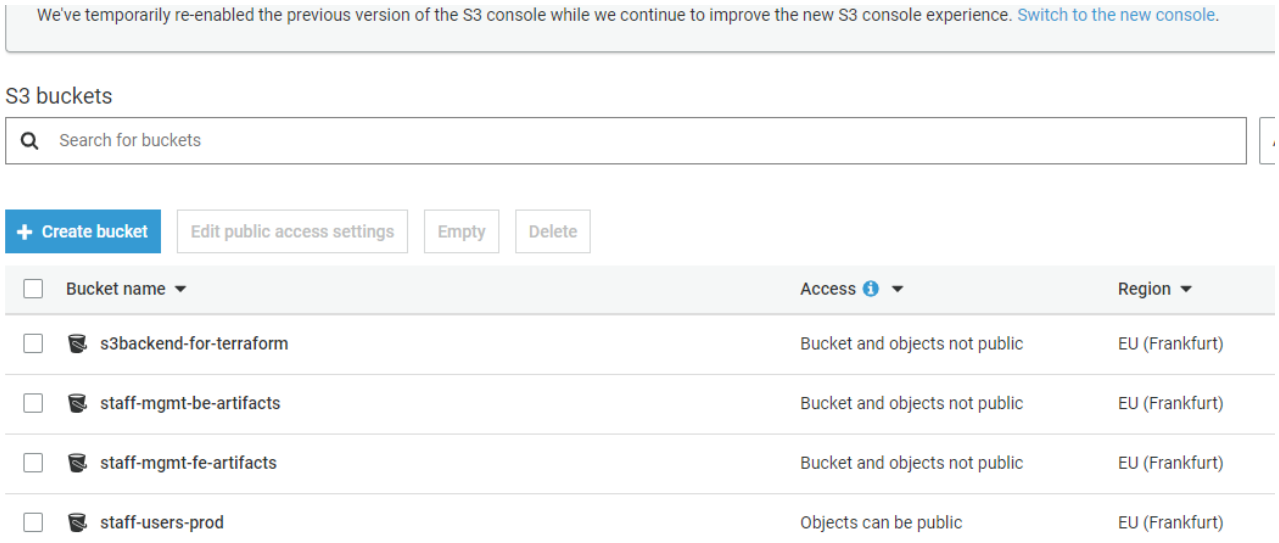

Рисунок 4.7 Налаштовані Buckets для додатку

4.6 Налаштування IAM

AWS Identity and Access Management (IAM) - це веб-сервіс для безпечного контролю доступу до сервісів AWS. За допомогою IAM можна централізовано керувати користувачами, обліковими даними безпеки, такими як ключі доступу, дозволами, які контролюють доступ користувачів і додатків до ресурсів AWS.

Для нашого додатку буде створено 3 юзера:

1. jenkins-s3-publisher – має доступ до S3
2. staff-backend-user – має доступ до S3, Cognito та Secrets Manager
3. staff-mgmt-s3-user – має доступ до S3

4.7 Налаштування Cognito

Amazon Cognito дозволяє швидко і легко додати користувачів в систему, увійти в систему і управляти доступом до своїх веб і мобільних додатків. Amazon Cognito масштабується до мільйонів користувачів і підтримує вхід з постачальниками соціальних ідентифікаторів, такими як Facebook, Google і Amazon, а також з постачальниками корпоративних ідентифікаційних даних через SAML 2.0.

Для нашого додатку буде створено User Pool staff та App Clients StaffBE.

