МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«ТЕХНОЛОГІЇ DEVOPS»

ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТЕЙ: 122 – «КОМП’ЮТЕРНІ НАУКИ»,
123 – «КОМП’ЮТЕРНА ІНЖЕНЕРІЯ»
ОСВІТНЬО-ПРОФЕСІЙНИХ ПРОГРАМ:
«КОМП’ЮТЕРНІ НАУКИ», «КОМП’ЮТЕРНА ІНЖЕНЕРІЯ»
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»
(ЧАСТИНА 1)

КРЕМЕНЧУК 2024

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Технології DevOps» для студентів денної форми навчання зі спеціальностей: 122 – «Комп'ютерні науки», 123 – «Комп'ютерна інженерія» освітньо-професійних програм: «Комп'ютерні науки», «Комп'ютерна інженерія» освітнього ступеня «Бакалавр» (частина 1)

Укладачі: д. т. н., проф. А. Л. Перекрест;
асист. К. О. Вадурін
Рецензент к. т. н., доц. В. М. Сидоренко

Кафедра комп'ютерної інженерії та електроніки

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського
Протокол № 7 від 25.04. 2024 року
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Метою навчальної дисципліни «Технології DevOps» є оволодіння знаннями та навичками, необхідними для впровадження та застосування DevOps-практик у процесі розробки та експлуатації програмного забезпечення, що дозволить їм ефективно співпрацювати з іншими фахівцями в рамках DevOps-команди для забезпечення швидкого та якісного випуску програмних продуктів.

У результаті вивчення навчальної дисципліни студент повинен

знати:

− принципи та концепції руху DevOps, включаючи розуміння життєвого циклу DevOps;

− інструменти та технології, пов’язані з DevOps, такі як Jenkins, Docker, та Chef;

− процеси інтеграції з Jenkins, включаючи створення та налаштування веб-додатків JEE з використанням Maven;

− створення коду та налаштування конвеєра збірки, використовуючи плагін Building Pipeline та розгортання файлу WAR;

− процес огляду та встановлення контейнерів Docker;

− розуміти відмінностей між віртуальними машинами та контейнерами;

− принципи хмарних обчислень та використання інструменту керування конфігурацією Chef;

уміти:

− розгортати та обслуговувати веб-додатки JEE з використанням Jenkins;

− створювати та налаштовувати конвеєри збірки, використовуючи плагіни та інструменти DevOps;

− встановлювати та налаштовувати контейнери Docker для ефективного розгортання додатків;

− виконувати автоматизоване тестування, включаючи функціональне тестування з використанням Selenium та навантажувальне тестування з використанням Apache JMeter;

− налаштовувати наскрізну автоматизацію за допомогою Jenkins, Chef і AWS EC2;

− забезпечувати безпеку в Jenkins та VSTS, а також виконувати моніторинг Jenkins і Microsoft Azure.

Згідно з вимогами Стандарту вищої освіти України, першого (бакалаврського) рівня, галузі знань 12 Інформаційні технології, спеціальності 122 «Комп'ютерні науки», навчальна дисципліна «Технології DevOps» забезпечує набуття здобувачами вищої освіти таких **компетентностей.**

ЗК 3. Знання та розуміння предметної області та розуміння професійної діяльності.

ЗК 9. Здатність працювати в команді.

ЗК 11. Здатність приймати обґрунтовані рішення.

ЗК 12. Здатність оцінювати та забезпечувати якість виконуваних робіт.

СК 8. Здатність проєктувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.

СК 9. Здатність реалізувати багаторівневу обчислювальну модель на основі архітектури клієнт-сервер, включаючи бази даних, знань і сховища даних, виконувати розподілену обробку великих наборів даних на кластерах стандартних серверів для забезпечення обчислювальних потреб користувачів, у тому числі на хмарних сервісах.

СК 10. Здатність застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника.

Програмні результати (ПР), які здобуває студент спеціальності 122 «Комп'ютерні науки» під час вивчення цієї навчальної дисципліни.

ПР 11. Володіти навичками управління життєвим циклом програмного забезпечення, продуктів і сервісів інформаційних технологій відповідно до вимог і обмежень замовника, вміти розробляти проєктну документацію (техніко-економічне обґрунтування, технічне завдання, бізнес-план, угоду, договір, контракт).

ПР 13. Володіти мовами системного програмування та методами розробки програм, що взаємодіють з компонентами комп'ютерних систем, знати мережні технології, архітектури комп'ютерних мереж, мати практичні навички технології адміністрування комп'ютерних мереж та їх програмного забезпечення.

ПР 15. Застосовувати знання методології та CASE-засобів проєктування складних систем, методів структурного аналізу систем, об'єктно-орієнтованої методології проєктування при розробці і дослідженні функціональних моделей організаційно-економічних і виробничо-технічних систем.

Згідно з вимогами Стандарту вищої освіти України, першого (бакалаврського) рівня, галузі знань 12 Інформаційні технології, спеціальності 123 «Комп'ютерна інженерія» навчальна дисципліна «Технології DevOps» забезпечує набуття здобувачами вищої освіти таких **компетентностей**.

ЗК 3. Здатність проводити дослідження на відповідному рівні.

ЗК 4. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.

ЗК 6. Здатність виявляти, ставити та вирішувати проблеми.

ЗК 7. Здатність приймати обґрунтовані рішення.

СК 2. Здатність розробляти алгоритмічне та програмне забезпечення, компоненти комп'ютерних систем та мереж, Інтернет додатків, кіберфізичних систем з використанням сучасних методів і мов програмування, а також засобів і систем автоматизації проєктування.

СК 5. Здатність будувати архітектуру та створювати системне і прикладне програмне забезпечення комп'ютерних систем та мереж.

СК 6. Здатність використовувати та впроваджувати нові технології, включаючи технології розумних, мобільних, зелених і безпечних обчислень, брати участь в модернізації та реконструкції комп'ютерних систем та мереж, різноманітних вбудованих і розподілених додатків, зокрема з метою підвищення їх ефективності.

СК 8. Здатність забезпечувати якість продуктів і сервісів інформаційних технологій на протязі їх життєвого циклу.

СК 11. Здатність обирати ефективні методи розв'язування складних задач комп'ютерної інженерії, критично оцінювати отримані результати та аргументувати прийняті рішення.

Програмні результати (РН), які здобуває студент спеціальності 123 «Комп'ютерна інженерія» під час вивчення цієї навчальної дисципліни:

РН 5. Розробляти і реалізовувати проєкти у сфері комп'ютерної інженерії та дотичні до неї міждисциплінарні проєкти з урахуванням інженерних, соціальних, економічних, правових та інших аспектів.

РН 8. Застосовувати знання технічних характеристик, конструктивних особливостей, призначення і правил експлуатації програмно-технічних засобів комп'ютерних систем та мереж для вирішення складних задач комп'ютерної інженерії та дотичних проблем.

РН 11. Приймати ефективні рішення з питань розроблення, впровадження та експлуатації комп'ютерних систем і мереж, аналізувати альтернативи, оцінювати ризики та імовірні наслідки рішень.

РН 13. Зрозуміло і недвозначно доносити власні знання, висновки та аргументацію з питань інформаційних технологій і дотичних між галузевих питань до фахівців і нефахівців, зокрема до осіб, які навчаються.

1 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Взаємодія та специфіка робота з GitHub. Створення першого репозиторію

Мета: ознайомитися із системою GitHub та навчитися користуватися терміналом Git Bash.

Короткі теоретичні відомості

GitHub – це інструмент, який дуже корисний для розробників програмного забезпечення. Він допомагає зберігати, спільно працювати та вдосконалювати код.

Переваги GitHub

1. Збереження коду: GitHub – це ніби сховище для вашого коду. Ви можете зберігати свій код тут і не більше не турбуватися про втрату даних. Якщо ваш комп’ютер зламається, ви все одно зможете отримати свій код з GitHub.

2. Спільна робота: Ви можете працювати над проєктами разом з іншими розробниками, навіть якщо ви далеко від них географічно. GitHub надає зручні інструменти для спільної роботи з кодом.

3. Історія змін: GitHub відстежує всі зміни, які ви робите у своєму коді. Це допомагає зрозуміти, як розвивався проєкт з часом, і відстежувати помилки.

4. Відкритий код: багато проєктів на GitHub є відкритими, і це означає, що вони вільно доступні для всіх. Це сприяє розвитку програмної галузі та навчанню.

5. Інструменти для автоматизації: GitHub надає можливості автоматизувати багато рутинних завдань, таких як тестування та розгортання коду.

Недоліки GitHub

Приватність даних: незважаючи на можливість створення приватних репозиторіїв, ваш код все одно може бути доступним для певних людей, якщо ви необережні з налагоджуванням безпеки.

Великий обсяг інформації: для новачків GitHub може здавптися складним через велику кількість функцій та інформації.

Чому всі використовують GitHub?

GitHub є стандартом щодо розробки програмного забезпечення через свою надійність, зручність і спрощення спільної роботи. Він допомагає розробникам у всьому світі створювати якісний код, спільно працювати з проєктами і навіть знайти роботу, демонструючи свої навички через свій профіль на GitHub.

Головні характеристики Git Bash

1. Командний рядок Git: Git Bash надає доступ до команд Git, що дозволяє розробникам використовувати всі функції Git для створення, коміту, гілок, злиття та інших операцій керування версіями.

2. Сумісність зі синтаксисом Unix: Git Bash побудований на основі Unix-подібного терміналу, що означає, що ви можете використовувати традиційні Unix-команди в середовищі Windows.

3. Підтримка скриптів: ви можете створювати скрипти та автоматизовані завдання за допомогою Git Bash, що робить його корисним для автоматизації рутинних завдань розробки.

4. Інтеграція з іншими інструментами розробки: Git Bash може бути інтегрований з іншими інструментами розробки, такими як редактори коду, для полегшення роботи з Git в контексті вашого проєкту.

Порядок виконання роботи

Крок 1. Спочатку необхідно зареєструватися на GitHub за посиланням (https://github.com/signup?return_to=https%3A%2F%2Fgithub.com%2Fsignup&source=login) та завантажити Git Bash за посиланням (https://git-scm.com/downloads).

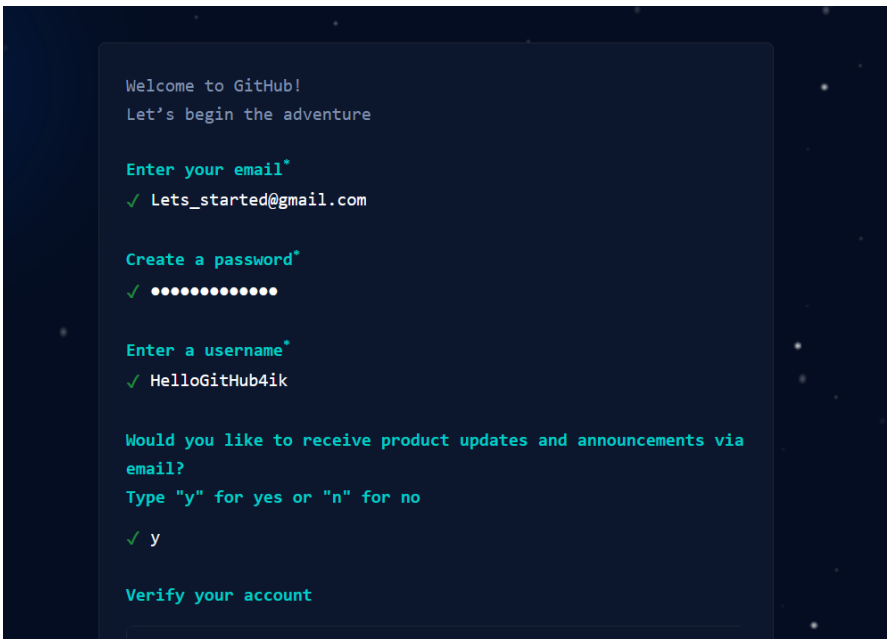

Рисунок 1.1 – Реєстрація на GitHub

Також установіть Git на свій комп’ютер за посиланням (https://desktop.github.com/).

Крок 2. Після того як створили обліковий запис та увійшли, необхідно створити перший репозиторій.

Репозиторій зазвичай використовують для організації одного проєкту. Репозиторії можуть містити теки та файли, зображення, відео, електронні таблиці та набори даних – усе, що потрібно для вашого проєкту. Зазвичай репозиторії містять файл README з інформацією про ваш проєкт. Файли README написані мовою Markdown у звичайному текстовому форматі. Ви можете скористатися такою «шпаргалкою» для ознайомлення з синтаксисом Markdown. GitHub дозволяє додати файл README в той час, коли ви створюєте новий репозиторій. Також GitHub пропонує інші загальні параметри, такі як ліцензійний файл, але ви не повинні вибирати їх зараз.

Ваш репозиторій «hello-world» може бути місцем, де ви зберігаєте ідеї, ресурси або навіть ділитесь та обговорюєте дещо з іншими.

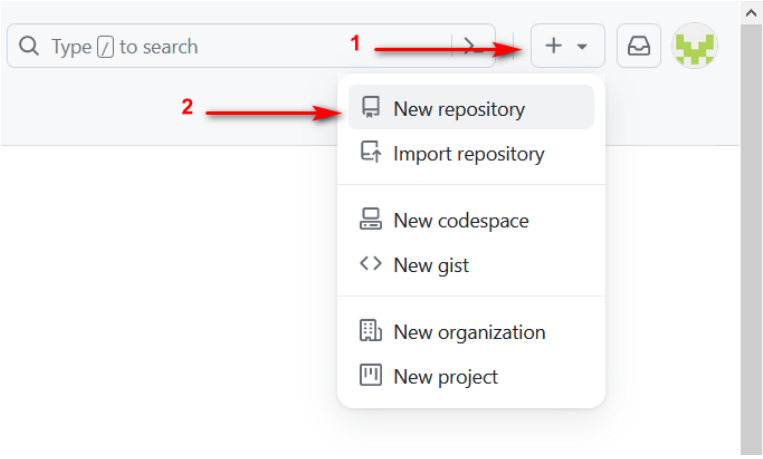

Рисунок 1.2 – Створення нового репозиторію

Щоб створити репозиторій, необхідно у верхньому правому кутку натиснути на випадне меню «+», а далі виберіть «**New repository**».

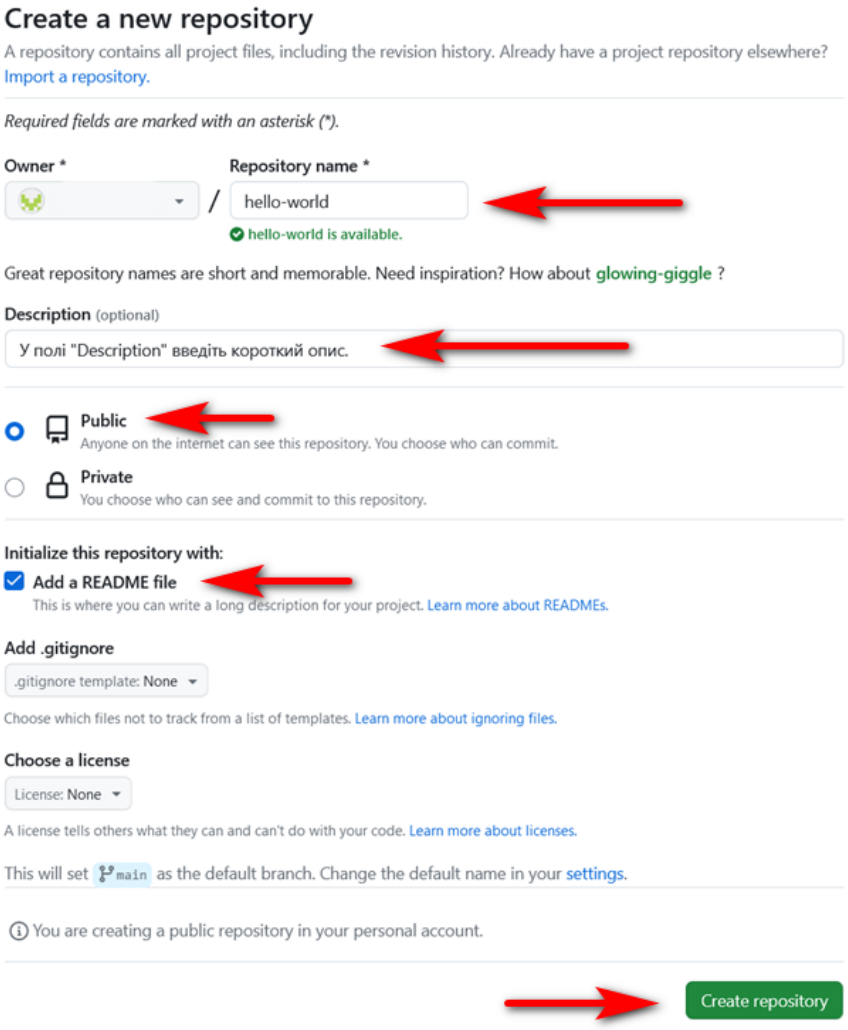

Рисунок 1.3 – Створення опису нового репозиторію

Після того як потрапили на сторінку для створення репозиторію, необхідно виконати такі дії.

1. У полі «Description» введіть короткий опис.

2. Виберіть, чи буде ваш репозиторій «**Public**» або «**Private**». В нашому випадку він буде «**Public**».

3. Виберіть «**Add a README file**».

4. Натисніть «**Create repository**».

Крок 3. Створення гілки «Creating a branch».

Створення гілки дозволяє мати кілька різних версій репозиторію одночасно.

За замовчуванням у вашому репозиторії є одна гілка, названа «main», яка вважається головною. Ви можете створити додаткові гілки від гілки «main» у вашому репозиторії. Гілки можна використовувати, щоб мати кілька різних версій проєкту одночасно. Це корисно, коли потрібно додати нові функції до проєкту, не змінюючи головний код. Робота, яка виконується в різних гілках, не відображатиметься в головній гілці, поки ви її не об'єднаєте. Гілки дозволяють експериментувати та вносити зміни до включення їх в головну гілку.

