

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ
Кафедра мережних технологій

КУРСОВА РОБОТА

на тему:

**«Побудова багаторівневого веб-застосування
на платформі Amazon Web Services(AWS)»**

Виконав:

Студент 3 року навчання

Бакалаврської програми

Бойцов С. В.

Керівник курсової роботи:

Черкасов Д. І.,

Кандидат технічних наук

Київ 2020

Міністерство освіти і науки України
 НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
 Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мережних технологій,
 проф., д.ф.-м.н.

_____ Г. І. Малашонок
 (підпис)

„_____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
 на курсову роботу

студенту Бойцову С.В. факультету ФІ 3 курсу
 ТЕМА Побудова багаторівневого веб-застосування
 на платформі Amazon Web Services

Вихідні дані:

- Конфігурація для розгортання застосунку

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

Огляд існуючих рішень

Структурна розробка власного рішення

Детальна розробка компонента

Висновки

Джерела

Додатки

Дата видачі „_____” _____ 2020 р. Керівник _____
 (підпис)

Завдання отримав _____
 (підпис)

Календарний план виконання курсової роботи

Тема: «Побудова багаторівневого веб-застосування на платформі Amazon Web Services(AWS)»

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	19.10.2019	
2.	Огляд технічної літератури за темою роботи.	02.12.2019	
3.	Огляд існуючих рішень	25.01.2020	
3.	Структурна розробка власного рішення	30.03.2020	
4.	Детальна розробка власного рішення	15.04.2020	
5.	Остаточне оформлення презентації	11.05.2020	
6.	Захист курсової роботи (проекту)	20.05.2020	

Студент _____

Керівник _____

“ ”

Зміст

1. Анотація	5
2. Вступ	6
3. Огляд існуючих рішень	7
3.1. Однорівневі застосування.....	7
3.2. Оренда дата-центру з розміщенням фізичного обладнання	10
3.3. Розміщення на власних ресурсах	11
3.4. Розміщення на хмарних ресурсах	12
3.5. 1-, 2-, 3-, ... рівневі застосування.....	9
4. Структурна розробка застосунку	13
4.1. Архітектура	13
4.2. Компоненти	14
4.3. Опис функціонування системи	15
5. Структурна розробка утиліти	16
6. Детальна розробка застосунку	17
7. Детальна розробка утиліти.....	19
8. Висновки	20
9. Джерела.....	21

1. Анотація

Метою роботи є дослідження та розробка багаторівневого веб-застосування, що дозволяє користувачам змінювати самостійно вміст сторінок через браузер, використовуючи спрощену і зручнішу, порівняно з HTML, розмітку тексту орієнтовану на розгортання в хмарі; та утиліти яка допомагала б розгорнути цей застосунок в хмарі.

Дана робота складається з 5 розділів:

У першому було розглянуто існуючі рішення. У другому була проведена структурна розробка застосунку. У третьому була проведена структурна розробка застосунку. У четвертому була описана детальна розробка застосунку. У п'ятому була описана детальна розробка утиліти.

2. Вступ

Тема роботи: «Побудова багаторівневого веб-застосування на платформі Amazon Web Services». Для цієї роботи, у ролі веб-застосування було обрано систему керування контентом, а саме вікі.

Вікі — вебсайт, що дозволяє користувачам змінювати самостійно вміст сторінок через браузер, використовуючи спрощену і зручнішу, порівняно з HTML, вікі-розмітку тексту.

На сьогодні існує більше 100 різних вікі сервісів, з різними видами оплати, монетизації, ліцензіями, редагування стилів, підтримки багатомовності, хостингу та іншого функціоналу. [1]

На AWS Marketplace є присутні декілька образів, після запуску одного з яких, вже можна було б користуватися власною вікі. Проте, ці образи надають нам лише окрему машину, тобто застосунок який є фактично 1-рівневим за структурою.

У n-рівневих же застосунків є переваги у масштабованості, конфігурованості, надійності, рівні безпеки та інші.

