Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Центр післядипломної освіти_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____

___Дослідження методів розгортання веб-додатків у приватних хмарах___
(тема)

Виконав: студент 2 курсу, групи ІПЗмзд-17-1___
спеціальності _121- Інженерія програмного забезпечення_
(код і повна назва спеціальності)
Освітньо-наукової програми
_____Інженерія програмного забезпечення_____
(повна назва освітньої програми)

_____Пержановський Микита Андрійович_____
(прізвище, ініціали)
Керівник _____доц. Лановий О. Ф._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет _____Центр післядипломної освіти_____
Кафедра _____Програмної інженерії_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _____121-Інженерія програмного забезпечення_____
(код і повна назва)
освітньо-наукова програма _Інженерія програмного забезпечення__
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____Пержановському Микиті Андрійовичу___ _____
(прізвище, ім'я, по батькові)

1. Тема роботи: _Дослідження методів розгортання веб-додатків у приватних хмарах_____
_____
затверджена наказом по університету від "___" _____20 ___ р № _____

2. Термін подання студентом роботи до екзаменаційної комісії: 05 червня 2019 р._____

3. Вихідні дані до роботи: *сукупність методів традційоного, автоматизованого та змішаних підходів до розгортання; сучасні інструменти розгортання.*_____

4. Перелік питань, що потрібно опрацювати в роботі *актуальність задачі, аналіз методів розгортання веб-додатків, порівняння методів розгортання веб-додатків, постановка задачі, огляд доступних рішень автоматизації розгортання, аналіз можливих типів архітектури системи, побудування UML діаграм системи, прототипування UI/UX інтрерфейсу.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): *Приклади систем автоматизації розгортання, діаграма прецедентів, діаграма послідовностей, архітектура системи, скриншоти інтерфейсу прототипу системи, приклад виводу з консолі, приклад архітектури веб-додатка; UML-діаграми, архітектура системи.*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Лановий О. Ф.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Постановка задачі на дослідження	19 квітня 2019р.	
2.	Аналіз методів розгортання	27 квітня 2019р.	
3.	Дослідження особливостей процесу розгортання	14 травня 2019р.	
4.	Підготовка пояснювальної записки	25 травня 2019р.	
5.	Спецчастина	26 травня 2019р.	
6.	Підготовка презентації та доповіді	28 травня 2019р.	
7.	Попередній захист	10 червня 2019р.	
8.	Нормоконтроль, рецензування	17 червня 2019р.	
9.	Занесення диплома в електронний архів	19 червня 2019р.	
10.	Допуск до захисту у зав. Кафедри	20 червня 2019р.	

Дата видачі завдання _11_ ____лютого___2019 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Лановий О. Ф._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 110 с., 31 рис., 1 табл., 17 джерел, 1 додаток.

АВТОМАТИЗАЦІЯ, ПРИВАТНІ ХМАРИ, ДІДЖИТАЛІЗАЦІЯ, РОЗГОРТАННЯ, WORDPRESS.

Метою роботи є аналіз існуючих традиційних та автоматизованих методів, інструментів та підходів до розгортання веб-додатків у приватних хмарах.

Результатом роботи є прототип інтерфейсу системи з швидкого та зручного розгортання веб-додатків, який надав би користувачу можливість повністю контролювати процес через графічний інтерфейс.

AUTOMATION, CLOUD COMPUTING, DIGITALIZATION, DEPLOYMENT, WORDPRESS.

The aim of the work is to analyze existing traditional and automated methods, tools and approaches to deploying web applications in private clouds.

The result of the work is a prototype system for the rapid and convenient deployment of web applications, which would give the user the opportunity to fully control the process through the graphical interface.

ЗМІСТ

ВСТУП

Сучасний світ існує в епоху діджиталізації – все більше та більше сучасних бізнесів, державних установ, різноманітних сервісів переходять у так званий «онлайн»-режим. З'являється велика кількість веб-застосунків, які надають змогу рядовим користувачам ПК взаємодіяти з веб-сайтами, онлайн-додаткамині, та навіть надають змогу створювати свої власні онлайн-ресурси.

Але необхідно розуміти, що таке веб-застосунок, та які особливості ховає кожен з них. Якщо говорити в загальному, то веб-застосунок - це онлайн-додаток, в якому клієнтом виступає браузер, а сервером - веб-сервер. Браузер може бути реалізацією так званих тонких клієнтів - логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є міжплатформовими сервісами.

Пониження рівня знань, які необхідні для створення веб-ресурсів, стало дуже помітним сьогодні. Тепер не потрібно мати освіту програміста, щоб створити примітивний веб-сайт – достатньо знайти декілька онлайн-білдерів та протестувати їх. Але для більш складних та дизайн-орієнтованих веб-сайтів – знадобляться знання хоча б у розмітці (HTML, CSS).

Дуже схожа тенденція «спрощення» спостерігається і у сучасному світі веб-розробки: за останні 15 років з'явилися велика кількість різноманітних систем для менеджменту контенту (WordPress, Drupal, Magento), які дозволяють будувати веб-сайти на своїй основі та надають кінцевому користувачу повний контроль над вмістом сторінок, форм, статті. Мови програмування теж розвивалися дуже стрімко та розпадались на окремі фреймворки (Laravel, React, Node.js), які, в свою чергу, мають чіткий спосіб застосування, як і CMS (Content Management System – Система Менеджменту Контенту).

Magento – це відмінна CMS для онлайн-магазинів, WordPress – дозволяє створювати прості, але в той же час дуже красиві веб-сайти чи блоги, Laravel – ідеальний та дуже потужний фреймворк для розробки «на замовлення», фреймворк JavaScript React – гарний вибір для створення інтерфейсів користувача, який покликаний вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків.

Але все рівно існує достатньо високий поріг знань, які необхідні, щоб повністю створити та розгорнути онлайн-додаток в Інтернеті, бо для цього необхідні знання у системному та серверному адмініструванні. Статистика показує, що не всі сучасні програмісти звертають на це увагу, а іноді і зовсім ігнорують дану область знань. Список дій, які необхідні для повноцінного розгортання веб-додатку в інтернеті виглядає так:

— установка чи налаштування операційної системи на сервері, де буде розгортатися веб-додаток;

— установка та налаштування веб-сервера, який буде отримувати та оброблювати запроси користувачів, які взаємодіють з веб-додатком;

— установка додаткових бібліотек, пакетів, інших програм для коректної роботи веб-застосунка;

— установка та налаштування самого веб-застосунка;

— налаштування DNS.

Існує багато інструментів, які дозволяють автоматизувати цей процес. У своїй більшості це скриптові мови програмування, які надають змогу закодувати деякий процес. Наприклад, установку та розгортання CMS базового пакету WordPress на приватній хмарі. Для їх володіння треба мати не тільки знання програмування, але і мати розуміння архітектури та сутності сервера та веб-сервера.

Але і це ще не все: з популяризацією діджиталізації в Інтернеті стали з’являтися онлайн-сервіси, які надавали користувальницький інтерфейс, який надавав клієнту змогу розгорнути необхідний веб-додаток на сервері чи приватній хмарі. Але основною проблемою цих інструментів було те, що вони лімітували користувачів у можливостях, кастомізаціях чи розширенні їх онлайн-бізнесу. У

деякий момент реальний бізнес клієнта потребував певного рівня кастомізацій, яких не міг надати продукт.

Таким чином, метою даної роботи є дослідження різних методів розгортання та установки веб-додатків на різні види приватних хмарних середовищ та прототипування онлайн-інструменту, який надав би користувачу можливість швидко та ефективно встановлювати, налаштовувати, обслуговувати та взаємодіяти з веб-додатком на певному рівні. Для цього було проведено набір досліджень різних підходів до розгортання, різних інструментів для розгортання, існуючих інструментів для розгортання та виділені їх сильні та слабкі риси.

Результати цього дослідження можуть бути практично корисними для рядових користувачів Інтернету, які хочуть адаптуватися до сучасних технологій та мати змогу виконувати певний набір дій для переведення свого бізнесу у онлайн-режим.

1 ПОСТАНОВКА ЗАДАЧІ НА ДОСЛІДЖЕННЯ

1.1 Актуальність задачі

Сьогодні такі країни як Сінгапур, Великобританія, Нова Зеландія, ОАЕ, Естонія, Японія, Ізраїль, Китай та інші взяли курс на цифровий розвиток не тільки на рівні діждиталізації державних установ, але і у сферах транспорту, освіти, електронних засобів і новітніх технологій. По всьому світові частка традиційних методів зменшується, а цифрових — збільшується, надаючи величезні переваги для бізнесу та людей.

Інтернет змінює спосіб роботи, спілкування, створення та обміну інформацією, а також організації людей, ідей та речей по всьому світу. Проте масштаби цієї трансформації все ще недооцінюються. Наприклад, у Китаї з 700 мільйонів користувачів Інтернету, продажі у онлайн-режимі досягли 350 мільярдів доларів у першій половині 2018 року, що становить 12% від загального обсягу роздрібних продажів усіх китайських споживачів у цей період. Операції з онлайн торгівлі між Китаєм та іншими країнами збільшилися на 33% до 2,8 трлн юанів у 2018 році і становили приблизно 9,6% від загальної міжнародної торгівлі Китаю [1]. Китай став найбільшим ринком електронної комерції у світі за величиною продажів, а в 2018 році оцінювався у 899 мільярдів долларів США.

Останні дослідження говорять, що сучасні люди більше віддають перевагу роботі з веб-додатком чи онлайн-системою, ніж реальному відвідуванню того ж магазину чи компанії, яка надає якийсь сервіс [2]. Тому малі та середні бізнеси почали активно інтегрувати веб-технології, бо зрозуміли, що Інтернет надає дуже гарні можливості для міжнародного партнерства, виробництва та торгівлі. А також буде сприяти швидкому розвитку та зростанню підприємств, конкуренції та технологій. За останні 10 років велика кількість компаній та бізнесів по усьому світу інвестували величезні обсяги грошей у розвиток веб- та мобільних додатків. Люди почали розуміти, що вони знімають усі обмеження у часі та просторі,

надають більше можливостей для спілкування з клієнтами, партнерами чи найманцями по всьому світу та просто полегшують життя.

Доля веб-присутності у сучасних компаній зростає дуже помітними темпами: дослідження говорять, що в наступні 2-3 роки вона зросте на 50-60%. Інший гарний приклад – це електронна комерція, у якої у 2017 році роздрібні продажі по всьому світі становили більше ніж 2,3 трильйони долларів США. В той же час традиційні ринки очікують зростання лише на 2%.

Діджиталізація у перспективі дозволить простим користувачам виконувати будь-яку дію в будь-який час і з будь-якого місця. Веб та традиційні способи ведення бізнесу є дуже різними. Класичний метод в більшості потребує беспосередньої присутності клієнта для отримання якогось сервісу, в той же час інтернет-сервіси дуже часто навіть не мають реального офісу, тому що весь необхідний функціонал присутній на платформі. А команда, що відповідальна за його розробку розподілена по всьому світу. Ціноутворення також є великою різницею для традиційних та онлайн-методів. Класичні бізнеси базують свої ціни на різних метриках: наприклад, класичний брокерський консалтинг базує свої ціни на досвіді та кількості вирішених справ своєї команди. В той ж час онлайн-брокер платформа базує ціни на швидкості отримання кінцевої цілі та зручності взаємодії користувача.

Сьогодні у більшості країн світу впроваджуються онлайн-сервіси для акредитації реальних бізнесів, які дозволяють пройти процес через веб-платформу [3]. У веб-додатків такого типу у більшості випадків є не тільки веб-сайт, але і велика інфраструктура із різноманітних сервісів, які оброблюють дії та інформацію клієнта. Front end частина такого веб-застосунку реалізована на технології WordPress, а Back end може мати набір з сервісів (MailChimp, Elastic Search) та фреймворків PHP Laravel і Vue.js. Для кожної з перерахованих технологій можна автоматизувати процес налаштування та розгортання. Це дозволить заощадити час, який необхідний для вивчення технологій та підходів до автоматизації, час необхідний для створення відповідних скриптів та час, який необхідний для повністю ручного розгортання.

Основною проблемою онлайн-систем сьогодні є безпека, надійність та відсутність довіри у більшої частини світу [4]. Інтернет наповнений речами та людьми, які хочуть використати результати чиєїсь роботи для власного прибутку. Сучасні власники веб-сайтів та електронної комерції стикаються з проблемами зловмисних спроб (як успішних, так і не зовсім) крадіжки даних, грабежу грошей і навіть використання свого сервера та веб-сайту в якості бази для програмного забезпечення з криптодобування, що працює у фоновому режимі. Ці випадки дійсно різноманітні і в більшості з них шкідливі програми приховують себе. Існує багато способів захисту та запобігання доступу хакерів чи шкідливих програм до вашої системи. Ці інструменти включають брандмауери, програмне забезпечення для шифрування, цифрові сертифікати та паролі. Іноді для цього необхідний окремий сервіс, взаємодіючий з компонентами інфраструктури веб-додатку. А іноді, наприклад, для малих онлайн-бізнесів достатньо слідувати базовим рекомендаціям з розробки та обслуговування веб-додатку.

Наприклад, брандмауер веб-додатків (WAF) - це інструмент для фільтрації, моніторингу та блокування HTTP / HTTPS трафіку, який йде до та з веб-додатку. Простими словами, це досить німий інструмент, який дозволяє обертати трафік через певні налаштовані правила. WAF пропускає трафік далі, якщо він не порушує жодного з зазначених правил і негайно блокує його, якщо він це зробить.

Потрібно визнати той факт, що різні WAF не є досконалими і їх можна обійти, коли, наприклад, вразливість нульового дня відкриває двері для атаки, яка ще не обробляється жодним з правил WAF.

Іншим важливим пунктом безпеки є відповідальний вибір розширень/плагінів для веб-додатку. Перше, на що потрібно звернути увагу, це частота оновлень цього плагіна. Якщо автор опублікував останнє оновлення близько 1 року тому - то слід підозрювати щось погане. Можливо, автор вже перестав працювати над цим розширенням, і не буде ніяких подальших оновлень не очікується. Потрібно розуміти, що в більшості випадків дії плагіна пов'язані з бізнес-логікою веб-сайту. Тобто установка потенційно ризикованої

функціональності може привести до відкриття бекдору для хакерів, що може стати великою проблемою для всього бізнесу.

Правильний підхід до розгортання, розробки, управління та обслуговування веб-додатку, а також своєчасно оновлення пакетів та модулів, знизить шанси проникнення в рази. Сервісне обслуговування - це складний термін, що складається з численних дій, які робляться для того, щоб не тільки захистити свій додаток та дані, але і запобігти виникненню помилок і вирішувати їх у їхньому джерелі, перш ніж вони переростуть у реальну проблему [5].

Спрощення процесу розгортання та обслуговування веб-додатків може бути практично використаний для подальших наукових досліджень та винаходів. Результати дослідження можуть бути використані для покращення вже існуючих інструментів та процесів розгортання, або навіть створення зовсім нового підходу до роботи з веб-додатками в Інтернеті, який не потребує широкого базису знань та навичок.

1.2 Аналіз традиційних методів розгортання веб-додатків

Перед тим, як почати безпосереднє розглядання методів розгортання, треба зрозуміти, що таке сам процес розгортання. Розгортання програмного забезпечення - це всі дії, які роблять систему програмним забезпеченням доступним для використання.

Загальний процес розгортання складається з декількох взаємопов'язаних дій з можливими переходами між ними. Оскільки кожна система програмного забезпечення є унікальною, точні процеси або процедури в межах кожної діяльності важко визначити. Таким чином, "розгортання" слід інтерпретувати як загальний процес, який необхідно налаштувати відповідно до конкретних вимог або характеристик.

Коли комп'ютери були надзвичайно великими, дорогими та громіздкими (мейнфрейми та мінікомп'ютери), програмне забезпечення часто постачалося виробниками разом з обладнанням. Якщо програмне забезпечення для бізнесу потрібно було встановлювати на вже існуючому комп'ютері, це потребувало дорогих та трудомістких візитів архітектора або консультанта. Для сучасних складних систем це все ще нормальна та розповсюджена практика.

Проте, з розвитком програмного забезпечення масового ринку для нового віку мікрокомп'ютерів у 1980-х роках з'явилися нові форми розповсюдження програмного забезпечення - перші картриджі, потім касети, потім дискети, потім (у 1990-х і пізніше) оптичні носії, інтернет і флеш-накопичувачі. Це означало, що розгортання програмного забезпечення доводилося робити клієнту. Проте з часом все більше усвідомлювалося, що конфігурація програмного забезпечення клієнтом є важливою, і що в ідеалі має бути зручний інтерфейс, який дозволяє контролювати цей процес (а не, наприклад, вимагати від клієнта редагування записів реєстру в Windows).

Розгортання програмного забезпечення до появи Інтернету було дорогою, складною, громіздкою та тривалою справою. Тому можна стверджувати, що розповсюдження Інтернету зробило можливим гнучке розробку програмного забезпечення. Дійсно, поява хмарних обчислень і програмного забезпечення як сервісу означало, що веб-додатку можуть бути розгорнуті для великої кількості клієнтів через Інтернет. Це також означало, що графіки розгортання тепер визначаються постачальником програмного забезпечення, а не клієнтами [6]. Така гнучкість призвела до зростання безперервної доставки як життєздатного варіанту, особливо для менш складних веб-додатків.

Складність і постійні зміни веб-додатків сприяли появі спеціалізованих ролей для координації та розробки процесу розгортання. Для настільних систем кінцеві користувачі часто стають "розробниками програмного забезпечення", коли вони встановлюють пакет програмного забезпечення на своїй машині. Розгортання корпоративного програмного забезпечення передбачає набагато більше ролей, і ці ролі зазвичай змінюються під час переходу програми від тестового середовища до

виробничого середовища. Типові ролі, які беруть участь у розгортанні програмного забезпечення для корпоративних додатків, можуть включати:

а) у тестових умовах:

1) розробники програмного забезпечення;
2) build- та release-інженери;
3) координатори розгортання (DevOps);

б) у виробничих умовах:

1) системний адміністратор;
2) адміністратор бази даних;
3) координатори випуску (DevOps);
4) керівники та менеджери проектів.

Тепер можна перейти до розглядання традиційних методів розгортання веб-додатків, які все ще є популярними, незважаючи на велику кількість нових інструментів.

Особливістю розгортання веб-додатків є те, що сам процес здається простим, але на ділі він може бути досить складним. Є дуже багато способів, які варіюються від спеціального та зручного для користувача програмного забезпечення та сервісів, до більш складних інструментів командного рядку. У цьому розділі я розгляну ці способи та їх особливості.

Commando - якщо ви редагуєте файли безпосередньо на сервері, у вас, по суті, відсутній так званий крок розгортання. Для цього можна використати редактор з підтримкою FTP, наприклад, Coda або Adobe Dreamweaver. Також є можливість використовувати щось подібне до інструменту WebDAV, який дозволяє змонтувати віддалений сервер чи приватну хмару, в якості жорсткого диску на локальному комп'ютері, який використовується для нашалтувань. Потім Ви зможете використовувати будь-який локальний текстовий чи кодовий редактори.

Даний підхід є досить сильним, тому що ви результати змін помітні одразу ж. Проте цей підхід не є поширеним сьогодні, оскільки:

— це небезпечно та не є гарною практикою, бо немає можливості протестувати веб-додаток перед запуском;

— не існує ніякого запису змін;

— внесені налаштування дуже складно скасувати;

— внесені налаштування можуть конфліктувати з іншими налаштуваннями;

— даний підхід зовсім не підходить до великих та довгострокових проектів, бо це спричиняє конфлікти.

Не треба забувати, що даний підхід має на увазі, що вже існує встановлений веб-сервер та усі пакети, необхідні для коректної роботи веб-додатку, присутні та налаштовані.

Наступною опцією розгортання є ручне використання FTP. Цей підхід розгортання полягає в тому, що користувач працює локально та по готовності переміщає файли на сервер використовуючи протокол FTP. File Transfer Protocol - це стандартний мережевий протокол, який використовується для передачі комп'ютерних файлів між клієнтом і сервером по комп'ютерній мережі. FTP побудований на архітектурній моделі клієнт-сервер, який використовує окремий канал з'єднання між клієнтом і сервером.

Користувачі FTP можуть аутентифікувати себе за допомогою імені користувача і пароля (налаштованих у веб-сервері окремо), та можуть підключатися анонімно, (якщо сервер налаштований на дозволення таких підключень). Для безпечної передачі, яка захищає ім'я користувача та пароль і шифрує вміст, FTP часто захищається SSL / TLS (FTPS) або замінюється протоколом передачі файлів SSH (SFTP).

Даний підхід є розповсюдженим та працюючим, але може створювати деякі конфлікти. Він не дозволяє відповісти на питання: «Які файли були змінені чи додані?». В основному проблеми цього підходу є дуже схожими з Commando – користувач ризикує перезаписати якісь важливі файли без легкої можливості повернення стану.

Також складністю даного підходу є те, що він також має на увазі, що у користувача є налаштований веб-сервер з пакетами, необхідними для правильної роботи веб-додатку. Це потребує знань системного та серверного адміністрування,

які дуже часто відсутні навіть у досвідчених розробників. Бо це зовсім інша область прикладних знань.

Наступним підходом до розгортання є контроль версій. Він є дуже важливим і буде відігравати певну роль у будь-яких великих розгортаннях. Але саме по собі керування версіями не обробляє саме розгортання за користувача. При використанні контролю версій код веб-додатків чи проектів зберігається у окремих репозиторіях. Git – є найбільш поширеним інструментом контролю версій сьогодні, але також існують більш старі аналоги Subversion і Mercurial. Git - це розподілена система контролю версій, яка використовується для відстеження змін вихідного коду під час розробки програмного забезпечення. Він призначений для координації роботи між програмістами, але може бути використаний для відстеження змін у будь-якому наборі файлів. Як і інші розподілені системи керування версіями, але на відміну від більшості систем з архітектурою «клієнт-сервер» , кожен каталог Git на кожному комп'ютері є повноцінним сховищем з повною історією та повною функцією відстеження версій, незалежно від доступу до мережі або центрального сервера.

Оскільки Git є розподіленою системою керування версіями, її можна використовувати як «сервер з коробки». Спеціальне серверне програмне забезпечення Git допомагає, крім інших функцій, додавати контроль доступу, відображати вміст сховища Git через Інтернет і допомагати керувати кількома сховищами одночасно.

Також можна поєднувати інструменти, для полегшення процесу розгортання та менеджменту веб-додатка. Наприклад якщо використовується Git і потрібно завантажити файли на FTP-сервер, команда git-ftp може заощадити час і пропускну здатність, завантаживши лише ті файли, які були змінені після останнього розгортання. Дана команда відстежує завантажені файли, зберігаючи ідентифікатор фіксації у файлі журналу на сервері. А сервер використовує Git, щоб визначити, які локальні файли були змінені, щоб завантажити тільки них.

Наступним варіантом розгортання є використання rsync. Це утиліта для ефективної передачі та синхронізації файлів між комп'ютером і зовнішнім

жорстким диском і мережевих комп'ютерів шляхом порівняння часу модифікації і розмірів файлів. Алгоритм rsync є типом дельта-кодування і використовується для мінімізації використання мережі. Є додаткові інструменти, які працюють разом с rsync, та можуть використовуватися для додаткового стиснення даних, безпеки і так далі.

Rsync діє шляхом спілкування з іншим процесом rsync, один з яких є відправником, а інший - приймачем. Під час запуску клієнт rsync підключається до однорангового процесу. Якщо передача є локальною (тобто між файловими системами, встановленими на тому ж хості), то вузол може бути створений за допомогою вилки, після установки відповідних каналів для з'єднання. Якщо задіяний віддалений хост, rsync запускає процес обробки з'єднання, зазвичай це - Secure Shell процес.

Після підключення видається команда для запуску процесу rsync на віддаленому хості, який використовує встановлене таким чином з'єднання. У rsync є численні параметри командного рядка та файли конфігурації для визначення альтернативних оболонок, параметрів, команд.

Традиційні методи розгортання є досить трудомісткими та потребують достатньо високого рівня знань та досвіду. Рядовий користувач не зможе одразу ж оперувати ними, бо для цього необхідно буде ознайомитися з великим об’ємом технічної документації, яка в свою чергу потребує фундаментальних знань програмування та системного адміністрування. Без цього оперування цими інструментами не є досить можливим.

Тому сьогодні з’являється велика кількість різноманітних інструментів, які дозволяють автоматизувати процес розгортання, тим самим роблячи його менш трудомістким. Вони дозволяють створити або сценарії розгортання, або автоматизують процес наданням користувачу можливості управління деякими функціями через інтерфейс, а не командну стрічку.

1.3 Аналіз автоматизованих методів розгортання веб-додатків

Більшість дій, які необхідно виконати при традиційних методах розгортання веб-додатків, повторюється з кожним розгортанням. Тому що сам процес розгортання для деяких веб-додатків потребує виконання чіткого та послідовного набору дій чи команд, які налаштують сервер та оточення для його подальшої коректної роботи. З розвитком технологій та складності веб-додатків почали з’являтися різні інструменти автоматизації, які дозволяють заскриптувати процес розгортання веб-додатку. Ці інструменти дають змогу закодувати послідовність дій, які будуть виконуватися одна за одною, та не деяких моментах будуть запитувати кінцевого користувача ввести чи надати якісь дані.

Автоматизація розгортання дозволяє розгортати веб-додатки в різних середовищах, що використовуються в процесі розробки, а також у кінцевих виробничих середовищах. Це призводить до більш ефективного, надійного та передбачуваного розгортання. Інструменти, які автоматизують процеси розгортання, підвищують продуктивність команд розробки та обслуговування, тим самим дозволяють їм і бізнесу розвиватися швидше, і в кінцевому підсумку створювати краще програмне забезпечення, яке швидко розгортається, функціонує більш надійно для кінцевого користувача та дозволяє розгортати веб-додатки частіше (бо процес закодований і потребує ручної взаємодії) [7].

Основними переваги автоматизації розгортання є:

— повторюваність - кожне розгортання відбувається так само, як і всі попередні;

— зменшення кількості помилок - видалення ручних кроків зменшує людські помилки;

— будь-хто зможе провести розгортання – бо усунуті вузькі місця, які потребували роботу експертів;

— фокус на те, що важливо – розробники, наприклад, можуть витрачати свій час на створення нових функцій, а не на виправленнях помилок після ручного розгортання

— частота оновлень - часті оновлення, які доповнюють веб-додатки новими можливостями задовільняють потреби клієнтів і підтримують конкурентоспроможність бізнесів;

— зниження витрат – при автоматизованому процесі допускається менша кількість помилок, витрачається менша кількість робочих годин, необхідних для розгортання, а це призводить до зниження витрат. Тепер можна розглянути різні методи та інструменти автоматизації розгортання веб-додатків.