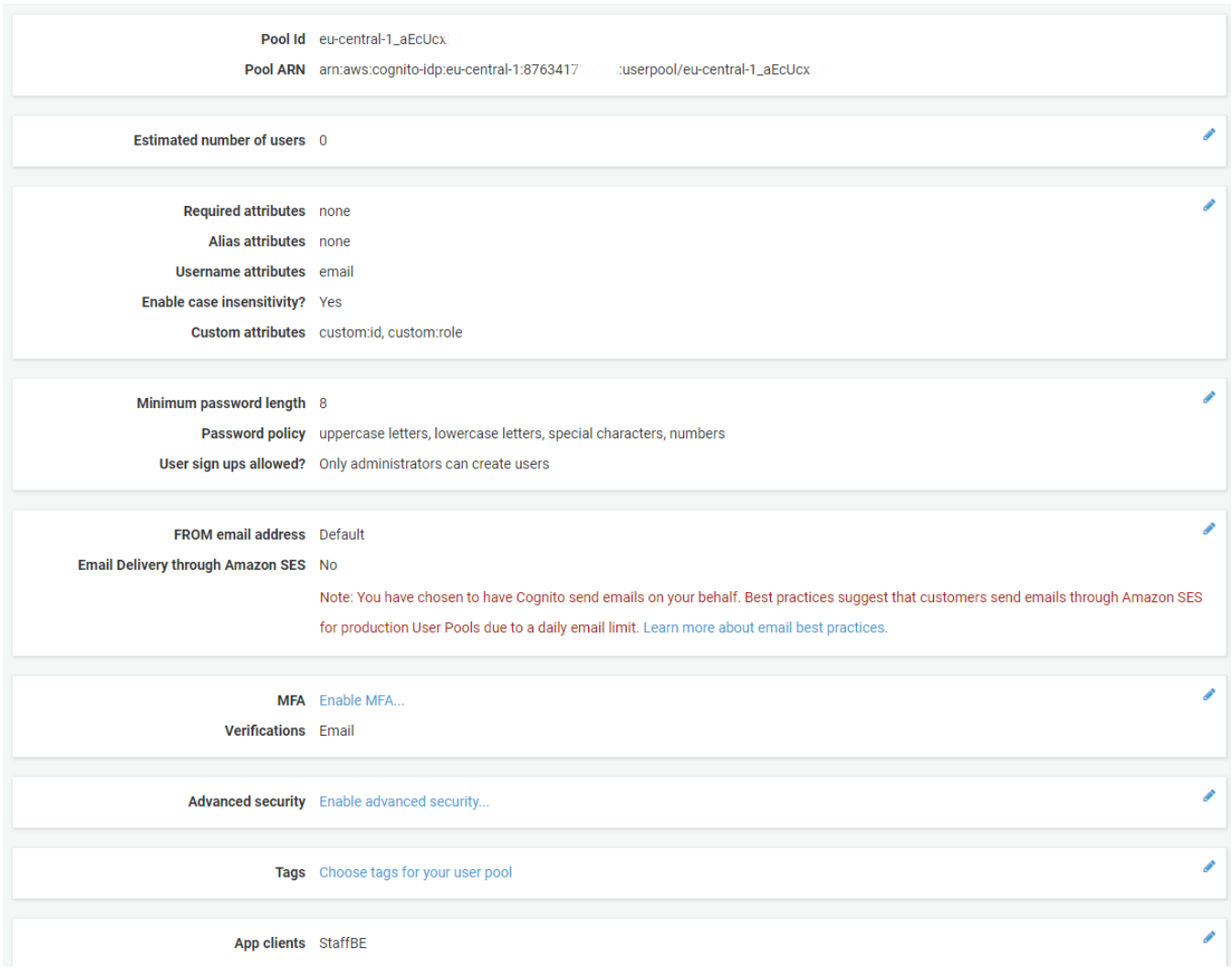

Рисунок 4.8 Зразок налаштування User Pool Cognito

5 ЕКОНОМІЧНА ЧАСТИНА

Проект, який розкривається у цієї роботі є архітектурою програмного застосунка, який розгорнутий у Amazon Web Services. Для технічного забезпечення розробки необхідно використовувати певні обчислювані засоби та деякі потрібні ліцензії на деяке програмне забезпечення.

При оцінці фінансових затрат та визначенні результатів було виявлено такі основні розділи:

- щомісячні затрати на підтримку хмарної інфраструктури
- заробітна плата співробітникам
- витрати на певні засоби розробки
- сплата за програмне забезпечення

5.1 Щомісячні затрати на підтримку хмарної інфраструктури

У Amazon Web Services кожен ресурс має свою ціну певно використанню.

Таблиця 5.1 – Витрати за хмарні ресурси

Тип ресурсу	Щомісячні затрати, $
EC2 – FrontEnd	140.16 (35.04 x 4)
EC2 – BackEnd	420 (140 x 3)
EC2 – Jenkins	9.78
ElasticSearch – Total	244.8 (122.5 x 2)
EBS – Total	15.47 (130 GiB x 0.119)
S3 – Total	1.55
RDS – Total	140.16 (70.08 x 2)

Загальна сума витрат за сервіси Amazon Web Services становить $971.92 (25 755,88 грн за курсом 26.5) щомісячно. За 3 місяці сума становить 77 267,64 грн.

5.2 Заробітна плата співробітникам

Над хмарною частиною застосунку працював один інженер, який був ще й тестувальником інфраструктури. Над програмною частиною застосунку працювало два програміста. На розробку проекту було витрачено 3 місяці, п’яти добовий робочий тиждень. Співробітники працювали 8 годин на добу, державні празники оплачувались і були робочими з стандартною ставкою.

Місячний оклад всіх працівників дорівнює 14 000 гривень.

Витрати на заробітну плату розраховуються за формулою:

$$З = (М * О) * К$$

Де:

М – кількість місяців

О – місячний посадковий оклад

К – кількість робітників

Розрахунок:

$$З = (3 * 14\,000) * 3 = 126\,000$$

Також від загальної суми заробітної плати потрібно виділити 14% на додаткові виплати та 22% на соціальне страхування. Кінцевий результат розрахунку витрат на заробітну плату робітникам представлено в таблиці 5.2.

Таблиця 5.2 – Витрати на заробітну плату розробникам

Тип витрат	Сума витрат (грн)
Сума заробітної плати за 3 місяці	126 000
Додаткові виплати та премії (14%)	17 640

Продовження таблиці 5.2

	Разом: 143 640
Соціальне страхування (22%)	31 600,80
	Загальна сума заробітної плати: 175 240, 80

5.3 Витрати на певні засоби розробки

Для розробки програмного забезпечення необхідно придбати робочу станцію. Так як при оренді на 2 місяці кошти витрачені на обладнання будуть становити, майже повну ціну комп'ютеру чи ноутбука, з точки зору економічної ефективності, обладнання потрібно купувати. В ході огляду ринку було обрано 3 ноутбука, основні параметри яких представлені в таблиці 5.3.

Таблиця 5.3 – Параметри обраного обладнання

№	Модель обладнання	Основні параметри	Кількість, шт	Ціна (грн)
1	Ноутбук LENOVO IdeaPad 330-15IKB (81DC009WRA)	Екран 15.6 "TN Full HD, / Intel Core i5-7200U / RAM 8 ГБ / HDD 1 TB / NVIDIA GeForce MX110	1	11 429
2	Ноутбук Lenovo V155-15API (81V5000CRA) Iron Gray	Екран 15.6 "TN Full HD, / AMD Ryzen 5 3500U / RAM 8 ГБ / SSD 256 ГБ / AMD Radeon Vega 8	2	18 012
				Загальна сума: 47 453

5.4 Сплата за програмне забезпечення

Усім розробникам потрібна IDE (Інтегроване середовище розробки). Усім розробникам підходить IntelliJ IDEA від JetBrains. Щомісячна вартість за це програмне забезпечення описано у таблиці 5.4.