Виходячи з попередніх міркувань за мету даної роботи було взято розробка конфігурацій веб-застосунку на основі одного із доступних сервісів, використавши широкий функціонал AWS для розподілення цього застосунку на декілька рівнів. Ще одним елементом розробки було обрано програму, яка збирає інформацію про потреби користувача і формує файли конфігурації для автоматизованого розгортання.

3. Огляд існуючих рішень

3.1.Архітектури мережеских застосувань

3.1.1. Однорівневі застосування

Однорівнева архітектура передбачає розміщення всіх необхідних компонентів для програмного додатку (як «бекенду», так і «фронт-енду») на одному сервері (схема на Рис. 1 – Схема однорівневого застосування). Це хороший спосіб перевірити свою програму в середовищі розробки, і це ідеальне рішення для невеликих сайтів з низьким попитом на трафік, які потребують ефективного використання ресурсів. Плюс також є зручність керувати та підтримки і економічність.

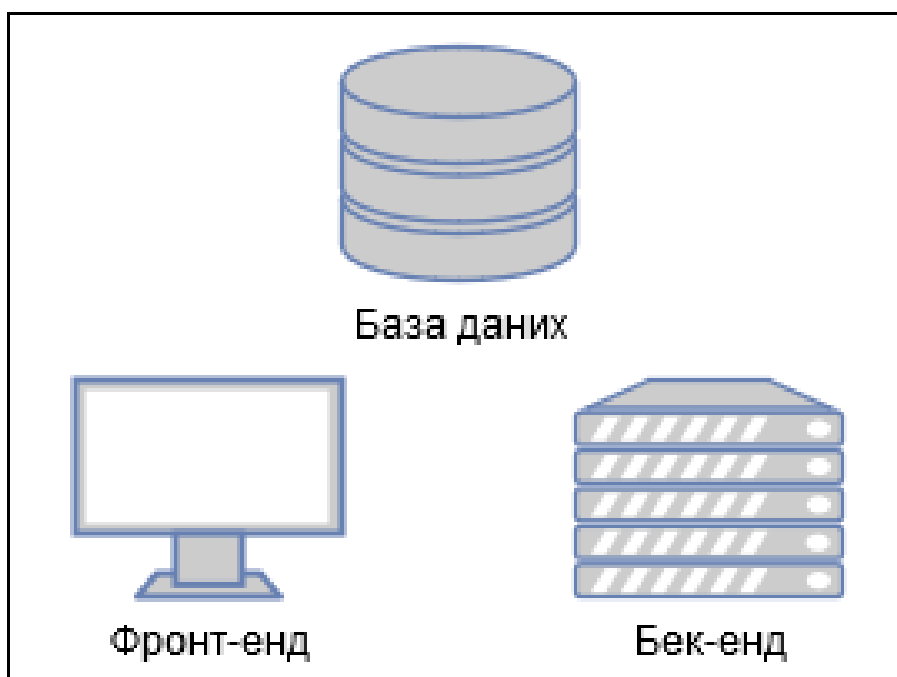


Рис. 1 – Схема однорівневого застосування

Водночас наявність усіх ресурсів на одній машині може створити ризик доступності та безпеки. Якщо сервер не працює, застосування теж не працюватиме, і не зможе спілкуватися з базою даних. Якщо сервер піддається атаці зовні, є ризик втратити дані, якщо немає репліки бази даних.[2]

3.1.2. 1-, 2-, 3-, ... рівневі застосування

«N»-рівневе застосування - це застосування, яка розподіляється між декількома окремими комп'ютерами в розподіленій мережі (n позначає кількість рівнів).

1-рівневе застосування найпростіше, оскільки він еквівалентний запуску програми на персональному комп'ютері. Усі необхідні компоненти для запуску програми знаходяться на одній програмі або сервері. Рівень презентації, рівень бізнес-логіки та рівень даних розташовані на одній машині.