1.3.1 Автоматизація за допомогою скриптування

Першим методом чи підходом до розгортання є створення сценаріїв розгортання (або скриптування). Розгортання веб-додатків можуть бути автоматизовані шляхом написання спеціальних скриптів, які автоматично виконують дії на певних середовищах у певному контексті. Підхід до написання сценаріїв є простим, але обмеженим, тому що, як тільки зміниться середовище, без ретельного технічного обслуговування, скрипти розірвуться, а розгортання можуть вийти з ладу.

Інсує декілька інструментів скриптування, які створені для різних типів та середовищ. Перший з таких - це Capistrano (https://capistranorb.com), який є засобом автоматизації для віддалених серверів. Коли веб-додаток стає більш складним, то традиційний метод розгортання за допомогою стягнення коду с репозиторія може не спрацювати. Наприклад, це може статися, коли веб-додаток декілька серверів. Або затягування цих файлів може зайняти дуже багато часу, і це може призвести до зависання програми. За допомогою Capistrano ви можете налаштувати підготовку серверного середовища, витягнути нові файли на необхідне місце,

оновити усі посилання, вказати останню версію, очистити чи оновити дозволи, перезапустити служби і зробити це на декількох серверах. Він підтримує різноманітні сценарії та виконання довільних завдань і включає в себе безліч робочих процесів розгортання за умовчанням. Capistrano можна використовувати для:

— надійного розгортання веб-додатків до будь-якої кількості хостів одночасно або послідовно;

— автоматизування перевірки будь-якої кількості хостів (перевірка журналів входу, застосування патчів безпеки);

— заскриптовувати довільні робочі процеси через SSH;

— автоматизування загальних завдань в групах веб-додатків;

— керування додатковими інструментами інфраструктурного забезпечення;

Capistrano може бути інтегрований з будь-яким програмним забезпеченням, але з самого початку він був створений для додатків написаних на мові Ruby / Ruby on Rails. Також Capistrano надає можливості для:

— зміни форматування вихідних даних;

— легкого інтегрування іншого програмного забезпечення керування контролем джерел;

— фільтрації хостів та ролів для часткового розгортання або часткового обслуговування кластерів веб-додатків;

— рецепти для конвеєрів ресурсів Rails і міграції баз даних;

— підтримки складних середовищ;

— розсудливого та виразного використання API інструмента.

Наступним інструментом скриптованої автоматизації є Ansible (https://www.ansible.com). Це кардинально простий движок автоматизації IT, який автоматизує надання хмарних даних, керування конфігураціями, розгортання додатків, внутрішньосервісного оркестрування та багато інших інформаційних потреб. Він був розроблений для в більшості для багаторівневих розгортань, тому Ansible моделює IT-інфраструктуру, описуючи, як всі системи/сервіси веб-додатка взаємопов'язані.

На відміну від іншого популярного програмного забезпечення для управління конфігураціями - наприклад, Chef чи Puppet (які будуть розглянуті далі) Ansible використовує архітектуру без використання спеціальних агентів, які потребують установки на контрольованому хості.

Замість цього, Ansible організовує вузол, встановлюючи і запускаючи модулі на вузлі тимчасово через SSH. Під час виконання оркестровального завдання процес, що виконується модулем, здійснює зв'язок з керуючою машиною через протокол на основі JSON. Якщо Ansible не керує вузлом, він не споживає ресурсів на вузлі, оскільки жоден з активних демонів не виконує встановлене програмне забезпечення. Ansible використовує мову YAML, у вигляді Ansible Playbooks. Основними перевагами Ansible вважають:

— мінімалістичність – системи управління не нав'язують додаткові залежності від навколишнього середовища.

— послідовність – усі дії виконуються послідовно, тому процес розгортання досить прозорий;

— безпечість – Ansible не розгортає додаткових агентів на хостах, бо для його роботи потрібні тільки пакети OpenSSH і Python;

— надійність – ретельно написаний скрипт чи Ansible playbook ідемпотентен для запобігання несподіваних побічних ефектів на керованих системах.

— доволі швидке освоєння інструменту – мова створення є доволі легкою для розуміння при наявності достатніх фундаментальних знань програмування та системного адміністрування.

Ще одним прикладом інструмента на основі скриптування є Puppet (https://puppet.com). Він призначений для управління конфігурацією у Unix-подібних і Microsoft Windows системах декларативно. Користувач описує системні ресурси та їх стан, використовуючи декларативну мову Puppet або Ruby DSL (доменну мову). Ця інформація зберігається у файлах під назвою "маніфести". Puppet виявляє системну інформацію через утиліту Facter і компілює маніфести в специфічний для системи каталог, що містить ресурси та їх залежності, які

застосовуються до цільових веб-додатків. Після цього будь-які дії здійснені Puppet записуються та логуються.

Puppet складається з користувальницької декларативної мови для опису конфігурації системи, яка може бути або застосована безпосередньо в системі, або скомпільована в каталог і розподілена до цільової системи через парадигму клієнт-сервер (за допомогою REST API).

Агент Puppet використовує систему конкретних провайдерів для забезпечення ресурсів, зазначених в маніфестах. Рівень абстракції ресурсів дозволяє адміністраторам описувати конфігурацію на високому рівні без необхідності вказувати специфічні команди ОС (наприклад, rpm, yum, apt). Puppet працює на архітектурі клієнт-сервер. Клієнт – це агент, а сервер – це так званий master. Для тестування і простих конфігурацій він також може бути використаний автономно через запуск з командного рядка.

Puppet сервер встановлюється на одному або декількох хостах, а Puppet Agent встановлюється на усих машинах, якими користувач хоче керувати. Агенти спілкуються з сервером і отримують інструкції з налаштуваннь. Потім агент застосовує отримані конфігурації чи команди на хості та надсилає на сервер звіт про стан виконання. Пристрої можуть запускати Puppet Agent як демон, який може періодично спрацьовувати як крон-процес або запускатися вручну, коли це необхідно.

Puppet доступний у 2 версіях: Puppet Enterprise і Open Source Puppet. Крім надання функцій Open Source Pupвpet, версія Puppet Enterprise також надає GUI, API і інструменти командного рядка для керування вузлами.

Вищеперераховані інструменти розгортання можуть бути використані, як для роботи безпосередньо з розгортанням веб-додаків, так і для автоматизування налаштування серверного середовища (веб-сервер, база даних, набір пакетів для коректного функціонування веб-додатка і т.д.)

1.3.2 Автоматизація за допомогою безперервної інтеграції

Тепер, коли були розглянуті інструменти розгортання на основі скриптування, треба ознайомитися з системами CI (Continuous Integration) – інструментів так званої безперервної інтеграції, які дозволяють розгортати веб-додатки автоматично, в якості частини процесу збирання веб-додатка. Це можна вважати розширенням до методу створення сценаріїв, оскільки системи CI зазвичай не знає деталей програми та серверного середовища. У більшості випадків CI система автоматично ініціює сценарії, які можуть розгортати програму.

Розглянемо CI системи розгортання та їх особливості. Одним з найбільш популярних інструментів сьогодні є Travis CI (рис. 1.1) - це служба безперервної інтеграції, що поширюється і використовується для побудови та тестування програмних продуктів, розміщених у сервісі GitHub. Проекти з відкритим кодом можуть бути протестовані безкоштовно за допомогою сервісу https://travis-ci.org. Приватні проекти можуть бути перевірені на https://travis-ci.com на платній основі. TravisPro надає користувальницьку версію, яку можна розгорнути на власному апаратному забезпеченні користувача.

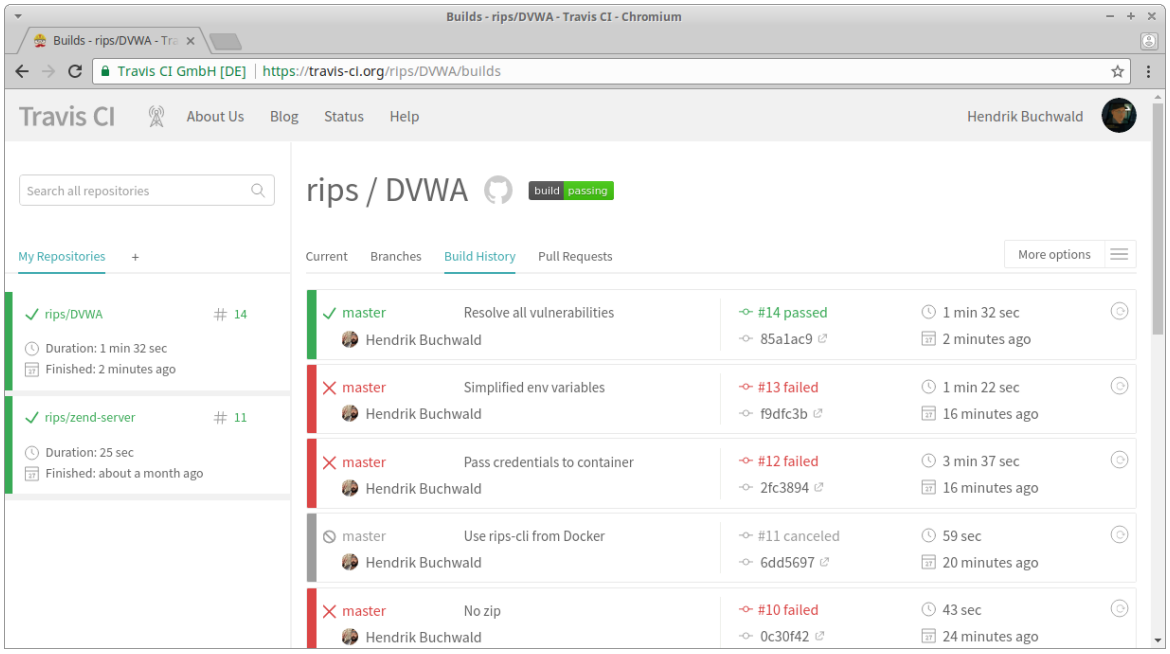


Рис. 1.1 – TravisCI

Travis CI (https://travis-ci.org) налаштовується шляхом додавання файлу з ім'ям .travis.yml, який є текстовим файлом у форматі YAML, до кореневого каталогу репозиторія, який використовується. Цей файл визначає використовувану мову програмування, бажане середовище побудови та тестування (включаючи залежності, які повинні бути встановлені до моменту створення та тестування програмного забезпечення), а також різні інші параметри.

Коли YML файл Travis CI було додано до репозиторію, GitHub повідомить користувача, коли нові компіляції будуть перенесені до цього сховища, або буде подано pull-запит (стягнення). Даний інструмент також може бути налаштований лише для конкретних гілок у сховищі або гілок, імена яких відповідають певному шаблону. Після цього Travis CI перевірить відповідну гілку і запустить команди, вказані в файлі .travis.yml. Зазвичай ці команди будують програмне забезпечення та виконують автоматизовані тести.

Після завершення процесу Travis повідомляє розробника(-ів) у вигляді, який був налаштований. Наприклад, надсилаючи повідомлення електронної пошти, що містить результати тестування (що показують успіх або невдачу), або публікуючи повідомлення на у месенджері, що використовує команда розробників. У випадку запитів стягування, команда pull буде анотуватися з результатом і посиланням на журнал побудови, використовуючи інтеграцію з сервісом GitHub.

Travis CI також може бути налаштований на тестування на різних хостах, з іншим встановленими веб-додатками (наприклад, старішими версіями, для перевірки на сумісність), і підтримує програмне забезпечення, створене на багатьох різних мовах, включаючи C , C++, C#, Clojure, D, Erlang, F#, Go, Apache Groovy, Haskell, Java, JavaScript, Julia, Perl, PHP, Python, R, Ruby, Rust, Scala і навіть Visual Basic.

Аналогом TravisCI є Jenkins (рис. 1.2). Це сервер автоматизації з відкритим кодом, написаний на Java. Jenkins (https://jenkins.io) допомагає автоматизувати частину процесу розробки програмного забезпечення, що постійно оновлюється, а також полегшує технічні аспекти безперервної доставки (Continuous Delivery). Цей інструмент є серверною системою, яка працює в контейнерах сервлетів, таких як

Apache Tomcat. Він підтримує засоби керування версіями, які включають AccuRev, CVS, Subversion, Git, Mercurial, Perforce, TD / OMS, ClearCase і RTC, а також може виконувати проекти Apache Ant, Apache Maven і sbt. Jenkins є вільним програмним забезпеченням, яка випущене під ліцензією MIT.

Збірки та розгортання програмного забезпечення або веб-додатків можуть спрацьовувати різними засобами, наприклад, за допомогою комміту у репозиторий, або шляхом запуску крон-механізму, або шляхом запиту конкретної URL-адреси збірки. Він також може бути запущений після завершення інших збірок у черзі.

Функціональність Jenkins може бути розширена за допомогою варіацій різноманітних плагінів. Вони розширюють його використання на проекти, написані на інших мовах, окрім Java. Плагіни доступні для інтеграції Jenkins з більшістю систем контролю версій і баз даних помилок. Багато інструментів збирання підтримуються за допомогою відповідних плагінів. Плагіни також можуть змінити вигляд Jenkins або додати нові функціональні можливості.

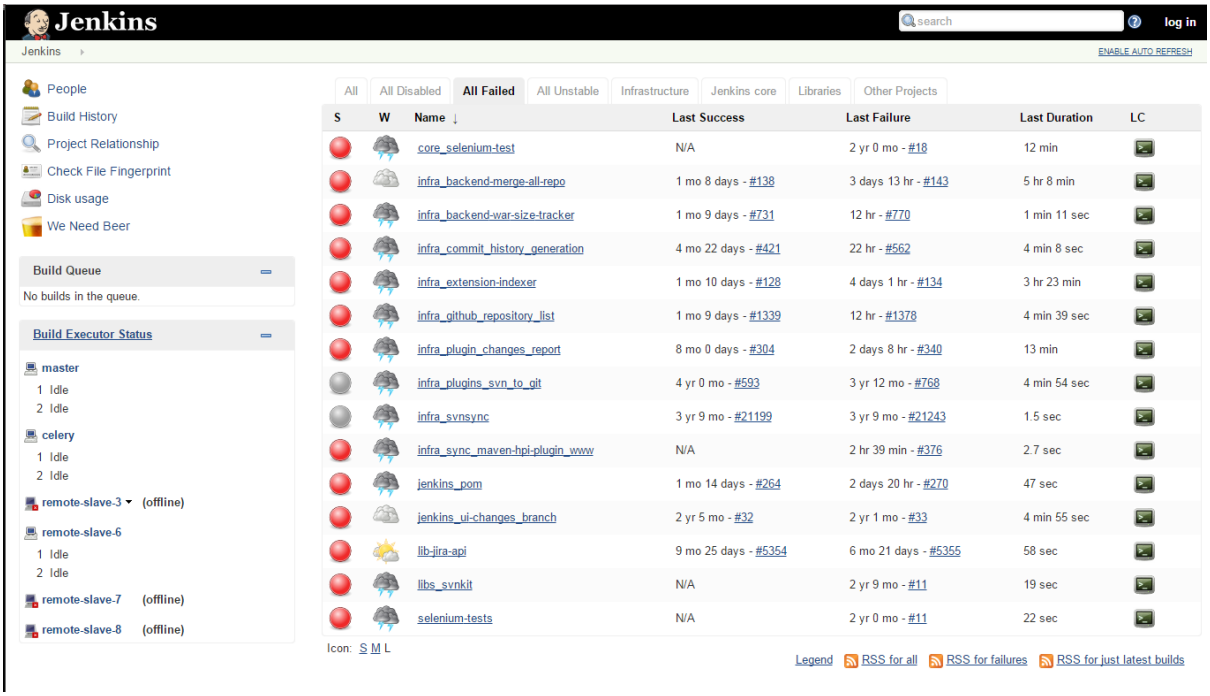


Рис. 1.2 - Jenkins

Є набір плагінів, призначених для цілей модульного тестування, які генерують тестові звіти в різних форматах і автоматизованого тестування, що підтримує автоматизовані тести. Збірки можуть генерувати тестові звіти в різних

форматах, що підтримуються плагінами (підтримка JUnit в даний час входить до комплекту), і Jenkins може відображати звіти і генерувати тенденції і відтворювати їх у графічному інтерфейсі.

Якщо говорити про безпеку, то для Jenkins залежить від двох факторів: контролю доступу та захисту від зовнішніх загроз. Контроль доступу може бути налаштований двома способами: аутентифікація користувача та авторизація. А для зовнішніх загроз, таких як атаки CSRF і зловмисні збірки, підтримується спеціальний захист.

Ще одним аналогом є TeamCity (рис. 1.3). Це система безперервної інтеграції та побудови системи управління. За допомогою TeamCity (https://www.jetbrains.com/teamcity/) користувач може налаштувати сервер збирання і налаштувати безперервне модульне тестування.

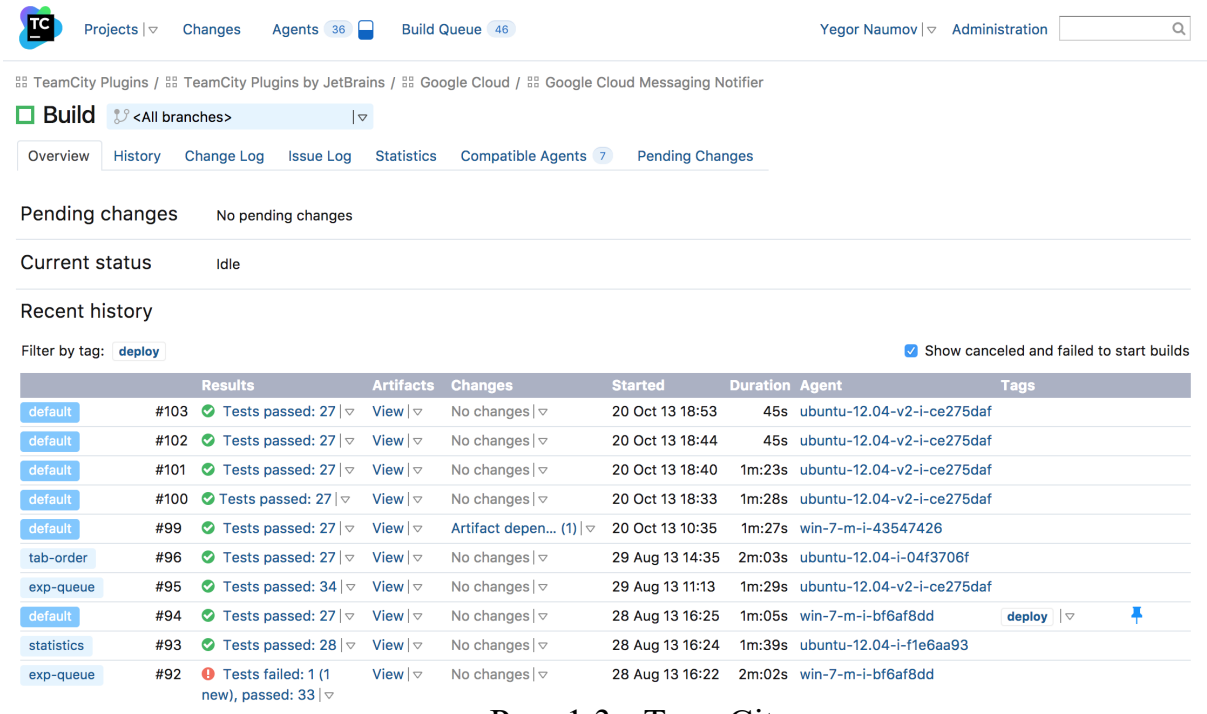


Рис. 1.3 - TeamCity

TeamCity пропонує легку криву навчання для фахівців, завдяки чому ви користувач с фундаментальними знаннями програмування та системного адміністрування може швидко покращити свою практику управління випуском, поступово використовуючи її розширені можливості. Інструмент створений на мові Java. TeamCity є комерційним програмним забезпеченням і ліцензовано під

продуктовою ліцензією. Також доступна безплатна ліцензія на строк до 100 розгортань. Проекти з відкритим кодом можуть запрошувати безкоштовну ліцензію у власників продукту.

Існує ще дуже багато різноманітних аналогів, які можуть бути встановлені на окремий сервер, так і використані повнісытю як Інтернет-сервіс. Більшість з них розрахована на досвідчених розробників, які розуміють, що традиційні підходи розгортання є доволі ризикованими в виробничих середовищах бізнесів. Шанс помилки в такому разі доволі великий, як і ціна цієї помилки.

1.3.3 Автоматизація за допомогою безперервного розгортання

Ще одним підходом є безперервне розгортання – доволі надзвичайна форма автоматизації розгортання, коли численні, або навіть усі зміни, внесені в розробку, проходять автоматичне тестування і негайно розгортаються на серверах чи приватних хмарах. Безперервне розгортання - це стратегія для випуску програмного забезпечення, в якій будь-який код, який проходить фазу автоматизованого тестування, автоматично вивільняється у виробниче середовище, роблячи зміни, видимі кінцевим користувачам програмного забезпечення. Безперервне розгортання усуває людські гарантії проти незадовільного коду в живому програмному забезпеченні. Даний підхід повинен бути реалізований тільки тоді, коли команди розробників суворо дотримуються практики розробки, що готується до виробництва, і ретельного тестування, і коли вони застосовують складний моніторинг у реальному часі у виробництві, щоб виявити будь-які проблеми з новими випусками.

Тепер розглянемо інструменти для безперервного розгортання та їх особливості. Перший з них це Terraform (https://www.terraform.io) – це інструмент з відкритим кодом, який дозволяє управляти інфраструктурою, як програмним кодом, створений HashiCorp. Він дозволяє користувачам визначати та надавати

інфраструктуру центру обробки даних за допомогою мови конфігурації високого рівня, відомої як мова налаштування Hashicorp (HCL), або, при необхідності, JSON. Terraform підтримує ряд постачальників хмарної інфраструктури, таких як веб-служби Amazon, IBM Cloud (раніше Bluemix), Google Cloud Platform, Linode, Microsoft Azure, Oracle Cloud або VMware vSphere, а також OpenStack.

Файли конфігурації описують для Terraform компоненти, необхідні для запуску однієї програми або навіть цілого набору сервісів чи веб-додатків. Terraform генерує план виконання, що описує, що він буде робити для досягнення бажаного стану, а потім виконує його для побудови описаної хмарної інфраструктури. При зміні конфігурації Terraform може визначити, що змінилося, і створити додаткові плани виконання, які можуть бути застосовані. Terraform може керувати всією інфраструктурою, включаючи компоненти низького рівня, такі як обчислювальні екземпляри, сховище та мережу, а також компоненти високого рівня, такі як записи DNS, функції SaaS тощо.

Основними перевагами Terraform є:

— плани виконання: Terraform має спеціальний етап планування, в якому він формує план виконання розгортання. Цей план показує, що буде робити Terraform під час запуску. Це дозволяє уникнути будь-яких помилок, коли Terraform маніпулює інфраструктурою;

— графік ресурсів: Terraform створює графік всіх ваших ресурсів і паралельно створює і змінює будь-які незалежні ресурси. Через це Terraform будує інфраструктуру якомога ефективніше, і оператори отримують інформацію про залежності в їхній інфраструктурі.

— автоматизація змін: комплексні набори змін можуть бути застосовані до вашої інфраструктури з мінімальною людською взаємодією. З попередньо згаданим планом виконання та графіком ресурсів, користувач точно знає, що Terraform буде змінювати і в якому порядку, уникаючи багатьох можливих людських помилок.

Ще одним інструментом с безперервного розгортання є Chef (https://www.chef.io). Це інструмент керування конфігурацією, написаним на Ruby

та Erlang. Він використовує Ruby, домен-специфічну мову (DSL) для написання конфігурації системи, які називаються "рецептами". Chef використовується для оптимізації задач налаштування та обслуговування серверів компанії, а також для інтеграції з хмаровими платформами, такими як Internap, AWS EC2, Google Cloud Platform, Oracle Cloud, OpenStack, SoftLayer, Microsoft Azure та Rackspace налаштувати нові машини. Chef містить рішення як для малих, так і великомасштабних систем з розширеними функціями за додаткову ціну.

Користувач створює спеціальні «рецепти» («receipts»), які описують, як Chef буде керувати налаштуванням серверних додатками і утиліт (наприклад, Apache HTTP Server, MySQL або Hadoop). Ці рецепти описують ряд ресурсів у вигляді пакетів, які повинні бути встановлені, служб, які повинні бути запущені, або файлів, які повинні бути записані. Ці різноманітні ресурси можуть бути налаштовані для запуску конкретних версій програмного забезпечення і можуть гарантувати, що програмне забезпечення встановлено у правильному порядку на основі необхідних залежностей. Chef гарантує, що кожен ресурс буде правильно налаштований і переустановлює будь-які ресурси, які не знаходяться у бажаному стані чи версії.

Chef може працювати в режимі «клієнт-сервер» або в окремій конфігурації під назвою «chef-solo». У режимі «клієнт-сервер» агент Chef посилає різні атрибути про хост, на якому працює, на сервер Chef-сервер. Він використовує Elasticsearch для індексування атрибутів і надає клієнтам API для можливостей запиту цієї інформації. Рецепти можуть запитувати ці атрибути і використовувати отримані дані, щоб допомогти налаштувати окремий вузол інфраструктури веб-додатка. Традиційно, Chef використовувався для управління тільки OS Linux, але самі нові версії також підтримують Microsoft Windows.

Сьогодні досвідченим користувачам Інтернету надається великій вибір різноманітних інструментів для спрощення процесу розгортання як простих так і складних веб-додатків. Але як рядовий користувач Інтернету, який не володіє базисом знань у системному адмініструванні, не зможе повноцінно використовувати функції, які надаються цими інструментами. Більш того на

великих та більш об’ємних інфраструктурах для цих задач виділяються окремі інженери, які займаються тільки задачами розгортання. Створення скриптів автоматизації розгортання також займає доволі великий час, тому для деяких задач це просто невиправдане витрачення ресурсів.