Коли ви створюєте гілку від гілки «main», ви створюєте копію або знімок «main» на той момент. Якщо хтось інший вніс зміни в гілку «main», поки ви працювали зі своєю гілкою, ви можете включити ці оновлення.

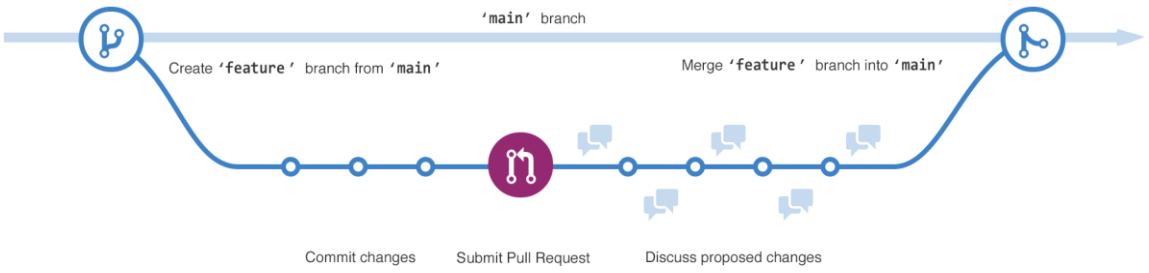

Рисунок 1.4 – Вигляд додаткової гілки

На цій діаграмі показано:

1. Гілку «main».

2. Нову гілку з назвою «feature».

3. Подорож, яку робить «feature» до її об’єднання в «main».

Різні версії файлу можуть мати різний вигляд. Наприклад:

– story.txt

– story-edit.txt

– story-edit-reviewed.txt

Гілки досягають схожих цілей у репозиторіях на GitHub.

У GitHub розробники, письменники та дизайнери використовують гілки для відокремлення виправлень помилок і роботи з функціями від головної (продукційної) гілки. Коли зміна готова, вони об’єднують свою гілку в головну.

Отже, коли зрозуміло для чого потрібні гілки, створюємо першу гілку:

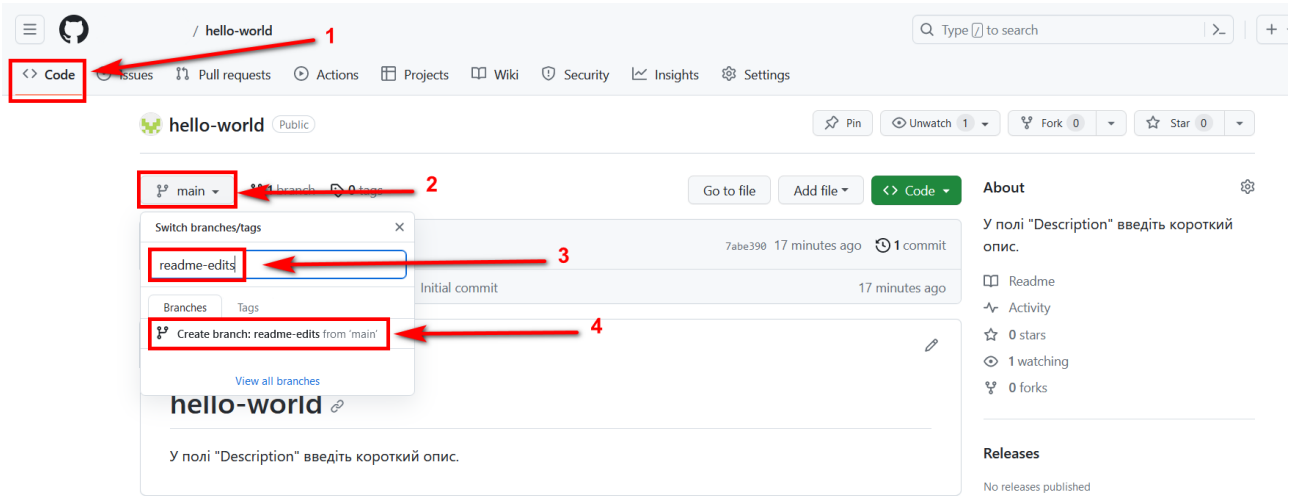

Рисунок 1.5 – Створення нової гілки

1. Натисніть на вкладці «Code» вашого репозиторію «hello-world».

2. Над списком файлів натисніть на розкривному меню, яке вказує «main».

3. Введіть назву гілки «readme-edits» у текстове поле.

4. Натисніть «Створити гілку: readme-edits з main».

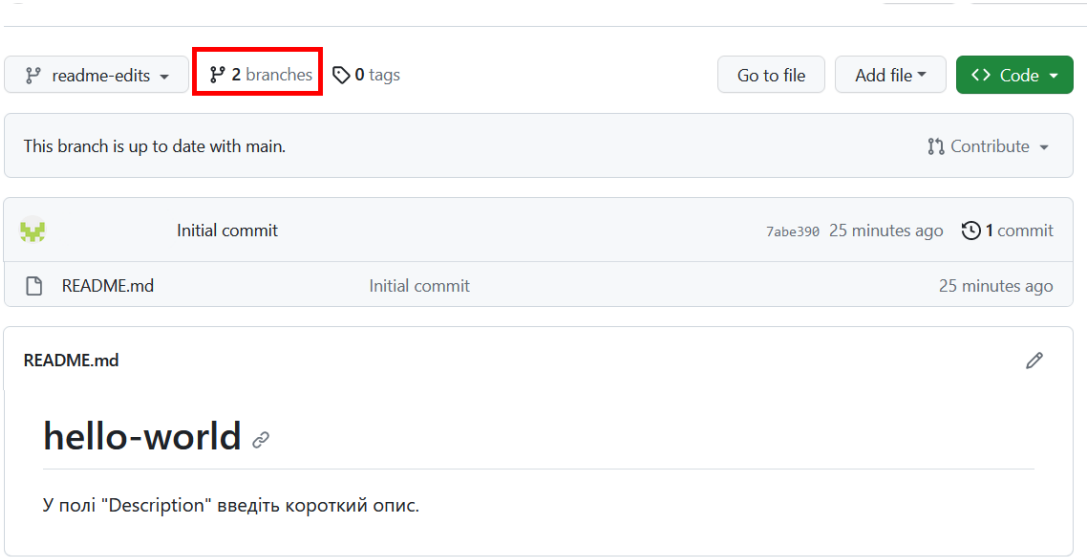

Рисунок 1.6 – Кількість гілок в репозиторії

Тепер у вас є дві гілки, «main» і «readme-edits». Зараз вони виглядають точно так само. Далі додамо зміни до нової гілки.

Крок 4. Створення та збереження змін.

Після створення нової гілки на попередньому кроці, GitHub перенаправив вас на сторінку «https://github.com/Ваш_нікнейм/hello-world/tree/readme-edits» коду вашої нової гілки «readme-edits», яка є копією «main».

Ви можете вносити та зберігати зміни в файлах свого репозиторію. На GitHub збережені зміни називаються комітами «commit».

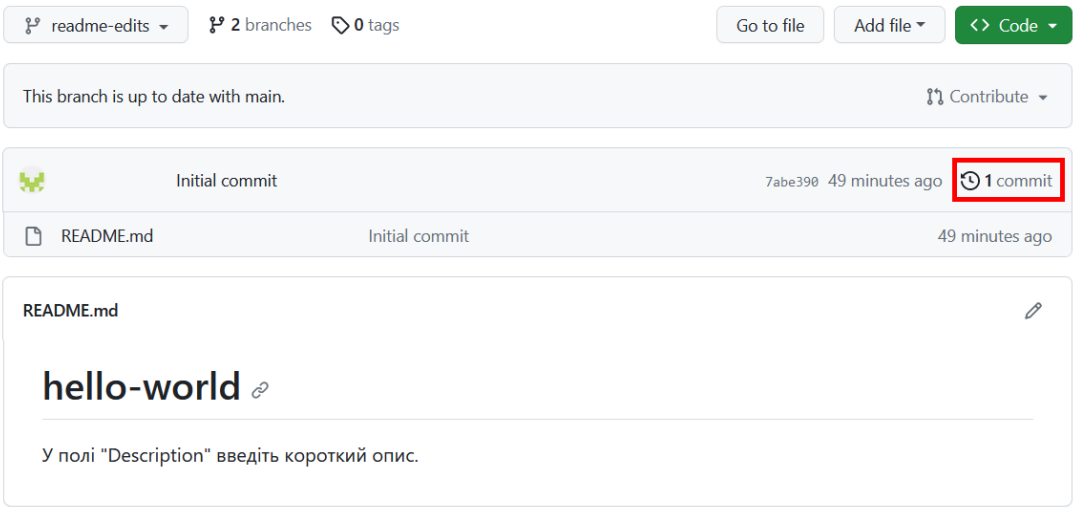

Рисунок 1.7 – Кількість комітів у репозиторії

Кожен коміт має пов’язаний з ним коментар, який є описом, що пояснює, чому була зроблена певна зміна. Коментарі до комітів зафіксовують історію ваших змін, щоб інші учасники могли зрозуміти, що ви зробили і чому.

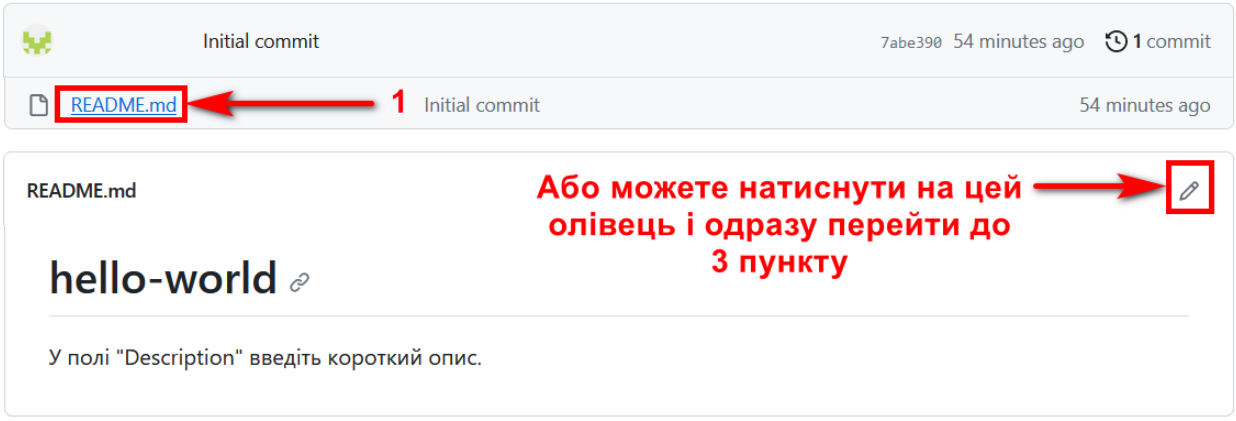

Рисунок 1.8 – Відкриття файлу для редагування

1. Під гілкою «readme-edits», яку ви створили, клацніть на файл README.md.

2. Щоб відредагувати файл, натисніть ✎.

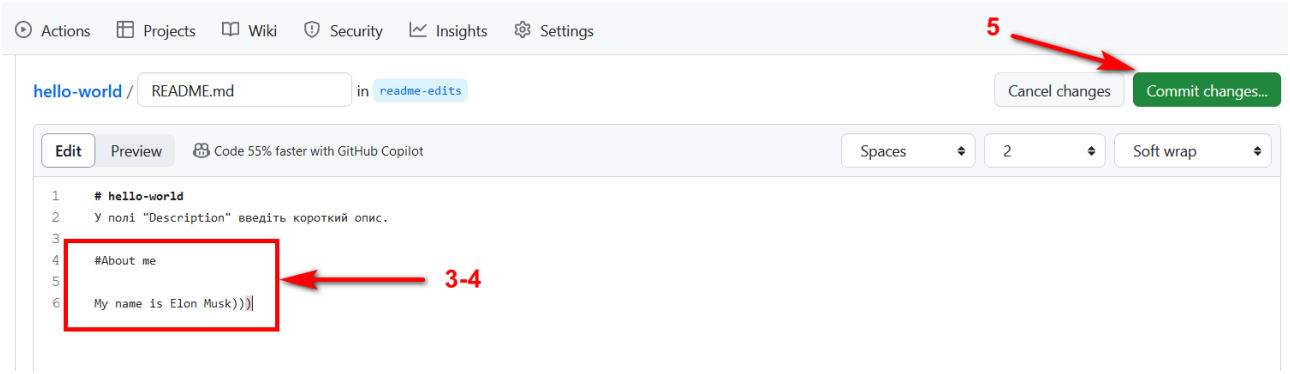

Рисунок 1.9 – Додавання нових змін у файл

3. У редакторі напишіть дещо про себе.

4. Спробуйте використовувати різні елементи Markdown.

5. Натисніть «Commit changes».

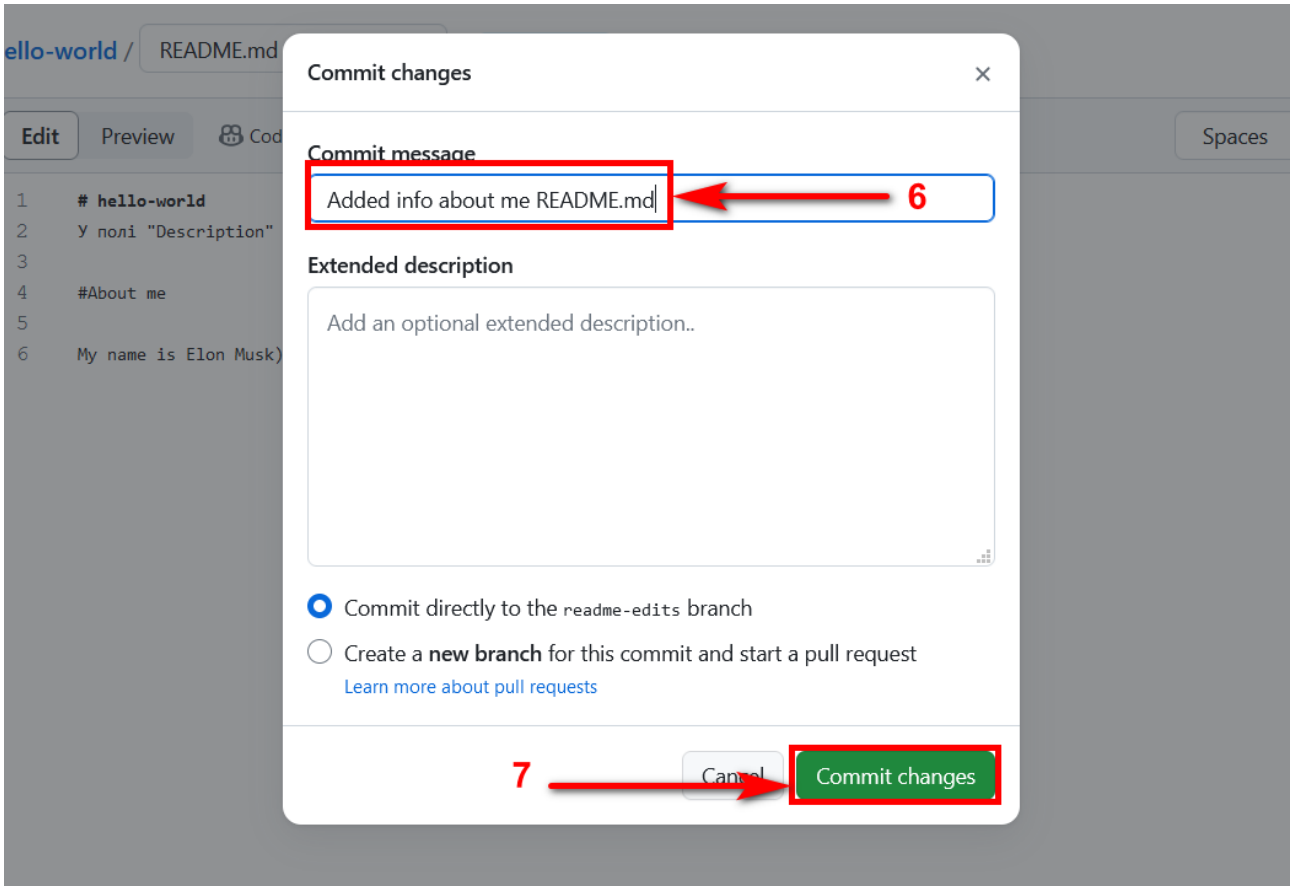

Рисунок 1.10 – Додавання нового коміту до створених змін

6. У полі «Commit changes» напишіть коментар до коміту, який описує ваші зміни.

7. Натисніть кнопку «Commit changes».

Ці зміни будуть внесені лише до файлу README у вашій гілці «readme-edits», тепер ця гілка містить контент, відмінний від головної («main»).

Крок 5. Відкриття запиту на включення змін «pull request».

Тепер, коли у вас є зміни в гілці, відокремленій від головної, ви можете відкрити запит на включення змін.

Запити на включення змін – це підґрунтя спільної роботи на GitHub. Коли ви відкриваєте запит на включення змін, ви пропонуєте свої зміни та запитуєте когось переглянути та включити ваш внесок та об’єднати їх у свою гілку. Запити на включення змін показують відмінності або різницю вмісту з обох гілок. Зміни, додавання та видалення показані різними кольорами.

Як тільки ви зробите коміт, ви можете відкрити запит на включення змін та розпочати обговорення навіть до завершення написання коду.

За допомогою функції @mention GitHub у вашому повідомленні запиту на включення змін ви можете попросити про зворотний зв'язок конкретних осіб чи команд, незалежно від того, чи знаходяться вони в сусідній кімнаті, чи в інших часових поясах.