Таблиця 5.4 – Вартість ПЗ

Постачальник ПЗ	Назва ПЗ	Кількість	Вартість, $
JetBrains.com	IntelliJ IDEA Ultimate	3	14.90
			Загальна вартість: 44,70

За 3 місяці сума сплати за програмне забезпечення становить 3 553,65 грн (за курсом 26.5)

Сумарні витрати на розробку програмного продукту та хмарної архітектури за 3 місяці у таблиці 5.5.

Таблиця 5.5 – Сумарні затрати за розробку

Тип затрат	Вартість, грн
Вартість AWS	77 267,64
Заробітна плата	175 240,80
Покупка обладнання	47 453
Програмне забезпечення	3 553,65
	Загальна сума: 303 515,09

Таким чином, за комплексну розробку трьорівневої архітектури, бекенду та фронтенду було витрачено 303 515,79 грн, що є дуже вигідним вкладенням для компанії щодо вартості інших готових рішень на дистанції.

6 ОХОРОНА ПРАЦІ

Виробниче середовище впливає на здоров'я робітника шкідливими і небезпечними факторами. Шкідливий виробничий фактор – це виробничий фактор, вплив якого на працівника може привести його до захворювання. Небезпечний виробничий фактор – це виробничий фактор, вплив якого на працівника може привести до його травми.

В кваліфікаційній роботі розглядається побудова комп'ютерної системи для додатку з керування персоналом. Об'єктом аналізу небезпечних і шкідливих факторів взято робоче місце розробника.

6.1 Аналіз шкідливих і небезпечних вражаючих факторів

У приміщенні з робочим місцем розробника на нього можуть впливати небезпечні і шкідливі виробничі фактори, які вказані в таблиці 6.1.

Таблиця 6.1 – Аналіз шкідливих і небезпечних вражаючих факторів

№ п/п	Найменування ШНВФ	Джерела ШНВФ	Норми
1	Можливість ураження електричним струмом	Електропроводка, блок живлення персонального комп'ютера	ДЕСТ 12.1.038-82.Електробезпека. Гранично допустимі значення напруг

			дотику і струмів
2	Тепловиділення від устаткування	Персональний комп'ютер	СанПиН 2.2.4.548-96

Продовження таблиці 6.1

3	Недостатня освітленість робочої зони	Робоче місце	СНіП II-4-79
4	Специфічний характер зорової роботи; вимушена локалізована поза	Монітор, робоче місце оператора ПЕОМ	СанПин 2.2.2/2.4.1340-03

6.2 Інженерно-технічні заходи з охорони праці

Відповідно до класифікації ПУЕ за небезпекою ураження електричним струмом приміщення операторів ПК відноситься до приміщень з підвищеною небезпекою, так як існує можливість одночасного дотику до опалювальних батарей приміщення, з'єднаних з землею і корпусів електрообладнання. В операторській використовується обладнання з напругою живлення 220 В.

Лінія електромережі для живлення ПК та периферійного обладнання виконується як окрема групова трьохпровідна мережу шляхом прокладки фазного і нульового робочого та захисного провідників. Нульовий захисний провідник служить для занулення електроприймачів. При напрузі до 1000 В застосовують трьохпровідну мережу з ізольованою нейтраллю.

Електропроводка в операторській повинна бути виконана прихованим методом, прокладена в гнучких металоруковах, що робить силові ланцюги недоступними для працюючих.

Струмовий захист реалізується з використанням автоматів, які розривають електричну мережу при високих струмах навантаження. Для забезпечення захисного відключення використовуємо УЗО ВД1-63, основним призначенням якого є забезпечення безпеки людини в разі дотику до занулення (заземленому) корпусу при замиканні на нього фази, а також при безпосередньому дотику до струмоведучих частини електроустановки.

Основні заходи, спрямовані на попередження випадків ураження електричним струмом в операторській, такі:

− щодня проводити очистку монітора від пилу;

− забороняється знімати захисну кришку системного блоку комп'ютера;

− усунення можливості випадкового дотику до струмоведучих частин електроустаткування, що знаходиться під напругою;

− малій напруги;

− надійна ізоляція струмоведучих частин електрообладнання і своєчасний його ремонт;

− захисне занулення;

− захисне відключення та застосування плавких запобіжників.

При виконанні робіт, пов'язаних з нервово-емоційним напруженням у приміщені з робочим місцем розробника повинні дотримуватися оптимальні умови мікроклімату (температура повітря 22 - 24 °С, відносна вологість 60 - 40%, швидкість руху повітря не більше 0,1 м/сек.).

Кімната повинна бути обладнана власною системою вентиляції та кондиціонування на базі спліт-системи LG A12LHR продуктивністю 510 $м^3$/год, яка стабілізує температуру повітря в регульованих межах 14 ... 32 ± 2°С.

В приміщенні з робочим місцем використовується поєднане природне і штучне освітлення. Згідно СНиП 23.05- 95 – середня точність зорової роботи, найменший розмір об'єкта розрізнення складає 0,3÷0,5 мм.