Перейдемо до більш детального розглядання проблеми та проведемо дослідження, результати якого можна буде практично використати для подальшого створення прототипу системи с швидкого розгортання веб-додатків. Треба проаналізувати за допомогою яких традиційних та автоматизованих інструменті можна провести повноцінне розгортання веб-додатку на декілька хмарних середовищ та у подальшому використати їх особливості у прототипуванні системи.

2 ОСОБЛИВОСТІ ПРОЦЕСУ РОЗГОРТАННЯ ВЕБ-ДОДАТКІВ

2.1 Аналіз проблеми

Як вже було зазначено раніше, у епоху діджиталізації проходить так званий «Інтернет»-бум, коли все більше країн, бізнесів у цих країнах та навіть деякі державні органи переходять у режим онлайн-взаємодії. Бо це легше, практичніше, потребує менших затрат на більшість операційних дій, але в той же час потребує доволі великих інвестицій. Сучасні IT компанії з розробки веб-додатків, які створюють дійсно якісний продукт, коштують дуже дорого. Для розробки веб-додатка у релизі MVP (Minimum Viable Product - мінімально життєздатний продукт) вони потребують від 3 до 4 місяців, а для більш складних систем – від півроку. Реальний чек за такий веб-додаток починається від 50 000 долларів США, та не всі користувачі можут це дозволити.

Розгортання також є доволі дорогим процесом, але дедалі коротшим. Деякі прості веб-додатки досвідчений експерт зможе розгорнути за 1 робочий день. Якщо ж йде мова про більш складні інфраструктури, де задіяно декілька різноманітних сервісів та компонентів, взаємодіючих між собою [8].

Зараз розглянемо приклад веб-додатка на основі CMS WordPress. Більшість сайтів створних на даному движку є або простими сайтами-візитками, або доволі складними системами с обробки великої дій. Так для WordPress існує спеціальний плагін WooCommerce, який дозволяє створювати онлайн магазини на своєму базисі.

Базова інфраструктура веб-додатка для доволі популярного онлайн-магазину, з великою кількістю одночасних користувачів та просто відвідувачів буде виглядати, як можна бачити на рисунку (рис. 2.1).

У якості провайдера приватних хмар розглянемо AWS – Amazon Web Services (https://aws.amazon.com). AWS надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки. Існує і безкоштовна підписка, яка доступна протягом перших 12 місяців. Технологія

дозволяє абонентам мати у своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет.

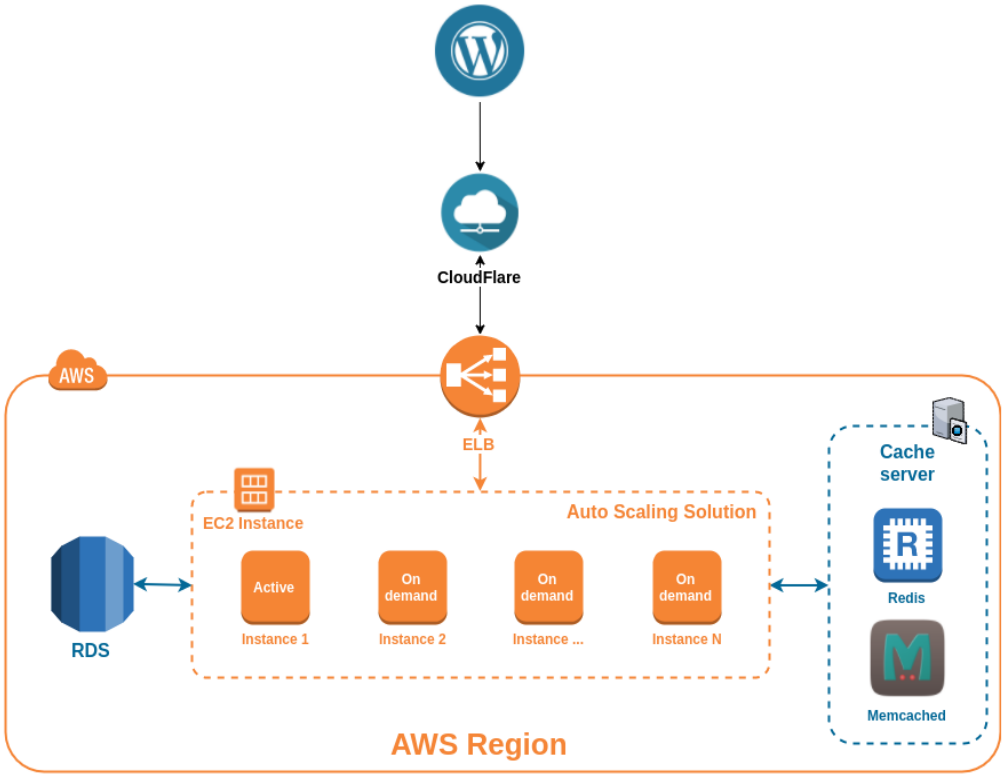


Рис. 2.1 – інфраструктура веб-додатка на основі WordPress у AWS

Віртуальні комп'ютери AWS мають більшість атрибутів реального комп'ютера, включаючи апаратні пристрої (процесор, відеокарту, локальну та оперативну пам'ять, жорсткий диск або SSD-накопичувач) [9]; операційну систему на вибір; мережу; і попередньо встановлені прикладні програми, такі як веб-сервер, база даних, CRM і т.д. Кожна система AWS також віртуалізує консольний ввід/вивід (клавіатура, дисплей і миша), що дозволяє користувачам AWS підключитися до своєї системи AWS за допомогою браузера. Браузер виступає як вікно у віртуальний комп'ютер, дозволяючи користувачу входити в систему, налаштовувати та використовувати свої віртуальні системи так само, як справжній, фізичний комп'ютер. Це дозволяє їм налаштувати систему так, щоб надавати інтернет-орієнтовані сервіси та послуги своїм клієнтам.

Розглянемо компоненти системи та їх функцій:

— CloudFlare (https://www.cloudflare.com/) - надає мережеві послуги доставки контенту, пом’якнешшя DDoS атак, службу безпеки в Інтернеті та сервіси розподілених доменних імен.

— AWS ELB – сервіс еластичного балансування навантаження автоматично розподіляє трафік вхідних додатків по декількох хостах, наприклад, Amazon EC2 хмарних контейнерах. Він може обробляти різне навантаження вашого трафіку додатків в одній зоні доступності або на декількох зонах доступності. Еластичне балансування навантаження пропонує три типи балансування навантаження, які мають високу доступність, автоматичне масштабування та надійну безпеку, необхідну для того, щоб зробити ваші програми стійкими до відмов.

— Amazon Elastic Compute Cloud (Amazon EC2) - веб-сервіс, котрий надає обчислювальні потужності в хмарі. Сервіс входить в інфраструктуру Amazon Web Services. Простий веб-інтерфейс сервісу дозволяє отримати доступ до обчислювальних потужностей і налаштувати ресурси з мінімальними затратами. Він надає користувачам повний контроль над обчислювальними ресурсами, а також доступне середовище для роботи. Служба скорочує час, необхідний для отримання і завантаження нового сервера. У даному прикладі ці хмарні контейнери об’єднані у одну групу, якою керує сервіс AWS ELB. Він створює чи знищує хмарні інстанси в залежності від навантаження на CPU, оперативну пам’ять, тощо. В його налаштуваннях є правила, якими описуються як повинно проходити розширення потужності, або її зменшення.

— Amazon Relational Database Service (Amazon RDS) дозволяє легко налаштовувати, керувати та масштабувати реляційні бази даних у хмарі. Сервіс забезпечує економічну та змінювану потужність, одночасно автоматизуючи тимчасові адміністративні завдання, такі як забезпечення обладнання, налаштування бази даних, виправлення та резервне копіювання. Користувач можете налаштувати їх продуктивність, високу доступність, безпеку та сумісність с іншими сервісами.

— Cache server – це окрема хмара, яка виконує функції кешування для запитів та даних веб-додатка WordPress. Цей сервер має 2 сервіси, яка працюють автономно один від одного: Memcached і Redis.

Memcached (https://memcached.org) – це сервіс кешування даних в оперативній пам'яті на основі парадигми розподіленої хеш-таблиці. З допомогою клієнтської бібліотеки (для Perl, PHP, Python, Java та ін.) дозволяє кешувати дані в ОЗП одного або декількох серверів. Розподіл даних реалізується по значенню хеш ключа. Використовуючи ключ даних, клієнтська бібліотека визначає його хеш і використовує його для вибору відповідного сервера. Ситуація збою сервера трактується як промах кеша. Це дозволяє, зокрема, проводити гарячу заміну серверів.

В API Memcached є тільки базові функції: вибір сервера, установка з'єднання, додання, видалення, оновлення і отримання об'єкта. Для кожного об'єкта встановлюється час актуальності, починаючи з 1 секунди до нескінченності. При переповненні пам'яті застарілі об'єкти кеша автоматично знищуються. Сервер Memcached було розроблено для сайту LiveJournal з метою зниження навантаження на сервери баз даних.

Redis (https://redis.io) - це розподілене сховище пар виду «ключ-значення», які зберігаються в оперативній пам'яті, з можливістю забезпечувати довговічність зберігання за бажанням користувача. Це програмне забезпечення з відкритим сирцевим кодом написане на ANSI C.

Redis надає схожі на Memcached функції для зберігання даних, розширені підтримкою структурованих даних, таких як списки, хеші і множини. На відміну від Memcached, Redis забезпечує постійне зберігання даних на диску і гарантує збереження БД у разі аварійного завершення роботи. Клієнтські бібліотеки доступні для більшості популярних мов, включаючи Perl, Python, PHP, Java, Ruby і Tcl.

Така інфраструктура веб-додатка на CMS WordPress с даною конфігурацією витримує навантаження у 3 000 000 користувачів щодня, це 2000 нових користувачів щохвилини! Для того, щоб розгорнути такий набір серверів на

сервісів на платформі AWS експертів (middle / senior DevOps engineer) дають такі середні оцінки.

Таблиця 1 – Середні оцінки задач на створення інфраструктури на AWS

№	Задача	Мін. (год)	Макс. (год)
1	Створення AMI образу для EC2	3	4
2	Створення та настройка групи авто-розширення	1	2
3	Настройка та створення ELB	0.5	1
4	Настройка правил розширення	1	3
5	Створення шаблона запуску для групи авто-розширення	1	2
6	Створення та налаштування RDS	0.5	1
7	Створення та налаштування Cache server	2	3
8	Установка та налаштування Redis	1	2
9	Установка та налаштування Memcached	1	2
10	Настройка CloudFlare (DNS, WAF, фільтрація трафіку)	2	3
11	Тестування	5	10
	Всього	18	33

Тобто досвідченому інженеру з розгортання веб-додатків потрібно від 4 до 5 робочих днів, щоб створити, налаштувати, протестувати (враховуючи усі ризики) та повноцінно розгорнути веб-додаток у виробниче середовище [10].

AWS економить час на налаштування деяких компонентів через свої сервіси, як от наприклад Amazon Machine Image (AMI) надає провайдеру інформацію, необхідну для запуску хмари на AWS. Можна запустити декілька хмар з одного AMI, якщо потрібно кілька хмар з однаковою конфігурацією. Ви можете використовувати різні AMI для запуску інстансів, коли вони вам потрібні з різними конфігураціями.

Дана інфраструктура є доволі гарним прикладом складного веб-додатку, який розрахований на різкі та великі приливи трафіку. Більшість традиційних налаштувань серверу чи сервісів автоматизовані завдяки панелі адміністратора AWS чи AWS Console (див. рис. 2.2).

Проблема криється у тому, що навіть надання доволі зручного GUI для управління хмаровим сховищем все ще не надає можливості повноцінно управління та менеджменту, як свого веб-додатка так і самого хмарового середовища, бо це більше розраховано на досвідчених реліз інженерів та відповідних спеціалістів, а не на рядового користувача. Для порівняння давайте розглянемо процес налаштування веб-додатку WordPress на одному сервері чи хмарі, щоб зрозуміти наскільки легший цей процес від налаштування попередньої інфраструктури.

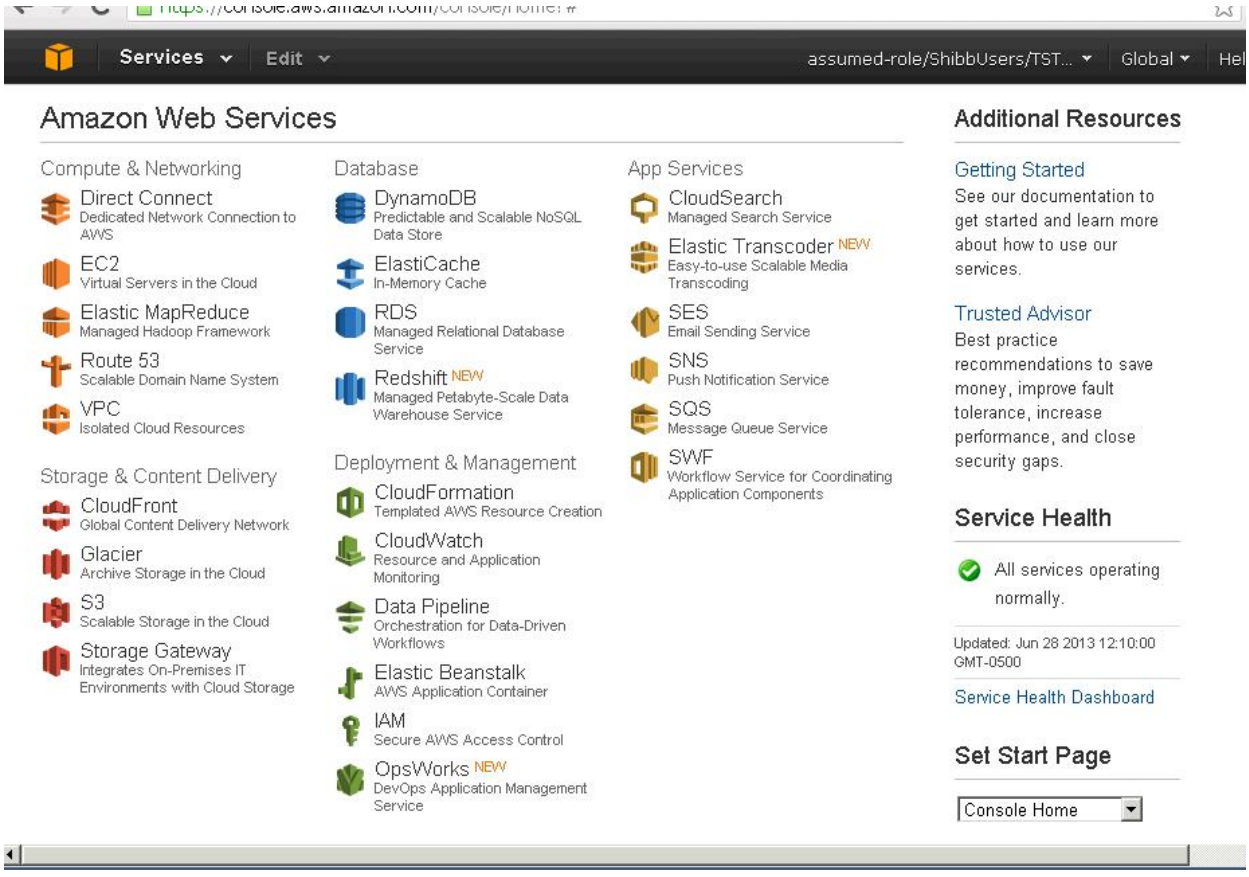


Рис. 2.2 – AWS Console

2.2 Дослідження традиційного методу розгортання на прикладі веб-додатка WordPress

Інакше це можна назвати – полу-автоматичний спосіб розгортання. У якості веб-додатку буде виступати одна з самих розповсюджених систем менеджменту контенту – WordPress (https://wordpress.org). Класичні методи розгортання у більшості випадків потребують знань та досвід роботи з інструментами командного рядку (CLI). Існує декілька інструментів та підходів, тому перед початком необхідно визначитися стратегією розгортання бази даних і медіа-бібліотеки.

Але перш ніж ми почнемо дослідження традиційного способу розгортання веб-додатку, нам потрібно мати сервер призначення, де ми розгорнемо наш сайт WordPress. Наприклад, це може бути приватна хмара провайдеру Digital Ocean (https://www.digitalocean.com), який надає розробникам хмарні різні хмарні сервіси, як AWS. Вони допомагають розгортати і масштабувати програми, які одночасно працюють на декількох комп'ютерах.

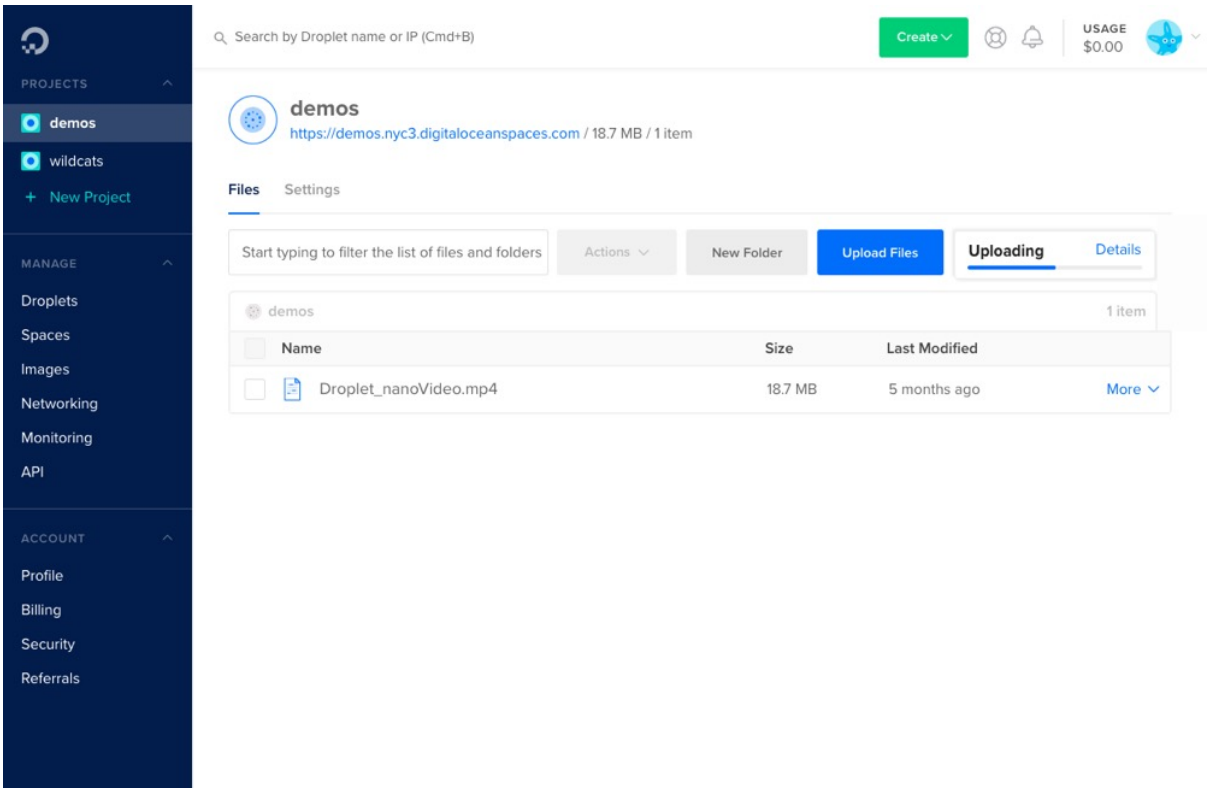


Рис. 2.3– інтерфейс Digital Ocean

Станом на січень 2018 року, компанія DigitalOcean стала третьою за величиною компанією з розміщення в Інтернеті з точки зору веб-комп'ютерів. Графічний інтерфейс провайдеру Digital Ocean (див. рис. 2.3) доволі сильно відрізняється від AWS та не надає таких широких функцій для користувачів. З іншого боку він дешевший, ніж AWS та розрахований на малі та середні бізнеси.

Підняття «чистої» хмари з операційною системою Ubuntu є доволі швидким процесом та дедалі легшим процесом ніж на AWS. Треба лише замовити хмару, яка отримає публічну IP адресу, на яку ви зможете ввійти через SSH. Окрім цього приватна хмара повинна мати певні налаштування, набір пакетів та установлений веб-сервер, який буде обробляти запити с веб-додатку. Традиційний метод розгортання веб-додатку можна поділити на такі кроки:

— установка та налаштування веб-сервера;

— установка та налаштування бази даних;

— установка та налаштування PHP;

— додаткові передналаштування серверу;

— установка та налаштування WordPress.

Перейдемо до розглядання кожного крока розгортання у деталях.

2.2.1 Установка та налаштування веб-сервера

Перш ніж розпочати установку веб-додатку WordPress, потрібно виконати кілька дій. Перша з них – це установка на налаштування веб-сервера - сутності, що має певне програмне забезпечення, підключене до інфраструктури для хостингу або обслуговування веб-додатків, які часто використовується разом з PHP, Perl та іншими скриптовими мовами сценаріїв. Клієнти дістаються веб-сервера за URL-адресою потрібної їм веб-сторінки або іншого ресурсу. Існує багато веб-серверів. Набір веб-серверів сьогодні доволі широкий, але не всі з них повноцінно функціонують, бо їх розробку та покращення вже не підтримують.

— NCSA HTTPd (http://hoohoo.ncsa.uiuc.edu/) - один із перших веб-серверів, розроблений Робертом Маккулом (англ. Robert McCool) та іншими у компанії NCSA;

— Apache HTTP-Server (https://httpd.apache.org) - найпоширеніший у світі веб-сервер з відкритим сирцевим кодом;

— IIS (https://www.iis.net) - веб-сервер компанії Microsoft, розповсюджується з ОС сімейства Windows NT;

— lighttpd (https://www.lighttpd.net) - open-source веб-сервер;

— Google Web Server - веб-сервер, створений на основі Apache компанією Google;

— Resin (https://caucho.com) - open-source сервер для застосувань java;

— Cherokee (http://cherokee-project.com) - вільний багатоплатформовий веб-сервер, написаний на C;

— Rootage (http://hostimum.com/rootage) - багатоплатформовий веб-сервер, написаний на java;

— THTTPD (https://acme.com/software/thttpd/) - простий, маленький, швидкий, переносний і добре захищений веб-сервер, розроблений для Unix-систем;

— GlassFish (https://javaee.github.io/glassfish/) - Java EE сервер застосунків з відкритим кодом, розроблений компанією Sun Microsystems.

Для веб-додатку WordPress найчайстіше використовується Apache HTTP-server, тому розглянемо процес його налаштування. Будемо вважати, що сервер, на який буде встановлений веб-сервер працює на OS Ubuntu. Для установки Apache2 на Ubuntu, у CLI операційної системи на сервері виконуються такі команди:

```
$ sudo apt install apache2
```

Дуже важливо вимкнути лістинг каталогів серверу, бо це може привести до несанкціонованого доступу до ваших даних:

```
$ sudo sed -i "s/Options Indexes FollowSymLinks/Options FollowSymLinks/" /etc/apache2/apache2.conf
```

Наступні кроки можуть бути дуже різноманітними, бо це залежить від певних налаштувань, необхідних для конкретних потреб. Наприклад, установка глобальної директиви 'ServerName', додаткові налаштування брандмауера для дозволу проходження HTTP/HTTPS трафіку і так далі.

Для перевірки коректності установки, щоб переконатися, що все пройшло так, як планується, треба відвідати загальнодоступну IP-адресу вашого сервера у веб-браузері: http: // your_server_IP_address

Ви повинні побачити стандартну веб-сторінку Ubuntu 16.04 Apache, яка призначена для інформаційних та тестових цілей (див. рис. 2.4).

.

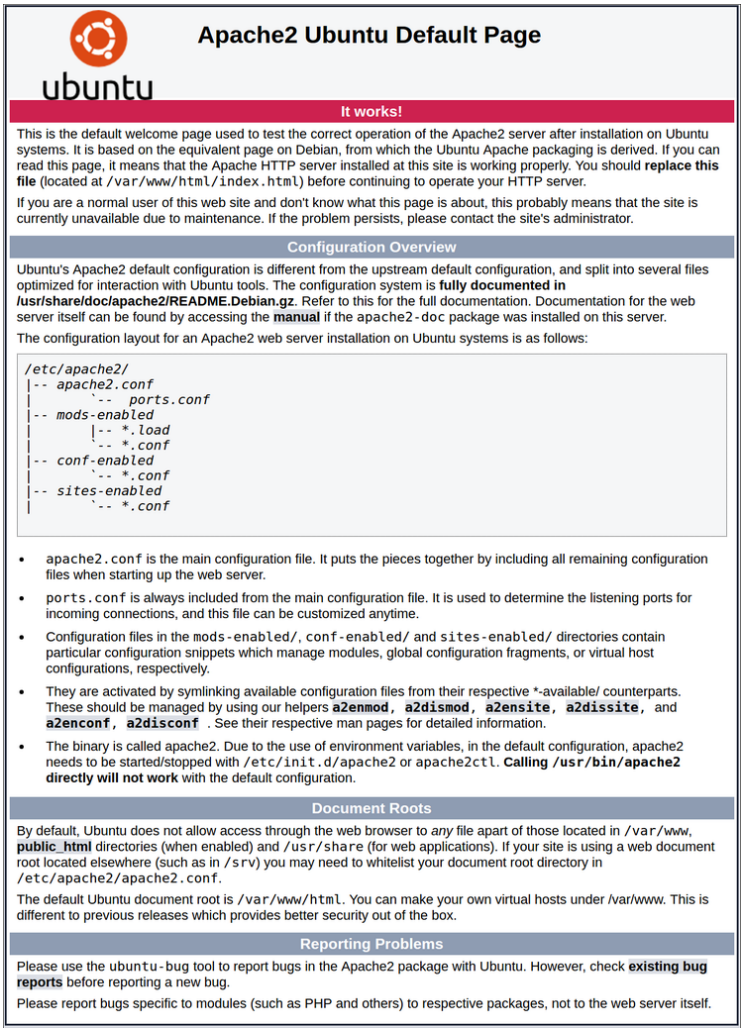


Рис. 2.4 – стандартна сторінка Apache2 на OS Ubuntu

2.2.2 Установка та налаштування бази даних

Після того, як веб-сервер налаштований, настав час встановити MySQL. MySQL - це система управління базами даних. В основному, вона організовуватиме та надаватиме доступ до баз даних, де наш сайт може зберігати інформацію. Сьогодні MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-додатків, оскільки має чудову підтримку з боку різноманітних мов програмування. MySQL сервер характеризується як компактний багатопотоковий інструмент з високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Основними можливостями сервера MySQL є:

— простота у встановленні та використанні;

— підтримка необмеженої кількісті користувачів, що одночасно працюють із БД;

— кількість рядків у таблицях може досягати 50 млн;

— висока швидкість виконання команд;

— наявність простої і ефективної системи безпеки.