Ви навіть можете відкривати запити на включення змін у своєму власному репозиторії та об'єднувати їх самостійно. Це чудовий спосіб вивчити робочий процес GitHub перед роботою з більшими проєктами.

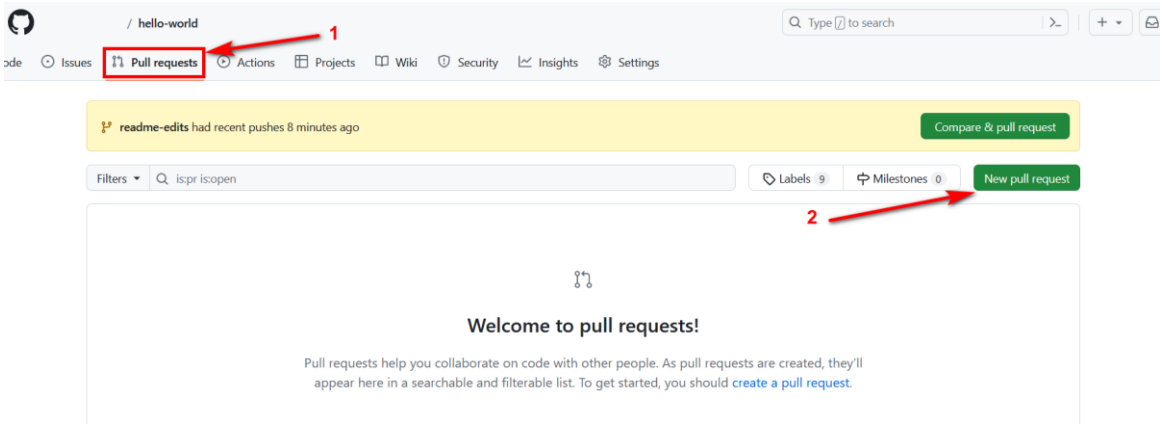

Рисунок 1.11 – Створення нового «**Pull requests**»

1. Натисніть вкладку «**Pull requests**» вашого репозиторію «hello-world».
2. Натисніть «New pull request».

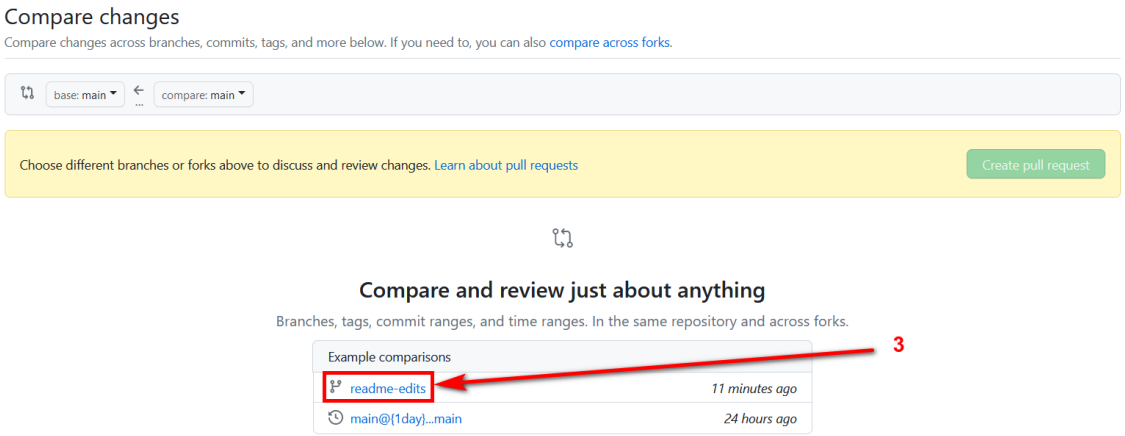

Рисунок 1.12 – Вибір гілки

3. У блоці «Example Comparisons» виберіть гілку, яку ви створили, «readme-edits», для порівняння з «main» (головною).

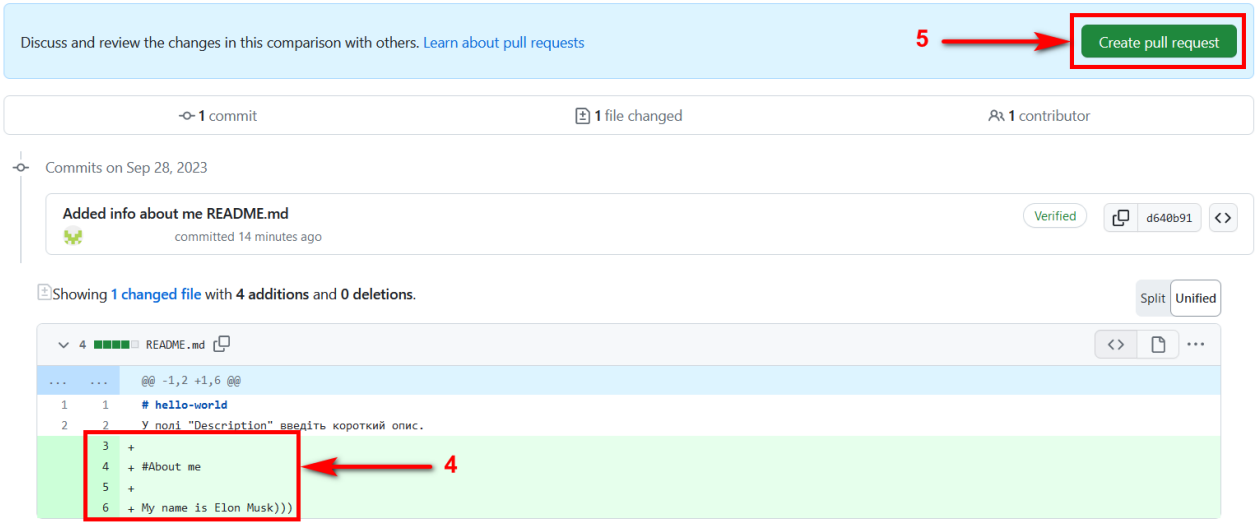

Рисунок 1.13 – Перегляд внесених змін

4. Перегляньте ваші зміни у різницях на сторінці порівняння і переконайтеся, що вони відповідають тому, що ви хочете подати.

5. Натисніть «Create pull request».

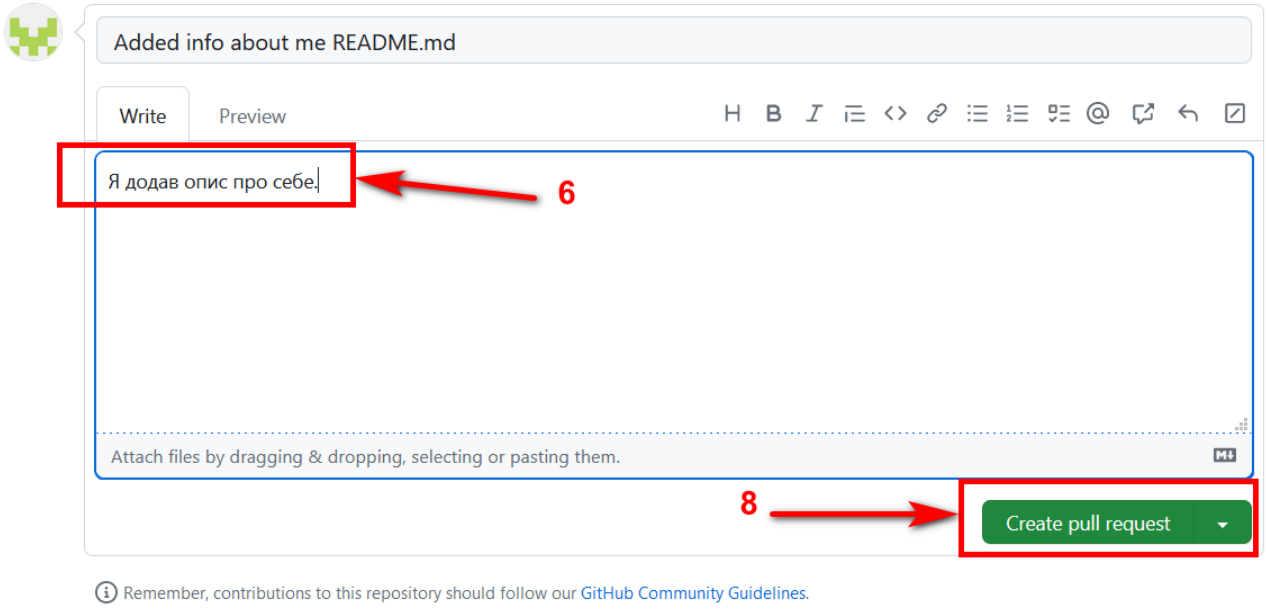

Рисунок 1.14 – Додавання опису та «Create pull request»

6. Дайте своєму запиту на включення змін заголовок і напишіть короткий опис ваших змін. Ви можете включити емодзі та перетягувати зображення та гіфки.

7. Не обов'язково, справа від вашого заголовка та опису, клацніть біля Рецензентів, Відділених, Міток, Проєктів чи Плану, щоб додати одну з цих опцій до вашого запиту на включення змін. Зараз не обов'язково додавати жодну з них, але ці опції пропонують різні способи співпраці через запити на включення змін. Докладніше про це дивіться в розділі «About pull requests».

8. Натисніть «Create pull request».

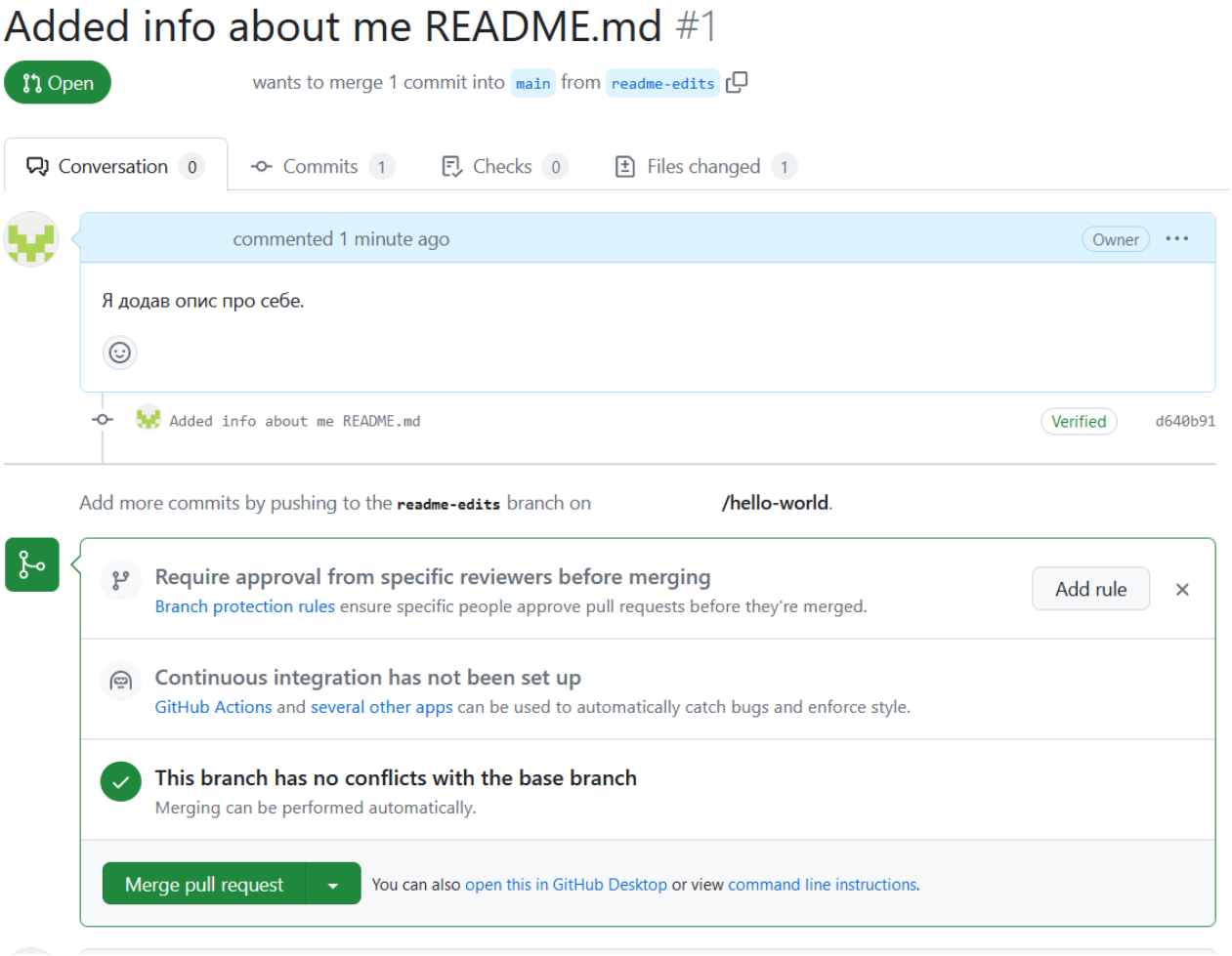

Рисунок 1.15 – Перевірка нових даних з актуальними в гілці «main»

Тепер ваші співробітники можуть переглянути ваші правки і надати пропозиції.

Крок 6. Об’єднання вашого запиту на включення змін «Merging your pull request».

На цьому останньому кроці ви об’єднаєте вашу гілку «readme-edits» з головною гілкою. Після об’єднання вашого запиту на включення змін зміни у вашій гілці «readme-edits» будуть включені в головну гілку.

Іноді запит на включення змін може внести зміни в код, які конфліктують з наявним кодом у головній гілці. Якщо є конфлікти, GitHub повідомить вас про конфліктний код і не дозволить об’єднання, поки конфлікти не будуть вирішені. Ви можете зробити коміт, який вирішує конфлікти, або використовувати коментарі в запиті на включення змін для обговорення конфліктів з членами вашої команди.

У цьому кроці не повинно бути конфліктів, тому ви готові об’єднати вашу гілку з головною гілкою.

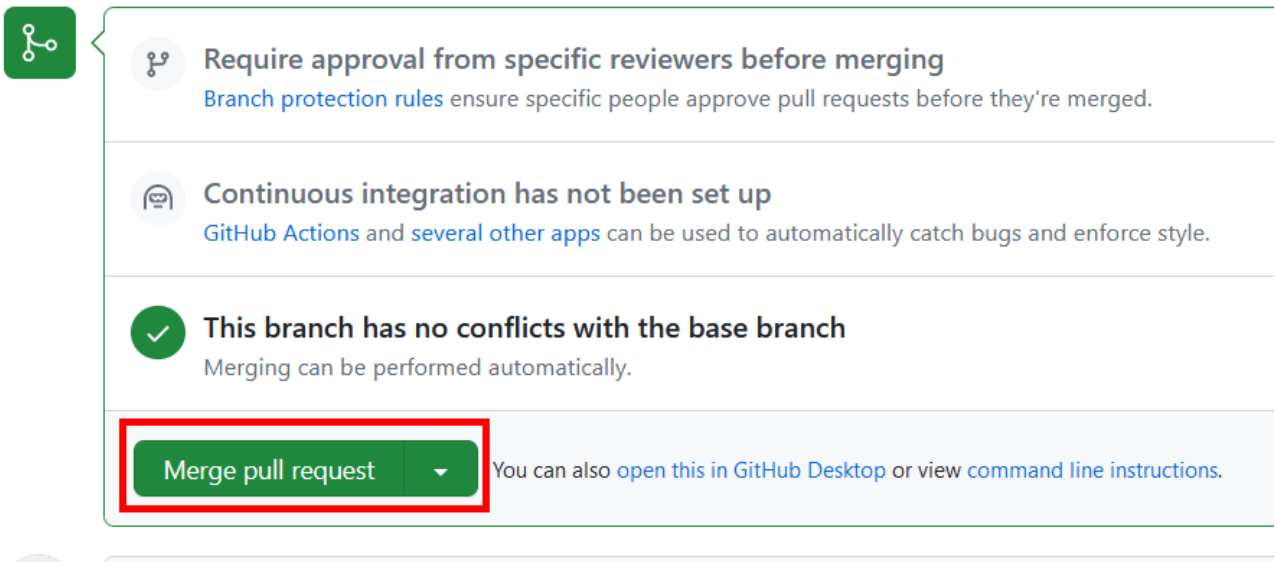

Рисунок 1.16 – Об’єднання змін з головною гілкою»Merge pull request»

1. Внизу запиту на включення змін натисніть «Merge pull request», щоб об’єднати зміни у головну гілку.

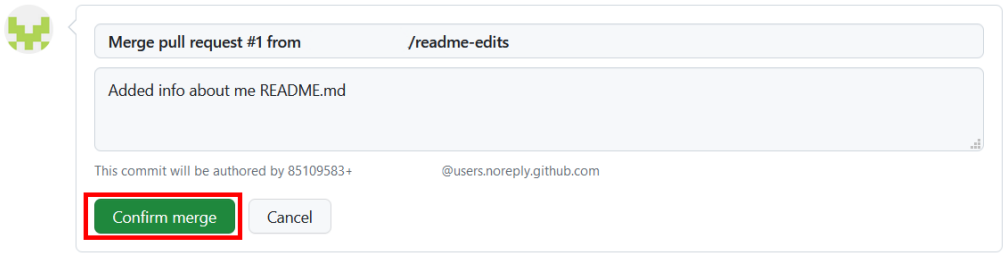

Рисунок 1.17 – Підтвердження змін «Confirm merge»

2. Після натискання на кнопку «Confirm merge» Ви отримаєте повідомлення про те, що запит був успішно об’єднаний і запит був закритий.