Штучне освітлення може бути двох систем – загальне і комбіноване.
Для освітлення приміщення використані, найбільш економічні люмінесцентні лампи типу ЛБ. Для місцевого освітлення використані лампи розжарювання.
На робочому місці відсутні різкі тіні.

Для внутрішнього оздоблення інтер'єру приміщень з ПК використані диффузионно-відбивні матеріали з коефіцієнтами відбиття світла для стелі 0,7÷0,8; для стін 0,5÷0,6; для підлоги 0,3÷0,5.
Розробники за характером роботи багато часу проводять з ПК. В даному випадку працівник схильний до впливу напруги зору. Це призводить до підвищеного стомлення зору і загального стомлення.

Для безпечної і комфортної роботи при експлуатації персонального комп'ютера розроблені наступні заходи:

1. Площа на одне робоче місце має становити не менше 6 м^2, об'єм – не менше 20 м^3.
2. Висота робочої поверхні для монітора повинна становити 680-800 мм (рекомендовані розміри столу: висота - 725 мм, ширина - 600-1400 мм, глибина - 800-1000 мм).
3. Робоче сидіння працівника має складатися з: сидіння, спинки і підлокітників.
4. Монітор і клавіатура повинні бути розміщені на поверхні столу або на спеціальній робочій поверхні окремо від столу, яка повинна бути відрегульована по висоті, на відстані 100-300мм від краю.

5. Щоб освітлення не створювало сліпучих відблисків, комп'ютер повинен бути розташований так, щоб пряме світло не попадало на екран.

6. Верхній край екрана слід розташовувати на рівні очей або трохи нижче.

7. Оптимальна відстань від очей до екрана 600-700 мм.

8. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5;

9. Вологе прибирання повинне проводитися на початку робочого дня, а також під час перерви.

6.3 Заходи з ергономіки

Проектування робочого місця оператора ПК має супроводжуватися прагненням поліпшити обстановку, яка буде сприяти збереженню високої працездатності, і створювати сприятливі умови для співпраці працівників.

Стандарт повинен обумовлювати те, що робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам.

Проектування робочих місць, забезпечених моніторами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки. При конструюванні робочих місць необхідно отримуватися наступні основні умови:

а) достатній робочий простір для оператора, що дозволяє здійснювати всі необхідні рухи і переміщення при експлуатації та технічному обслуговуванні обладнання;

б) достатні фізичні, зорові і слухові зв'язки між операторами та обладнанням, а також між операторами;

в) оптимальне розміщення робочих місць у приміщеннях для оперативної роботи, а також безпечні і достатні проходи для операторів;

г) оптимальне розміщення устаткування (головним чином засобів відображення інформації й органів керування), завдяки чому забезпечується зручне положення оператора при роботі;

д) чітке позначення органів керування, індикаторів та інших елементів обладнання, які потрібно знаходити, ідентифікувати і якими доводиться маніпулювати (маркування не є обов'язковим для органів управління або обладнання, призначення яких очевидно для оператора);

е) необхідне природне і штучне освітлення для виконання оперативних завдань,

технічного обслуговування" або тренувань;

ж) припустимий рівень акустичного шуму і вібрації, створюваних устаткуванням робочого місця або іншими джерелами шуму і вібрації;

з) достатню простоту і швидкість збірки і розбирання устаткування;

и) виключення можливості неправильної установки, заміни та монтажу блоків або елементів обладнання, а також неправильної ідентифікації, орієнтації і розташування кабелів і роз'ємів;

к) наявність необхідних інструкцій і попереджувальних знаків, застережливих про небезпеки, які можуть виникнути при роботі, і вказують на необхідні заходи безпеки;

л) необхідні опори і підставки для тимчасового розміщення вийнятих блоків або елементів обладнання, а також для випробувального обладнання, приладів, інструментів і технічних посібників;

м) надійну індикацію для випадків відмови електричного живлення, а також відмови обладнання або його функціонування з виходом за допустимі межі.

Згідно з ГОСТ 12.2.032-78 висота робочої поверхні при організації робочого місця повинна бути:

а) Для жінок – 630 мм.

б) Для чоловіків 680 мм.

Висота сидіння:

а) Для жінок – 400 мм.

б) Для чоловіків 430 мм.

Підставка для ніг повинна бути регульованою по висоті. Ширина повинна бути не менше 300 мм, довжина - не менше 400 мм. Поверхня підставки повинна бути рифленою. По передньому краю слід передбачати бортик висотою 10 мм