2-рівневе застосування по суті є архітектурою клієнт-сервер, де спілкування відбувається між клієнтом і сервером. У цьому типі архітектури програмного забезпечення рівень шару презентації або користувальницький інтерфейс працює на стороні клієнта, тоді як рівень набору даних виконується і зберігається на стороні сервера. Між клієнтом і сервером немає шару бізнес-логіки.

Найпоширенішою формою n-рівневої структури є 3-рівневий додаток, що поділяється на три рівні:

- Програмування інтерфейсу користувача на комп'ютері користувача
- Бізнес-логіка на більш централізованому комп'ютері
- Необхідні дані в комп'ютері, який управляє базою даних.

4 та більше-рівневі застосування розділяють основні рівні на під-рівні або додають нові за необхідністю.

Зі збільшенням кількості рівнів зростає складність розробки і заплутаність застосунку. Водночас покращується масштабованість, цілісність даних, універсальність, поліпшується безпека і доступність. [6]

3.2. Платформи для розгортань мережевих застосунків

3.2.1. Оренда дата-центру з розміщенням фізичного обладнання

Оренда центрів обробки даних - це послуга, що надається компаніями, які пропонують спільний захищений простір підприємствам для зберігання апаратних засобів, пов'язаних із зберіганням даних та іншим обладнанням.

Замовник підприємства взаємодіє з компанією оренди схожим чином, до відносин з менеджером оренди майна, при якому замовник орендує простір в приміщенні для зберігання обладнання.

Замовник зазвичай постачає обладнання: сервери та інше обладнання, необхідне для щоденних операцій. А компанія-орендатор зберігає це обладнання надійно в охолоджуваному середовищі під наглядом, забезпечуючи задоволення потреб у пропускну здатності. Центр обробки даних пропонує рівні обслуговування, які гарантують майже постійний час роботи.

Однією з переваг оренди є час роботи сервера. Клієнтові гарантується певний відсоток тривалості роботи без витрат на оплату праці для обслуговування або інших зборів за обслуговування. Підприємства, які використовують центри обробки даних як стратегію відновлення даних, роблять це для зменшення ризику у випадку «зовнішньої» події (наприклад, стихійного лиха) або відключення. Використання об'єкта оренди забезпечує безперервність бізнесу у випадку катастрофи.

Незважаючи на те, що аутсорсинг центрів обробки даних пропонує багато переваг, деякі підприємства віддають перевагу керуванню власними центрами обробки даних з кількох причин.

Коли важливе обладнання є під контролем третьої особи, є ризик пошкоджень обладнання і випадкових втрат даних. Також власники підприємств можуть бути сильно обмежені договором між їхньою компанією та компанією-орендатором. [3]

3.2.2. Розміщення на власних ресурсах

Для розгортання будь-якого застосунка за моделлю на власних ресурсах, потрібен власний фізичний сервер, який має бути розташований в стінах власної організації, так і на орендованому або поставленому компанією сервері в будь-якому дата-центрі.

У разі, якщо сервери розташовуються на території організації, витрати на обслуговування, безпеку, електроенергію та утримання обладнання на балансі компанії залишаються за вами.

Рішення розміщення на власних ресурсах гарантують своїм власникам максимальний рівень збереження і безпеки корпоративних даних, а також - можливість цілодобового фізичного доступу до будь-якої інформації, розміщеної на сервері в власному дата-центрі. Такі рішення затребувані компаніями-представниками середнього та великого бізнесу, в яких на перше місце ставиться контроль і безпеку внутрішніх даних. [4]

3.2.3. Розміщення на хмарних ресурсах

Хмарні обчислення - це наявність на замовлення ресурсів комп'ютерної системи, особливо для зберігання даних та обчислювальної потужності, без безпосереднього активного управління користувачем. Термін зазвичай використовується для опису центрів обробки даних, доступних багатьом користувачам через Інтернет. Великі хмари, що переважають сьогодні, часто мають функції, розподілені по декількох місцях від центральних серверів.

Хмари можуть бути обмежені однією організацією (хмари підприємств) або бути доступними багатьом організаціям (громадська хмара). Хмарні обчислення покладаються на обмін ресурсами для досягнення узгодженості та економії масштабу.