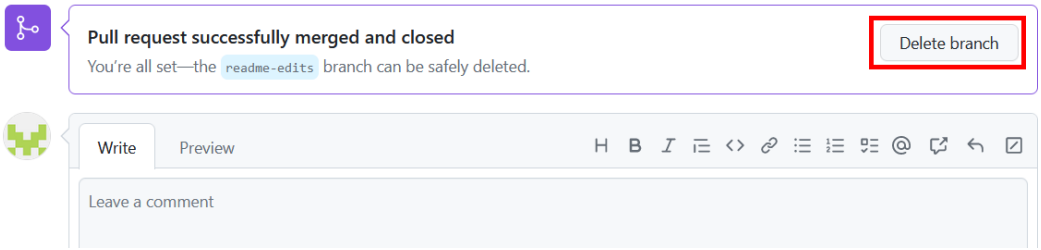

Рисунок 1.18 – Видалення гілки «Delete branch»

3. Натисніть «Delete branch». Тепер, коли ваш запит на включення змін об’єднаний, і ваші зміни знаходяться в головній гілці, ви можете безпечно видалити гілку «readme-edits».

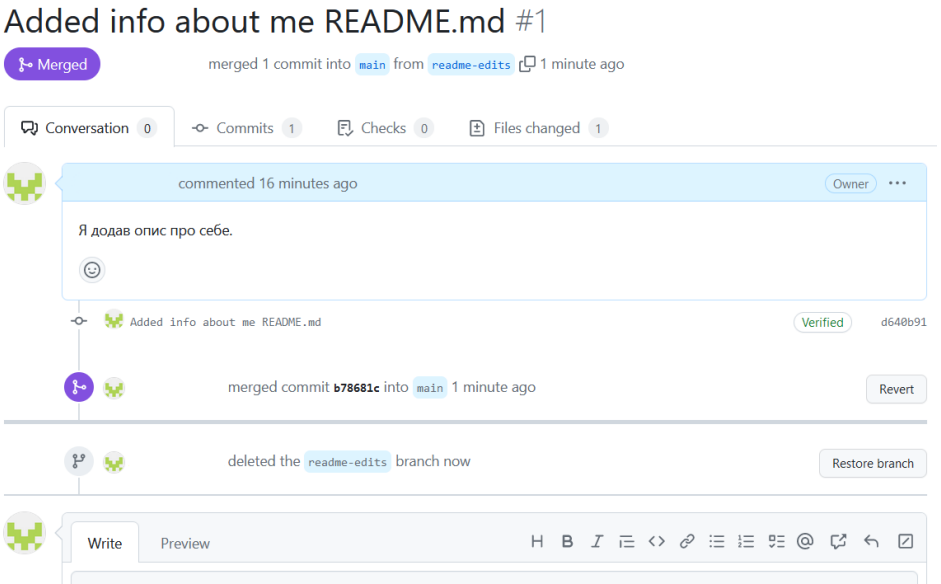

Рисунок 1.19 – Інформація про успішне об’єднання та видалення гілки

Якщо ви хочете внести більше змін у свій проєкт, ви завжди можете створити нову гілку і повторити цей процес.

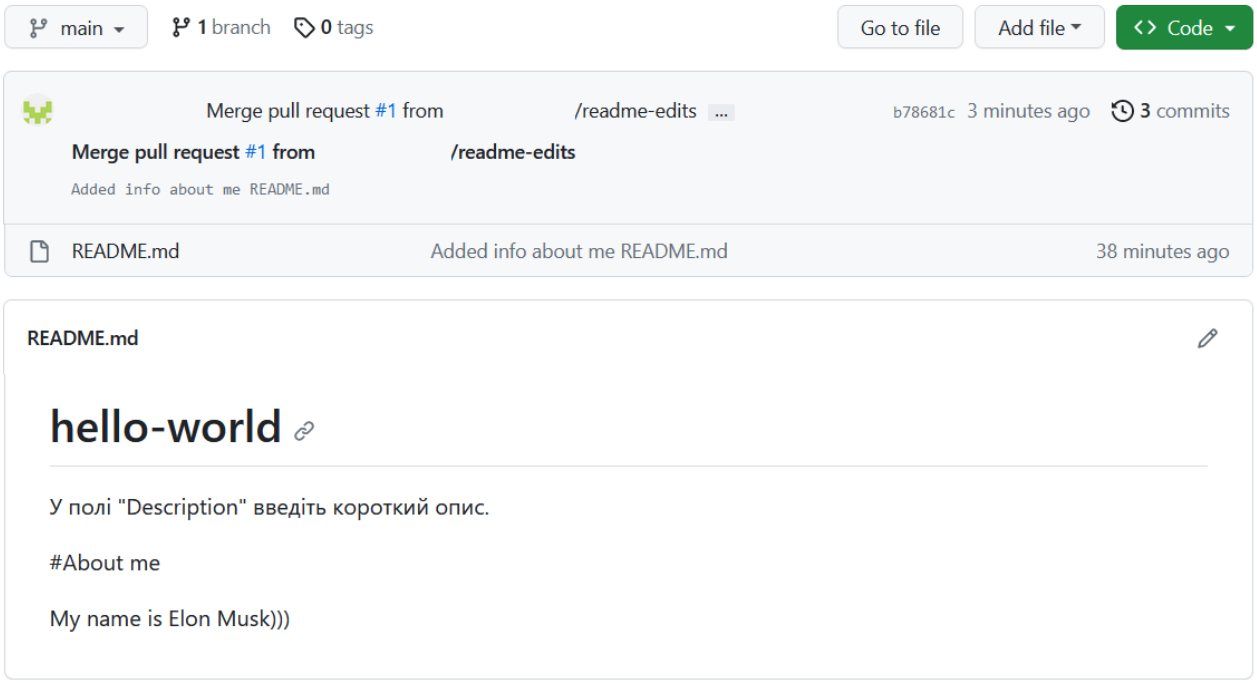

Рисунок 1.20 – Додаткова інформація про об’єднання на вкладці Code

Крок 7. Тепер необхідно перейти у термінал. Це можна зробити двома способами:

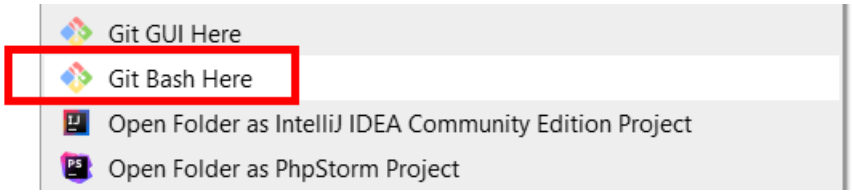

Рисунок 1.21 – Запуск Git Bash в папці з необхідним для завантаження файлом

– відкрити Git Bash через exe файл та через команду cd перейти в необхідну папку, в якій знаходиться код, який необхідно завантажити в репозиторій;

– або відрийте необхідну папку, в якій міститься код для завантаження в репозиторій. Далі на вільному місці натисніть праву кнопку миші та виберіть варіант «Git Bash Here».

Після того як ми опинилися в необхідній папці, потрібно обрати, якщо хочете задати одне (в нашому випадку одне) ім’я користувача для всіх репозиторіїв на комп’ютері, напишіть в командному рядку команду:

git config – global user.name «<ваше_ім’я>»

Замініть <ваше_ім’я> на своє ім’я в лапках. Можете написати будь-що. Якщо хочете задати ім’я лише одному репозиторію, то видаліть з команди слово global.

Тепер напишіть свою адресу електронної пошти. Простежте, щоб вона збігалася з адресою, вказаною під час реєстрації на GitHub.

git config – global user.email «електронна_адреса@email.com»

Далі необхідно ініціалізувати папку, для цього виконайте таку команду:

git init

Після ініціалізації необхідно завантажити наш файл README.md з репозиторію GitHub у папку, для цього використайте команду «git clone» та посилання на ваш репозиторій.

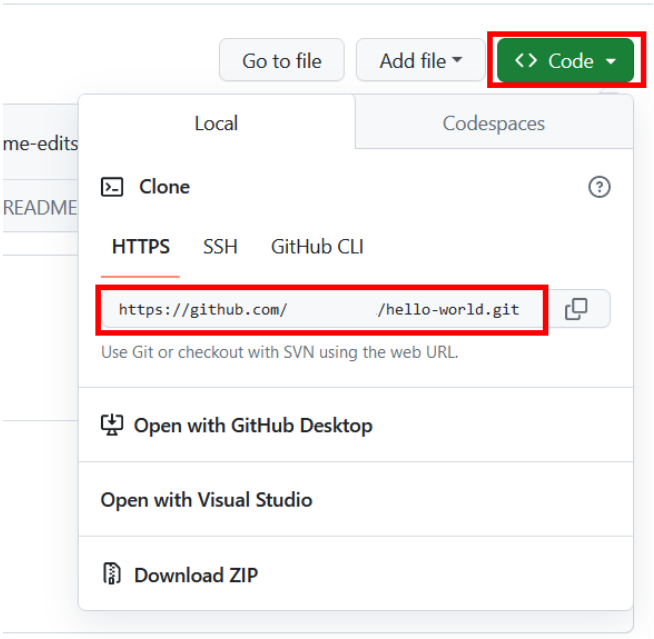

Рисунок 1.22 – Місцезнаходження посилання на репозиторій

Щоб узяти посилання на репозиторій, можна натиснути на зелену кнопку code та скопіювати посилання в один клік.

git clone https://github.com/Ваш_нікнейм/hello-world.git

Коли файл буде завантажений до вашої папки, його необхідно відкрити та видалити напис «У полі «Description» введіть короткий опис». Замість нього впишіть назву своє групи та нижче після «#About me» додайте ще певну інформацію, наприклад, останні новини минулого дня, і збережіть зміни.

Завдання для самостійного виконання

Необхідно нижче ознайомитися з додатковою інформацією про команди. Також потрібно виконати такі завдання.

1. Створити нову гілку в репозиторії.

2. Перейти до цієї гілки.

3. Завантажити на GitHub файл README.md зі змінами та перевірити його в репозиторії.

4. Створити файл або файли в папці, наповнити його, можливо, якимось кодом з лабораторної роботи з іншої навчальної дисципліни.

5. Завантажити файл або файли в цю саму нову гілку.

6. Виконати кроки, які робили раніше, з кроку № 5 по № 7.

Додаткова інформація:

Початкове налаштування. Налаштування інформації про користувача для всіх локальних репозиторіїв

$ git config --global user.name "[ім'я]"

Встановлює ім'я, яке відображатиметься в полі автора у комітів, які ви виконуєте.

$ git config --global user.email "[адреса електронної пошти]"

Встановлює адресу електронної пошти, яка відображатиметься в інформації про коміти, які ви виконуєте.

Внесення змін в репозиторій. Перегляд змін та створення коммітів (фіксація змін).

$ git status

Перелічує всі нові або змінені файли, які потребують фіксації.

$ git diff

Показує відмінності щодо внесених змін у ще не проіндексованих файлах.

$ git add [файл]

Індексує вказаний файл для наступного коміту.

$ git diff --staged

Показує різницю між проіндексованою та останньою зафіксованою версіями файлів.

$ git reset [файл]

Скасовує індексацію вказаного файлу, зберігаючи його вміст.

$ git commit -m "[повідомлення з описом]"

Фіксує проіндексовані зміни та зберігає їх в історію версій.

Створення репозиторію. Створення нового репозиторію або отримання його за наявною URL-адресою.

$ git init [назва проекту]

Створює новий локальний репозиторій із заданим ім’ям.

$ git clone [url-адреса]

Завантажує репозиторій разом з усією його історією змін.

Колективна робота. Іменовані серії комітів та поєднання результатів роботи.

$ git branch

Список іменованих гілок комітів із зазначенням обраної гілки.

$ git branch [ім'я гілки]

Створює нову гілку.

$ git switch -c [ім'я гілки]

Перемикається на вибрану гілку та оновлює робочу директорію до її стану.

$ git merge [ім'я гілки]

Вносить зміни вказаної гілки в поточну гілку.

$ git branch -d [ім'я гілки]

Видаляє вибрану гілку.

Операції з файлами. Переміщення та видалення версій файлів репозиторію.

$ git rm [файл]

Видаляє конкретний файл з робочої директорії та індексує його видалення.

$ git rm --cached [файл]

Забирає конкретний файл з контролю версій, але фізично залишає його на своєму місці.

$ git mv [оригінальний файл] [нове ім'я]

Переміщує та перейменовує вказаний файл, одразу індексуючи його для наступного коміту.

Перегляд історії. Перегляд та вивчення історії змін файлів проекту.

$ git log

Історія коммітів для поточної гілки.

$ git log --follow [файл]

Історія змін конкретного файлу, у тому числійого перейменування.

$ git diff [перша гілка]...[друга гілка]

Показує різницю між змістом коммітів двох гілок.

$ git show [комiт]

Виводить інформацію та показує зміни у вибраному коментарі.

Ігнорування деяких файлів. Виключення тимчасових та вторинних файлів та директорій.

**.log*

build/

*temp-**

Git ігноруватиме файли та директорії, перелічені у файлі .gitignore за допомогою wildcard синтаксису.

$ git ls-files --others --ignored --exclude-standard

Список усіх ігнорованих файлів у поточному проєкті.

Відкат комітів. Видалення помилок та коригування створеної історії.

$ git reset [комітт]

Скасує всі коміти після заданого, залишаючи всі зміни у робочій директорії.

$ git reset --hard [комітт]

Скидає всю історію разом зі станом робочої директорії до вказаного комміту.

Збереження фрагментів. Збереження та відновлення незавершених змін.

$ git stash

Тимчасово зберігає всі незафіксовані зміни файлів, що відстежуються.

$ git stash pop

Відновлює стан раніше збережених версій файлів.

$ git stash list

Виводить список усіх тимчасових збережень.

$ git stash drop

Скидає останні тимчасово збережені зміни.

Синхронізація з віддаленим репозиторієм. Реєстрація видаленого репозиторію та обмін змінами.

$ git fetch [віддалений репозиторій]

Завантажує всю історію з віддаленого репозиторію.

$ git merge [віддалений репозиторій]/[гілка]

Вносить зміни з гілки віддаленого репозиторію у поточну гілку локального репозиторію.

$ git push [віддалений репозиторій] [гілка]

Завантажує всі зміни локальної гілки у віддалений репозиторій.

$ git pull

Завантажує історію з віддаленого репозиторію та поєднує її з локальною. pull = fetch + merge

Зміст звіту

1. Номер, тема та мета лабораторної роботи.

2. Відповіді на контрольні питання.

3. Порядок і результати виконання самостійного завдання (скріншоти та текстовий опис кроків виконання завдань).

4. Висновки до роботи.

Контрольні питання

1. Що таке Git Bash і як він пов’язаний із GitHub?

2. Які головні переваги використання GitHub для спільної роботи з проєктами?

3. Які кроки потрібно виконати для створення нового репозиторію на GitHub і додавання змін до нього?

4. Які головні компоненти GitHub, що використовуються для роботи з репозиторіями та змінами в них?

5. Як можна відкрити запит на включення змін у GitHub і як це сприяє співпраці у розробці програмного забезпечення?

Література: [1, 5, 13].

Лабораторна робота № 2

Тема. AWS, розгортання типової серверної архітектури

Мета роботи: навчитися працювати з хмарними сервісами AWS та розгортати типову серверну архітектуру.

Короткі теоретичні відомості

1. Front-End: Apache.

Apache – це веб-сервер з відкритим кодом, який використовується для обробки запитів HTTP та HTTPS. Він є одним з найпопулярніших веб-серверів у світі, що використовується для розміщення веб-сайтів, веб-додатків та API.

Головні характеристики Apache:

– доступний безкоштовно та може бути вільно модифікований та розповсюджуваний;

– може бути налагоджений для роботи з різними типами веб-контенту, мовами програмування та платформами;

– має репутацію стабільного та безпечного веб-сервера;

– може обробляти велику кількість одночасних з'єднань.

Apache використовується як Front-End у DevOps, тому що:

– може бути легко інтегрований з іншими інструментами DevOps, такими як Ansible, Puppet, Chef та Docker;

– має широкий спектр функцій, які можуть бути корисними для розробників та операторів, таких як балансування навантаження, віртуальний хостинг та модульна система;

– має велику та активну спільноту користувачів і розробників, які можуть допомогти з будь-якими проблемами.

2. Back-End: PHP та Wordpress.

PHP – це динамічна мова програмування з відкритим кодом, яка використовується для створення веб-сайтів та веб-додатків. PHP може бути вбудований в HTML-код, що робить його зручним для створення динамічних веб-сторінок.

Wordpress – це система керування контентом (CMS) з відкритим кодом, яка використовується для створення веб-сайтів та блогів. Wordpress написаний на PHP і використовує MySQL як базу даних.

Головні характеристики Wordpress:

– простий у вивченні та використанні, навіть для людей без досвіду програмування;

– може бути розширений за допомогою плагінів та тем, щоб додати нові функції та змінити дизайн веб-сайту;

– має репутацію безпечної CMS, але важливо регулярно оновлювати програмне забезпечення та використовувати надійні паролі.

PHP та Wordpress використовуються як Back-End у DevOps, тому що:

– PHP – це динамічна мова програмування, яка може бути легко

інтегрована з іншими інструментами DevOps.

– Wordpress – це популярна CMS, яка має велику спільноту користувачів та розробників.

– Wordpress має широкий спектр плагінів та тем, які можуть бути корисними для розробників та операторів.

3. Database: MySQL.

MySQL – це реляційна система керування базами даних (RDBMS) з відкритим кодом, яка використовується для зберігання даних. MySQL є однією з найпопулярніших RDBMS у світі, що використовується для веб-сайтів, веб-додатків та інших систем.

Головні характеристики MySQL:

– доступний безкоштовно та може бути вільно модифікований та розповсюджуваний;

– може обробляти великі обсяги даних з високою швидкістю;

– має репутацію стабільної та безпечної RDBMS;

– може бути налагоджений для роботи з різними типами даних та платформами.