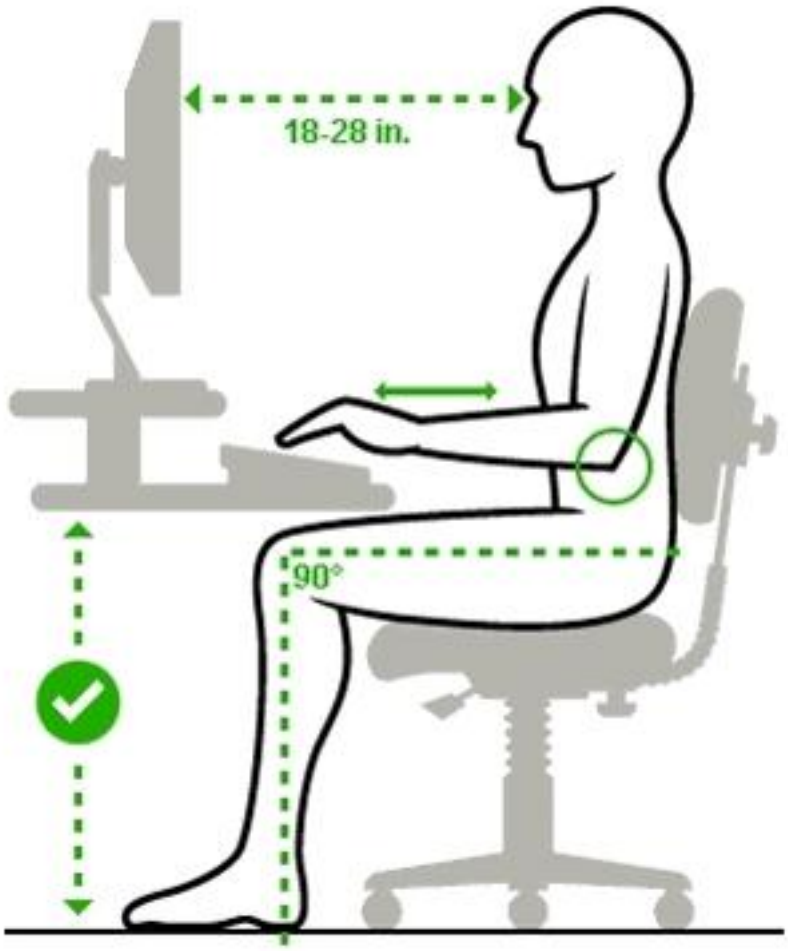

Рисунок 6.1 – Положення робочого місця

ВИСНОВКИ

В рамках кваліфікаційної роботи бакалавра було досліджено сучасний ринок обчислюваних потужностей та зокрема провайдерів, які надають послуги хмарного обчислювання. Проаналізовані типи сервісів, існуючи на ринку, їх характеристики та переваги. Основні критерії дослідження – це високий рівень доступності системи, відмовостійкість усіх компонентів та гнучкість налаштування основних ресурсів комп'ютерної системи. Також було розглянуто потребу використання засобів опису інфраструктури як коду.

У результаті аналізу сучасних провайдерів інфраструктури як сервісу було виявлено, що Amazon Web Services відповідає усім критеріям та має дуже великий вибір інструментів для гнучкого налаштування додатків. Також AWS має можливість створити внутрішню мережу для майже усіх ресурсів, що добре позначається на безпеці системи.

Для розробки інфраструктури як коду було обране рішення від компанії Hashicorp – Terraform. За допомогою цього інструменту можливо кратне прискорення внесення змін у існуючу комп'ютерну систему.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дипломування. Методичні вказівки для бакалаврів галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, С.М. Ткаченко, Я.В. Панферова, Д.О. Бешта, Л.В. Бешта. – Д.: НТУ «Дніпровська політехніка», 2020. – 69 с.
2. Методичні вказівки до виконання розділу „Охорона праці" в дипломних проектах (роботах) бакалаврів інституту електроенергетики / В.І. Голінько, В.Ю. Фрундін, Ю.І. Чеберячко, М.Ю. Іконніков. – Д.: Державний ВНЗ «Національний гірничий університет», 2012. – 8 с.
3. AWS Certified Solutions Architect Official Study Guide: Associate Exam, by Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut, Kevin E. Kelly, Sean Senior, John Stamper, 2017. – 437 с.
4. Дослідження SRGResearch [Електронний ресурс] - https://www.srgresearch.com/articles/incremental-growth-cloud-spending-hits-new-high-while-amazon-and-microsoft-maintain-clear-lead-reno-nv-february-4-2020
5. Jenkins [Електронний ресурс] - https://www.jenkins.io/doc/book/
6. Terraform [Електронний ресурс] - https://www.terraform.io/docs/
7. AWS VPC [Електронний ресурс] - https://aws.amazon.com/blogs/apn/amazon-vpc-for-on-premises-network-engineers-part-one/
8. RFC 1918 [Електронний ресурс] - https://tools.ietf.org/html/rfc1918
9. AWS Regions & AZs [Електронний ресурс] - https://aws.amazon.com/about-aws/global-infrastructure/

10. Amazon resources [Електронний ресурс] - https://github.com/open-guides/og-aws

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

СХЕМИ
ТРЬОХ РІВНІВ ХМАРОВОЇ ІНФРАСТРУКТУРИ

Листів 4

2020

АНОТАЦІЯ

Дані схеми візуально представляють логіку мережевого налаштування інфраструктури. Також візуалізовано використання додаткових сервісів Amazon Web Services. Описано, як мережевий трафік потрапляє у різні рівні додатку.