Прихильники громадських та гібридних хмар відзначають, що хмарні обчислення дозволяють компаніям уникати або мінімізувати авансові витрати на ІТ-інфраструктуру. Прихильники також стверджують, що хмарні обчислення дозволяють підприємствам швидше запускати свої програми та працювати з покращеною керованістю та меншим обслуговуванням, а також це дає можливість ІТ-командам швидше налаштувати ресурси для задоволення коливаючих та непередбачуваних потреб, забезпечуючи широкі можливості обчислень: висока обчислювана потужність у певні періоди пікового попиту. [5]

4. Структурна розробка застосунку

4.1. Архітектура

Загалом, архітектура застосунку є варіативною і залежить від потреб користувача. Максимальний варіант архітектури зображений на схемі нижче.

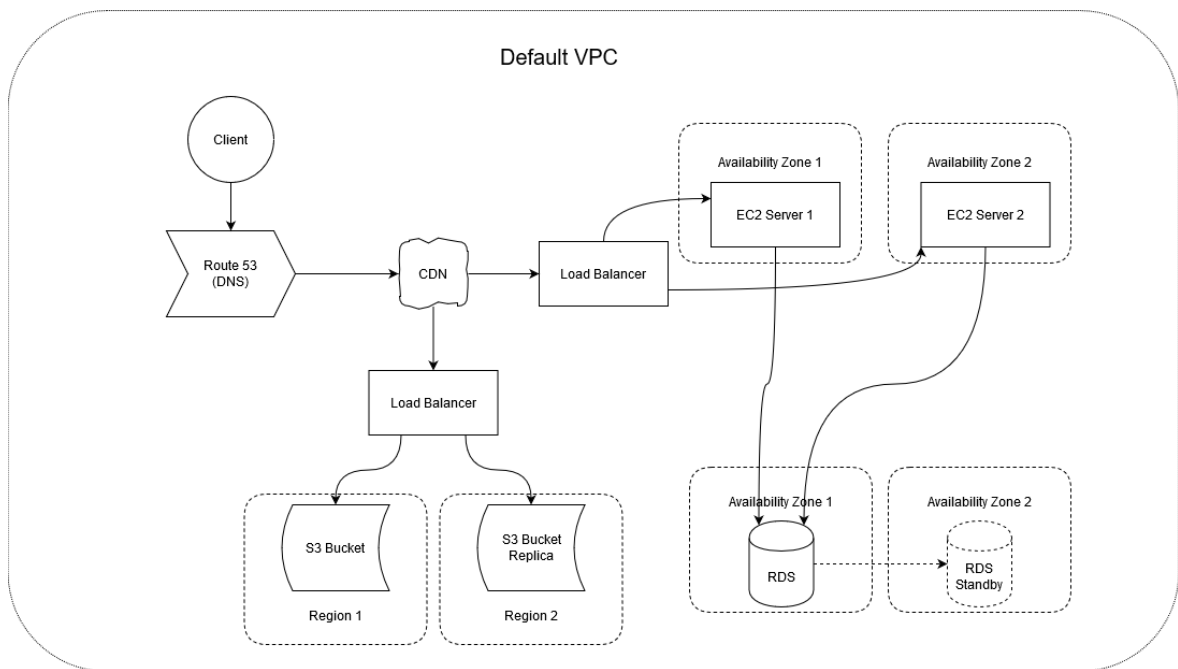


Рис. 2 – Структурна схема архітектури

4.2. Компоненти

В застосунку використовуються наступні компоненти:

- Amazon Elastic Compute Cloud (EC2) дозволяє користувачам орендувати віртуальні комп'ютери, на яких можна запускати власні комп'ютерні програми. EC2 заохочує масштабоване розгортання програм, надаючи веб-сервіс, за допомогою якого користувач може завантажувати зображення Amazon Machine (AMI) для налаштування віртуальної машини, яку Amazon називає "екземпляром", що містить будь-яке потрібне програмне забезпечення.
- Amazon Elastic Load Balancer (ELB) - послуга з балансування навантаження для розгортання веб-служб AWS. ELB автоматично розподіляє вхідний трафік додатків та масштабує ресурси для задоволення потреб у трафіку. ELB допомагає ІТ-команді налаштувати потужність відповідно до вхідних програм та мережевого трафіку.
- Amazon Simple Storage Service (S3) – послуга, яка забезпечує зберігання об'єктів через інтерфейс веб-сервісу. Amazon S3 використовує ту саму масштабовану інфраструктуру зберігання, яку Amazon.com використовує для запуску глобальної мережі електронної комерції.
- Amazon CloudFront - мережа доставки вмісту (CDN), пропонована Amazon Web Services. Мережі доставки вмісту забезпечують глобально розподілену мережу проксі-серверів, які кешують вміст, наприклад веб-відео чи інші об'ємні медіа, локальніше для споживачів, тим самим покращуючи швидкість доступу для завантаження вмісту.
- Amazon Relational Database Service - служба розподілених реляційних баз даних AWS. Це веб-сервіс, призначений для спрощення налаштування, роботи та масштабування реляційної бази даних для використання в додатках.
- Amazon Route 53 (Route 53) - це масштабована та високодоступна послуга системи доменних імен (DNS). На додаток до можливості перенаправляти користувачів до різних служб AWS, включаючи екземпляри EC2, Route 53 також дозволяє клієнтам AWS спрямовувати користувачів до сервісів що не належать AWS, а також стежити за станом застосунку та його кінцевими точками.

4.3.Опис функціонування системи

Система функціонує наступним чином. На всіх «екземплярах» EC2, що є в ролі серверу встановлена ОС Ubuntu та встановлене middleware MediaWiki. Між користувачем та застосунком знаходиться DNS сервіс Route 53. Далі знаходиться CDN , що згідно с правилами або передає статичні файли з S3, або динамічні дані з Load Balancer. За Load Balancer, що розподіляє навантаження , розташовані декілька EC2 в різних зонах доступності. Текстові дані (тексти статей, профілі користувачів) зберігаються на RDS, а саме MySQL Community, і в RDS завжди є «екземпляр в очікуванні», до якого звертаються EC2 у разі втрати з'єднання з основною RDS.

Коли з'явиться необхідність оновити застосування (наприклад підключити нові розширення для MediaWiki), можна на одній з EC2 провести модифікацію, зробити образ цього екземпляру, і замінити старі EC2 новими створеними з образу.

5. Структурна розробка утиліти DeployHelper

Утиліта, що дозволяє користувачу створювати конфігурацію та розгортати цю конфігурацію на AWS є саме веб-застосунком. Цей застосунок побудований на Express з допомогою шаблонізатора EJS.

6. Детальна розробка застосунку

Розробка застосунку почалась через веб-інтерфейс AWS. Спочатку був розгорнутий образ MediaWiki з AWS Marketplace. Після розгортання його на EC2, можна було одразу користуватись вікі без подальшого налаштування. Однак оскільки, це було по суті 1-рівневе застосування, з встановленим веб-сервером на цій же машині, було б недоцільно брати за основу такий образ.

Тому було розгорнуто образ Ubuntu від Amazon на EC2, при цьому через Security Groups до машини було надано дозвіл SSH та HTTP(S) трафіку, і через IAM було надано доступ до RDS та S3. Далі було під'єднано до серверу через SSH. Згідно з інструкціями по ручному встановленню MediaWiki було встановлено необхідні «пакунки» для запуску, такі як apache і php. Використовуючи curl було завантажено останню версію MediaWiki, розпаковано її, та переведено файли до папки apache.

Через RDS було запущено базу даних типу MySQL. При цьому було через Security Groups відкрито вхідний трафік MySQL.

За допомогою браузеру було під'єднано до EC2, та за документацією було виконано початкову ініціалізацію сайту, вказавши при цьому зовнішній ір мережного пристрою бази даних замість localhost. Після ініціалізації було завантажено файл конфігурації на сервер за допомогою curl.