MySQL використовується як Database у DevOps, тому що:

– може бути легко інтегрований з іншими інструментами DevOps, такими як Ansible, Puppet, Chef та Docker;

– має широкий спектр функцій, які можуть бути корисними для розробників та операторів, таких як реплікація даних, балансування навантаження та резервне копіювання;

– має велику та активну спільноту користувачів та розробників, які можуть допомогти з будь-якими проблемами.

Порядок виконання роботи

Підготовка AWS акаунту до наступної роботи.

Для того щоб зареєструвати новий AWS акаунт слід перейти на сайт: https://aws.amazon.com/.

1. Натиснути кнопку «Create an AWS Account».

2. Ввести свою адресу електронної пошти та назву для нового акаунту (також сайт запропонує залогінитись у наявні акаунт).

3. Отримати лист з кодом підтвердження.

4. Внести його (код) у відповідне поле. Якщо код не надійшов, слід перевірити правильність написання пошти і папку зі спамом.

5. Після того, як пошту підтверджено, форма запропонує створити пароль.

Зверніть увагу, що пароль має бути достатньо складний.

6. Коли пароль успішно створено, система запитає про тип акаунту та контактну інформацію. Слід вибрати персональний (Personal) та заповнити контактну інформацію.

7. Наступним кроком Амазон запитає вашу банківську картку. Так, це обов’язково, але без додаткового запиту він нічого не зніматиме (максимум заморозить на кілька днів 1$ на карті, щоб перевірити, що вона реальна). Картку слід вказувати справжню. Валюта картки не має значення. Cardholder’s name – це просто назва для картки – введіть щось. щоб легше було її впізнати.

8. Наступним кроком потрібно буде підтвердити свою ідентичність шляхом унаслідок SMS на телефон. Виберіть країну, вкажіть телефон та капчу. SMS прийде майже миттєво.

9. Введіть отриманий код та натисніть «Продовжити».

10. Оберіть тарифний план для підтримки – нас цікавить безкоштовний тариф.

Розгортання трирівневої серверної інфраструктури на прикладі Wordpress.

Частина 1. Створення інстанса у Amazon EC2.

Для виконання цього завдання необхідно увійти в обліковий запис в Amazon Web Services.

На ілюстрації блакитним виділено секцію з нещодавно відвіданими сервісами. У цій секції необхідно знайти сервіс EC2.

У випадку, якщо секція порожня, EC2 можна знайти у меню сервісів (вгорі зліва, виділено червоним).

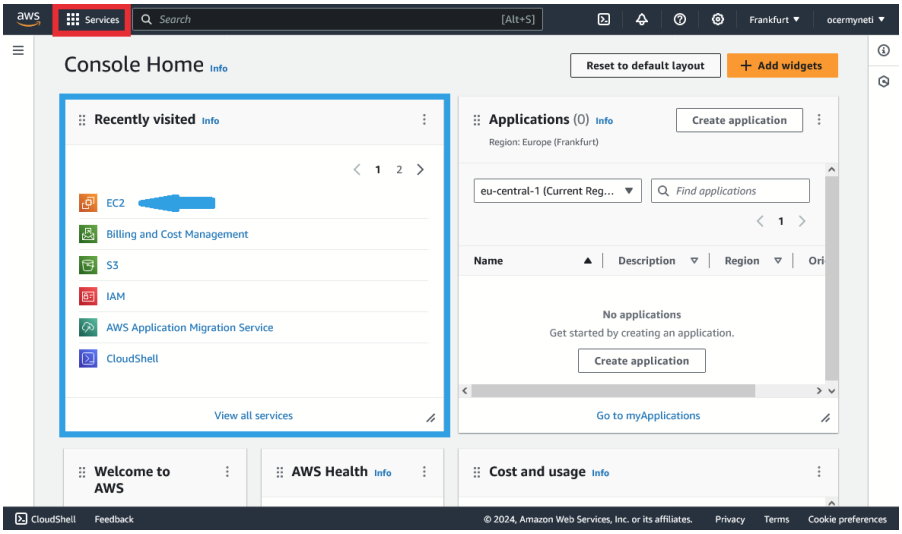

Рисунок 2.1 – Секція з нещодавно відвіданими сервісами

В меню сервісів можна скористатися пошуком або вибрати у лівій секції «Compute», а в правій – «EC2».

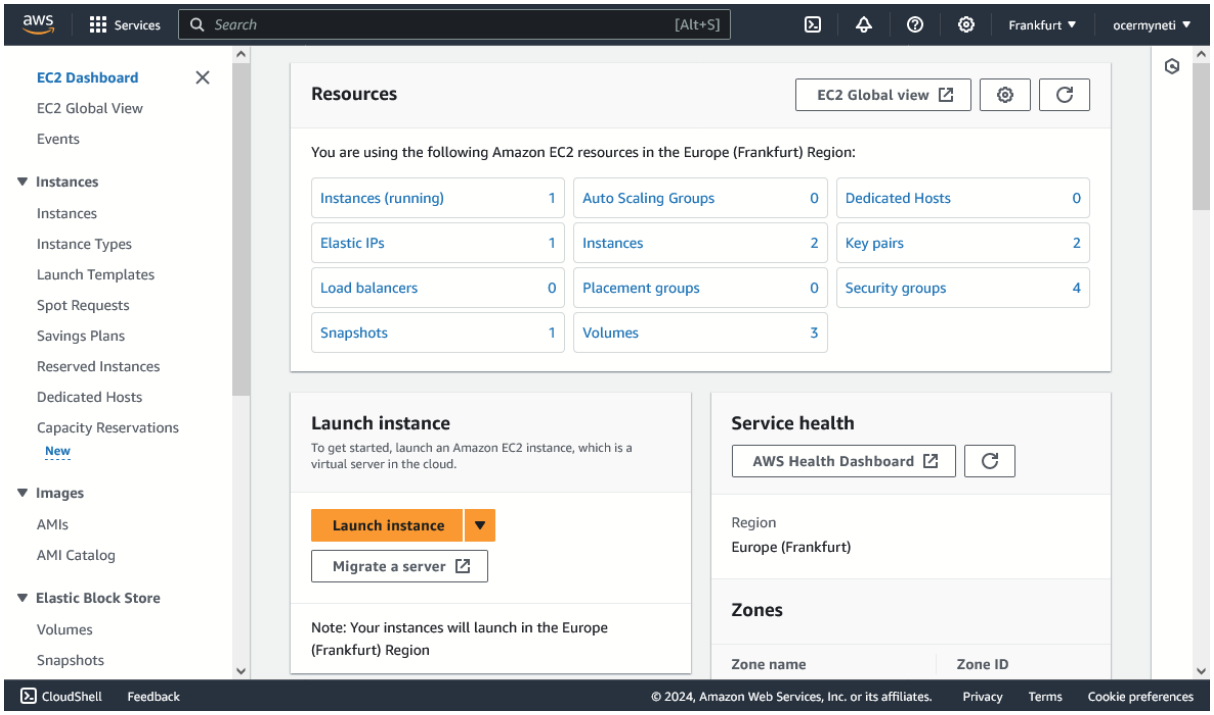

Рисунок 2.2 – Вибір «Compute» і «EC2»

Зайшовши на сторінку EC2, виберіть потрібний регіон у випадному списку справа вгорі. Укажіть Frankfurt (eu-central-1). Після цього потрібно натиснути «Launch instance», щоб відкрити діалог запуску нового інстанса (віртуальної машини).

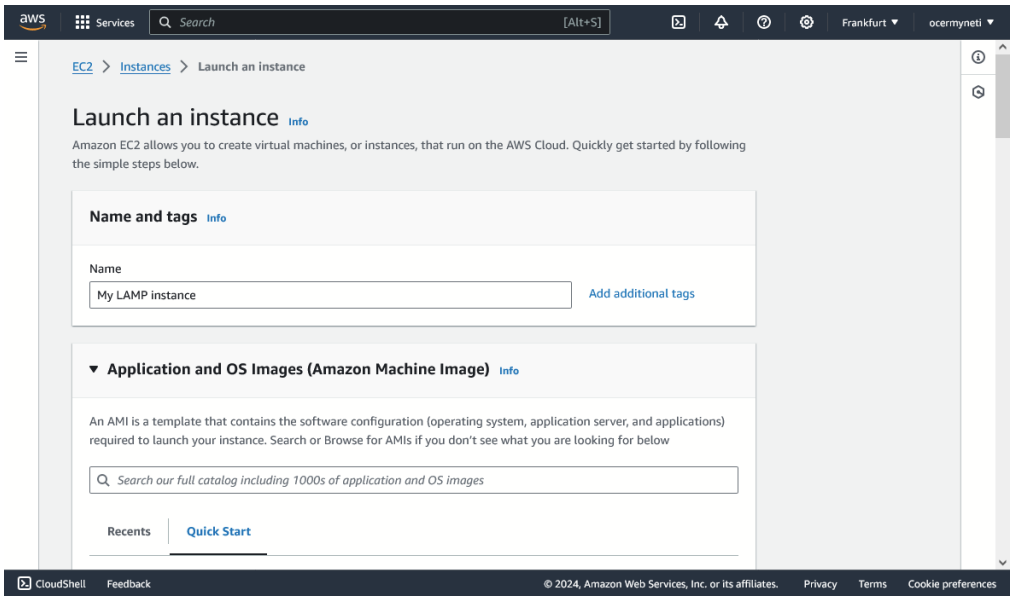

Рисунок 2.3 – Діалог запуску нового інстанса

У полі Name необхідно вказати ім'я для цього інстанса. Воно буде відображатись у списку ваших інстансів на сторінці EC2. Це ім'я не впливає на *hostname* машини, тобто настроєне в операційній системі мережне ім'я.

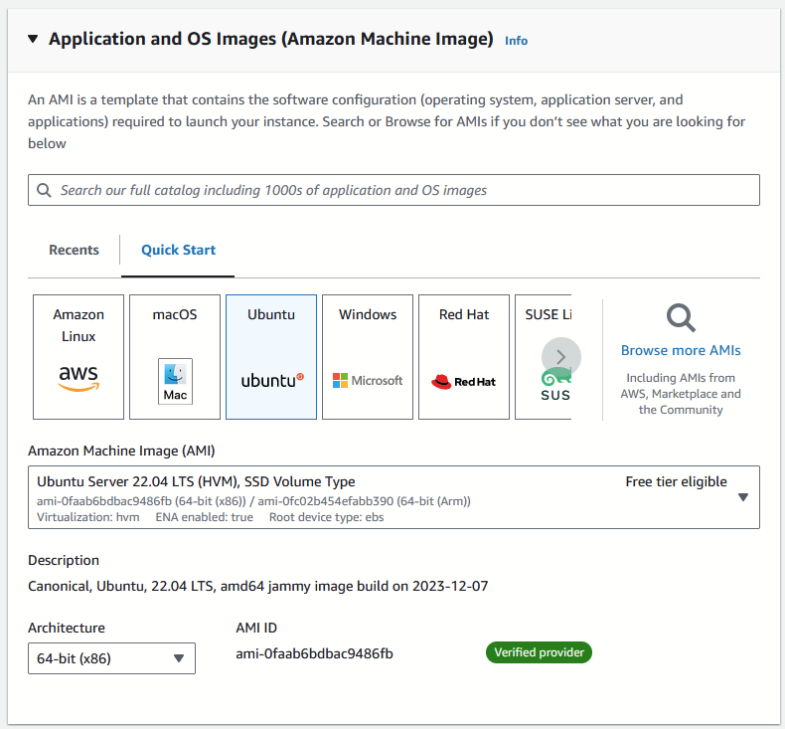

Рисунок 2.4 – Вибір ім'я для інстанса

Наступним етапом є вибір AMI (образу операційної системи). Для цієї роботи потрібно вибрати Ubuntu Server. При виборі Ubuntu версія 22.04 LTS наразі вибрана за замовчуванням, тому вона буде використана у цьому завданні.

Архітектура: 64-біт.

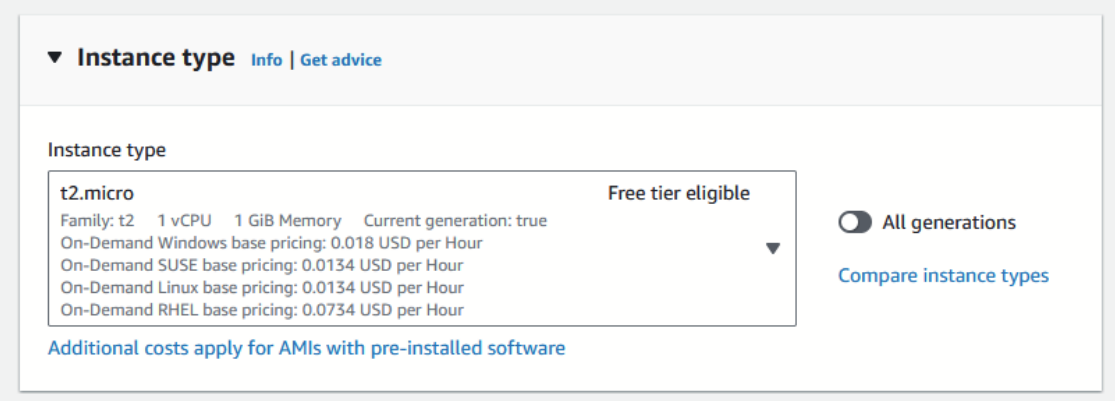

Рисунок 2.5 – Настроювання інстанса

Тип інстанса: t2.micro.

Зверніть увагу: напис Free Tier Eligible означає, що цей тип віртуальної машини можна використовувати безплатно в межах чинних обмежень (сумарно не більше 750 годин роботи машин на місяць – якщо, наприклад, працює дві машини то вони вичерпують години вдвічі швидше).

Інстанси поділяють на типи за декількома критеріями:

– ресурси: кількість ядер процесора та їхня частота, об’єм ОЗП.

– призначення: обчислення (compute optimized), використання ОЗП (memory optimized), використання графічних процесорів (GPU optimized).

– архітектура процесора: x86, arm та виробник: Intel, Amd, Amazon, Apple.

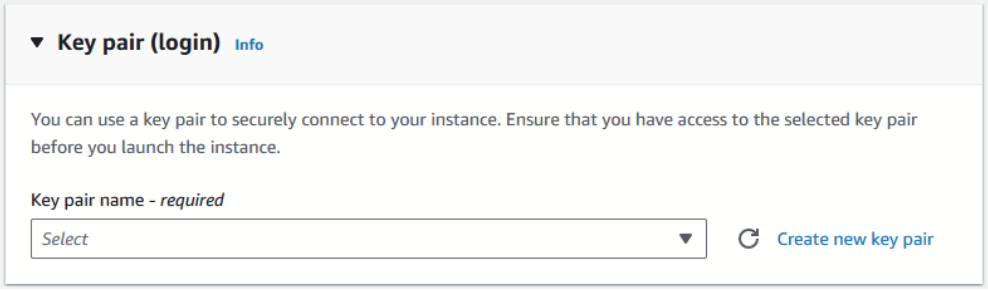

Рисунок 2.6 – Назва пари ключів

Для доступу до інстанса необхідно створити пару ключів або вибрати наявну. У даному випадку потрібно створити нову. Натисніть Create new key pair.

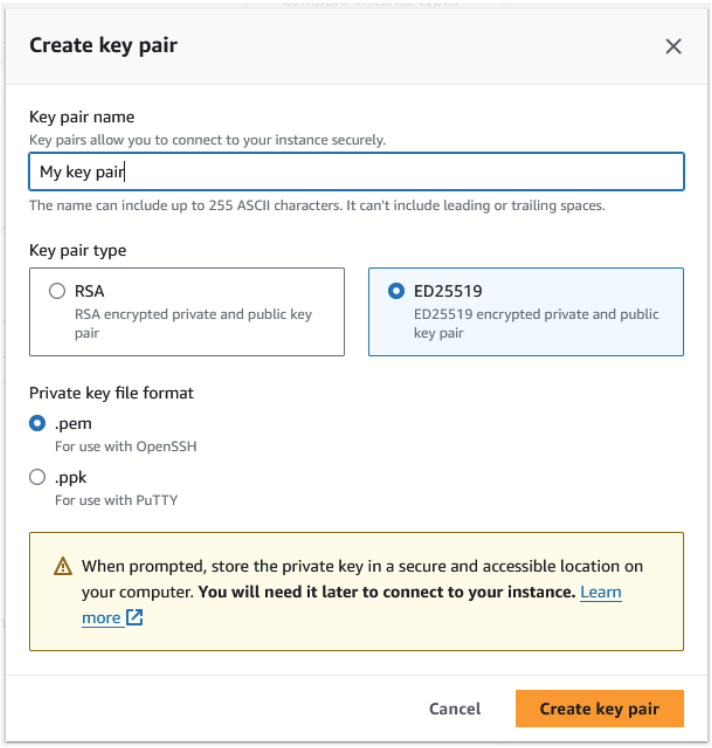

Рисунок 2.7 – Назва пари ключів

У наступному діалоговому вікні потрібно вказати параметри нового ключа: ім’я, тип та формат експорту.

Задайте ім’я, тип укажіть як ED25519, а формат як .pem.

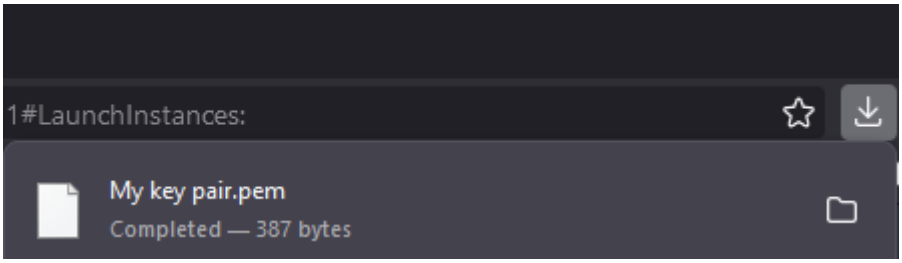

Рисунок 2.8 – Збережена пара ключів

Одразу після натискання кнопки Create key pair AWS «віддасть» новий ключ, і він буде завантажений на ваш локальний комп’ютер. Не видаляйте цей файл.