ЗМІСТ

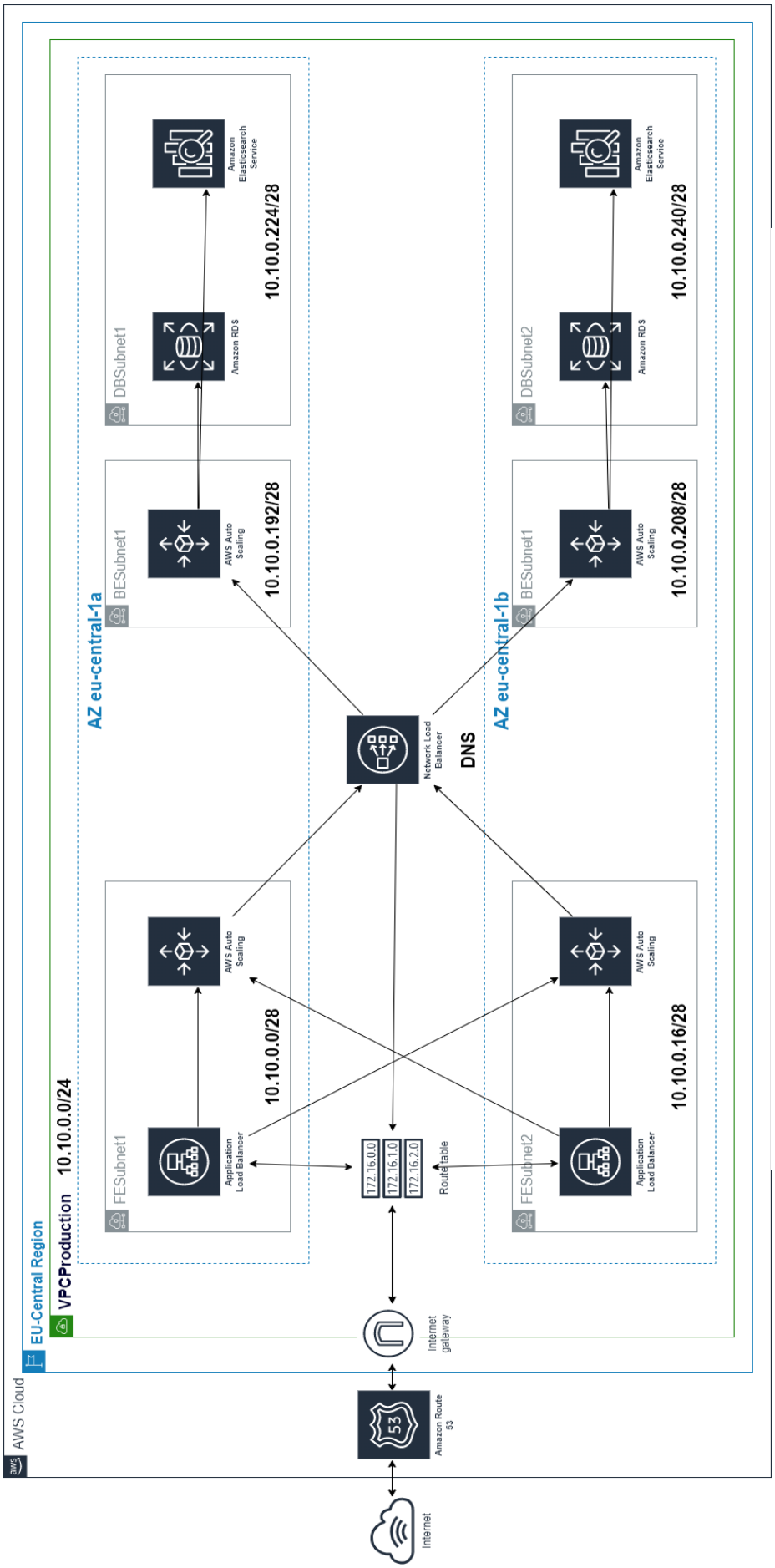

Рисунок А1 – Схема мережевого налаштування інфраструктури

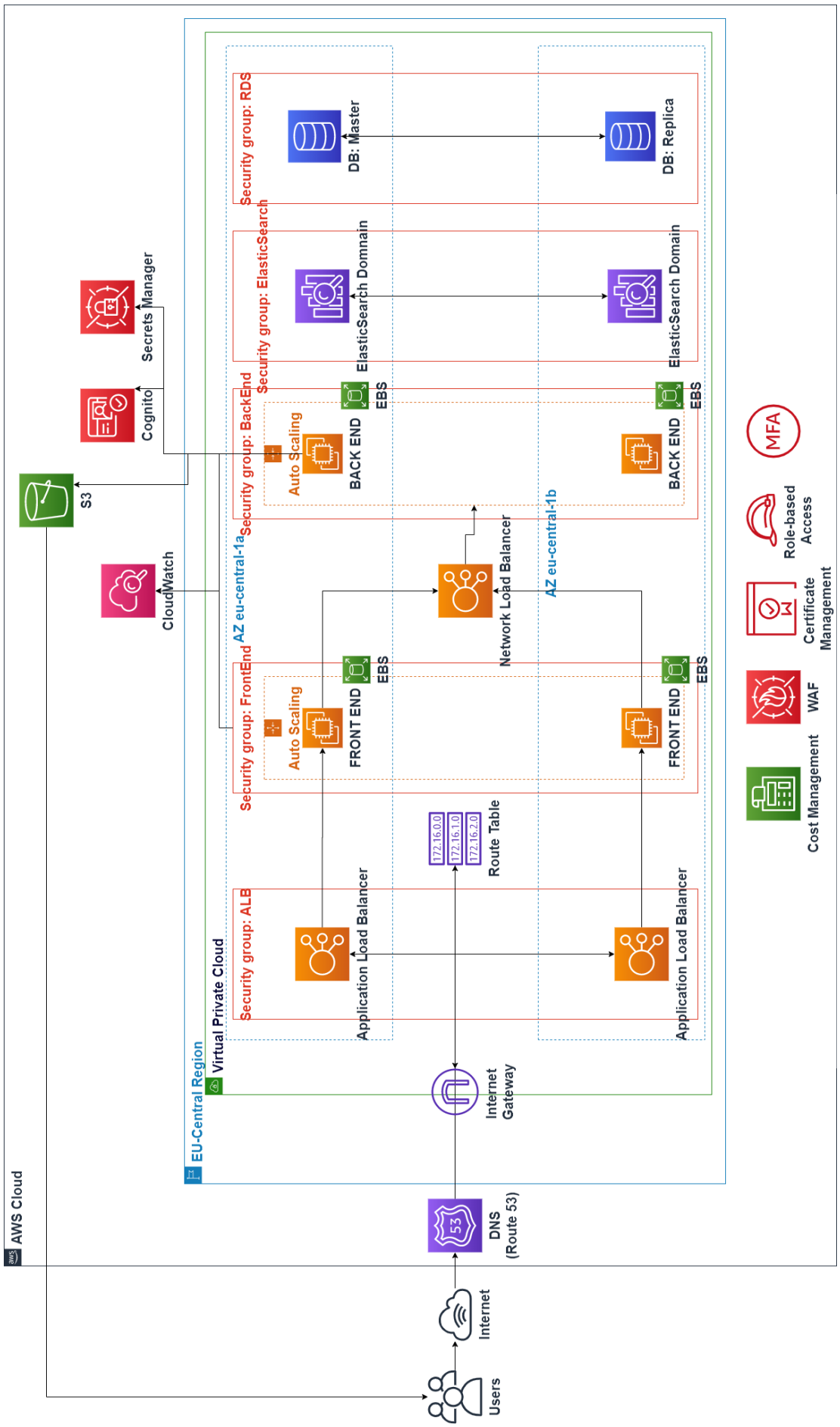

Рисунок А2 – Схема апаратного налаштування інфраструктури

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ЛІСТІНГ
ІНФРАСТРУКТУРИ ЯК КОДУ

Листів 12

2020

АНОТАЦІЯ

Дана інфраструктура як код містить в собі частину автоматичного налаштування Terrform для подальшої праці з хмарою. Також описано налаштування деяких компонентів Amazon Web Services, наприклад, мережевих: Virtual Private Cloud, Subnets, Internet Gateway; апаратних: Auto Scaling Group (EC2).

ЗМІСТ

2 Загальне налаштування Terraform та ініціалізація модулів

```
provider "aws" {
  region = var.aws_region
  shared_credentials_file = var.credentials_path
  profile = var.aws_profile_name
}

terraform {
  backend "s3" {
    bucket = "s3backend-for-terraform"
    key = "terraform.tfstate"
    region = "eu-central-1"
  }
}

module "network" {
  source = "./modules/network"
  vpc_cidr_block = var.vpc_cidr_block
}

module "database" {
  source = "./modules/database"
  vpc_id = module.network.vpc_id
  be_subnet_cidr = module.network.be_subnet_cidr