Для установки ми можемо використовувати інструмент apt. Цього разу ми також встановимо інші пакунки "помічників", які допоможуть нам налаштувати компоненти сервера на спілкування один з одним:

```
$ sudo apt-get install mysql-server
```

Індекс упаковки на нашому комп'ютері має бути вже актуальним, бо при установці веб-сервера ми обновили усі необхідні пакети.

Під час інсталяції сервер попросить вас вибрати та підтвердити пароль для користувача "root" MySQL. Це адміністративний обліковий запис в MySQL, що має підвищені привілеї. Думайте про те, що він схожий на кореневий обліковий запис для самого сервера (той, який ви зараз налаштовуєте, є обліковим записом MySQL). Переконайтеся, що це сильний, унікальний пароль, і ні я якому разі не залишайте його порожнім.

Якщо ви використовуєте слабкий пароль у поєднанні з програмним забезпеченням, яке автоматично налаштовує облікові дані користувача MySQL, це призведе до виникнення проблем з пакетами Ubuntu для phpMyAdmin. Завжди слід використовувати сильні, унікальні паролі для облікових даних бази даних. Подальше налаштування бази даних у більшості пов’язані з безпекою, тому вони специфічні для кожного окремого веб-сайту.

2.2.3 Установка та налаштування PHP

На даний момент ваша система баз даних налаштована, і ми можемо рухатися далі. Крок 3 – це установка PHP на сервер. PHP є компонентом нашої установки, яка відповідає за обробляння коду для відображення динамічного вмісту. Він може запускати скрипти, підключатися до баз даних MySQL для отримання інформації, і передавати оброблений вміст на наш веб-сервер для подальшого відображення.

Для установки цього пакету виконати декілька apt команду у інтерфесі командної стрічки. При налаштуванні можна включити деякі допоміжні пакети, щоб PHP-код міг працювати під сервером Apache і говорити з нашою базою даних MySQL:

```
$ sudo apt-get ins php libapache2-mod-php php-mcrypt php-mysql
```

Це має встановити PHP без будь-яких проблем. У більшості випадків ми хочемо змінити спосіб, у який Apache подає файли, коли запитується певний

каталог. В даний час, якщо користувач запитує каталог з сервера, Apache спочатку шукає файл під назвою index.html. Ми хочемо сказати нашому веб-серверу, що він віддав перевагу файлам PHP, тому ми спочатку налаштуємо Apache на пошук файлу index.php. Для необхідно виконати команду, щоб відкрити файл dir.conf у текстовому редакторі з привілеями root:

```
$ sudo nano /etc/apache2/mods-enabled/dir.conf
```

Це буде виглядати так:

```
/etc/apache2/mods-enabled/dir.conf

<IfModule mod_dir.c>

DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm

</IfModule>
```

Тепер наша задача перемістити файл index.php, виділений вище, до першої позиції після специфікації DirectoryIndex, наприклад:

```
/etc/apache2/mods-enabled/dir.conf

<IfModule mod_dir.c>

    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm

</IfModule>
```

Після цього нам потрібно перезапустити веб-сервер Apache для того, щоб наші зміни були розпізнані. Для цього необхідно виконати команду:

```
$ sudo systemctl reload apache2
```

Ми також можемо перевірити статус служби apache2 за допомогою systemctl:

```
$ sudo systemctl status apache2
```

Sample Output

```
apache2.service - LSB: Apache2 web server

   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)

  Drop-In: /lib/systemd/system/apache2.service.d

           └─apache2-systemd.conf

   Active: active (running) since Wed 2016-04-13 14:28:43 EDT; 45s ago

     Docs: man:systemd-sysv-generator(8)

  Process: 13581 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)

  Process: 13605 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)

    Tasks: 6 (limit: 512)

   CGroup: /system.slice/apache2.service

           ├─13623 /usr/sbin/apache2 -k start

           ├─13626 /usr/sbin/apache2 -k start

           ├─13627 /usr/sbin/apache2 -k start

           ├─13628 /usr/sbin/apache2 -k start

           ├─13629 /usr/sbin/apache2 -k start

           └─13630 /usr/sbin/apache2 -k start


Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Stopped LSB: Apache2 web server.

Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Starting LSB: Apache2 web server...
```

```
Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]: * Starting Apache httpd web server apache2

Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerNam

Apr 13 14:28:43 ubuntu-16-lamp apache2[13605]: *

Apr 13 14:28:43 ubuntu-16-lamp systemd[1]: Started LSB: Apache2 web server.
```

Остання стрічка виводу команди показує, що Apache2 сервер запустився без помилок та можна переходити до наступного етапу.

Четвертою частиною установки та налаштування сервера Apache є установка додаткових модулів PHP, щоб підвищити функціональність мови. Щоб побачити доступні параметри для модулів і бібліотек PHP, ви можете передати результати пошуку apt-cache:

```
$ apt-cache search php- | less
```

Результатом цієї команди буде вивід всіх додаткових компонентів, які можна встановити:

```
php-all-dev - package depending on all supported PHP development packages

php-cgi - server-side, HTML-embedded scripting language (CGI binary) (default)

php-cli - command-line interpreter for the PHP scripting language (default)

php-common - Common files for PHP packages

php-curl - CURL module for PHP [default]

php-dev - Files for PHP module development (default)

cakephp-scripts - MVC rapid application development framework for PHP (scripts)
```

doctrine - transitional dummy package for php-doctrine-orm

kdevelop-php-dbg - debugging symbols for the KDevelop PHP plugin

kdevelop-php-docs - PHP documentation plugin for KDevelop

kdevelop-php-l10n - localization files for KDevelop PHP plugin

php-gd - GD module for PHP [default]

php-gmp - GMP module for PHP [default]

Під час дослідження нам я вирішив встановити певний набір пакетів. Це можна зробити використовуючи команду apt-get install, як ми робили для нашого іншого програмного забезпечення в цій роботі. Я вирішив встановити пакети php-cli, php-curl, php-dev, php-cgi та php-all-dev. Для їх установки необхідно виконати команду:

$ sudo apt-get install php-curl php-gd php-mbstring php-mcrypt php-xml php-xmlrpc

На цьому етапі встановлено та налаштовано стек LAMP. LAMP — абревіатура набору програмного забезпечення з відкритим кодом, в який входять ОС Linux, веб-сервер Apache, СКБД MySQL, та інтерпретатор PHP (в нашому випадку)/Python/Perl.

2.2.4 Тестування

Як ми вже зрозуміли, це основні компоненти для побудови життєздатного багатоцільового веб-сервера для веб-додатків. Проте ми повинні перевірити коректність роботи модулів PHP. Щоб перевірити, що наша система налаштована належним чином для PHP, ми можемо створити дуже простий скрипт PHP.

Цей скрипт будемо називати info.php. Для того, щоб Apache знаходив файл і міг коректно його обробити, він повинен бути збережений в дуже специфічній директорії, яка називається "web root". У Ubuntu 16.04 цей каталог розташований за адресою /var/www/html/. Ми можемо створити файл у цьому місці, набравши:

```
$ sudo nano /var/www/html/info.php
```

Відкриється порожній файл. Ми хочемо розмістити у файлі наступний текст, який є дійсним кодом PHP:

```
info.php
<? php
phpinfo ();?>
```

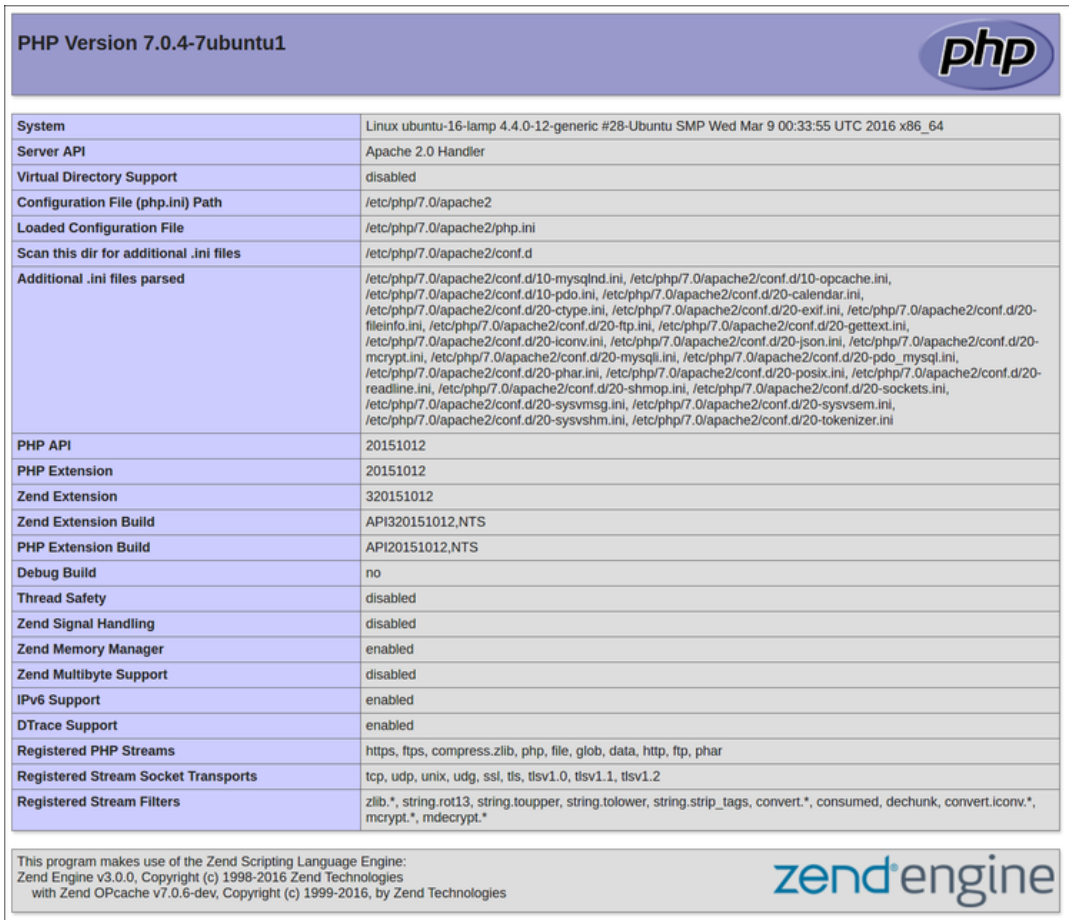
Після завершення збережіть та закрийте файл. Тепер ми можемо перевірити, чи може наш веб-сервер правильно відображати вміст, створений скриптом PHP. Для цього нам потрібно відвідати цю сторінку в нашому веб-браузері через загальнодоступну IP-адресу вашого сервера http: //your_server_IP_address/info.php. Створена сторінка (див. рис. 2.5) в основному дає вам інформацію про ваш сервер з точки зору PHP. Це корисно для налагодження та для забезпечення правильного застосування налаштувань. Якщо сторінка відкрилась успішно, то ваш PHP працює як слід. Можливо, ви захочете видалити цей файл після цього тесту, оскільки він може фактично надавати інформацію про ваш сервер неавторизованим користувачам.

Ви завжди можете відтворити цю сторінку, якщо вам потрібно отримати доступ до інформації пізніше. Для видалення сторінки необхідно виконати команду:

```
$ sudo rm /var/www/html/info.php
```

На цьому установка та перевірка LAMP пакету закінчена. Тепер можна перейти к установці та налаштуванню веб-додатку WordPress. Але перед цим існує декілька рекомендацій, які допоможуть уникнути проблем у майбутньому. Одна з головних рекомендацій - це покупка SSL сертифікату: WordPress обслуговує

динамічний контент і обробляє аутентифікацію та авторизацію користувачів. TLS / SSL - це технологія, яка дозволяє шифрувати трафік з веб-додатку, щоб з'єднання користувачів було безпечним. Але перед покупкою сертифікату треба придбати доменне ім’я. Існує також можливість використання безкоштовних сертифікатів, але вони дійсні лише місяць, тому власнику веб-додатку доведеться постійно обновлювати SSL. Платний сертифікат діє 1 рік.



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

Рис. 2.5 – сторінка index.php

2.2.6 Додаткові налаштування серверу

Першим кроком установки WordPress на сервер є створення бази даних MySQL і користувача для WordPress. Це підготовчий крок, бо WordPress використовує MySQL для керування та зберігання інформації про сайт і користувача. У нас вже встановлений MySQL, але тепер необхідно створити базу

даних і користувача для використання WordPress. Щоб розпочати роботу, треба увійти до кореневого облікового запису MySQL, ввівши цю команду:

```
$ mysql -u root -p
```

Під час установки програмного забезпечення вам буде запропоновано встановити пароль для облікового запису кореневої системи MySQL. По-перше, ми можемо створити окрему базу даних з назвою wordpress, якою може керувати веб-додаток. Для цього необхідно виконати команду:

```
CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

Далі необхідно створити окремий обліковий запис користувача MySQL, який буде використано виключно для роботи з новою базою даних. Створення однофункціональних баз даних і облікових записів є гарною ідеєю з точки зору управління та безпеки. Назвемо його wordpressuser. Тепер треба створити цей обліковий запис, встановити пароль і надати доступ до створеної бази даних. Це можна зробити за допомогою команди:

```
GRANT ALL ON wordpress.* TO'wordpressuser'@'localhost' IDENTIFIED BY 'password';
```

Тепер база даних і обліковий запис користувача створені. Та кожен з них створений спеціально для WordPress. Потрібно видалити привілеї, щоб поточний екземпляр MySQL знав про нещодавні зміни, які ми внесли. Використаємо команду:

```
$ FLUSH PRIVILEGES;
```

Наступним кроком буде налаштування конфігурації Apache, таким чином, щоб дозволити перевизначення та перезапису файлу .htaccess. Для цього необхідно зробити декілька незначних коригувань конфігурації Apache. Наразі використання файлів .htaccess вимкнено. А WordPress і багато плагіни, створені для WordPress, широко використовують ці файли для налаштування в каталозі поведінки веб-сервера. Крім того, ми необхідно ввімкнути mod_rewrite, який необхідний для того,

щоб постійні посилання працювали у WordPress працювали так як треба. Щоб увімкнути так зване заміщення .htaccess необхідно відкрити основний конфігураційний файл Apache, щоб зробити першу зміну:

```
$ sudo nano /etc/apache2/apache2.conf
```

Щоб дозволити файли .htaccess, нам потрібно встановити директиву AllowOverride в блоці Directory, що вказує на наш кореневий документ. Донизу файлу додайте наступний блок:

```
/etc/apache2/apache2.conf
. . .
<Directory / var / www / html />
    AllowOverride All
</Directory>
. . .
```

Наступним кроком буде ввімкнення mod_rewrite. Модуль mod_rewrite використовує механізм перезапису на основі правил, заснований на аналізаторі регулярних виразів PCRE, щоб надавати можливість переписувати потрібні URL-адреси на льоту. За замовчуванням, mod_rewrite відображає URL на шлях файлової системи. Однак його можна також використовувати для перенаправлення однієї URL-адреси на іншу URL-адресу або для виклику внутрішнього вибору проксі-сервера.

mod_rewrite надає гнучкий і потужний спосіб керувати URL-адресами за допомогою необмеженої кількості правил. Кожне правило може мати необмежену кількість умов доданого правила, щоб дозволити URL-адресу на основі серверних змінних вам перезаписувати, змінних оточення, заголовків HTTP або міток часу.

mod_rewrite працює на повному шляху URL, включаючи розділ info-шляху. Правило перезапису можна викликати в файлах httpd.conf або в .htaccess. Шлях, згенерований правилом перезапису, може включати рядок запиту, або може призводити до внутрішньої обробки, перенаправлення зовнішнього запиту або

внутрішньої проксі. Щоб ввімкнути mod_rewrite, який буде використаний для перманентних посилань у Wordpress:

```
$ sudo a2enmod rewrite
```

Перш ніж реалізувати внесені зміни, треба переконатися, що зміні файлів не було внесено жодних синтаксичних помилок:

```
$ sudo apache2ctl configtest
```

Вихідні дані можуть мати таке повідомлення:

```
Output

AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message

Syntax OK

If you wish to suppress the top line, just add a ServerName directive to the /etc/apache2/apache2.conf file pointing to your server's domain or IP address. This is just a message however and doesn't affect the functionality of our site. As long as the output contains Syntax OK, you are ready to continue.
```

Останнім кроком тут буде перезапуск веб-сервера Apache:

```
$ sudo systemctl reload apache2
```

2.2.7 Установка та налаштування WordPress

На цьому налаштування веб-сервера закінчуються та можна перейти к установці веб-додатку WordPress. З міркувань безпеки, зокрема, завжди рекомендується отримати останню версію WordPress з офіційного сайту. Для цього

необхідно перейти до каталогу для запису, а потім завантажити стиснений реліз, ввівши:

```
$ cd / tmp
$ curl -O https://wordpress.org/latest.tar.gz
```

Теперь треба витягнути стислий файл для створення структури каталогів WordPress:

```
$ tar xzvf latest.tar.gz
```

Команда вище миттєво перемістить ці файли в кореневий документ. Перед тим, як це зробити, ми можемо додати фіктивний файл .htaccess і встановити його дозволи, щоб це було доступним для використання додатком WordPress пізніше. Також скопіюємо примірний файл конфігурації до імені файлу, який WordPress фактично читає. Ще однією рекомендацією є створення каталогу оновлення, так що WordPress не зможе зіткнутися з проблемами дозволів при спробі зробити це самостійно після оновлення програмного забезпечення:

Щоб створити файл та встановити дозволи треба виконати команду:

```
$ touch /tmp/wordpress/.htaccess
$ chmod 660 /tmp/wordpress/.htaccess
$ cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php
$ mkdir / tmp / wordpress / wp-content / upgrade
```

Тепер ми можемо скопіювати весь вміст каталогу в кореневий документ. Обов’язково треба використати прапорець -a, щоб переконатися, що дозволи зберігаються. А крапку в кінці нашого вихідного каталогу використано, щоб вказати, що все в каталозі має бути скопійовано, включаючи приховані файли (наприклад, файл .htaccess, створений раніше):

```
$ sudo cp -a / tmp / wordpress /. / var /www / html
```

Тепер перейдемо до налаштувань каталогу WordPress, а саме до налаштування прав власності та дозволів. Створення розумних дозволів на доступ до файлів та володіння ними – є дуже важливим кроком у даній установці. Я повинен вміти писати до цих файлів як звичайний користувач, і а веб-сервер повинен надати можливість доступу і налаштування певних файлів і каталогів для правильної роботи. Починати необхідно з призначення права власності на всі файли в корені до імені користувача. В цьому дослідженні будемо використовувати sammy як ім'я користувача. Групі даних www буде присвоєно право власності на групи:

```
$ sudo chown -R sammy: www-data / var / www / html
```

Далі буде встановлений біт setgid на кожен з каталогів у корені документа. Це призводить до того, що нові файли, створені в цих каталогах, успадковують групу батьківського каталогу (яку ми просто встановили на www-data) замість початкової групи користувача. Це просто гарантує, що всякий раз, коли буде створений файл у каталозі в командному рядку, веб-сервер все одно матиме групове володіння над ним. Для того, щоб встановити біт setgid для кожного каталогу в установці WordPress необхідно ввести:

```
$ sudo find /var/www/html -type d -exec chmod g+s {} \;
```

Є ще кілька дозволів, які треба налаштувати По-перше, ми необхідно надати групі доступ до запису до каталогу wp-content, щоб веб-інтерфейс мав змогу змінювати теми та плагіни. Також треба надати веб-серверу доступ до запису до всього вмісту цих двох каталогів:

```
$ sudo chmod g + w / var / www / html / wp-content
$ sudo chmod -R g+w /var/www/html/wp-content/themes
$ sudo chmod -R g+w /var/www/html/wp-content/plugins
```

Цих налаштувань буде достатньо для дослідження, що проводиться. Тепер можна перейти до налаштування конфігураційного файлу WordPress. Тепер нам потрібно внести деякі зміни до головного конфігураційного файлу WordPress. При

відкритті файлу, першим завданням буде конфігурація деяких секретних ключів для забезпечення певного рівня безпеки для інсталяції. WordPress має генератор цих значень. Щоб отримати безпечні значення з генератора секретного ключа WordPress треба ввести команду:

```
$ curl -s https://api.wordpress.org/secret-key/1.1/salt/
```

```
Output

define('AUTH_KEY',         '1jl/vqfs<XhdXoAPz9 DO NOT COPY THESE VALUES c_j{iwqD^<+c9.k<J@4H');

define('SECURE_AUTH_KEY',  'E2N-h2]Dcvp+aS/p7X DO NOT COPY THESE VALUES {Ka(f;rv?Pxf})CgLi-3');

define('LOGGED_IN_KEY',    'W(50,{W^,OPB%PB<JF DO NOT COPY THESE VALUES 2;y&,2m%3]R6DUth[;88');

define('NONCE_KEY',        'll,4UC)7ua+8<!4VM+ DO NOT COPY THESE VALUES #`DXF+[$atzM7 o^-C7g');

define('AUTH_SALT',        'koMrurzOA+|L_lG}kf DO NOT COPY THESE VALUES  07VC*Lj*lD&?3w!BT#-');

define('SECURE_AUTH_SALT', 'p32*p,]z%LZ+pAu:VY DO NOT COPY THESE VALUES C-?y+K0DK_+F|0h{!_xY');

define('LOGGED_IN_SALT',   'i^/G2W7!-1H2OQ+t$3 DO NOT COPY THESE VALUES t6**bRVFSD[Hi])-qS`|');

define('NONCE_SALT',       'Q6]U:K?j4L%Z]}h^q7 DO NOT COPY THESE VALUES 1% ^qUswWgn+6&xqHN&%');
```

Дуже важливо щоразу запитувати унікальні значення. Ні в якому разі не треба намагатися копіювати наведені вище значення. Цей вивід можна вставити безпосередньо в конфігураційний файл, щоб встановити безпечні ключі. Тепер необхідно відкрити файл налаштування WordPress:

```
$ nano /var/www/html/wp-config.php
```

Зараз перейдемо у розділ, який містить фіктивні значення для цих налаштувань та замінимо існуючі рядки з ключами на значення, скопійовані з командного рядка раніше. Щоб перейти до розділу файлу треба виконати команду:

```
$ /var/www/html/wp-config.php
```

Далі потрібно змінити деякі параметри підключення до бази даних на початку файлу. Потрібно налаштувати назву бази даних, користувача бази даних і відповідний пароль, який був налаштований в MySQL раніше.

Іншою необхідною зміною є установка методу, який WordPress повинен використовувати для запису у файлову систему. Оскільки раніше я надав веб-серверу дозвіл на запис там, де це потрібно, я можу явно встановити метод файлової системи на "direct". Це налаштування можна додати під конфігурацією підключення до бази даних або в іншому місці файлу:

```
$ /var/www/html/wp-config.php
. . .
define('DB_NAME', 'wordpress');
/** MySQL database username */
define('DB_USER', 'wordpressuser');
/** MySQL database password */
define('DB_PASSWORD', 'password');
. . .
define('FS_METHOD', 'direct');
```

Тепер, коли налаштування сервера завершено, можна завершити установка та налаштування веб-додатку через зрозумілий веб-інтерфейс додатку WordPress. Для цього необхідно перейти доменного імені чи загальнодоступної адреси сервера, на якому працює веб-додаток. Перше, що він запропонує це вибір язику. Далі CMS направляє на головну сторінку налаштування (див. рис. 2.6).

Веб-додаток запропонує вибрати назву для сайту WordPress та ім'я користувача. Сильний пароль буде згенерований автоматично. Також треба ввести адресу електронної пошти та вибрати, чи потрібно відмовляти пошукові системи від індексації вашого сайту.

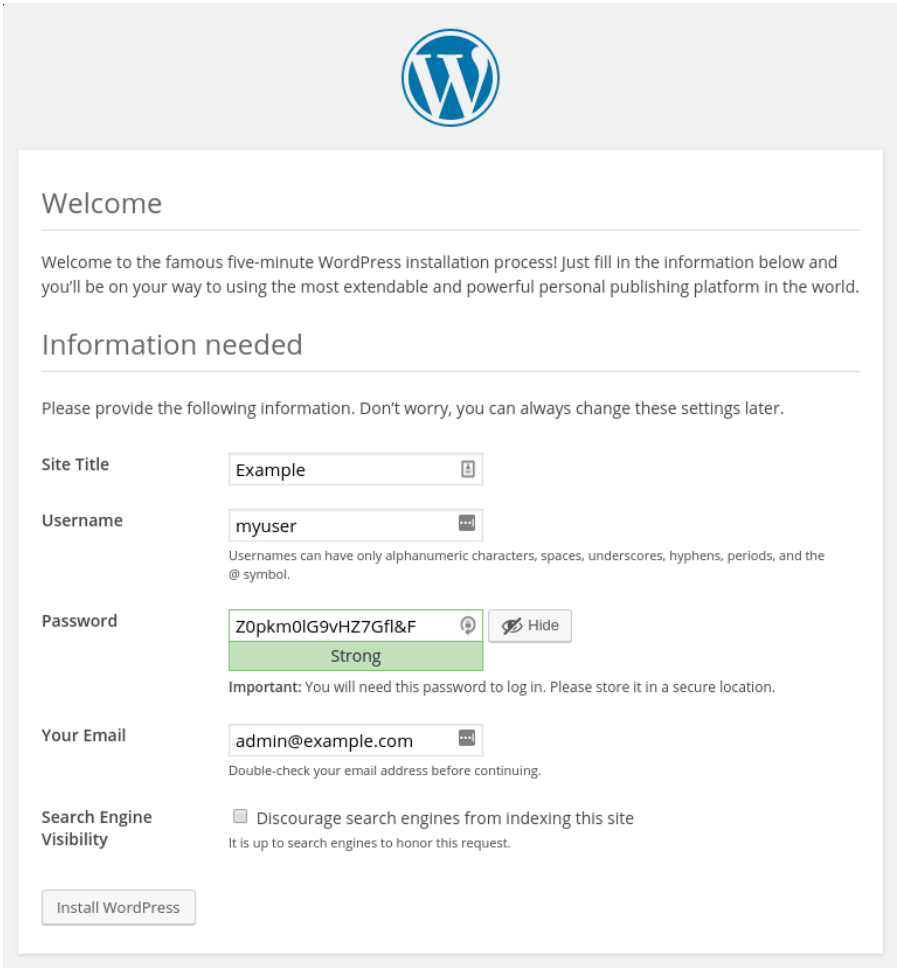


Рис. 2.6 – головна сторінка налаштування WordPress

Після завершення усих налаштуваннь веб-додаток перенаправить мене до авторизування, після якого повинна з’явитися головна сторінка CMS WordPress (рис. 2.7), на якій користувачу надається інтерфейс для кастомізації самого веб-додатка, контенту його сторінок та форм, а також для повноцінного налаштування плагінів, які розширюють фунціонал та можуть змінювати чи допопвнювати інтерфейс веб-додатка.

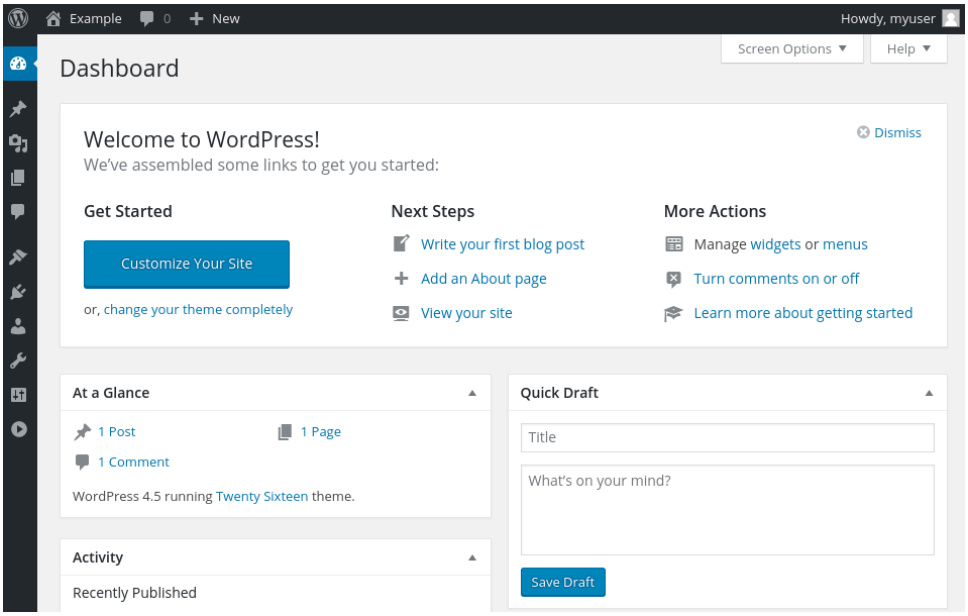


Рис. 2.7 – головна сторінка CMS WordPress

2.3 Дослідження автоматизації розгортання WordPress за допомогою Ansible playbook

В термінології Ansible є термін «playbook», який є файлом виконання. Він складається з набору хостів, на яких повинна виконуватися автоматизація, сервісів, які повинні бути створені. Як от наприклад, сервера MySQL і так далі. У нашому випадку єдиний хост буде відповідальним за всі ці ролі.

Спочатку потрібно створити каталог на хості, де будуть зберігатися всі наші конфігурації:

```
$ mkdir wordpress-playbook
$ cd wordpress-playbook
$ mkdir roles
$ touch hosts
$ touch playbook.yml
```

Каталог ролей і два пусті файли створені: hosts і playbook. Перше, що потрібно зробити - це відкрити файл hosts і додати потрібне ім'я (це не обов'язково повинно бути ім'ям цільового сервера, просто розумною міткою) разом з IP-адресою сервера, де буде встановлено WordPress.

В файл hosts потрібно додати наступний запис:

```
[wordpress]
wp_server_ip
```

Для установки на декількох серверах можна встановити декілька IP-адрес під [wordpress]. Наступним кроком буде визначення різних ролей в підкаталозі ролей:

```
$ cd roles
$ ansible-galaxy init server
$ ansible-galaxy init php
$ ansible-galaxy init mysql
$ ansible-galaxy init wordpress
```

Ці команди вносять шаблонні конфігурації для окремих компонентів з Ansible-galaxy, яка є сховищем для багатьох стандартних конфігурацій Ansible. Після виконання цих команд у папці wordpress-playbook/roles буде створено 4 папки: MySQL, PHP, server і WordPress. Тепер потрібно їх налаштувати.

Наступним кроком є створення ролей. Ansible інструкції написані в форматі .yml, які є доволі зрозумілі та читабельні навіть для людини без широких знань Ansible. Ролі Ansible будуть створені для кожної окремої сутності: стеку LAMP і та веб-додатку WordPress.

Спочатку проходить модифікація файлу Playbook.yml, який є файлом виконання та інструкцією для запуску автоматизації. Туди записано наступний вміст:

```
- hosts: all
  gather_facts: False

  tasks:
  - name: install python 2
    raw: test -e /usr/bin/python || (apt -y update && apt install -y python-minimal)

- hosts: wordpress

  roles:
    - server
    - php
    - mysql
    - wordpress
```

Дані команди встановлюють Python 2.7 на всі ваші цільові хости (що будуть керуватися Ansible), а потім переходить до призначення 4 ролей.

Першою буде налаштована роль для сервера. Для цього необхідно відкрити файл wordpress-playbook / roles / server / tasks / main.yml і додати наступний код:

```
# tasks file for server
- name: Update apt cache
  apt: update_cache=yes cache_valid_time=3600
become: yes
- name: Install required software
  apt: name={{ item }} state=present
```

```
  become: yes
  with_items:
    - apache2
    - mysql-server
    - php7.2-mysql
    - php7.2
    - libapache2-mod-php7.2
    - python-mysqldb
```

Тепер необхідно налаштувати PHP та встановити додаткові модулі Щоб це зробити необхідно додати наступний вміст у той самий файл wordpress-playbook / roles / php / tasks / main.yml:

```
# tasks file for php
- name: Install php extensions
  apt: name={{ item }} state=present
  become: yes
  with_items:
    - php7.2-gd
    - php7.2-ssh2
- php-curl
- php-gd
- php-mbstring
- php-mcrypt
- php-xml
- php-xmlrpc
```

Тепер можна перейти до налаштування MySQL, але перед початком створення бази даних MySQL потребуватиме додаткової інформації. У файлі wordpress-playbook / roles / mysql / defaults / main.yml треба додати наступне:

```
# defaults file for mysql
wp_mysql_db: wordpress
wp_mysql_user: wordpress
wp_mysql_password: #randompassword#
```

Після цього треба налаштовувати основну задачу для MySQL ролі, а саме: створення користувача, бази даних MySQL і надання користувачеві доступу до новоствореної бази даних. У файлі wordpress-playbook / roles / mysql / tasks / main.yml додамо наступні рядки:

```
- name: Create mysql database
  mysql_db: name={{ wp_mysql_db }} state=present
  become: yes

- name: Create mysql user
  mysql_user:
    name={{ wp_mysql_user }}
    password={{ wp_mysql_password }}
    priv=*.*:ALL
  become: yes
```

Останнім та найскладнішим кроком є настройка ролі для WordPress. По сценарію виконання ця роль повинна буде отримати файл установки WordPress з офіційного сайту, витягти його і змінити wp-config.php з відповідними даними. Для цього необхідно додати наступний код у wordpress-playbook / roles / wordpress / tasks / main.yml:

```
    - name: Download WordPress
      get_url:
        url=https://wordpress.org/latest.tar.gz
        dest=/tmp/wordpress.tar.gz
        validate_certs=no

    - name: Extract WordPress
      unarchive: src=/tmp/wordpress.tar.gz dest=/var/www/
copy=no
      become: yes

    - name: Update default Apache site
      become: yes

      lineinfile:
        dest=/etc/apache2/sites-enabled/000-default.conf
        regexp="(.)+DocumentRoot /var/www/html"
        line="DocumentRoot /var/www/wordpress"
      notify:
        - restart apache

    - name: Copy sample config file
      command: mv /var/www/wordpress/wp-config-sample.php
/var/www/wordpress/wp-config.php
creates=/var/www/wordpress/wp-config.php
```

```
    become: yes


- name: Update WordPress config file
    lineinfile:
      dest=/var/www/wordpress/wp-config.php
      regexp="{{ item.regexp }}"
      line="{{ item.line }}"
    with_items:
      - {'regexp': "define\\('DB_NAME', '(.)+'\\);",
'line': "define('DB_NAME', '{{wp_mysql_db}}');"}
      - {'regexp': "define\\('DB_USER', '(.)+'\\);",
'line': "define('DB_USER', '{{wp_mysql_user}}');"}
      - {'regexp': "define\\('DB_PASSWORD', '(.)+'\\);",
'line': "define('DB_PASSWORD', '{{wp_mysql_password}}');"}
    become: yes
```

І, нарешті, щоб перезапустити Apache та прийняти усі налаштування для WordPress, треба додати наступний фрагмент до файлу wordpress-playbook / roles / wordpress / handlers / main.yml:

```
# handlers file for wordpress
- name: restart apache
    service: name=apache2 state=restarted
    become: yes
```

Тепер, коли усі приготування завершені можна перейти до запуску Ansible playbook, яка встановить WordPress на цільовий хост. Для цього необхідно виконати наступну команду у CLI цільового хоста:

$ ansible-playbook playbook.yml -i hosts -u «ім'я користувача» -K

Ім’я користувача має бути замінене на буквальне ім'я користувача на хост-системі, що має привілеї sudo. Якщо все йде правильно, ви побачите такий вивід (рис. 2.8).

Рис. 2.8 – вивід після завершення виконання створеного Ansible Playbook

Тепер якщо у будь-який час вам потрібно встановити WordPress на сервер, достатньо додати його IP-файл у файлі wordpress-playbook / hosts, а потім знову запустити вказану вище команду. Тепер після відвідування публічної IP-адреси сервера, де розгортався веб-додаток, за допомогою веб-браузера, ви побачите головну сторінку налаштувань WordPress.

2.4 Постановка задачі

У попередніх розділах були розглянуті методи, підходи та інструменти розгортання простих та більш складних веб-додатків. Були визначені їх переваги, недоліки, особливості, можливості та вартість використання. Сучасні автоматизовані методи дозволяють значно скоротити час та трудоємкість процесу розгортання веб-додатків, але в той же час потребують доволі широкого базису фундаментальних та навіть більш глибоких знань з програмування та системного/серверного адміністрування. Тому сучасним малим та середнім

бізнесам, які не готови інвестувати великі гроші експертів та мають доволі лімітовані знання дуже розвиватися та виходити на більший рівень, бо прогрес йде перед ними. Беручі всі ці фактори до уваги, задачу дослідження можна сформувати таким чином:

— змоделювати прототип системи з швидкого розгортання веб-додатків;

— змоделювати архітектуру прототипу системи;

— визначити функціональні вимоги до системи;

— обрати оптимальне поєднання інструментів автоматизованого розгортання у рамках системи, що проектується;

— розробити прототип інтерфейсу системи, яка надала б користувачу можливість швидко розгортати різні веб-додатки на різні приватні хмари за допомогою GUI;

— дослідити процес розгортання веб-додатку на основі CMS WordPress на приватну хмару, використовуючи лише прототип платформи.

3 ФОРМУВАННЯ ВИМОГ

Перед початком роботи над прототипуванням системи необхідно визначитися з вимогами до неї. У загальному випадку під вимогами розуміють сукупність властивостей, які повинна мати система, що реалізується. У цій роботі необхідно спроектувати прототип системи, яка надала б користувачу можливість швидко розгортати різні веб-додатки на різні приватні хмари за допомогою графічного інтерфейсу користувача.

3.1 Мета розробки

Програмна система, що прототипується, розрахована на аудиторії різних категорій, бо власники сучасних малих та середніх бізнесів, у більшості не мають або достатніх технічних знань для ручного розгортання, або не готові інвестувати великі гроші на експертів.

Як вже зазначалося, останні дослідження говорять, що сучасні люди більше віддають перевагу роботі з веб-додатком чи онлайн-системою, ніж реальному відвідуванню того ж магазину чи компанії, яка надає якийсь сервіс. Тому інтерес малих та середніх бізнесів до Інтернету почав активно рости, бо він надає дуже гарні можливості для міжнародного партнерства, виробництва та торгівлі. А також буде сприяти швидкому розвитку та зростанню підприємств, конкуренції та технологій. За останні 10 років велика кількість компаній та бізнесів по усьому світу інвестували величезні обсяги грошей у розвиток веб- та мобільних додатків. Люди почали розуміти, що вони знімають усі обмеження у часі та просторі, надають більше можливостей для спілкування з клієнтами, партнерами чи найманцями по всьому світу та просто полегшують життя.

3.2 Загальні відомості

Веб-система може бути реалізована на різних комбінаціях мов програмування, бо дане прототипування спрямоване на розробку бізнес логіки платформи, а не її реалізацію.

Система повинна бути доступною для усіх користувачів та за концептом не повинна потребувати якихось додаткових інсталяцій чи конфігурацій на клієнтській машині.

Доступ до платформи буде відбуватися через будь-який веб-браузер та потребуватиме лише реєстрацію користувача та підключення відповідних хмарних провайдерів, на які користувач буде розгортати свої веб-додатки.

Система, що розробляється буде мати бізнес-модель «Software as a service» або SaaS («Програма як послуга) - це модель ліцензування та постачання програмного забезпечення, в якій програмне забезпечення ліцензується на основі передплати та розміщується централізовано. Його іноді називають «програмним забезпеченням за вимогою». SaaS зазвичай доступний користувачам, що використовують тонкий клієнт, наприклад, веб-браузер. SaaS стала загальною моделлю доставки для багатьох бізнес-додатків, включаючи офісне програмне забезпечення, програмне забезпечення для обміну повідомленнями, програмне забезпечення для обробки заробітної плати, програмне забезпечення СУБД, програмне забезпечення для управління, програмне забезпечення САПР, програмне забезпечення для розробки, гейміфікація, віртуалізація, бухгалтерський облік, співпраця, управління відносинами з клієнтами, інформаційні системи управління (MIS), планування ресурсів підприємства (ERP), виставлення рахунків, управління людськими ресурсами (HRM), придбання талантів, системи управління навчанням, управління контентом (CMS), географічні інформаційні системи (ГІС), управління робочими столами і так далі [11]. SaaS була включена в стратегію майже всіх провідних компаній з програмного забезпечення для підприємств.

SaaS не має фізичної потреби в непрямому розповсюдженні, оскільки вона

фізично не розподіляється і розгортається майже миттєво, тим самим заперечуючи необхідність традиційних партнерів і посередників. Однак, як ринок зростав, гравці SaaS та керовані сервіси були змушені намагатися перевизначити свою роль.

Ключовим фактором, що пояснює економічну доцільність SaaS, є «ефект масштабу» - провайдер SaaS обслуговує єдине програмне ядро, яким користуються всі клієнти, і тому витрачає меншу кількість ресурсів у порівнянні з управлінням окремими копіями програм для кожного замовника. Крім того, використання єдиного програмного ядра дозволяє планувати обчислювальні потужності і зменшує проблему пікових навантажень для окремих замовників. Все це дозволяє постачальникам SaaS рішень істотно знизити вартість обслуговування ПЗ. У кінцевому результаті, періодична вартість послуг для кінцевого замовника стає нижче витрат, що виникають при використанні класичної моделі ліцензування.

Переважна більшість рішень SaaS базується на багатопрофільній архітектурі. За допомогою цієї моделі використовується одна версія програми, що має єдину конфігурацію (для апаратного забезпечення, мережи, операційної системи), яка доступна для всіх клієнтів («орендарів»). Для підтримки масштабованості додаток встановлюється на декількох машинах (так зване горизонтальне масштабування). У деяких випадках встановлюється друга версія програми, яка пропонує вибраній групі клієнтів доступ до попередніх версій програм (наприклад, бета-версії) для цілей тестування. Це контрастує з традиційним програмним забезпеченням, де декілька фізичних копій програмного забезпечення - кожен потенційно іншої версії, з потенційно різною конфігурацією і часто налаштованими - встановлюються на різних сайтах клієнтів. У цій традиційній моделі кожна версія програми базується на унікальному коді.

Хоча виняток, а не норма, деякі SaaS-рішення не використовують багатозадачність, або використовують інші механізми, такі як віртуалізація, для економічного управління великою кількістю клієнтів замість багатозадачності. Чи є багатозадачність необхідним компонентом для програмного забезпечення як послуги, є предметом суперечок.

Існує дві основні різновиди SaaS:

— вертикальна SaaS модель - це програмне забезпечення, яке відповідає потребам певної галузі (наприклад, програмне забезпечення для охорони здоров'я, сільського господарства, нерухомості, фінансів);

— горизонтальна SaaS модель - це продукти, які орієнтовані на категорію програмного забезпечення (маркетинг, продаж, інструменти розробників, HR), але не прив’язані до певної галузі.

Система, що прототипується, буде мати горизонтальну SaaS модель, адже вона не призначена до певної галузі, а буде спрямована на велику кількість різноманітних малих та середніх бізнесів, які потребують своє онлайн-рішення у вигляді того, чи іншого веб-додатку.

3.3 Функціональні вимоги

До прототипу поставлені такі функціональні вимоги:

— авторизація у системі. Будь-який зареєстрований користувач буде мати доступ до сервісу після авторизації. Якщо він ще не був зареєстрований, йому буде запропонована форма для реєстрації;

— реєстрація у системі;

— авторизація хмарного провайдера послуг у системі;

— розгортання обраного веб-додаток на хмару авторизованого хмарного середовища;

— обрання типу хмари, на який буде розгорнутий веб-додаток;

— переглядання списку розгорнутих веб-додатків;

— обрання версії та варіацій додаткових інструментів для коректної роботи веб-додатка (наприклад версія та тип БД);

— налаштування доступів користувачів для БД, FTP, SSH та веб-додатка в цілому;

— авторизація та налаштування DNS провайдера;

— установка та налаштування безкоштовного SSL сертифікату для веб-додатка;

— створення та налаштування cron задач;

— переглядання метрик моніторингу веб-додатка (використання CPU та RAM, середня загрузка та ін.).

Даний прототип розрахований на розгортання веб-додатку на єдиний сервер, однак можна розширити цей функціонал та надати користувачу можливість розгортати більш складні веб-додатки, які потребують більш комплексного налаштування та декілька різних хмар та хмарних сервісів, які будуть взаємодіяти між собою.

3.4 UML проектування прототипу

Для проектування програмної системи було використано засоби проектування draw.io та створено наступні набір UML діаграм.

3.4.1 Діаграма прецедентів

Діаграма прецедентів (див. рис. 3.1) використання являє собою уявлення про взаємодію користувача з системою, яка показує взаємозв'язок між користувачем і різними випадками використання, в яких задіяний користувач. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, а також часто супроводжуватиметься іншими типами діаграм.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із

системою за допомогою так званих варіантів використання. Варіант використання використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором [12]. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

Діаграма прецедентів системи, що прототипується, має декількох акторів: користувач, хмарне середовище та DNS реєстратор. Перш ніж, користувачу зможе мати можливість взаємодії з системою, він повинен або авторизуватися, або зареєструватися, а потім авторизуватися. Після авторизації, користувач зможе авторизувати хмарного провайдера, який буде використаний для створення хмари та подальшого розгортання на ній веб-додатку, а також зареєструвати DNS реєстратора, який буде відповідати за контроль доменів, якими володіє користувач.

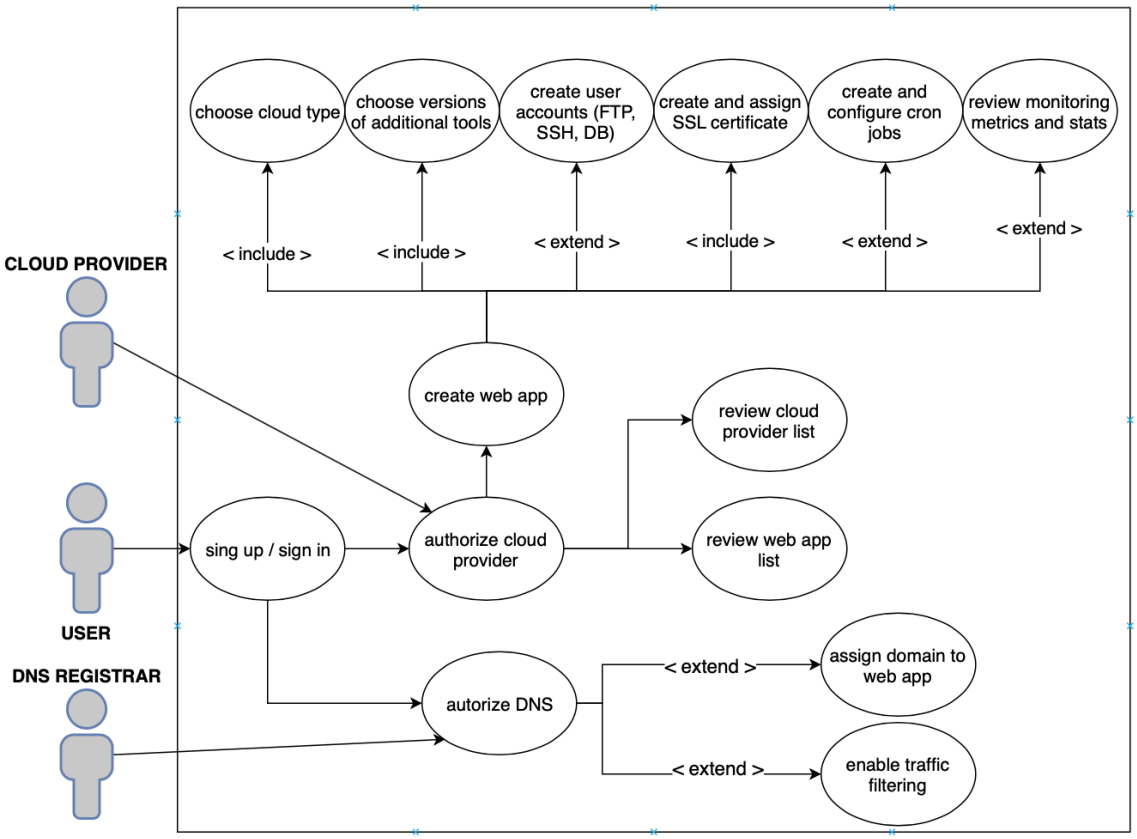


Рис. 3.1 – діаграма прецедентів

Після авторизації хмарного провайдера, у користувача з’являється

можливість створити сутність веб-додатка у системі та виконати розгортання веб-додатка.

При створенні веб-додатка користувач повинен:

— обрати тип хмари, на яку він буде розгорнутий;

— додаткові пакети та інструменти, які будуть встановлені;

— необхідність установки SSL сертифікату.

Після створення веб-додатка та його розгортання користувачу надається можливість на:

— створення додаткових SSH, FTP доступів та користувачів для бази даних;

— створення cron задач;

— перегляд моніторинг метрик та статистики веб-додатка.

Також після авторизації хмарного провайдера, користувачу надається можливість перегляду списку авторизованих провайдерів (адже їх може буті декілька) та можливість перегляду веб-додатків, які розгорнуті на кожному провайдері.

Після авторизації DNS реєстратора користувач має можливість створити та призначити доменне ім’я до розгорнутого веб-додатка, а також налаштувати фільтрацію трафіка.

3.4.2 Діаграма послідовностей

Діаграма послідовностей (див. рис. 3.2) відображає взаємодію об’єктів у часі. На ній присутні тільки ті об’єкти, які приймають участь у процесі взаємодії. Ключевий момент – це динаміка взаємодії об’єктів у часі.

В UML діаграма послідовності має виміри. Перший зліва направо у вигляді вертикальних ліній, кожна з яких зображує лінію життя окремого об'єкта, який бере участь у взаємодії. Крайнім зліва на діаграмі зображується об'єкт, який є

ініціатором взаємодії. Праворуч зображується інший об'єкт, який безпосередньо взаємодіє з першим. Таким чином, всі об'єкти на діаграмі послідовності утворюють деякий порядок, який визначається черговістю або ступенем активності об'єктів при взаємодії один з одним [13].

Другим виміром діаграми послідовності є вертикальна тимчасова вісь, спрямована зверху вниз. Початкового моменту часу відповідає сама верхня частина діаграми. Взаємодії об'єктів реалізуються за допомогою повідомлень, які надсилаються одними об'єктами іншим. Повідомлення зображуються у вигляді горизонтальних стрілок з ім'ям повідомлення, а їх порядок визначається часом виникнення. Тобто, повідомлення, розташовані на діаграмі послідовності вище, ініціюються раніше тих, які розташовані нижче. Масштаб на осі часу не вказується, оскільки діаграма послідовності моделює лише тимчасову упорядкованість взаємодій типу «раніше-пізніше».

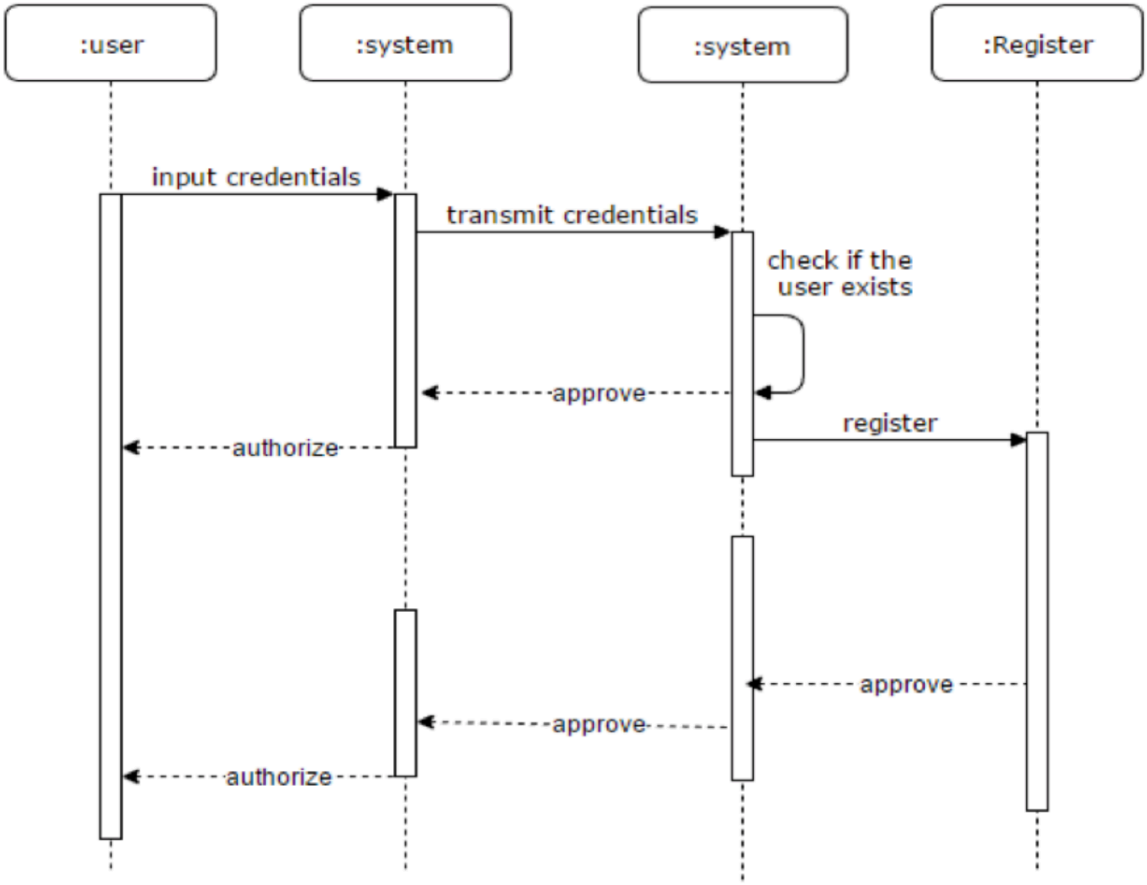


Рис. 3.2 – діаграма послідовностей

UML вніс значні поліпшення можливостей діаграм послідовності. Більшість цих удосконалень ґрунтуються на ідеї фрагментів взаємодії, які являють собою менші фрагменти взаємодіючих об'єктів. Кілька фрагментів взаємодії об'єднуються для створення різноманітних комбінованих фрагментів, які потім використовуються для моделювання взаємодій, які включають паралелізм, умовні гілки, необов'язкові взаємодії.

У цьому розділі були розглянуті була визначена мета розробки, описані загальні відомості про систему, визначені функціональні вимоги та було проведено UML моделювання системи, а саме створена діаграма прецедентів та послідовностей. На основі цих вимог можна перейти до безпосереднього створення прототипу системи та дослідження розгортання веб-додатка за її допомогою.

4 ПРОТОТИПУВАННЯ СИСТЕМИ

4.1 Архітектура системи

Система, що прототипується складається з декількох різних частин, які взаємодіють між собою. Діаграма архітектури системи (див. рис. 4.1) показує, що при зверненні до системи користувач потрапляє до сайту (FRONTEND), який надає йому можливості управління.

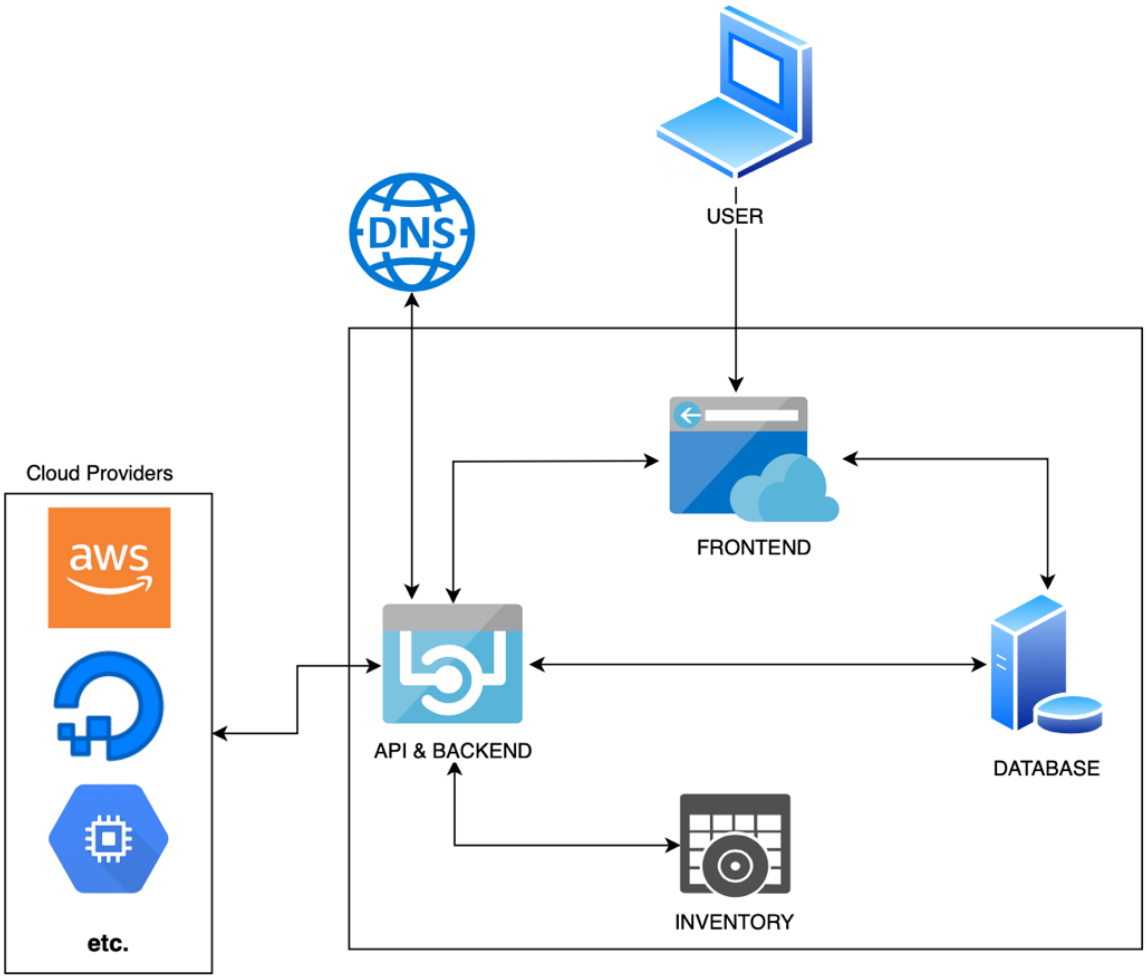


Рис. 4.1 – архітектура системи

Ті дії та дані, які користувач виконує чи вводить пересилаються на обробку у внутрішній сервіс (BACKEND) системи, де реалізована бізнес логіка. Для виконання певних дій сервісу бізнес логіки необхідно звертатися до інвентару (INVENTORY) – це сховище усих скриптів та додаткових заздалегідь створених

функцій, які використовує система, щоб автоматизовано розгортати веб-додатки та ін. У інвентарі зберігаються скрипти для розгортання різних веб-додатків (WordPress, Magento, Magento 2, Drupal, тощо), інструкції для виконання певних команд та інше. База даних зберігає дані про користувача, його веб-додатки, налаштування в системі, тощо.

З зовнішніми системами, а саме хмарними провайдерами, DNS реєстраторами, система взаємодіє через API, який надає BACKEND. Повноцінний процес розгортання веб-додатку можна описати наступним чином:

— користувачу треба виконати певні налаштування системи, які необхідні для початку роботи (авторизація хмарних провайдерів, авторизація DNS, тощо) на FRONTEND, зберігає налаштування у базі даних;

— користувач запускає процес установки та налаштування веб-додатку на FRONTEND, яка посилає запит до BACKEND;

— BACKEND опрацьовує запит користувача та підключається до INVENTORY, щоб знайти необхідний скрипт для розгортання;

— BACKEND посилає запит до хмарного провайдера, який авторизує систему для виконання дій на хмарній платформі та запускає обраний з INVENTORY скрипт на кінцевій хмарі;

— система спостерігає за процесом налаштування та повідомляє користувача про кожний наступний шаг на FRONTEND;

— після успішного завершення чи помилки процесу на хмарі BACKEND оновлює статус веб-додатка та надсилає повідомлення до користувача на FRONTEND;

— система оновлює дані про користувача та про сутності, які з ним зв’язані у DATABASE.

Для повноцінної та швидкої роботи при середній загрузці системі буде необхідно мати декілька серверів, на яких вона зможе повноцінно працювати. При необхідності деякі компоненти системи можна розширити або вертикально, або горизонтально в залежності від потреб, бізнес моделі впровадження системи та трафіку, на який вона буде розрахована [14].

4.2 Приклад роботи

У даному прикладі ми розглянемо розгортання веб-додатку на основі CMS WordPress у хмарне середовище Amazon Web Services.

4.2.1 Авторизація хмарного провайдера

Для створення хмари, установки та розгортання веб-додатку на хмару необхідно мати:

— зареєстрований акаунт у AWS;

— зареєстрований акаунт у CloudFlare (або у іншому DNS провайдері).

Будемо вважати, що користувач вже зареєстрований у системі, тому пропустимо процес реєстрації, верифікації та авторизації користувача. Наступним кроком для розгортання веб-додатку буде авторизація хмарного провайдера у системі та у профілі користувача. Щоб це зробити необхідно перейти до секції «Server Providers» (див. рис. 4.2), обрати AWS та натиснути на кнопку «Connect».