Наступний етап створення це мережні налаштування. Тут достатньо буде проставити галочки на опціях за замовчуванням:

Allow SSH Traffic from Anywhere

Allow HTTPS Traffic from the Internet

Allow HTTPS Traffic from the Internet

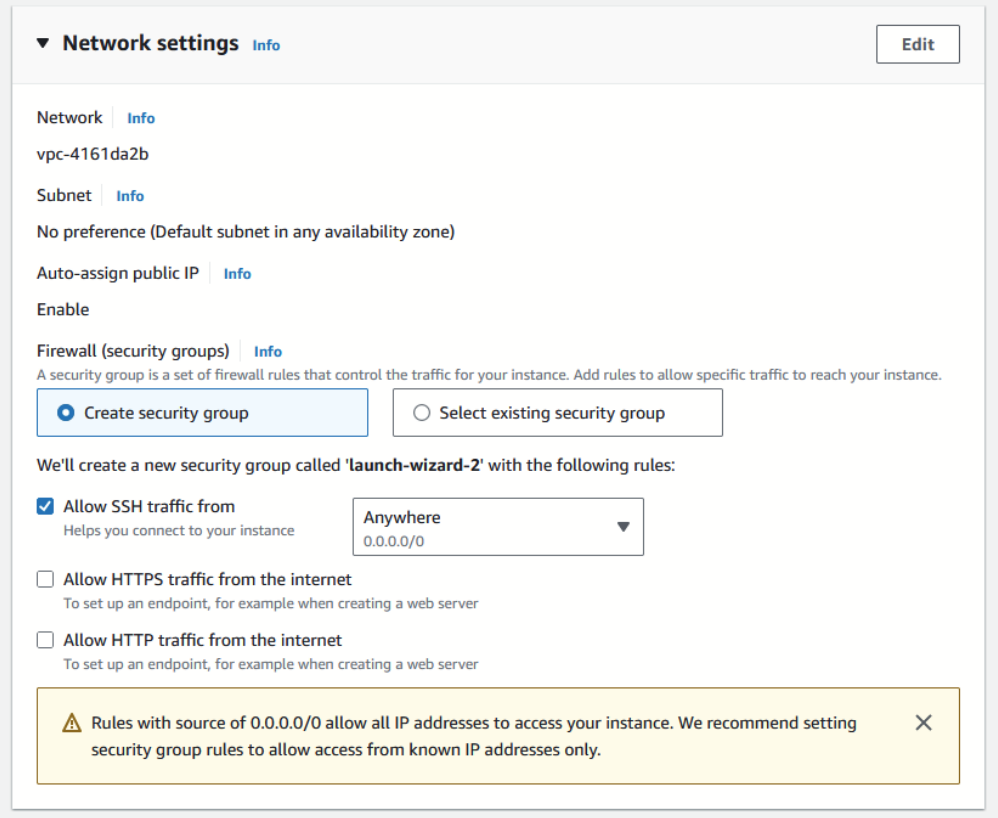

Рисунок 2.9 – Налаштування мережі

Потім необхідно налаштувати storage (диск) для інстанса. Безкоштовна опція AWS дозволяє створити диск до 30 Гб. Установіть 16 Гб і залиште тип gp2, як він встановлений за замовчуванням.

В останньому розділі, Summary можна вибрати кількість інстансів, які будуть запущені з цими настройками і перевірити, які саме настройки будуть застосовані.

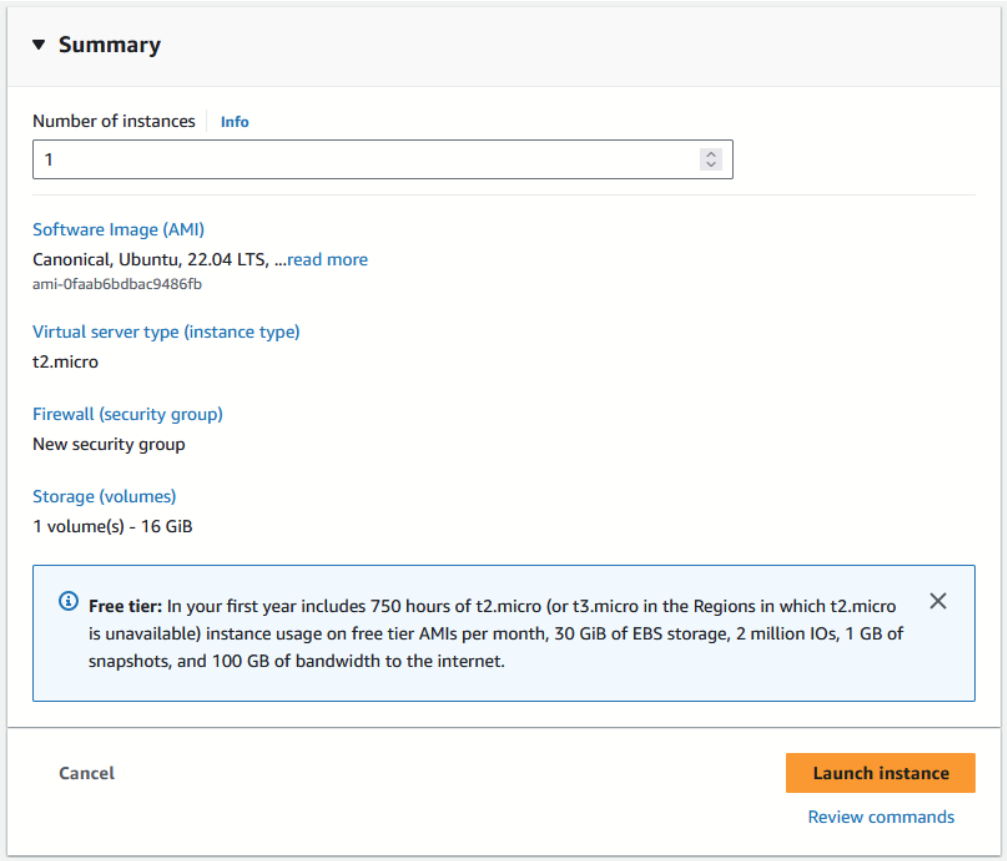

Рисунок 2.10 – Розділ Summary

Натисніть Launch instance. Ви побачите екран із прогресом створення необхідного ресурсу.

Натисніть View all instances, щоб перейти у список ваших інстансів.

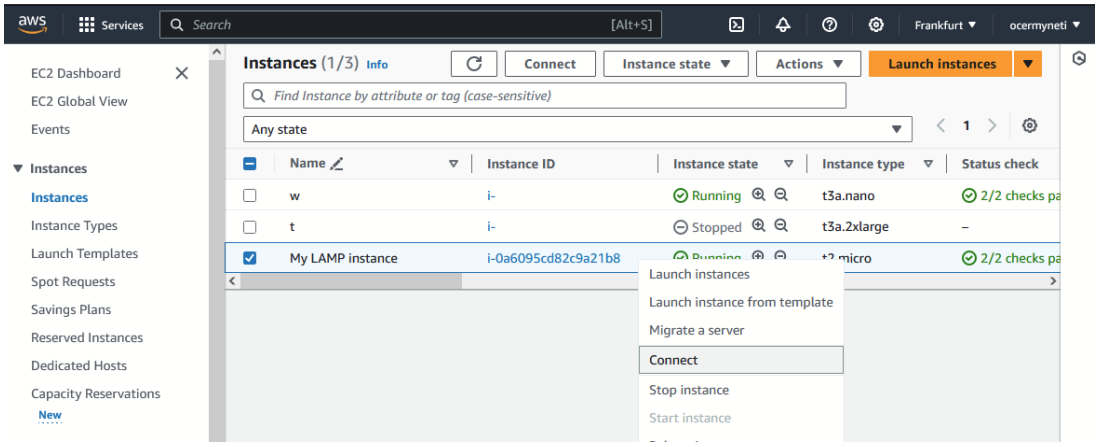

Рисунок 2.11 – Перегляд списку instances

Зробіть правий клік по щойно створеному інстансу і натисніть Connect у контекстному меню.

Виберіть «Connect using EC2 Instance Connect» і натисніть Connect. Оскільки це стандартне встановлення Ubuntu, то ім’я користувача буде теж стандартним.

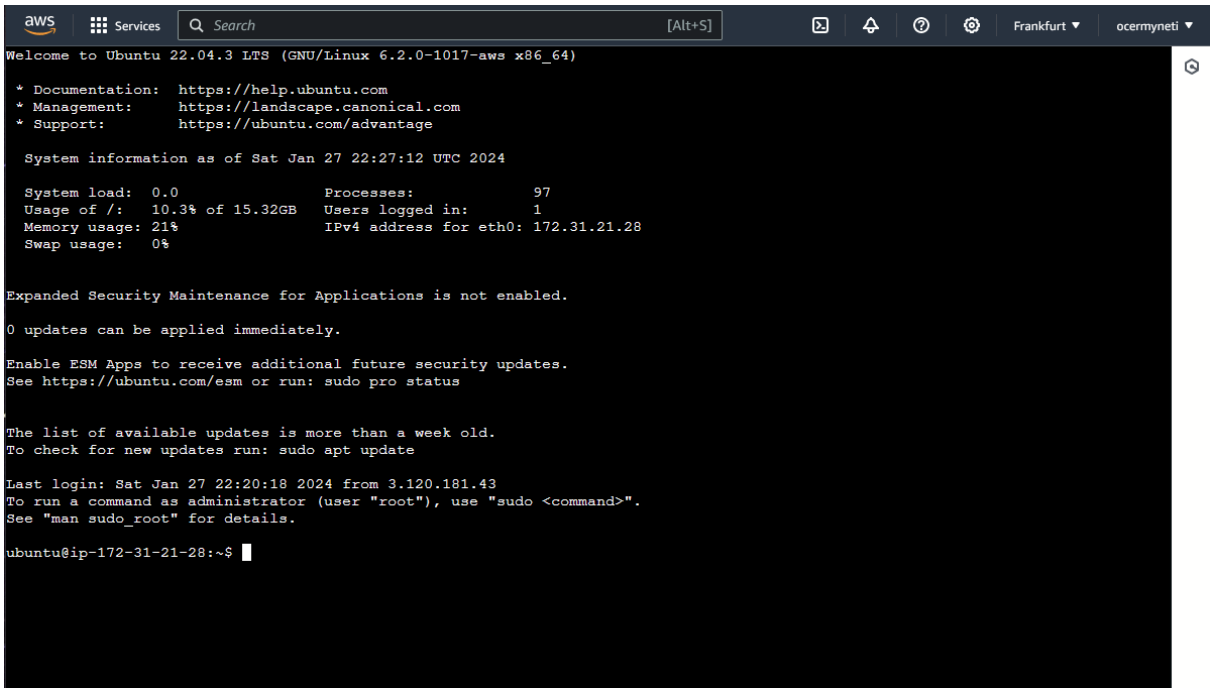

Рисунок 2.12 – Перегляд терміналу Ubuntu

Ви маєте опинитись у командному рядку інстанса, який щойно створили.

Частина 2. Установлення LAMP-стеку (Linux, Apache, MySQL, PHP).

У цьому розділі потрібно навчитися встановлювати веб-сервер Apache, середовище управління базами даних MySQL та інтерпретатор мови PHP.

У всіх наведених тут командах різниця між великою та малою літерою важлива. Їх не можна заміняти між собою. Команда з ключем -R і команда з ключем -r виконуються по-різному.

Спочатку потрібно оновити кеш пакетів. У командному рядку введіть команду (зеленим кольором буде позначено команди, які потрібно вводити у командний рядок):

sudo apt update

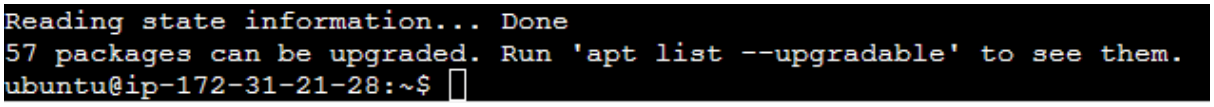

Рисунок 2.13 – Виконання команди sudo apt update

Якщо в результаті виконання цієї команди ви отримаєте повідомлення про те, що «x packages can be upgraded», це означає що є оновлення для встановлених компонентів системи (пакетів). Щоб ці оновлення завантажити і встановити, виконайте наступну команду.

sudo apt upgrade --yes

Далі необхідно встановити веб-сервер Apache2. Виконайте команду:

sudo apt install apache2 --yes

Після встановлення Apache необхідно пересвідчитись, що все запрацювало. Публічну IP-адресу вашого інстанса можна побачити внизу під консоллю в браузері.

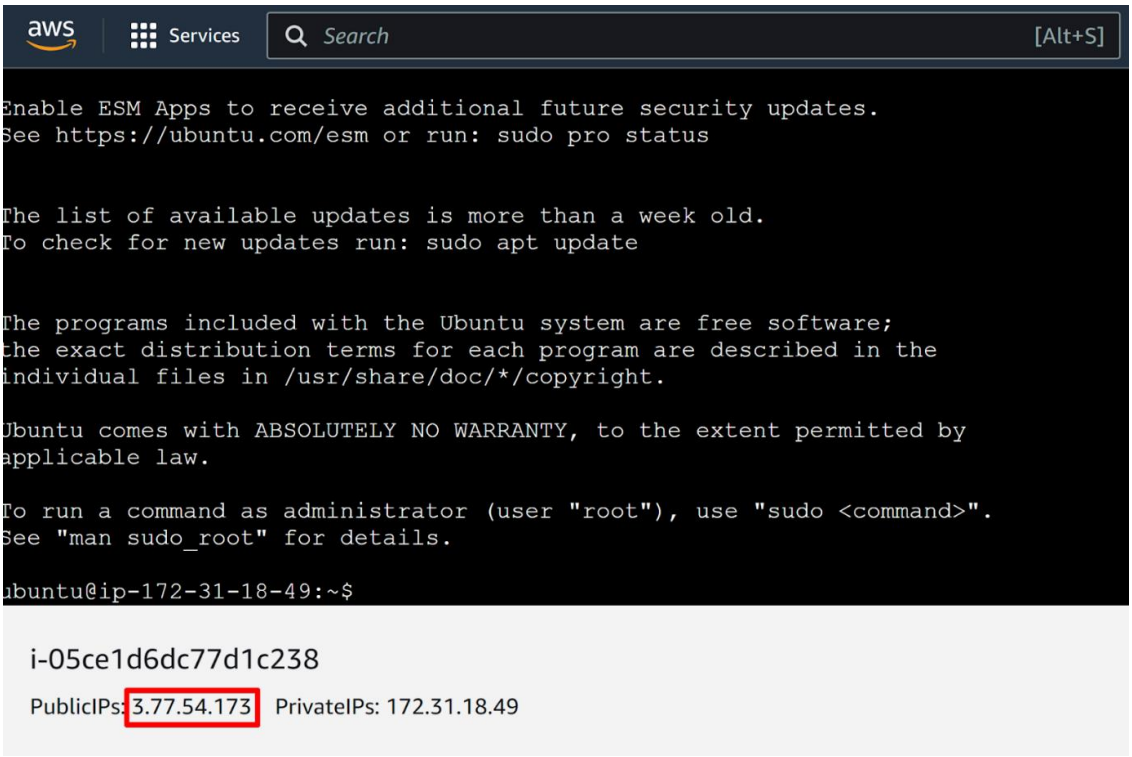

Рисунок 2.14 – Виконання команди sudo apt update

Знайшовши цю адресу, скопіюйте її в адресний рядок у браузері.

Якщо все виконано правильно, то у браузері ви побачите початкову сторінку веб-сервера Apache.

Рисунок 2.15 – Початкова сторінка веб-сервера Apache

Це наш компонент Front-End, який представлено веб-сервером apache на нашому сервері.

Тепер перейдемо до Database layer компоненту.

Для цього нам потрібно встановити MySQL сервер. Для цього виконайте команду:

sudo apt install mysql-server --yes

Після того, як процедура встановлення завершиться, виконайте наступну команду для налаштування сервера MySQL

sudo mysql_secure_installation

Ця команда виконує скріпт початкового налаштування MySQL сервера з огляду безпеки.

Для цілей поточного завдання введіть відповіді на запити скріпта (три крапки позначають пропущений текст):

...

Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: *n*

...

Remove anonymous users? (Press y|Y for Yes, any other key for No) : *y*

...

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : *y*

...

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : *y*

...

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : *y*

Success.

All done!