Далі було створено S3 bucket з громадським доступом читання. Було завантажено останню версію «розширення» AWS для MediaWiki, та «пакунок» composer, за допомогою якого було завантажено всі необхідні модулі для розширення, та додано необхідні дані в конфігурацію сервера(в тому числі назву створеного S3 bucket).

Для функціонування CloudFront була взята назва S3 bucket та доданий отриманий endpoint до конфігурації сервера.

Для LoadBalancer було створено з першого EC2 екземпляру образ, і розгорнуто цей образ в іншій зоні доступності Amazon. Було створено Target Group і додано до нього 2 створених екземпляри EC2. Нарешті було створено Load Balancer і додано до нього Target Group.

Результат розробки можна побачити на Рис. 3 – Початкова сторінка Рис. 4 – Вигляд утиліти.

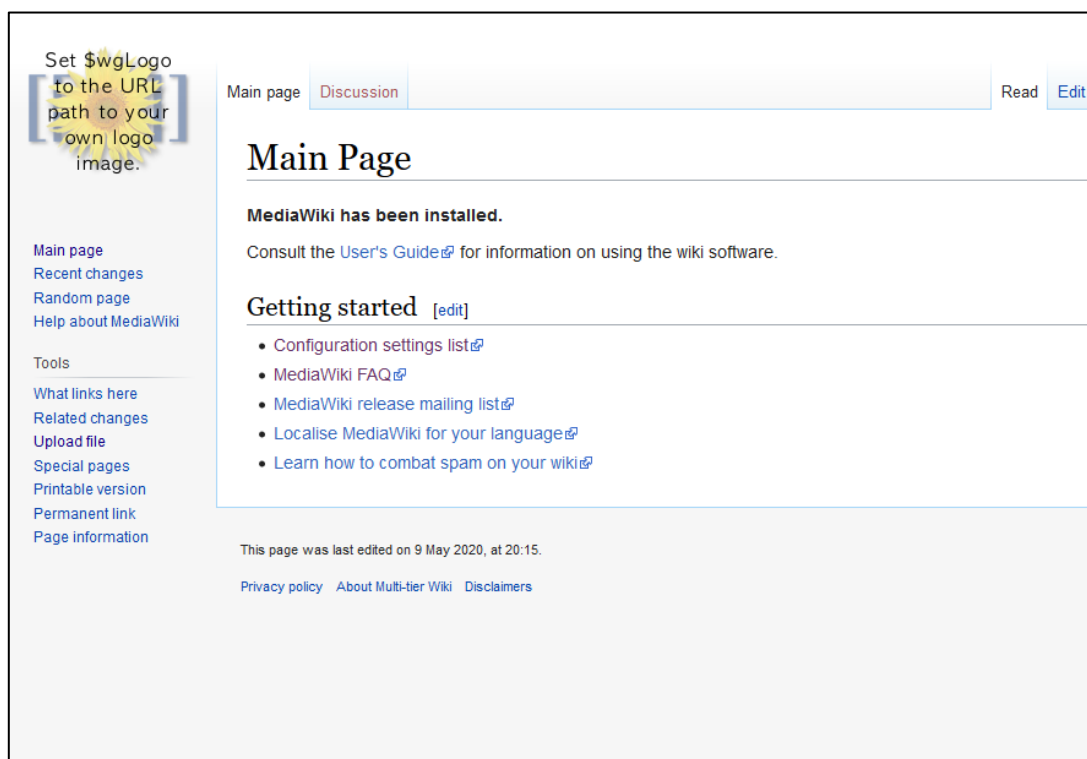


Рис. 3 – Початкова сторінка

7. Детальна розробка утиліти DeployHelper

За основу було взято скелет middleware Express. Було створено початкову сторінку, на якій користувача запитується які з сервісів він хоче використати в застосунку.