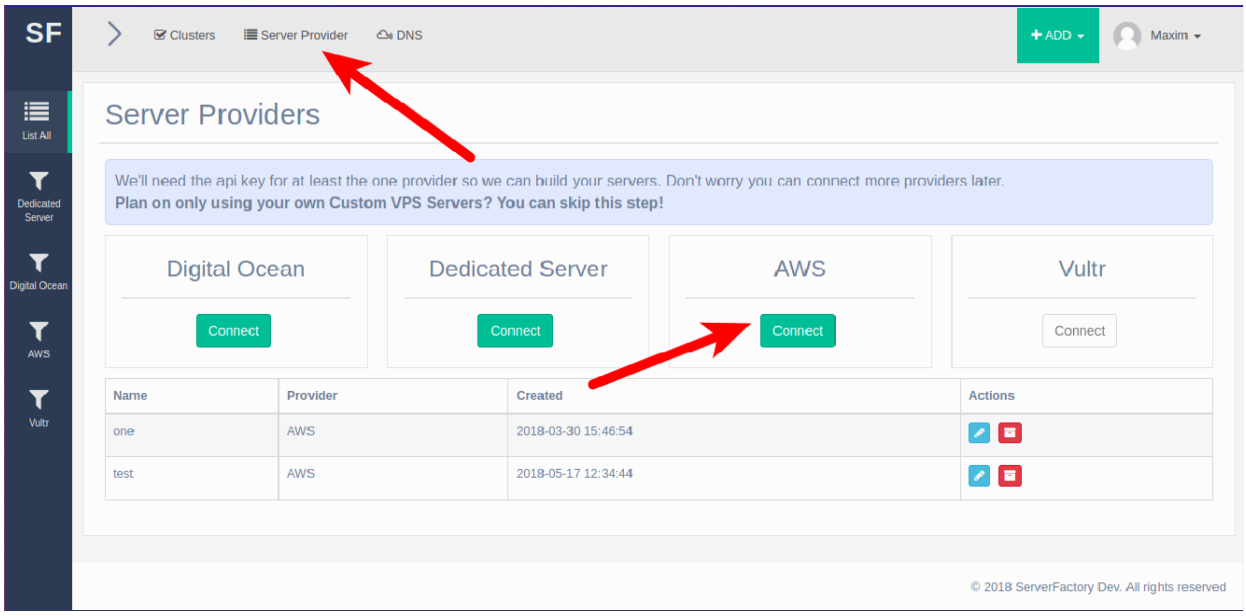


Рис. 4.2 – сторінка «Server providers»

Система переведе користувача на наступну сторінку, де користувачу потрібно буде ввести AWS access key та AWS secret key (див. рис. 4.3) , які надаються при створенні користувача у середовищі AWS, а також створити назву для ідентифікації провайдера на платформі.

Наступного разу, коли користувач захоче авторизувати те саме AWS середовище, але з іншим користувачем, він зможе надати цій сутності іншу назву, та таким чином контролювати своїх користувачів, як на різних так і на однакових хмарних середовищах. Якщо дані були введені вірно, система авторизує AWS провайдера під унікальним ім’ям.

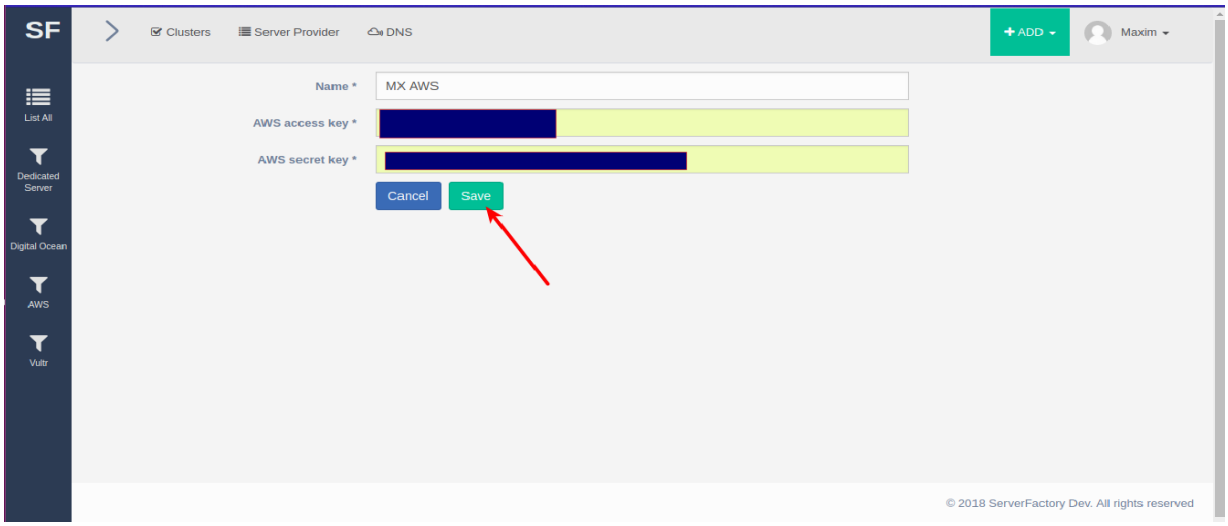


Рис. 4.3 – сторінка авторизації хмарного провайдера

Також користувач має можливість управляти хмарними провайдерами, яких він авторизував у системі. Таким чином йому надається можливість контролювати свої профілі у різних хмарних середовищах та розгортати веб-додатки до декількох провайдерів одночасно. Функціонал списку авторизованих хмарних середовищ може бути розширений, що може надати користувачу більше можливостей для управління свого акаунта у системі хмарного середовища, тим самим розширивши його контроль та надавши легкість управління.

4.2.2 Створення сутності веб-додатка у системі

Наступним кроком буде створення сутності веб-додатка у системі. Для цього необхідно натиснути на кнопку «Add» та обрати «Web app». (див. рис. 4.4)

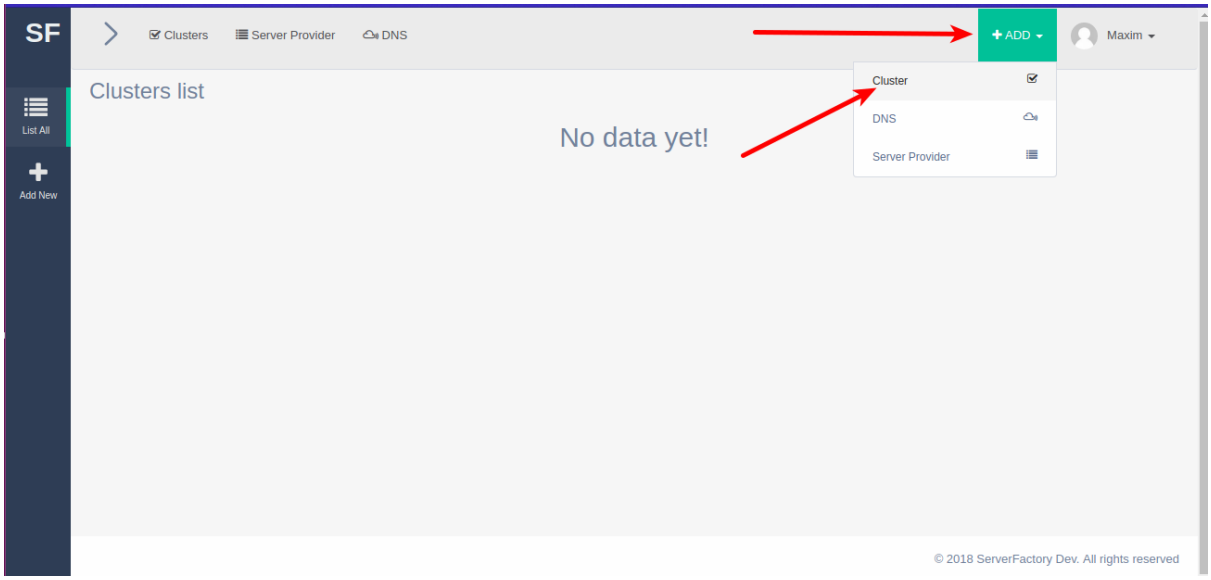


Рис. 4.4 – сторінка лістінгу веб-додатків

Система запропонує обрати (див. рис. 4.5):

— назву для веб-додатку;

— хмарного провайдера, який вже повинен бути авторизований у профілі користувача;

— регіон, де буде піднята та налаштована приватна хмара;

— тип та розмір хмари, які буде налаштована на боці хмарного середовища;

— обрати версію БД, яка буде налаштована на сервері.

Після натискання на кнопку «Save», платформа почне обробляти дані, які надав користувач та почне взаємодіяти з AWS, щоб підняти обрану користувачем хмару у відповідному регіоні. Регіони AWS знижують затримку даних, тому більшість веб-служб Amazon пропонують регіональні кінцеві точки для виконання запитів. Якщо сервіс підтримує регіони, ресурси в кожному регіоні є повністю незалежними. Наприклад, якщо ви створюєте примірник Amazon EC2 або чергу

Amazon SQS в одному регіоні, то екземпляр або чергу незалежні від примірників або черг в іншому регіоні.

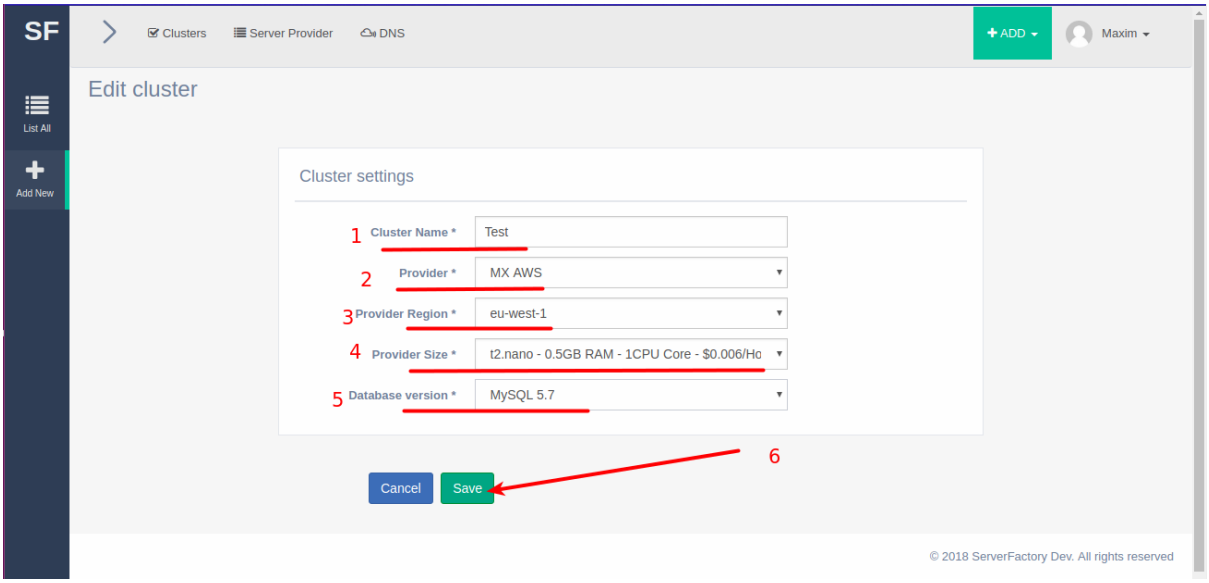


Рис. 4.5 – сторінка створення сутності веб-додатка

4.2.3 Створення бази даних

Наступним етапом розгортання буде створення та налаштування бази даних на кінцевій хмарі, бо майже будь-якому веб-додатку необхідний доступ до існуючої бази даних. Для того, щоб налаштувати базу даних на кінцевому сервері необхідно перейти до списку веб-додатків та натиснути на кнопку «View» (див. рис. 4.6).

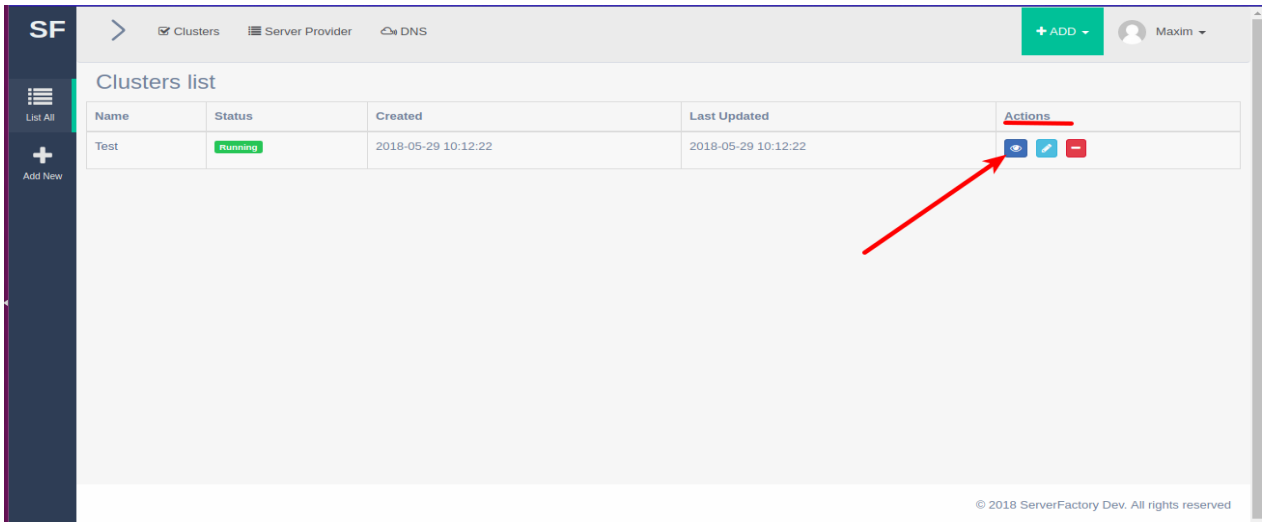


Рис. 4.6 – сторінка зі списком створених веб-додатків та кнопка «View»

Система відкриє детальну сторінку веб-додатка (див. рис. 4.7) та надасть користувачу можливості для його управління.

Для створення бази даних на сервері необхідно перейти до пункту «Databases» (див. рис. 4.7) та надати БД ідентифікуючу назву, бо на одному сервері може бути розгорнуто декілька веб-додатків, які використовують різни бази даних.

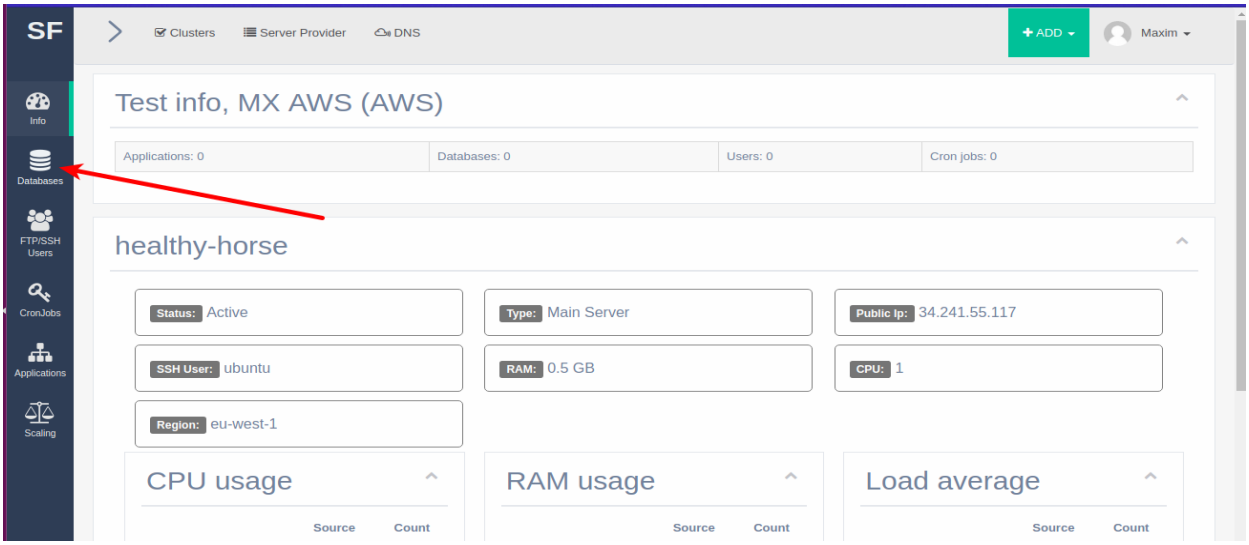


Рис. 4.7 – сторінка створеної сутності веб-додатка

Тому для ідентифікації, яка БД створена для якого веб-додатка, користувач має змогу надати їм назви (див. рис. 4.8).

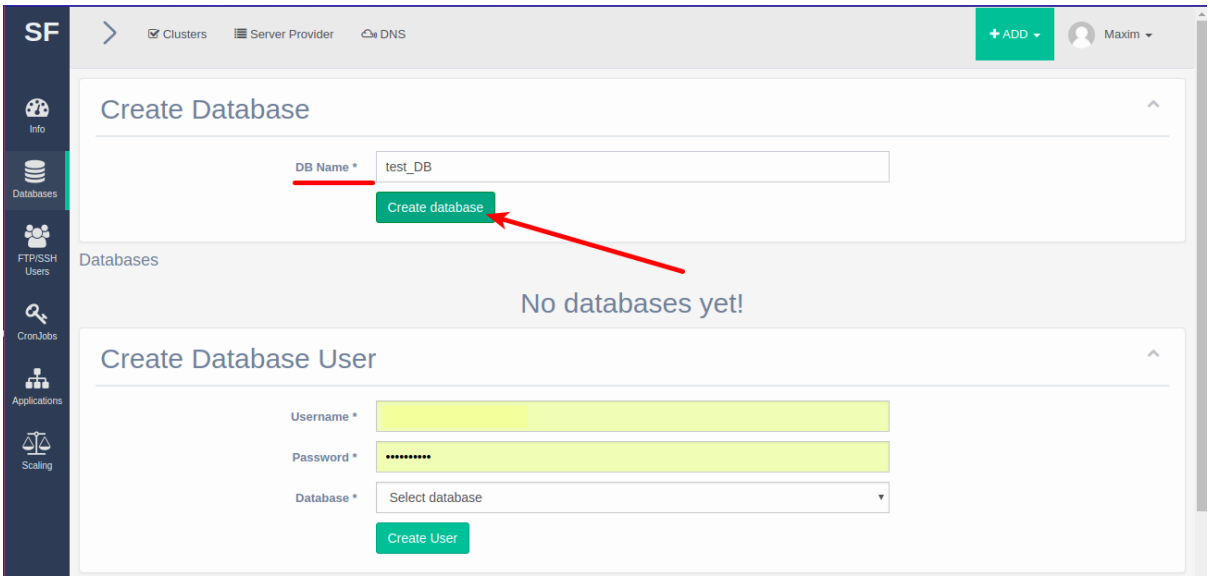


Рис. 4.8 – створення бази даних

Наступним кроком буде створення користувача бази даних (див. рис. 4.9), через якого веб-додаток буде взаємодіяти з базою даних. Система запропонує користувачу ввести ім’я користувача, пароль для доступу, а також обрати базу даних до якої треба створити цього користувача.

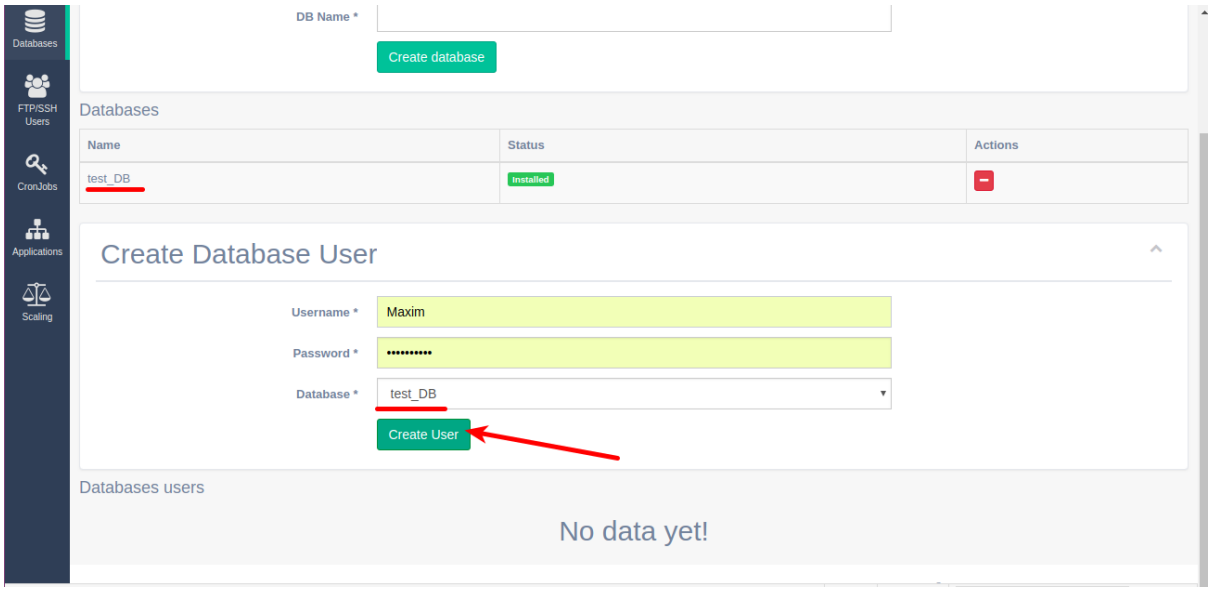


Рис. 4.9 – створення користувача для бази даних

4.2.4 Налаштування DNS

Коли база даних та користувач створені можна перейти до налаштування DNS. DNS – це система доменних імен, яка визначає присвоєння доменних імен та відображення цих імен до ресурсів Інтернету [15]. Мережеві адміністратори можуть делегувати повноваження над субдоменами свого виділеного простору імен іншим серверам імен. Цей механізм забезпечує розподілений і відмовостійкий сервіс і розроблений для того, щоб уникнути єдиної великої центральної бази даних.

Інтернет підтримує дві основні простори імен: ієрархію доменних імен і систему IP-адресів. Система доменних імен підтримує простір імен домену і надає послуги перекладу між цими двома просторами імен. Сервери імен Інтернету

впроваджують систему доменних імен. Верхня ієрархія системи доменних імен обслуговується серверами кореневих імен, що підтримуються делегуванням корпорацією Internet для призначених імен і номерів (ICANN).

Під корінням Інтернет-ресурси організовані в ієрархію доменів, керованих відповідними реєстраторами та власниками доменних імен. Сервер імен DNS - це сервер, який зберігає записи DNS, такі як записи адрес (A, AAAA), записів серверів імен (NS) і записи обмінника пошти (MX) для доменного імені і відповідає відповідями на запити щодо своєї бази даних.

Право на використання доменного імені делегується реєстраторами доменних імен, які акредитовані Інтернет-корпорацією для присвоєних імен та номерів (ICANN) або іншими організаціями, такими як OpenNIC, які наглядають за системою імен та номерів Інтернету. Окрім ICANN, кожен домен верхнього рівня (Top Level Domain, TLD) технічно підтримується та обслуговується адміністративною організацією, що керує реєстром. Реєстр відповідає за управління базою даних імен у межах своєї авторитетної зони, хоча цей термін найчастіше використовується для TLD.

Клієнт у цьому випадку виступає реєстрантом - це особа або організація, які звернулися з проханням про реєстрацію домену. Реєстр отримує реєстраційну інформацію від кожного реєстратора доменних імен, який уповноважений (акредитований) призначати імена у відповідній зоні та публікує інформацію за допомогою протоколу WHOIS.

ICANN публікує повний список TLD, реєстрів TLD і реєстраторів доменних імен. Інформація реєстратора, пов'язана з доменними іменами, зберігається в онлайновій базі даних, доступній за допомогою служби WHOIS. Для більшості з більш ніж 290 доменів верхнього рівня коду країни (ccTLD) реєстри доменів підтримують інформацію WHOIS (реєстратор, сервери імен, терміни дії і т.д.). Наприклад, DENIC, Німеччина NIC, зберігає дані домену DE. З приблизно 2001 року більшість реєстрів домену верхнього рівня (gTLD) прийняли цей так званий товстий підхід до реєстру, тобто збереження даних WHOIS в центральних регістрах замість реєстраційних баз даних.

Для доменів верхнього рівня в COM і NET використовується тонка модель реєстру. Реєстр доменів (наприклад, GoDaddy, BigRock і PDR, VeriSign тощо) містить основні дані WHOIS (тобто, реєстратор і сервери імен тощо). З іншого боку, організації або реєстранти, які використовують ORG, знаходяться в реєстрі державних інтересів виключно.

Деякі реєстри доменних імен, часто називаються мережевими інформаційними центрами (NIC), також функціонують як реєстратори кінцевим користувачам, крім того, що надають доступ до наборів даних WHOIS. Реєстри доменів верхнього рівня, такі як для доменів COM, NET та ORG, використовують модель реєстратора-реєстратора, що складається з багатьох реєстраторів доменних імен. У цьому методі управління реєстр керує тільки базою даних доменних імен і відносинами з реєстраторами. Реєстранти (користувачі доменного імені) є клієнтами реєстратора, в деяких випадках шляхом додаткового підрядного контракту з реселлерами.

Одним із цих реєстраторів є компанія CloudFlare, яка надає мережеві послуги доставки контенту, зм’якення DDoS атак, службу безпеки в Інтернеті та сервіси сервера розподілених доменних імен. Служби Cloudflare перебувають між відвідувачем і хостинг-провайдером Cloudflare, який діє як зворотний проксі для веб-сайтів.

CloudFlare пропонує багато сервісів, одним з яких є безкоштовна авторизована система доменних імен для всіх клієнтів, які працюють від мережі anycast. Anycast - це методологія мережевої адресації та маршрутизації, в якій один адреса призначення має декілька шляхів маршрутизації до двох або більше кінцевих пунктів. Маршрутизатори виберуть потрібний шлях на основі кількості стрибків, відстані, мінімальної вартості, затримки або на основі найменш перевантаженого маршруту. Мережі Anycast широко використовуються для продуктів мережі доставки контенту (CDN), щоб наблизити їх зміст до кінцевого користувача, що є також одним з основних сервісів компанії CloudFlare.

SolveDNS оцініли Cloudflare як одного з найшвидших DNS швидкість пошуків в усьому світі зі швидкістю пошуку 5,28 мс в квітні 2019 року [16].

У системі, що розробляється CloudFlare може мати декілька різних аналогів, такі як AWS Route53, KeyCDN, GoogleCloudDNS та багато інших.

Щоб налаштувати DNS необхідно отримати публічний адрес сервера, до якого буде обрано доменне ім’я. Щоб це зробити необхідно перейти до вкладки Info (див. рис. 4.7), та перевірити публічну IP-адресу сервера. Користувачу потрібно зберегти цей IP-адрес, бо він буде в подальшому використаний при налаштуванні веб-додатку.

Треба натиснути кнопку DNS (див. рис. 4.7), яка переведе користувача у обрання DNS реєстраторів. В нашому випадку таким буде виступати CloudFlare (див. рис. 4.10). Для авторизації реєстратора DNS у системі користувачу необхідно перш за все створити профіль у системі CloudFlare, а потім надати системі ім’я користувача та API ключ (див. рис. 4.11), щоб система мала можливість передавати та отримувати дані від CloudFlare.

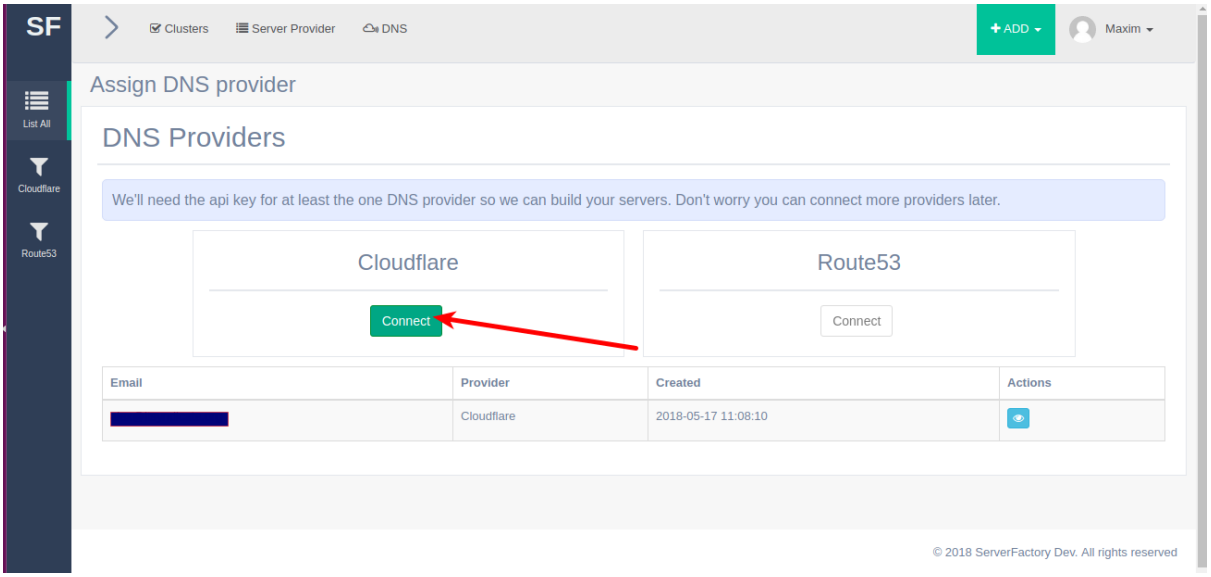


Рис. 4.10 – сторінка DNS реєстраторів

Після того, як CloudFlare авторизувалася у системі (див. рис. 4.12), треба створити зону доменних імен. Зона DNS - це будь-яка окрема, суміжна частина простору доменних імен у системі доменних імен (DNS), для якої адміністративна відповідальність делегована одному менеджеру.

Простір доменних імен в Інтернеті організований в ієрархічне розташування субдоменів нижче кореневого домену DNS. Окремі домени цього дерева можуть служити пунктами делегування для адміністративних органів та управління.

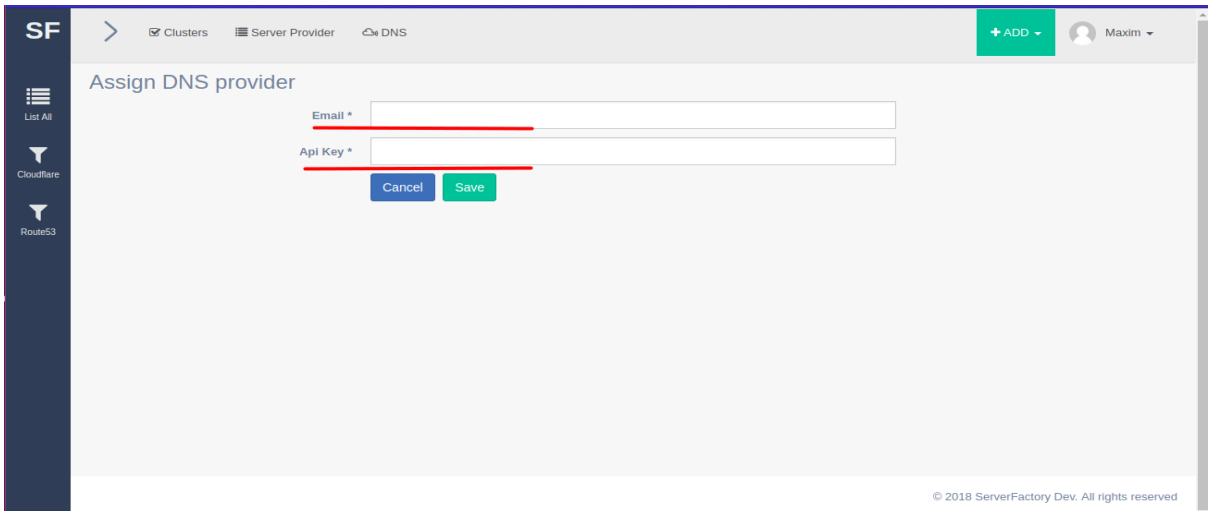


Рис. 4.11 – сторінка авторизації DNS реєстраторів

Однак, як правило, бажано реалізувати дрібні межі делегування, так що кілька підрівнів домену можна керувати незалежно. Тому простір доменних імен розбивається на області (зони) для цієї мети.

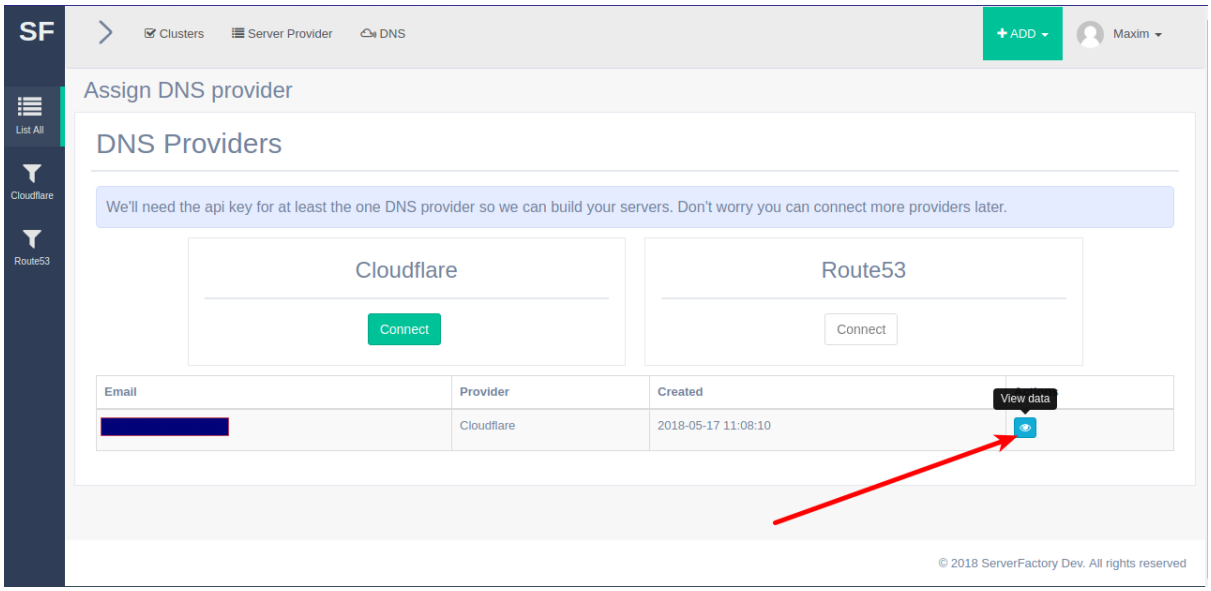


Рис. 4.12 – список авторизованих DNS реєстраторів

Зона починається з домену і простягається вниз по дереву до вузлів листя або до верхнього рівня субдоменів, де починаються інші зони. Система, що

проєктується повинна надавати можливість контролювати декілька веб-додатків, тому можливість контролювати їх DNS теж дуже важлива.

Користувачу пропонується придумати ім’я для DNS зони та створити її (див. рис. 4.13). Після створення треба перейти на рівень нижче та створити доменне ім’я у рамках доменної зони, що тільки но була створена, та прив’язати його до серверу, що був створений. Для цього йому потрібно натиснути кнопку «View» (див. рис. 4.13) та перейти до наступної сторінки (див. рис. 4.14), де користувачу необхідно обрати:

— тип доменного ім’я;

— назву або сам домен;

— IP-адресу сервера, яку користувач скопіював раніше.

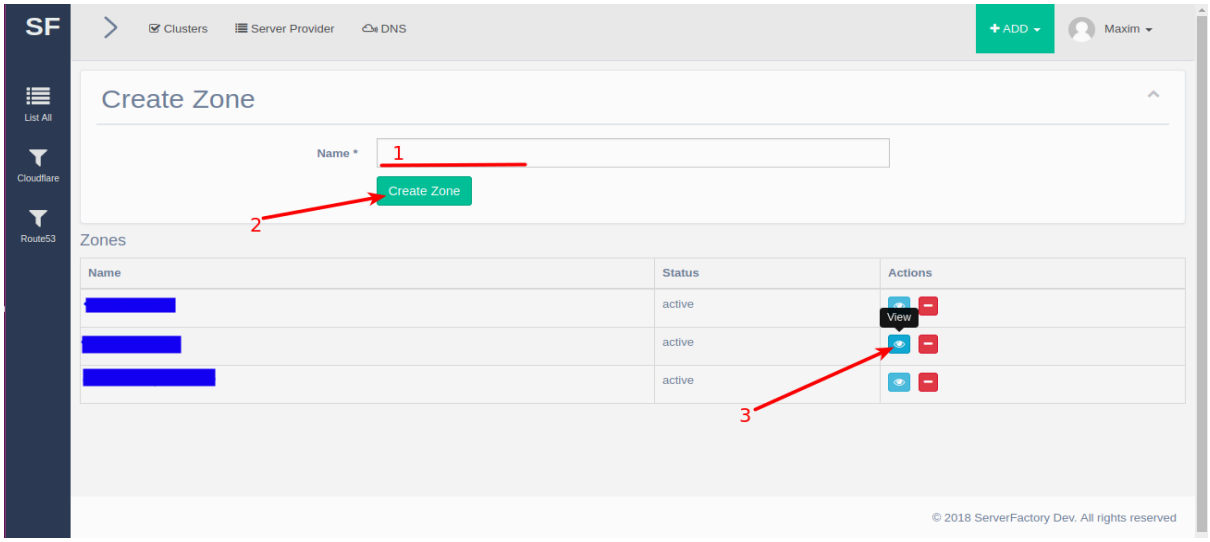


Рис. 4.13 – створення доменної зони

Після натискання на кнопку «Add record» (див. рис. 4.14) система сповістить користувача про створення доменного ім’я.

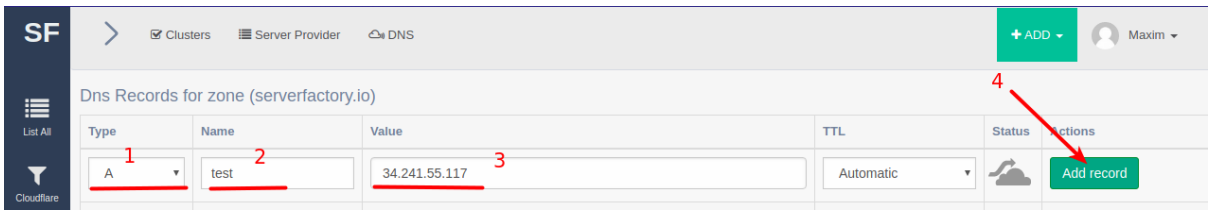


Рис. 4.14 – призначення доменного ім’я

Також користувачу надається можливість налаштування пропуску трафіка у CloudFlare – користувач може це змінити, натиснувши на значок хмаринки, тим самим змінивши спосіб проксінгу трафіку.

4.2.5 Установка та налаштування веб-додатку

Тепер коли налаштування DNS завершено, можна перейти до установки веб-додатку. Для цього користувачу необхідно натиснути на кнопку «Clusters» (див. рис. 4.6), обрати сервер, який користувач хоче змінити, натиснути на кнопку «View», та перейти до вкладки «Applications» у лівому нижньому кутку. Система запропонує користувачу ввести певні дані для подальших налаштувань, а саме (див. рис. 4.15):

— домен, до якого треба прив’язати веб-додаток (це той самий домен, який був створений раніше у CloudFlare);

— обрати веб-додаток, який необхідно розгорнути – у нашому випадку це WordPress, але вибір користувача може бути необмежним;

— веб-директорію на сервері, яку буде використати веб-додаток для своїх даних;

— обрати попередньо створену базу даних;

— обрати версію PHP;

— обрати необхідність установки та автоматичного налаштування Let’s Encrypt SSL.

Let’s Encrypt (https://letsencrypt.org) - це некомерційний центр сертифікації, який управляє Internet Security Research Group (ISRG), який безкоштовно надає сертифікати X.509 для шифрування TLS. Сертифікат дійсний протягом 90 днів, протягом яких може відбутися оновлення в будь-який час. Пропозиція супроводжується автоматизованим процесом, спрямованим на подолання ручного

створення, перевірки, підписання, установки та поновлення сертифікатів для безпечних веб-сайтів.

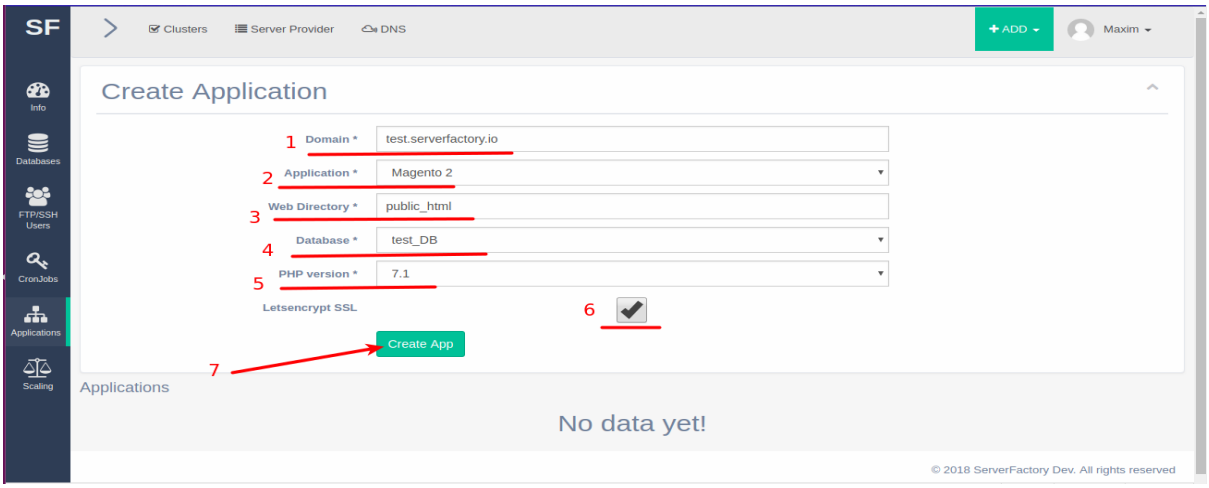


Рис. 4.15 – створення веб-додатка

Проект стверджує, що його мета полягає в тому, щоб зробити шифровані з'єднання на серверах World Wide Web повсюдними. Завдяки усуненню: платежів, додаткових налаштуваннь веб-сервера, перевірок через електронну пошту та необхідності оновлення сертифіката, даний підхід має значно зменшити складність налаштування та підтримки шифрування TLS. На веб-сервері Linux достатньо виконати лише декілька команд, щоб встановити шифрування HTTPS.

Після натискання кнопки «Create App», система почне процес установки та налаштування веб-додатка. Після завершення налаштувань внизу екрана з’явиться новий елемент у таблиці (див. рис. 4.16) – це і є веб-додаток, що тільки но був розгорнутий на сервер та його доменне ім’я.

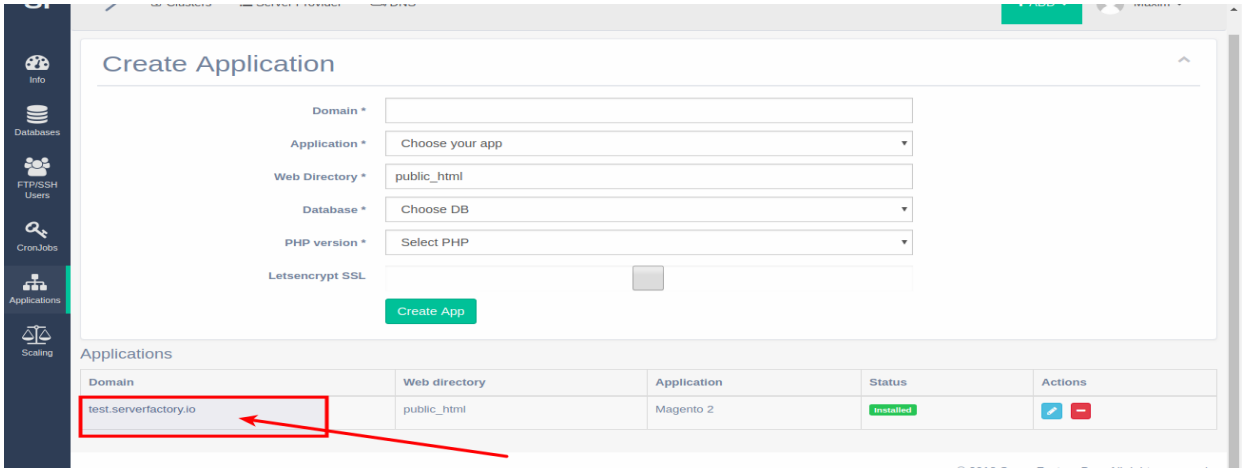


Рис. 4.16 – створений веб-додаток

Перевірити працездатність веб-додатка можна через звернення до доменого імені через веб-браузер. Користувач повинен побачити головну сторінку налаштування CMS WordPress (див. рис. 4.17) На цьому процес створення та налаштування веб-додатка завершено.

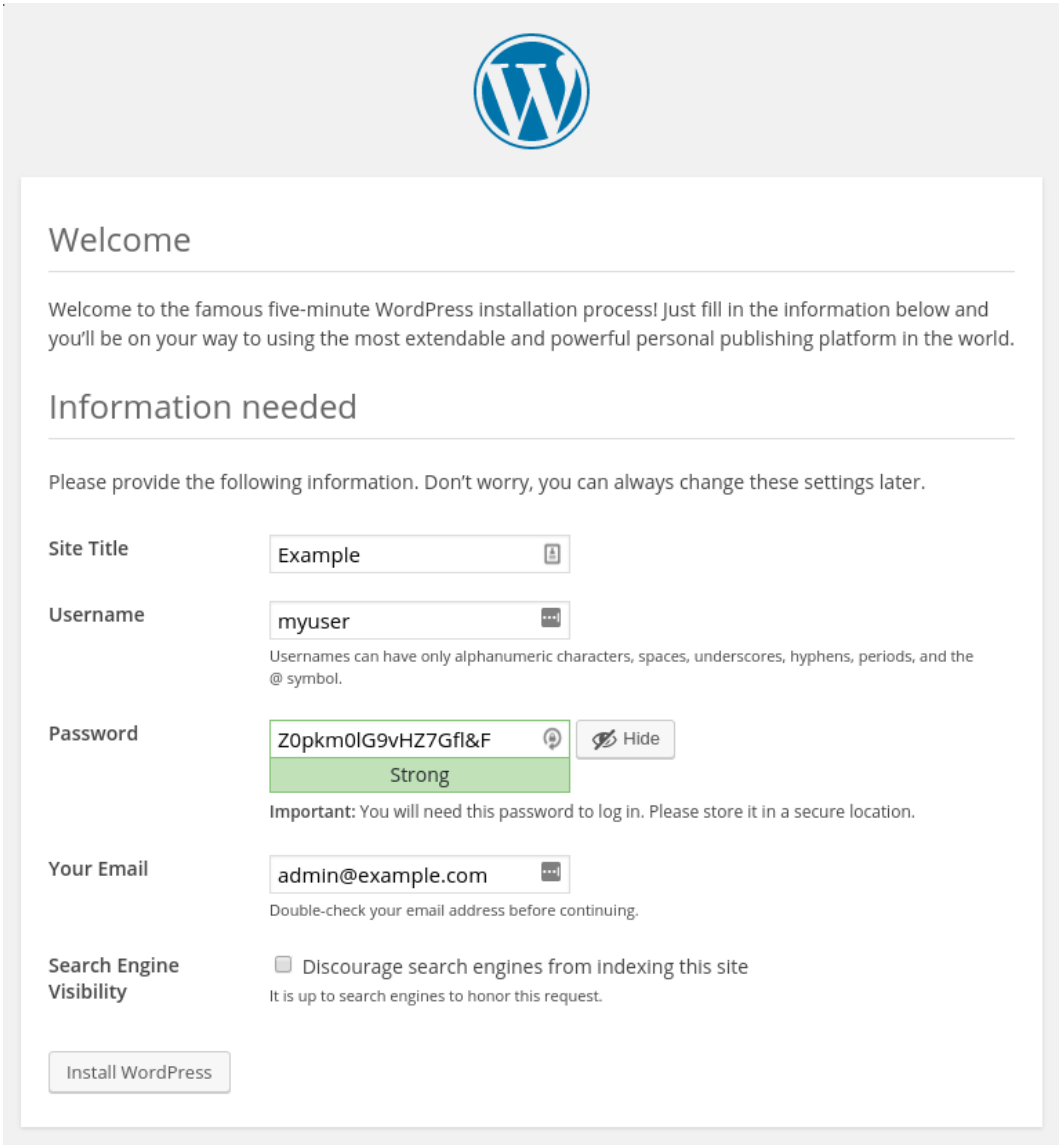


Рис. 4.17 – головна сторінка налаштування WordPress

На цьому процес розгортання можна вважати завершеним, бо наступні налаштування будуть специфічні для веб-додатка, а не для його розгортання. Перевірка показала, що система вдало розгорнула веб-додаток на приватній хмарі AWS.

4.2.6 Додаткові можливості

Додатковими можливостями для користувача у рамках системи, що проектується є можливість створення та управління FTP/SSH користувачами, а також створення та управління crons. Для створення нового FTP/SSH користувача необхідно перейти до вкладки «FTP/SSH users» (див. рис. 4.6) та заповнити поля, які пропонує система. Щоб створити нового FTP користувача (див. рис. 4.18) необхідно ввести:

— ім’я користувача;

— пароль для доступу;

— кінцеву директорію для доступу.

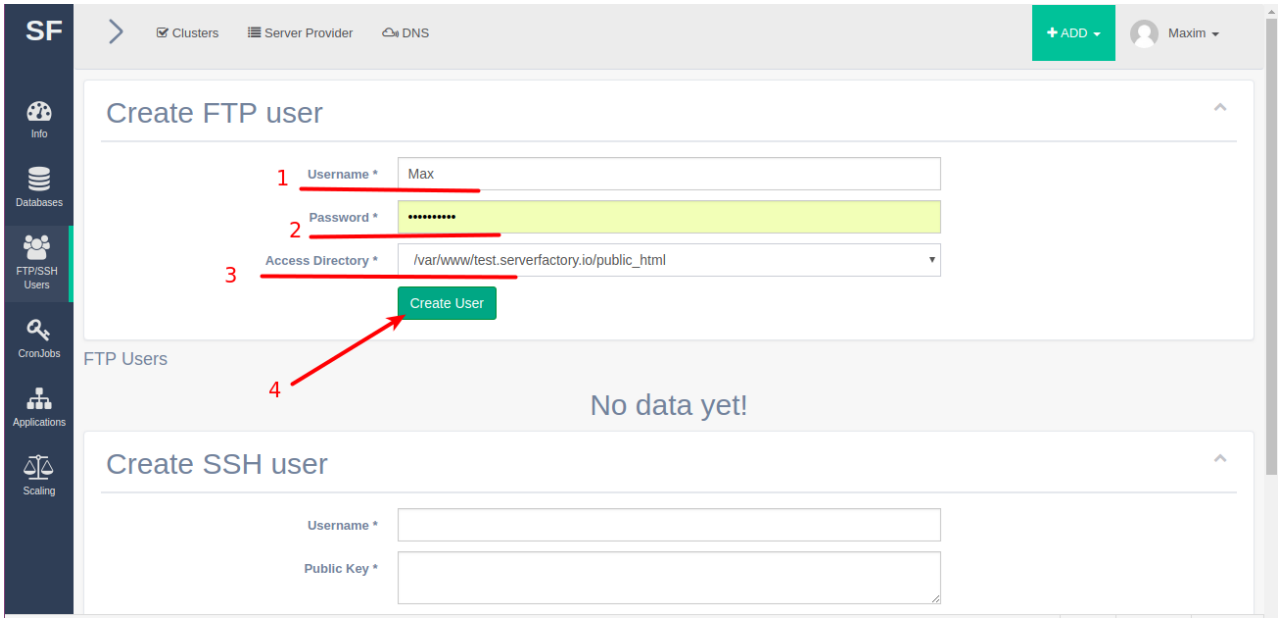


Рис. 4.18 – створення FTP користувача

Після натискання кнопки Create user система створить користувача на кінцевому сервері та повідомить користувача про успіх чи невдачу. Перевірити FTP користувача можна через будь-який FTP клієнт, найпоширенішим з яких є FileZilla.

Щоб надати доступ SSH користувачу (див. рис. 4.19) необхідно ввести його:

— ім’я;

— публічний ключ, який був згенерований заздалегідь;

— кінцеву директорію для доступу.

Ще однією можливістю для користувача є створення та управління cron. Програмна утиліта cron є тимчасовим планувальником завдань в Unix-подібних операційних системах комп'ютера. Люди, які створюють та підтримують середовища програмного забезпечення, використовують cron для планування завдань (команд або скриптів оболонки) для періодичного запуску на фіксований час, дату або інтервали.

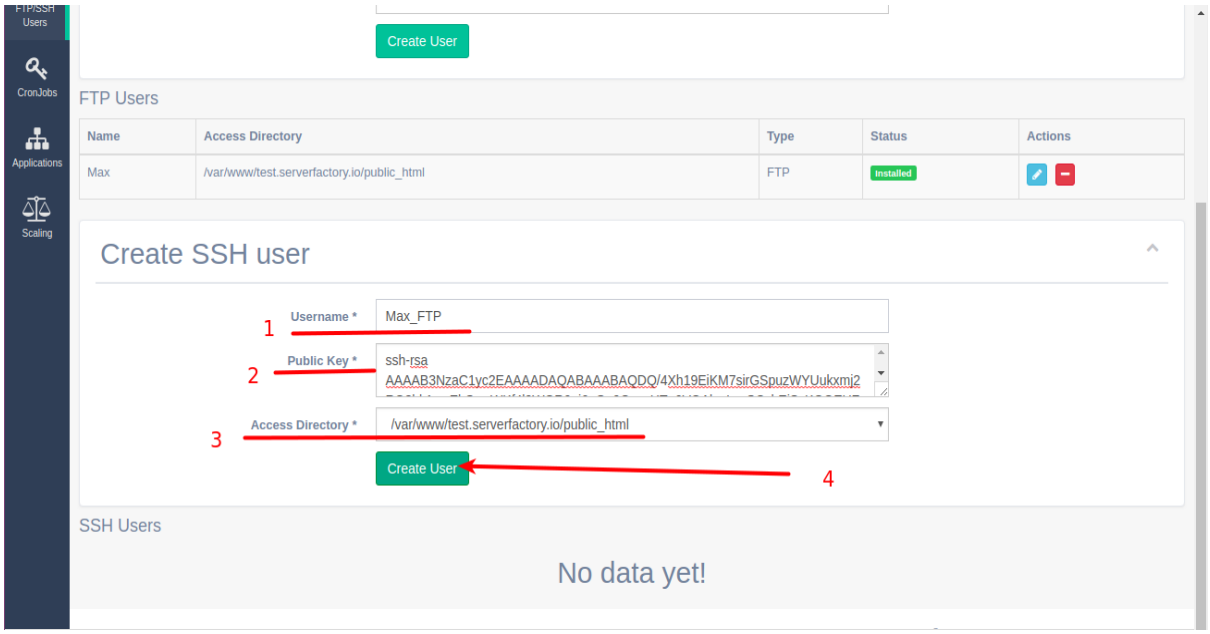


Рис. 4.19 – створення SSH користувача

Як правило, це автоматизує обслуговування або адміністрування системи, хоча її загальний характер робить його корисним для таких речей, як завантаження файлів з Інтернету та завантаження електронної пошти через регулярні проміжки часу. Походження назви cron - від грецького слова для часу, χρόνος (chronos).

Cron є найбільш придатним для планування повторюваних завдань. Планування одноразових завдань може бути здійснено за допомогою асоційованої утиліти.

Cron керується файлом crontab (таблиця cron), файлом конфігурації, який визначає команди оболонки для періодичної роботи за заданим графіком. Файли crontab зберігаються там, де зберігаються списки завдань та інші інструкції для демона cron [17]. Користувачі можуть мати свої власні файли crontab і часто є загальносистемний файл crontab (зазвичай в / etc або підкаталог / etc), який можуть редагувати в більшості тільки системні адміністратори.

Для створення cron задачі у системі необхідно перейти до вкладки «Cron jobs» (див. рис. 4.6) у меню веб-додатка, де система запропонує користувачу кнопку «Create cron job», а також можливість переглянути вже створені на сервері cron задачі.

Для прикладу розглянемо команду, яка буде записувати дату з позначкою ОК до файлу /tmp/testcron.log:

```
echo "`date` - OK" >> /tmp/testcron.log
```

Цю команду треба ввести у текстове поле (див. рис. 4.20), а також обрати наскільки часто системі потрібно запускати цю cron задачу. Після натискання кнопки «Save» система сповістить користувача про створення задачі, перевірити виконання якої можна вікривши файл /tmp/testcron.log.

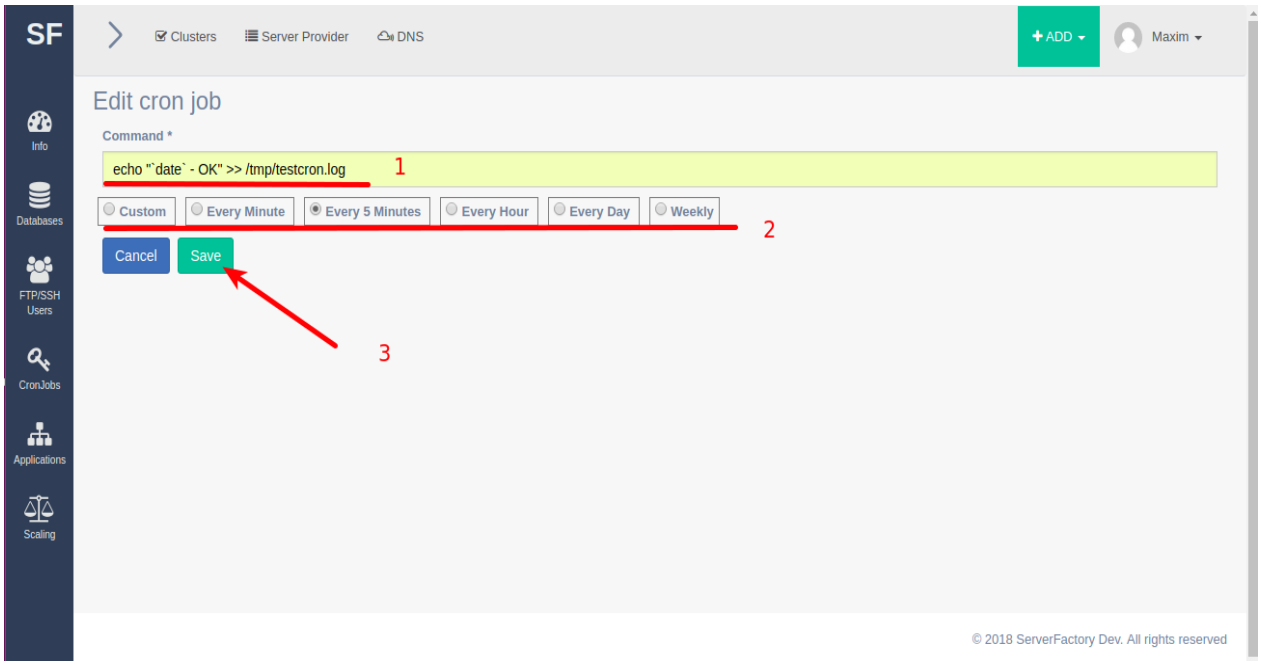


Рис. 4.20 – створення cron задачі

На цьому процес розгортання веб-додатку WordPress до хмарного провайдера AWS завершено. Тепер користувачу надається можливість детальних налаштувань веб-додатка та його необхідних кастомізацій, які або потребують наймання розробника, який буде відповідати за зовнішний вигляд сайту веб-додатка, його додаткові конфігурації і так далі.

У майбутньому планується розширити функціонал системи, додавши можливість управління та розгортання більш складних веб-додатків, які будуть мати розподілену інфраструктуру та декілька різноманітних сервісів/веб-додатків/фреймворків. У цілому систему можна розширювати ще дуже довгий час, бо треба постійно розширювати базу хмарних провайдерів та DNS реєстраторів, щоб покривати більший прошарок користувачів. Результати дослідження та прототип системи, який об’єднює у собі функції та властивості декількох інструментів, можуть бути корисними у подальших наукових та практичних дослідженнях з розгортання веб-додатків, а також з спрощення та ускорення цього процесу.

ВИСНОВКИ

У роботі були проаналізовані традиційний та автоматизований методи до розгортання веб-додатків у приватних хмарах. Були виявлені переваги та недоліки обох методів та умови, в яких доцільно використовувати тий чи інший. Також були розглянуті існуючі інструменти для розгортання веб-додатків у приватних хмарах, на прикладі веб-додатку WordPress, який є безкоштовною системою управління контентом (CMS) з відкритим вихідним кодом на основі PHP і MySQL.

У якості альтернативи був розглянутий змішаний підхід до розгортання веб-додатків у приватних хмарах з використанням поєднання різних інструментів у рамках однієї платформи. Ця платформа повинна надати рядовому користувачу можливості для легкого підключення різних хмарних провайдерів, автоматизованого розгортання там різних веб-додатків, поєднання їх у більш складні та інших налаштуваннь. Користувач зовсім не повинен думати про розгортання та інфраструктуру веб-додатка та довіритися платформи, яка може автоматично обслуговувати та налаштовувати додаток, і також мати доступ до всіх найважливіших конфігураці. Таким чином, для того щоб підтвердити перевагу запропонованого для веб-додатків у приватних хмарах був обраний один із найпопулярніших веб-додатків сьогодні – WordPress.

У ході роботи були проаналізовані різні інструменти та їх недоліки та метрики, такі як складність, трудоємкість та час. Був розглянутий приклад традиційного методу розгортання веб-додатку WordPress, і приведений автоматизований метод за допомогою Ansible. Було також розглянені різні підходи до автоматизованого розгортання, які відповідають за різні функції.

Результатом роботи став прототип інтерфейсу системи з швидкого розгортання веб-додатків, який надає користувачу можливості для:

— реєстрації, авторизації різних існуючих сторонніх систем (хмарні провайдери, DNS реєстратори);

— швидкого розгортання веб-додатка на хмари популярних хмарних провайдерів;

— додання SSL сертифікату для свого веб-додатка;

— вибору версій пакетів та додаткових інструментів веб-додатка;

— перегляду моніторинг метрик та статистики хмари, на яку був розгорнутий веб-додаток;

— створення доступів для SSH, FTP та БД користувачів;

— створення cron задач для дій, які повторюються;

— призначення домену до розгорнутого веб-додатка;

— перегляду та контролювання набору веб-додатків, DNS реєстраторів та хмарних середовищ, авторизованих у системі.

Надання можливості контролювати, змінювати процес розгортання, а також загальне спрощення цього процесу, створює більше можливостей для роботи с веб-додатками та залучить більшу аудиторію користувачів. Розвиток цього інструменту потенційно може надати велику перевагу малим та середнім бізнесам, навіть якщо вони лімітовані у коштах, які вони могли б інвестувати у професійних спеціалістів з розгортання.

Ці дані будуть практично корисними для подальшого створення та проектування платформи, які надала б користувачу можливість розгортати як прости так і більш складні веб-додатки у більшості популярних хмарних провайдерів сьогодні. Крім того вони можуть бути використані у подальших наукових та практичних дослідженнях з розгортання веб-додатків, а також з спрощення та оптимізації цього процесу.

ПЕРЕЛІК ПОСИЛАНЬ

1) Kyungyiung Ohk, Miyea Kim «Who's leading China's E-Commerce industry? The antecedents and consequences of e-Wom focusing on one person media» [Електронний ресурс] // Journal of Theoretical and Applied Information Technology 2018. Vol.96. No 5, ISSN: 1992-8645 URL: http://www.jatit.org/volumes/Vol96No5/15Vol96No5.pdf (дата звернення 19.03.2019)

2) Xiaohua Cai, Guibin Zhang «Innovation and Reform of Logistics Management in E-commerce Environment» [Електронний ресурс] // 2019 5th International Conference on Education Technology, Management and Humanities Science (ETMHS 2019) URL: https://webofproceedings.org/proceedings_series/ESSP/ETMHS%202019/ETMHS19274.pdf (дата звернення: 19.03.2019)

3) Abhilash Sreeramaneni, Bokuk Seo, and Chan KOH «A Business Driven Scalable Cloud Computing Service Platform (PaaSXpert), January 2018» [Електронний ресурс] // Engpaper Engineering Rerearch papers library URL: https://www.researchgate.net/profile/Abhilash_Sreeramaneni/publication/313812753_A_Business_Driven_Scalable_Cloud_Computing_Service_Platform_PaaSXpert/links/58ad6694aca272af0666fd49/A-Business-Driven-Scalable-Cloud-Computing-Service-Platform-PaaSXpert.pdf (дата звернення: 10.03.2019)

4) Prof. Dhanashri Patil, Ms. Pranita Patil, Ms. Priyanka Patil «Establishing Cloud Computing Security in trust-based Cloud Service Provider» [Електронний ресурс] // International Journal of Engineering Technology Science and Research, ISSN 2394 – 3386 Volume 4 URL: http://www.ijetsr.com/images/short_pdf/1491647371_dmce878_venue.pdf (дата звернення: 19.03.2019)

5) A Venkatesh, Marrynal S Easteff «A Study of Data Storage Security Issues in Cloud Computing» [Електронний ресурс] // International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2018 IJSRCSEIT, ISSN

: 2456-3307 Volume 3 URL: http://ijsrcseit.com/paper/CSEIT1831389.pdf (дата звернення 23.03.2019)

6) Nelson Tavares de Sousa, Wilhelm Hasselbring, Tobias Weber, Dieter Kranzlmuller «Designing a Generic Research Data Infrastructure Architecture with Continuous Software Engineering» [Електронний ресурс] // Research Group Software Construction URL: https://www2.swc.rwth-aachen.de/docs/CSE_2018_paper_3.pdf (дата звернення 25.03.2019)

7) Dirk van der Linden, Georg Neugschwandtner, Herwig Mannaert "Industrial Automation Software: Using the Web as a Design Guide" [Електронний ресурс] // ICIW 2015 : The Seventh International Conference on Internet and Web Applications and Services URL: https://www.researchgate.net/profile/Herwig_Mannaert/publication/259590189_Industrial_Automation_Software_Using_the_Web_as_a_Design_Guide/links/54d487c80cf246475805ff4b.pdf (дата звернення: 23.03.2019)

8) María Luz Alvarez ; Isabel Sarachaga ; Arantzazu Burgos ; Elisabet Estévez ; Marga Marcos «A Methodological Approach to Model-Driven Design and Development of Automation Systems» [Електронний ресурс] // 2017 Innovations in Power and Advanced Computing Technologies (i-PACT) ISSN: 1558-3783 URL: https://ieeexplore.ieee.org/document/7496983 (дата звернення 25.03.2019)

9) AWS documentation [Електронний ресурс] // Офіціальний сайт Amazon Web Services URL: https://docs.aws.amazon.com/index.html?nc2=h_ql_doc (дата звернення: 17.04.2019)

10) Christian A. Rodríguez ; Lía Molinari ; Fernando G. Tinetti «Evolution of Handling Web Applications Up to the Current DevOps Tools» [Електронний ресурс] // 2017 International Conference on Computational Science and Computational Intelligence (CSCI) e-ISBN: 978-1-5386-2652-8 URL: https://ieeexplore.ieee.org/document/8560930 (дата звернення 18.04.2019)

11) Dr. Birajkumar V. Patel, Dr. Dipti B. Shah «Implementation of Cloud Computing (Software as a Service) for Product based Search Engine» [Електронний

ресурс] // International Journal of Scientific Research in Science and Technology ISSN: 2395-602X URL: http://ijsrst.com/paper/2584.pdf (дата звернення 18.04.2019)

12) Darothi Sarkar, Monika Bhalla, Swati Mittal Singal «Darothi Sarkar ; Monika Bhalla ; Swati Mittal Singal» [Електронниий ресурс] // 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence e-ISBN: 978-1-5090-3519-9 URL: https://ieeexplore.ieee.org/document/7943257 (дата звернення 18.04.2019)

13) Wassana Naiyapo, Watcharee Jumpamule "An Event Driven Approach to Create UML Models" [Електронний ресурс] // 2018 22nd International Computer Science and Engineering Conference (ICSEC) SBN: 978-1-5386-8164-0 URL: https://ieeexplore.ieee.org/document/8712621 (дата звернення 25.04.2019)

14) Sagar Narendrasing Deshmukh, H. P. Khandagale «A system for application deployment automation on cloud environment» [Електронний ресурс] // 2017 Innovations in Power and Advanced Computing Technologies (i-PACT) e-ISBN: 978-1-5090-5682-8 URL: https://ieeexplore.ieee.org/document/8245025 (дата звернення 25.04.2019)

15) Michael Dooley, Timothy Rooney «Introduction to the Domain Name System (DNS)» // Wiley Online Library e-ISBN: 9781119328292 URL: https://doi.org/10.1002/9781119328292.ch2 (дата звернення 27.04.2019)

16) SolveDNS - April 2019 DNS Speed Comparison Report [Електронний ресурс] URL: http://www.solvedns.com/dns-comparison/2019/04 (дата звернення: 27.04.2019)

17) Gerion Entrup, Felix Herrmann, Sophie Matter, Elias Entrup, Matthias Jakob, Jan Eberhardt, Mathias Casselt «Architecture of the Linux kernel» [Електронний ресурс] // Systems Research and Architecture Group (SRA) URL: https://sra.uni-hannover.de/Lehre/WS17/S_AKSI/preview/document.pdf