  db_subnet_cidr1 = module.network.db_subnet_cidr1
  db_subnet_cidr2 = module.network.db_subnet_cidr2
  db_subnet_id1 = module.network.db_subnet_id1
  db_subnet_id2 = module.network.db_subnet_id2
}
```

```
module "elasticsearch" {
  source = "./modules/elasticsearch"
  vpc_id = module.network.vpc_id
  be_subnet_cidr = module.network.be_subnet_cidr
  db_subnet_id1 = module.network.db_subnet_id1
}

module "frontend" {
  source = "./modules/frontend"
  ec2_ip = module.backend.ec2_ip
  vpc_id = module.network.vpc_id
  fe_subnet_id = module.network.fe_subnet_id
  vpc_cidr_block = var.vpc_cidr_block
  fe_subnet_cidr = module.network.fe_subnet_cidr
  fe_subnet_id2 = module.network.fe_subnet_id2
}
module "backend" {
  source = "./modules/backend"
  vpc_id = module.network.vpc_id
  vpc_cidr_block = var.vpc_cidr_block
  be_subnet_id = module.network.be_subnet_id
  fe_lb_dns = module.frontend.fe_lb_dns
  be_subnet_cidr = module.network.be_subnet_cidr
  fe_subnet_cidr = module.network.fe_subnet_cidr
  db_subnet_cidr1 = module.network.db_subnet_cidr1
  db_subnet_cidr2 = module.network.db_subnet_cidr2
  es_endpoint = module.elasticsearch.es_endpoint
  es_sg_id = module.elasticsearch.es_sg_id
}
```