Далі, оскільки дослідження застосунка було проведено через веб інтерфейс, було перевірено як виконувати аналогічні дії за допомогою AWS CLI. Далі всі операції що треба виконати для розгортання було додано до програми, додано скриншоти та пояснення. Для коду було додано окремий стиль, щоб виділити його візуально від звичайного тексту.

Результат вигляду утиліти можна побачити на Рис. 4 – Вигляд утиліти.

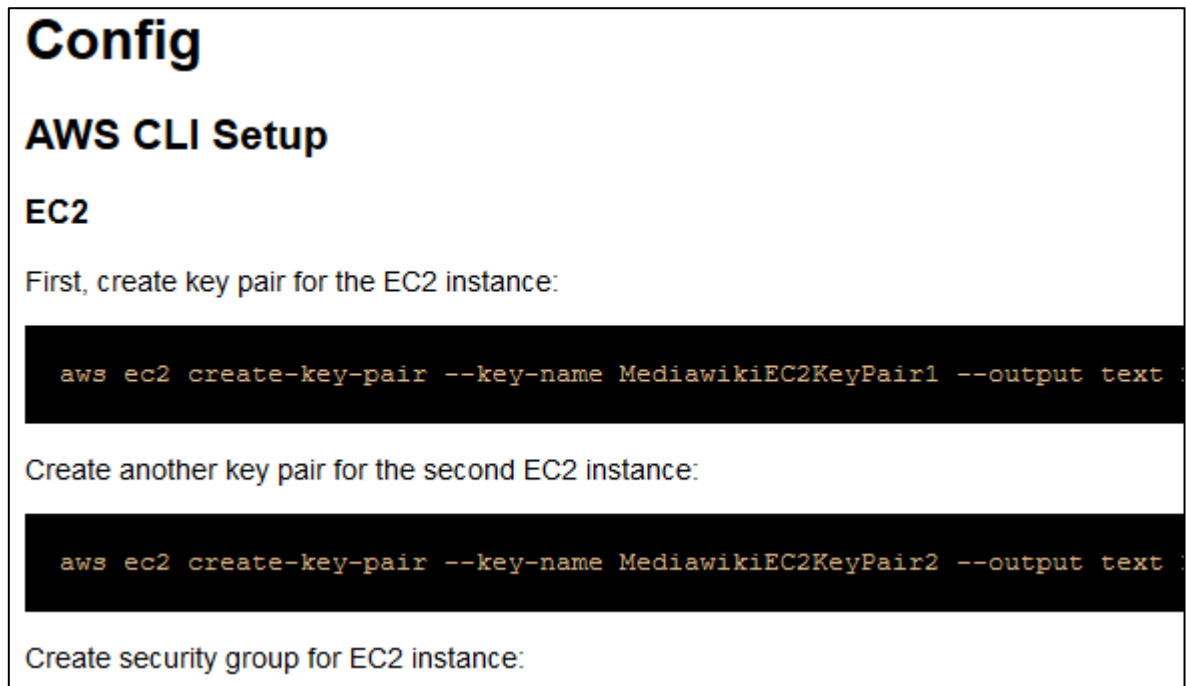


Рис. 4 – Вигляд утиліти

8. Висновки

В результаті роботи було отримано програму-утиліту яка виводить інструкції та код для розгортання багаторівневого застосування-вікі на AWS.

Не вдалося повністю автоматизувати розгортання застосунку, адже потрібно довантажувати файли та налаштовувати систему. Також не було реалізовано DNS, оскільки цей сервіс є платним.

З складних та моментів можна виокремити розв'язання проблеми недоступності бази даних с серверу – для доступу потрібна ір адреса, яка напряду не відома у RDS, і знаходиться в мережевих інтерфейсах.

9. Джерела

1. <https://wikimatrix.org>
2. <https://docs.bitnami.com/google-templates/singletier-vs-multitier/>
3. <https://www.bmc.com/blogs/data-center-colocation/>
4. <https://dlg.im/ru/blog/on-premise/>
5. <https://aws.amazon.com/what-is-cloud-computing/>
6. <https://www.guru99.com/n-tier-architecture-system-concepts-tips.html#4>