3 Модуль налаштування мережі та компонентів VPC

```
resource "aws_vpc" "VPC" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "VPCProduction"
  }
}

resource "aws_subnet" "FrontEndSubnet" {
  depends_on = [aws_vpc.VPC]
  vpc_id = aws_vpc.VPC.id
  cidr_block = "10.10.0.0/28"
  availability_zone = "eu-central-1a"
  map_public_ip_on_launch = false
  tags = {
    Name = "FrontEndSubnet"
  }
}
resource "aws_subnet" "FrontEndSubnet2" {
  depends_on = [aws_vpc.VPC]
  vpc_id = aws_vpc.VPC.id
  cidr_block = "10.10.0.16/28"
  availability_zone = "eu-central-1b"
  map_public_ip_on_launch = false
  tags = {
    Name = "FrontEndSubnet"
  }
}

resource "aws_subnet" "BackEndSubnet" {
  depends_on = [aws_vpc.VPC]
```

```
  vpc_id = aws_vpc.VPC.id
  cidr_block = "10.10.0.192/28"
  availability_zone = "eu-central-1a"
  map_public_ip_on_launch = false
  tags = {
    Name = "BackEndSubnet"
  }
}

resource "aws_subnet" "DBSubnet1" {
  depends_on = [aws_vpc.VPC]
  vpc_id = aws_vpc.VPC.id
  cidr_block = "10.10.0.224/28"
  availability_zone = "eu-central-1a"
  map_public_ip_on_launch = false
  tags = {
    Name = "DBSubnet1"
  }
}
resource "aws_subnet" "DBSubnet2" {
  depends_on = [aws_vpc.VPC]
  vpc_id = aws_vpc.VPC.id
  cidr_block = "10.10.0.240/28"
  availability_zone = "eu-central-1b"
  map_public_ip_on_launch = false
  tags = {
    Name = "DBSubnet2"
  }
}

resource "aws_internet_gateway" "InternetGateway" {
  depends_on = [aws_vpc.VPC]
```

```
  vpc_id = aws_vpc.VPC.id
  tags = {
    Name = "IGProduction"
  }
}

resource "aws_route_table" "RT" {
  vpc_id = aws_vpc.VPC.id
  tags = {
    Name = "RTProduction"
  }
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.InternetGateway.id
  }
}

resource "aws_route_table_association" "RTAFE1" {
  route_table_id = aws_route_table.RT.id
  subnet_id = aws_subnet.FrontEndSubnet.id
}
resource "aws_route_table_association" "RTAFE2" {
  route_table_id = aws_route_table.RT.id
  subnet_id = aws_subnet.FrontEndSubnet2.id
}

resource "aws_route_table_association" "RTABE1" {
  route_table_id = aws_route_table.RT.id
  subnet_id = aws_subnet.BackEndSubnet.id
}

output "vpc_id" {
```

```
  value = aws_vpc.VPC.id
}

output "fe_subnet_id" {
  value = aws_subnet.FrontEndSubnet.id
}
output "fe_subnet_id2" {
  value = aws_subnet.FrontEndSubnet2.id
}
output "fe_subnet_cidr" {
  value = aws_subnet.FrontEndSubnet.cidr_block
}

output "be_subnet_id" {
  value = aws_subnet.BackEndSubnet.id
}
output "be_subnet_cidr" {
  value = aws_subnet.BackEndSubnet.cidr_block
}

output "db_subnet_id1" {
  value = aws_subnet.DBSubnet1.id
}
output "db_subnet_id2" {
  value = aws_subnet.DBSubnet2.id
}
output "db_subnet_cidr1" {
  value = aws_subnet.DBSubnet1.cidr_block
}
output "db_subnet_cidr2" {
  value = aws_subnet.DBSubnet2.cidr_block
}
```

```
variable "vpc_cidr_block" {}
```

4 Модуль налаштування Auto Scaling Group для Front End

```
resource "aws_instance" "FrontEndEC2" {
  ami = "ami-0e342d72b12109f91"
  instance_type = "t2.micro"

  associate_public_ip_address = false
  key_name = "diploma-dev"
  security_groups = [aws_security_group.FrontEndSG.id]
  subnet_id = var.fe_subnet_id
  user_data = <<-USERDATA
#!/bin/bash
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
apt-get update
apt-get install nginx awscli yarn nodejs -y
aws configure set aws_access_key_id ${ACCESS_KEY_ID}
aws configure set aws_secret_access_key ${SECRET_ACCESS_KEY_ID}
aws s3api get-object --bucket staff-mgmt-fe-artifacts --key stuff-management-
master.tar.gz ./stuff-management-master.tar.gz
tar xf stuff-management-master.tar.gz -C ./
cd ./stuff-management-master
touch .env
echo "`cat <<EOF
REACT_APP_VERSION=${APP_VERSION}
REACT_APP_API_URL=http://${var.ec2_ip}:1234/staff
EOF`" > .env
yarn install
yarn build
```

```
mv build/* /var/www/html/
rm /var/www/html/index.nginx-debian.html
sed -i 's/try_files \$uri \$uri\/ =404;/try_files \$uri \/index.html;/' /etc/nginx/sites-available/default
service nginx restart
USERDATA
  tags = {
    Name = "FrontEndEC2"
  }
}

output "fe_lb_dns" {
  value = aws_lb.FEApplicationLoadBalancer.dns_name
}

resource "aws_security_group" "FrontEndSG" {
  name = "FrontEndSG"
  vpc_id = var.vpc_id
  ingress {
    description = "Accept all from local network"
    from_port = 0
    protocol = "-1"
    to_port = 0
    cidr_blocks = [var.vpc_cidr_block]
  }
}
```

5 Модуль налаштування Application Load Balancer

```
resource "aws_lb" "FEApplicationLoadBalancer" {
  name = "FELoadBalancer"
  internal = false
  load_balancer_type = "application"
```

```
  security_groups = [aws_security_group.ELBSG.id]
  subnets = [var.fe_subnet_id, var.fe_subnet_id2]
  idle_timeout = 120
}
resource "aws_lb_target_group" "FEALBTargetGroup" {
  name = "FEALBTargetGroup"
  port = 80
  protocol = "HTTP"
  target_type = "instance"
  slow_start = 100
  vpc_id = var.vpc_id
  health_check {
    healthy_threshold = 3
    unhealthy_threshold = 3
    timeout = 10
    path = "/"
    matcher = "200-399"
  }
}

resource "aws_lb_listener" "FEALBListener" {
  load_balancer_arn = aws_lb.FEApplicationLoadBalancer.arn
  port = 80
  protocol = "HTTP"
  default_action {
    type = "forward"
    target_group_arn = aws_lb_target_group.FEALBTargetGroup.arn
  }
}
resource "aws_alb_target_group_attachment" "FEEC2Target" {
  target_group_arn = aws_lb_target_group.FEALBTargetGroup.arn
  target_id = aws_instance.FrontEndEC2.id
```

```
  port = 80
}
```