Тепер потрібно перевірити виконану роботу:

sudo mysql

Ви маєте побачити привітання і командний рядок.

```
ubuntu@ip-172-31-21-28:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.35-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Рисунок 2.16 – Привітання у командному рядку

У командному рядку введіть команду exit щоб вийти з mysql.

Front-end та Database компоненти встановлені. Як Back-End буде виступатиме PHP і движок Wordpress.

Спочатку потрібно встановити PHP. Виконайте команду:

sudo apt install php libapache2-mod-php php-mysql --yes

Щоб перевірити що встановлення php відбулося вдало, виконайте перевірку:

php -v

```
ubuntu@ip-172-31-21-28:~$ php -v
PHP 8.1.2-1ubuntu2.14 (cli) (built: Aug 18 2023 11:41:11) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2-1ubuntu2.14, Copyright (c), by Zend Technologies
```

Рисунок 2.17 – Результат виконання команди php -v

Частина 3. Установлення WordPress.

Увійдіть у mysql знову:

sudo mysql

Створіть нову базу даних, з якою буде працювати WordPress (блакитним позначено команди, які вводяться у командний рядок MySQL):

CREATE DATABASE wordpress

Створіть користувача для цієї бази даних. У цій інструкції буде використано ім’я користувача wordpressuser. Замість слова password у лапках, укажіть пароль, який ви створите. Бажано згенерувати його за допомогою генератора паролів:

CREATE USER 'wordpressuser'@'%' IDENTIFIED WITH mysql_native_password BY 'password';

Видайте новому користувачу права доступу до створеної бази даних:

GRANT ALL ON wordpress. TO 'wordpressuser'@'%';*

Для того, щоб застосувати ці нові налаштування, необхідно виконати оновлення прав доступу в mysql. Для цього виконайте наступну команду:

FLUSH PRIVILEGES;

Після цього можна вийти з mysql:

exit

Настав час завантажити та встановити WordPress.

Перейдіть у директорію /tmp

cd /tmp

Завантажте архів з останньою версією WordPress з офіційного сайту:

curl -O https://wordpress.org/latest.tar.gz

Розпакуйте цей архів:

tar xzvf latest.tar.gz

Зробивши це, потрібно скопіювати зразок конфіг-файлу (файлу налаштувань) WordPress як робочий конфіг-файл:

cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php

Після виконання всієї цієї підготовки можна перенести ці файли в безпосередньо директорію, з якої читатиме файли Apache:

sudo cp -a /tmp/wordpress/. /var/www/wordpress

Коли файли були перенесені, потрібно виставити правильний набір прав на них. Потрібно зробити власником усіх файлів користувача “www-data”:

sudo chown -R www-data:www-data /var/www/wordpress

Також потрібно виставити дозволи на файли та директорії. Для цього виконайте команди:

sudo find /var/www/wordpress/ -type d -exec chmod 750 {} \;

sudo find /var/www/wordpress/ -type f -exec chmod 640 {} \;

Тепер час перейти до редагування файлу налаштувань WordPress який називається wp-config.php:

sudo nano /var/www/wordpress/wp-config.php

У цьому файлі потрібно вказати назву бази даних для WordPress (у цьому прикладі wordpress), користувача (wordpressuser) та пароль (ви його вказували під час налаштування користувача бази даних).

```
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'wordpressuser' );

/** Database password */
define( 'DB_PASSWORD', '$UEFw                                    ' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

Рисунок 2.18 – Задання ключових параметрів WordPress

Останнє налаштування, яке потрібно зробити перш ніж запрацює WordPress, це змінити налаштування DirectoryIndex у Apache. Для цього відредагуйте файл:

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

Після DirectoryIndex і перед index.html додайте запис index.php. Якщо він уже є в цьому рядку на іншому місці, видаліть його звідти, щоб такий запис залишився один і він був на самому початку в списку.

```
  GNU nano 6.2                          /etc/apache2/mods-enabled/dir.conf *
<IfModule mod_dir.c>
        DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

Рисунок 2.19 – Додавання запису index.php

Тепер уведіть IP-адресу вашого сервера в адресний рядок браузера, і ви повинні побачити сторінку налаштування WordPress.

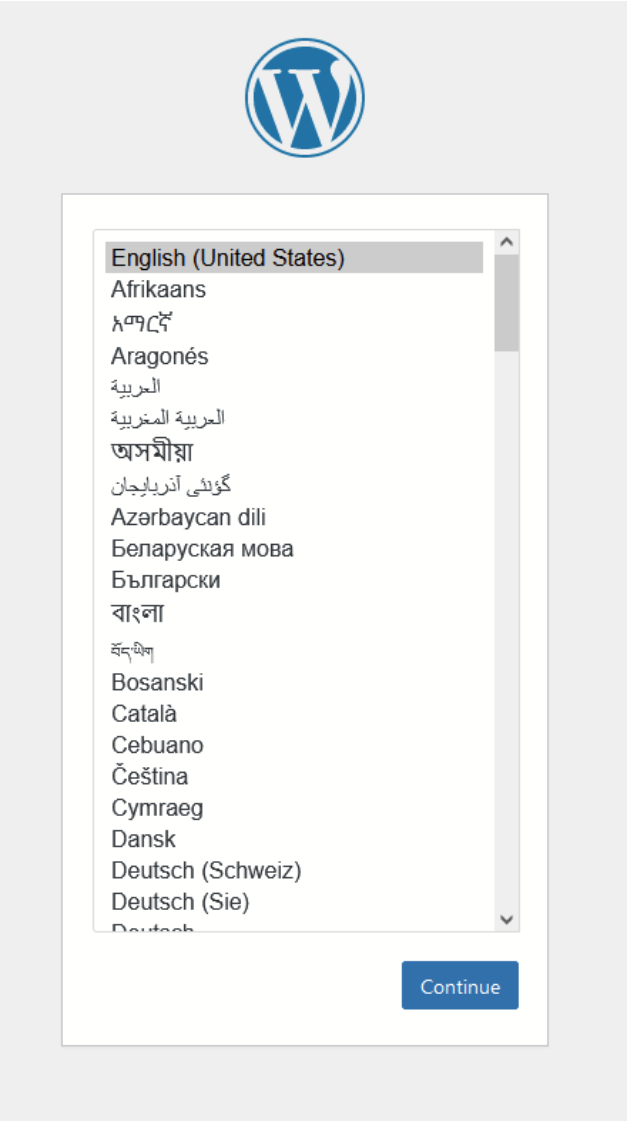

Рисунок 2.20 – Сторінка налаштувань WordPress

Завдання для самостійного виконання

Установити усі три рівні роботи веб-сервера:

1. Front-End у вигляді Apache.

2. Back-End у вигляді PHP та движка Wordpress.

3. Database у вигляді MySQL.

Зміст звіту

1. Номер, тема та мета лабораторної роботи.

2. Відповіді на контрольні питання.

3. Порядок і результати виконання самостійного завдання (скріншоти та текстовий опис кроків виконання завдань).

4. Висновки до роботи.

Контрольні питання

1. Які кроки потрібно виконати, щоб зареєструвати новий AWS акаунт?

2. Яку інформацію потрібно надати для реєстрації AWS акаунту?

3. Як підтвердити свою адресу електронної пошти під час реєстрації AWS акаунту?

4. Які типи акаунтів AWS доступні?

5. Як додати банківську картку до AWS акаунту?

6. Як підтвердити свою ідентичність під час реєстрації AWS акаунту?

7. Які безкоштовні тарифні плани доступні в AWS?

Література: [2, 7, 8].

Лабораторна робота № 3

Тема. Огляд контейнерів Docker. Установлення та налаштування Docker

Мета роботи: ознайомитися з програмою Docker і навчитися створювати Dockerfile.

Короткі теоретичні відомості

Docker – інструментарій для керування ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів.

Початковий код Docker написаний мовою Goi поширюється під ліцензією Apache 2.0. Інструментарій ґрунтується на застосуванні вбудованих в ядро Linux штатних механізмів ізоляції на підставі просторів імен (namespaces) і груп керування (cgroups). Для створення контейнерів використовуються скрипти lxc. Для формування контейнера досить завантажити базовий образ оточення (команда docker pull base), після чого можна запускати в ізольованих оточеннях довільні програми (наприклад, для запуску bash можна виконати docker run -i -t base/bin/bash)

Docker складається з двох процесів:

− демона Docker, який запускається на гостьовій машині (якщо це Лінукс), або всередині VirtualBox середовища boot2docker (якщо це Windows або OS X);

− клієнта, через який можна взаємодіяти з демоном;

− образ Docker (Docker image) – містить операційну систему, застосунок і всі його залежності. Образи в Docker складаються з шарів. Якщо потрібний образ з веб-сервером, то ми беремо образ з дистрибутивом операційної системи, додаємо залежність – веб-сервер, і записуємо це як новий образ, який матиме два шари – один з ОС, наступний з веб-сервером. Образами можна обмінюватись через DockerHub;

Контейнер Docker – це запущений образ. Контейнери Docker можна запускати, спиняти, переміщувати і видаляти. Також можна зробити docker commit контейнера, що створить образ з поточного стану контейнера.

Порядок виконання роботи

Установлення Docker. Тепер дізнаємось, як встановити програмне забезпечення, яке дозволяє розміщувати контейнери Docker.

Для цього рецепту знадобиться сервер Ubuntu 14.04.

Розробники Docker доклали чимало зусиль, щоб зробити встановлення Docker максимально простим, і цей рецепт допоможе запустити роботу за лічені хвилини.

1. Спочатку переконаємося, що встановлено інструмент wget, виконавши таку команду:

$ sudo apt-get install wget

2. Після встановлення wget виконаємо таку команду, щоб запустити програму встановлення Docker:

$ wget -qO- https://get.docker.com/ | sh

3. Інсталятор запропонує ввести пароль sudo, після введення якого буде встановлено Docker і всі його залежності. Після завершення інсталяції можемо переконатися, що Docker правильно встановлено, виконавши таку команду:

$ docker -v

Маємо отримати результат, подібний до такого знімка екрана.

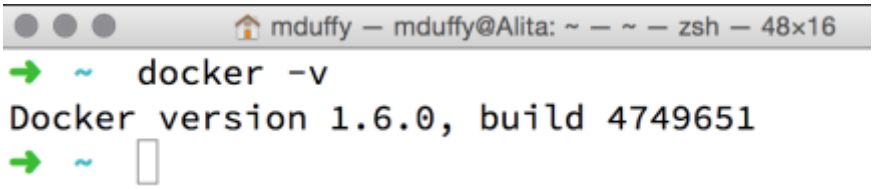

Рисунок 3.1 – Результат установлення

Отримання зображення з публічного реєстру Docker. Тепер, коли встановлено Docker, можемо використовувати його для запуску контейнера з публічного реєстру Docker. Загальнодоступний реєстр Docker містить тисячі готових до використання образів, які охоплюють сотні різних пакетів, від баз даних до серверів програм. Загальнодоступний реєстр також містить офіційні зображення від певних постачальників програмного забезпечення, що пропонує швидкий спосіб почати розробку з цими пакетами, а також гарантію, що зображення є правильним і безпечним.

У цьому рецепті використаємо комбінацію двох різних зображень, щоб запустити базовий блог WordPress.

У цьому рецепті використовуватимуться деякі прості команди Docker і використовуватимуться загальнодоступні зображення Docker для MySQL і WordPress для встановлення блогу:

1. Перше завдання, яке потрібно виконати, – це створити контейнер MySQL для зберігання наших даних. Можемо зробити це за допомогою такої команди:

$ sudo docker run --name test-mysql -e MYSQL_ROOT_ PASSWORD=password -d mysql:latest

2. Ця команда підключиться до загальнодоступного реєстру Docker і перетягне зображення контейнера з тегом mysql:latest на сервер. Після завантаження новий контейнер Docker з назвою test-mysql буде запущено з паролем root MySQL password. Можемо підтвердити, що він працює, виконавши таку команду:

$ docker ps

Рисунок 3.2 – Проміжний результат отримання зображення з публічного реєстру Docker

3. Тепер, коли є контейнер MySQL, можемо звернути увагу на WordPress. Як і у випадку з MySQL, розробники WordPress створили офіційний образ контейнера, і можемо використовувати його для запуску WordPress за допомогою такої команди:

$ docker run --name test-wordpress -p 80:80 --link testmysql:mysql -d wordpress.

4. Ця команда отримає та запустить офіційне зображення WordPress і назве його test-Wordpress. Зверніть увагу на параметр --link ; це пов’язує контейнер MySQL з контейнером WordPress без створення явної мережі між ними.

Зауважимо, що можна пов’язати лише два контейнери на одному хості; якщо контейнери знаходяться на різних хостах, потрібно буде зіставити порти, щоб вони могли спілкуватися.

5. Зверніть увагу на параметр -p , це експортує TCP-порт 80 з контейнера в порт 80 на хості, роблячи інсталяцію WordPress доступною. Він не буде доступний для зовнішнього світу без зіставлення портів, навіть якщо контейнер має відкритий порт 80. Відображення по суті створює правило брандмауера на хості Docker, яке з’єднує мережу хоста та віртуальну мережу, створену для роботи контейнерів Docker.

6. Відкриємо браузер, наведемо в ньому адресу хосту Docker і побачимо таку сторінку.

Рисунок 3.3 – Результат отримання зображення з публічного реєстру Docker

Виконання головних операцій Docker. Тепер, коли маємо можливість створювати контейнери Docker, потрібно знати, як ними керувати. Docker має повний набір інструментів, які дозволяють запускати, зупиняти та видаляти контейнери.

Цей рецепт демонструє головні команди, які використовуються для керування контейнерами Docker. Використовуючи ці команди, можемо керувати повним життєвим циклом контейнера.

1. Використаємо таку команду, щоб отримати список запущених контейнерів у вашій системі:

$ sudo docker ps

Показує лише запущені контейнери. Щоб побачити контейнери, які були зупинені, скористайтеся такою командою:

$ sudo docker ps -a

2. Щоб зупинити запущений контейнер, скористаємося такою командою:

$ docker stop

Тут ідентифікатор отримується після запуску докера ps і вибору ідентифікатора зображення, яке потрібно зупинити.

3. Щоб видалити контейнер докерів, скористаємося такою командою:

$ sudo docker rm

Пам’ятайте, що це видаляє лише КОНТЕЙНЕР , а не базове зображення.

4. Скористаємося такою командою, щоб отримати список зображень Docker, завантажених на хост:

$ sudo docker images

Це повинно створити вихід, який має приблизно такий вигляд:

Рисунок 3.4 – Результат виконання головних операцій Docker

5. Використаємо таку команду, щоб видалити зображення:

$ sudo docker rmi < IMAGE ID >

Видалення зображень є безпечною операцією; Docker використовує підрахунок посилань, щоб відстежувати контейнери, які мають залежності зображення. Якщо спробуємо видалити зображення, яке використовується контейнером (запущеним або зупиненим) на вашому хості, то отримаємо попередження, і воно не буде видалено.

Запуск контейнера в інтерактивному режимі. Іноді дуже корисно мати можливість інтерактивно запускати контейнер для діагностики проблем.

Запуск контейнера в інтерактивному режимі по суті дає оболонку для контейнера, і всередині контейнера можемо працювати так само, як і з будь-якою іншою системою Linux.

Можемо запустити будь-який контейнер Docker в інтерактивному режимі за допомогою команди:

$ sudo docker run -i -t ubuntu /bin/bash

Створення файлу Docker. Хоча в реєстрі Docker доступно багато готових зображень, ви неминуче захочете створити власні зображення як підґрунтя для своїх контейнерів. Однією з видатних особливостей Docker є його прості інструменти створення. Можемо легко створювати нові зображення за допомогою простого текстового файлу.

У цьому рецепті дізнаємося, як за допомогою Docker запакувати програмне забезпечення Gollum Wiki та відправити його в загальнодоступне сховище Docker.

Наступні кроки описують, як створити новий Dockerfile, зібрати його та, нарешті, відправити в реєстр Docker.

1. По-перше, необхідно створити новий обліковий запис реєстру Docker. Почнемо з відвідування https://registry.hub.docker.com з дотриманням інструкцій, щоб створити новий обліковий запис. Створюючи обліковий запис, створюємо простір імен у реєстрі Docker, який дозволяє завантажувати власні зображення.

2. Створимо обліковий запис реєстру Docker, тож тепер можемо звернути увагу на створення нашого першого Dockerfile. Файл Docker – це список кроків, які використовуються для створення повного образу Docker. Ці кроки можуть передбачати копіювання файлів в образ або виконання команд, але кожен Dockerfile повинен починатися з команди FROM. Команда FROM дозволяє вибрати образ Docker, який буде основою для цього контейнера; загалом, це буде образ ОС, і багато дистрибутивів Linux тепер постачають офіційний образ, який можна використовувати.

Може здатися дещо рекурсивним, що для створення зображення потрібно використовувати зображення. Також можна створити власний образ ОС, який буде основою, слідкуйте за інструкціям, наведеним на: https://docs.docker.com/articles/baseimages/.

3. Використаємо образ Ubuntu для контейнера Gollum. Для цього створимо новий файл під назвою Dockerfile і вставимо такий код:

FROM ubuntu:14.04

Будемо використовувати Ubuntu 14:04 як базовий образ.

4. Далі можемо вставити невеликі метадані, які дозволять людям побачити, хто зараз підтримує зображення. Це дозволяє людям бачити, хто є автором зображення, і до кого звертатися, якщо у них виникнуть запитання чи проблеми. Це має форму текстового поля, у якому зазвичай прийнято вказувати

своє ім’я та адресу електронної пошти. Можемо зробити це, додавши таку команду у файл Docker:

MAINTAINER Example User example@adomain.com

5. Тепер, подбаємо про метадані для контейнера, можемо звернути увагу на встановлення програмного забезпечення. Оскільки використовуємо базовий образ Ubuntu, то менеджер пакунків apt для встановлення програмного забезпечення; однак базове зображення може мати застарілий список пакетів, збережений у кеші, тому краще оновити його. Щоб оновити список пакетів, додамо директиву RUN. У свій Dockerfile вставимо такий код:

*RUN apt-get update && *

Директиву RUN будемо бачити часто, оскільки вона дозволяє виконувати команди всередині контейнера. Однак будемо обережні, оскільки потрібно переконатися, що команди, які виконуємо, не є інтерактивними; інтерактивні команди призведуть до збою створення образу, оскільки не зможемо взаємодіяти з ним під час створення.

6. Зверніть увагу на && \ ; це функція оболонки, яка виконує наступну команду, якщо попередня команда була успішною, дозволяє об’єднувати команди в один рядок. Це корисно, щоб зменшити кількість шарів Docker. \ є розривом рядка, що дозволяє зберегти Dockerfile

Коли запускаємо збірку Docker, кожна команда створює новий шар, і кожен шар розміщується поверх наступного, створюючи остаточне зображення Docker. Однак кожен рівень містить невелику кількість внутрішніх метаданих, які, хоча й невеликі, можуть складатися. Можливо, що важливіше, існує обмеження на кількість шарів, які може містити зображення, обмеження базової файлової системи AUFS; на момент написання статті обмеження становить 127 шарів. Хоча можемо використовувати альтернативні файлові системи з Docker, які можуть усунути це обмеження, варто розробляти це з урахуванням.

7. Тепер можемо почати інсталювати необхідне програмне забезпечення. Оскільки Gollum є Ruby-додатком, йому потрібен Ruby, а також деякі додаткові інструменти збірки. Знову збираємося використати команду RUN і матимемо можливість установити ці пакунки для нас. Вставте наступний код у свій Dockerfile:

RUN apt-get update && apt-get install -y ruby1.9.1 ruby1.9.1-dev make zlib1g-dev libicu-dev build-essential git

Це встановить програмне забезпечення, яке потрібно для встановлення Gollum.

8. Тепер установимо саме Gollum. Gollum розповсюджується як дорогоцінний камінь Ruby, тому можемо використовувати менеджер пакетів Gem, щоб встановити його. Для цього додамо такий код:

*RUN apt-get update && *
*apt-get install -y ruby1.9.1 ruby1.9.1-dev make zlib1g-dev libicu-dev build-essential git && *
gem install gollum

Очевидно, виконуємо встановлення як ланцюжковий набір команд, а не використовуємо окрему директиву RUN для кожного нового рядка.

9. Тепер потрібний каталог для зберігання нашого вікі-вмісту. На відміну від багатьох вікі, які покладаються на базу даних для зберігання вмісту, Gollum використовує репозиторій Git як постійне сховище. Усе, що потрібно, це файлова система для зберігання репозиторію Git, і вона піклується про версії. Створимо його зараз; вставимо наступний код у свій Dockerfile:

RUN mkdir -p /usr/local/gollum

10. Тепер налагодимо робочий каталог. За замовчуванням Docker виконує всі директиви в кореневому каталозі контейнера; установивши робочий каталог, можемо виконувати команди в каталозі за нашим вибором. Щоб установити робочий каталог, додамо таку директиву до вашого Dockerfile**:**

WORKDIR /usr/local/gollum

11. З установленим робочим каталогом можемо створити початкове сховище для зберігання вікі-вмісту; це досягається за допомогою команди Git. Додамо цей код до свого Dockerfile:

RUN git init.

Ця команда буде запущена в робочому каталозі, який встановлено в попередній команді, і створить порожнє сховище Git, готове для вмісту.

12. Потрібно відкрити мережний порт. Виставивши порт, зможемо підключитися до сервісу з мережі; це також дозволяє іншим контейнерам підключатися до служби через зв'язування. Gollum за замовчуванням працює на порту TCP 4567 ; додамо наступний код, щоб відкрити його:

EXPOSE 4567

13. Нарешті додаємо команду за замовчуванням, яка запускатиметься під час запуску контейнера. У цьому випадку пакет Gollum встановлює двійковий файл, який можна використовувати для запуску вікі. Додамо таку команду, щоб виконати її під час запуску контейнера:

CMD ["gollum"]

14. Тепер готові створити наш контейнер Docker. У командному рядку перейдемо до каталогу, що містить Dockerfile, і введемо таку команду:

$ sudo docker build -t /gollum:4.0.0

Де <username> — це ім'я користувача реєстру Docker, яке встановили раніше. Зверніть увагу на -t : це тег. Тег використовується як для назви зображення, так і для його версії. У цьому випадку використовували версію програмного забезпечення.

Контроль версій зазвичай – це спірне питання, і якщо сумніваємося, найкраще використовувати наявні стандарти. Зазвичай, створюємо версію контейнера, яка відповідає версії програми, яку пакуємо, оскільки це дозволяє з першого погляду побачити, на якому хості та яка версія цього програмного забезпечення.

15. Щойно запустимо цю команду, побачимо результат, схожий на наступний знімок екрана:

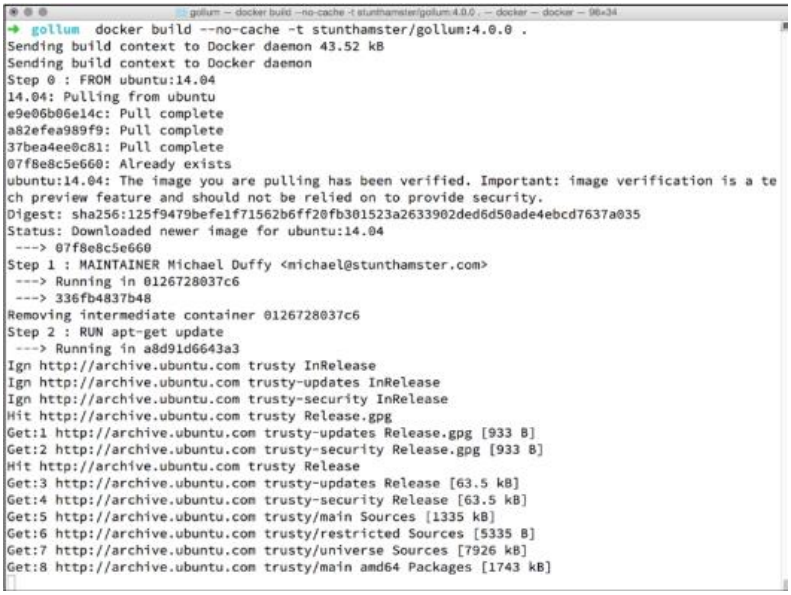

Рисунок 3.5 – Результат роботи створення файлу Docker

16. Після завершення збірки можемо відправити її в репозиторій Docker. Розмістивши образ у сховищі, спростимо його розгортання на інших машинах. Щоб проштовхнути контейнер, виконаємо таку команду:

$ sudo docker push /gollum:4.0.0

Це надішле зображення до репозиторію Docker і підготує його до розповсюдження. Якщо бажаємо зробити його приватним, то можемо зареєструвати преміум-акаунт і скористатися функцією приватного сховища; альтернативно, можемо розмістити свій власний реєстр Docker.

Завдання для самостійного виконання

Необхідно нижче ознайомитися з додатковою інформацією про команди. Також потрібно виконати такі завдання.

1. **Встановіть Docker.**

Переконайтеся, що на вашому сервері Ubuntu 14.04 встановлено інструмент wget.

Виконайте команду:

sudo apt-get install wget.

Завантажте та запустіть програму встановлення Docker:

wget -qO- https://get.docker.com/ | sh.

Введіть пароль sudo, коли буде запропоновано.

Переконайтеся, що Docker правильно встановлено:

docker -v.

2. **Отримайте зображення з публічного реєстру Docker.**

Запустіть контейнер MySQL:

sudo docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=password -d mysql:latest.

Запустіть контейнер WordPress:

sudo docker run --name test-wordpress -p 80:80 --link testmysql:mysql -d wordpress.

Перевірте, чи працює WordPress, відкривши браузер і ввівши адресу хосту Docker.

3. **Виконайте головні операції Docker.**

Перегляньте список запущених контейнерів:

sudo docker ps.

Зупиніть контейнер:

docker stop <container_id>.

Видаліть контейнер:

sudo docker rm <container_id>.

Перегляньте список зображень Docker:

sudo docker images.

Видаліть зображення:

sudo docker rmi <image_id>.

4. **Запустіть контейнер в інтерактивному режимі.**

Запустіть контейнер Ubuntu в інтерактивному режимі:

sudo docker run -i -t ubuntu /bin/bash.

5. **Створіть Dockerfile.**

Створіть Dockerfile для Gollum Wiki.

Додайте в Dockerfile такі кроки:

1) Виберіть базовий образ (Ubuntu 14.04).

2) Встановіть Ruby, Git та інші необхідні інструменти.

3) Створіть каталог для зберігання вікі-вмісту.

4) Ініціалізуйте репозиторій Git.

5) Відкрийте порт TCP 4567.

6) Встановіть Gollum Wiki.

7) Запустіть Gollum Wiki.

6. **Зберіть та відправте Dockerfile.**

− Зберіть Dockerfile:

sudo docker build -t <username>/gollum:4.0.0.

− Відправте зображення в реєстр Docker:

sudo docker push <username>/gollum:4.0.0.

Зміст звіту

1. Номер, тема та мета лабораторної роботи.

2. Відповіді на контрольні запитання.

3. Порядок і результати виконання самостійного завдання (скріншоти та текстовий опис кроків розв’язання завдань).

4. Висновки до роботи.

Контрольні питання

1. Що таке Docker?

2. Docker складається з двох процесів, яких саме?

3. Які кроки потрібно виконати для отримання зображення з публічного реєстру Docker?

4. Які головні дії для запуску контейнера в інтерактивному режимі?

5. Як можна створити файл Docker?

Література: [3, 9, 10].

Лабораторна робота № 4

Тема. Створення коду та налаштування конвеєра складання

Мета роботи: ознайомитися із системою вбудованих конвеєрів доставки, плагіном Building Pipeline та навчитися розгортати файл WAR.

Короткі теоретичні відомості

Jenkins – це інструмент з відкритим кодом для автоматизації завдань безперервної інтеграції та безперервної доставки (CI/CD). Він використовується для автоматизації різних етапів життєвого циклу програмного забезпечення, таких як: компіляція; тестування; складання; розгортання.

Jenkins може бути налаштований для автоматизації практично будь-якого завдання, пов’язаного з розробкою програмного забезпечення.

Jenkins має ряд переваг, таких як:

– відкритий код;

– простий у використанні інтерфейс користувача;

– Jenkins може бути налаштований для автоматизації практично будь-якого завдання;

– велика та активна спільнота користувачів і розробників.

Jenkins використовується завдяки створенню **конвеєрів**. Конвеєр – це серія завдань, які виконуються послідовно. Завдання можуть бути будь-якими, пов’язаними з розробкою програмного забезпечення, наприклад, компіляція, тестування, складання та розгортання.

Jenkins має безліч **плагінів**, які можна використовувати для розширення його функціональності. Плагіни доступні для будь-якого завдання, пов’язаного з розробкою програмного забезпечення.

Jenkins Build Pipeline – це потужний інструмент, який використовується для автоматизації процесів розробки програмного забезпечення. Він дозволяє візуалізувати та оркеструвати ваші CI/CD pipelines, роблячи їх більш ефективними та надійними.

Плагін надає зручний графічний інтерфейс для візуалізації вашого pipeline. Це робить його зрозумілим та простішим для розуміння, що робить pipeline, і, як різні кроки, пов’язані між собою.

Плагін дозволяє вам оркеструвати всі кроки вашого pipeline, у тому числі код, тестування, складання, артефакти та розгортання. Ви можете легко

додавати, видаляти та змінювати кроки pipeline, а також налаштовувати їх поведінку.

Плагін автоматизує всі кроки вашого pipeline, що економить час і зусилля. Вам не потрібно вручну запускати кожен крок, pipeline може запускатися автоматично внаслідок внесенні змін до коду або за розкладом.

Плагін легко інтегрується з іншими інструментами CI/CD, такими як Git, Maven, Docker, Kubernetes та іншими. Це робить його гнучким та розширюваним, щоб відповідати вашим потребам.

Порядок виконання роботи

1. Створення вбудованих конвеєрів доставки.

Jenkins 2 надає спосіб створення конвеєрів доставки за допомогою доменно-специфічної мови (DSL).

Кроки для створення вбудованого pipeline доставки такію

1. Перейдіть на інформаційну панель Jenkins і натисніть **«Новий елемент»**.

2. Уведіть назву елемента та виберіть **pipeline**, як показано на зображенні нижче.

3. Натисніть **OK**.

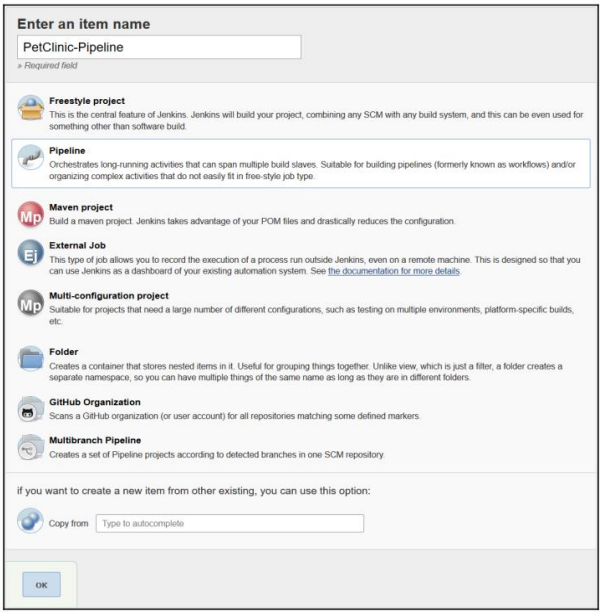

Рисунок 4.1 – Уведення назви елемента та вибір pipeline

4. Якщо у вас є конвеєр, ви можете створити новий конвеєр, скопіювавши з нього.

5. Перейдіть до **додаткових параметрів проєкту**. З метою навчання вхідне exo «Привіт із Pipeline Demo» у вікні «Сценарій».

6. Натисніть **«Зберегти»**, щоб зберегти конфігурацію.

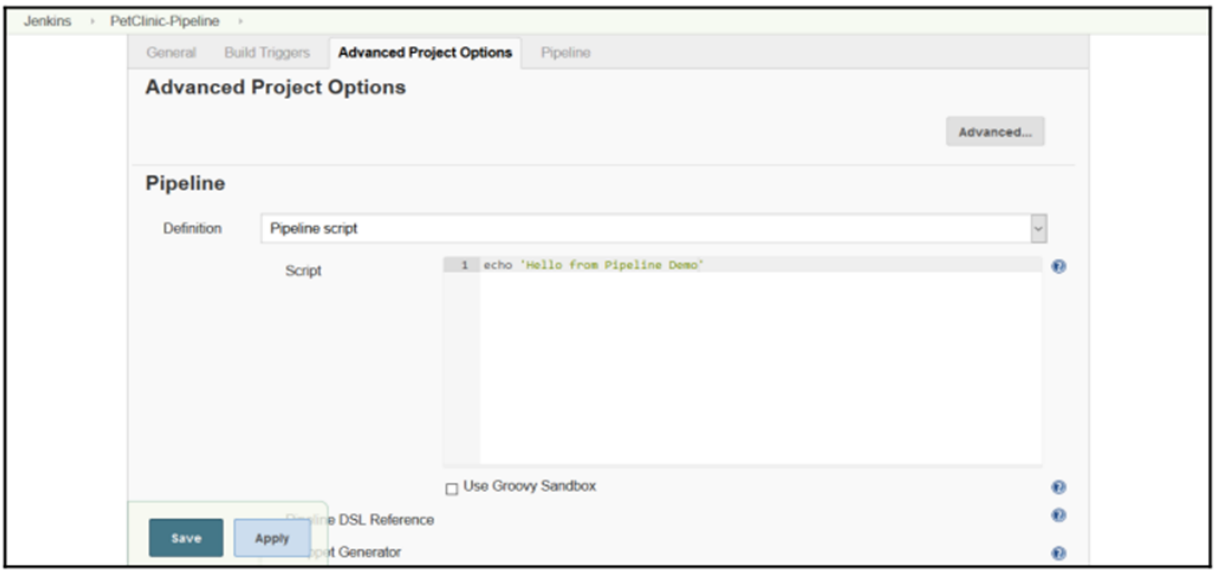

Рисунок 4.2 – Створення конвеєра

7. Оскільки ми не створили жодного етапу, ми отримаємо попередження, як показано на рисунку 4.3. Однак ми можемо виконати конвеєр для демонстраційних цілей.

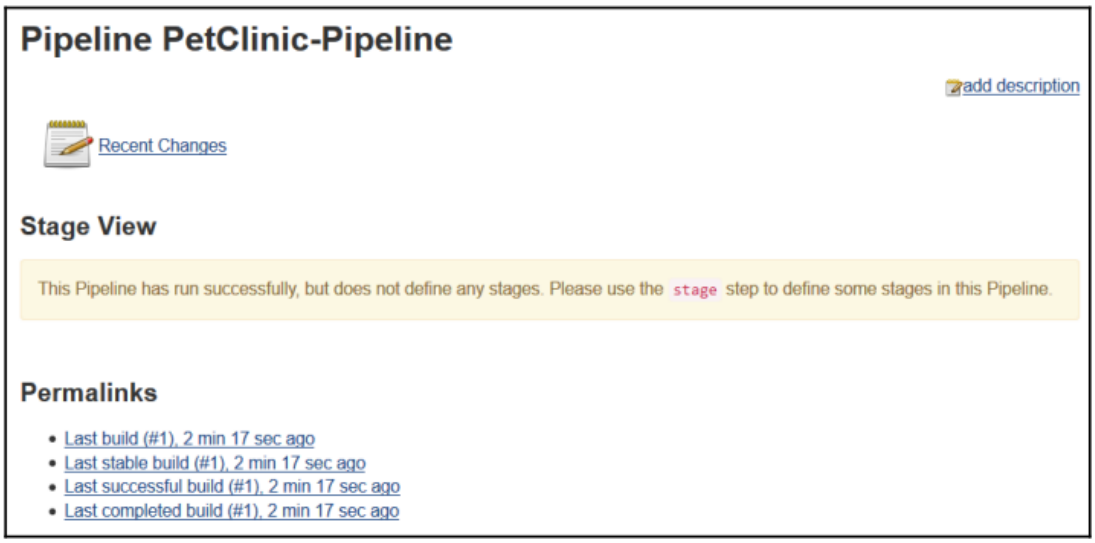

Рисунок 4.3 – Попередження про виконання порожнього Pipeline

8. Натисніть **«Побудувати зараз»**. Перевірте **вихід консолі**. Ми бачимо успішне завершення виконання сценарію.

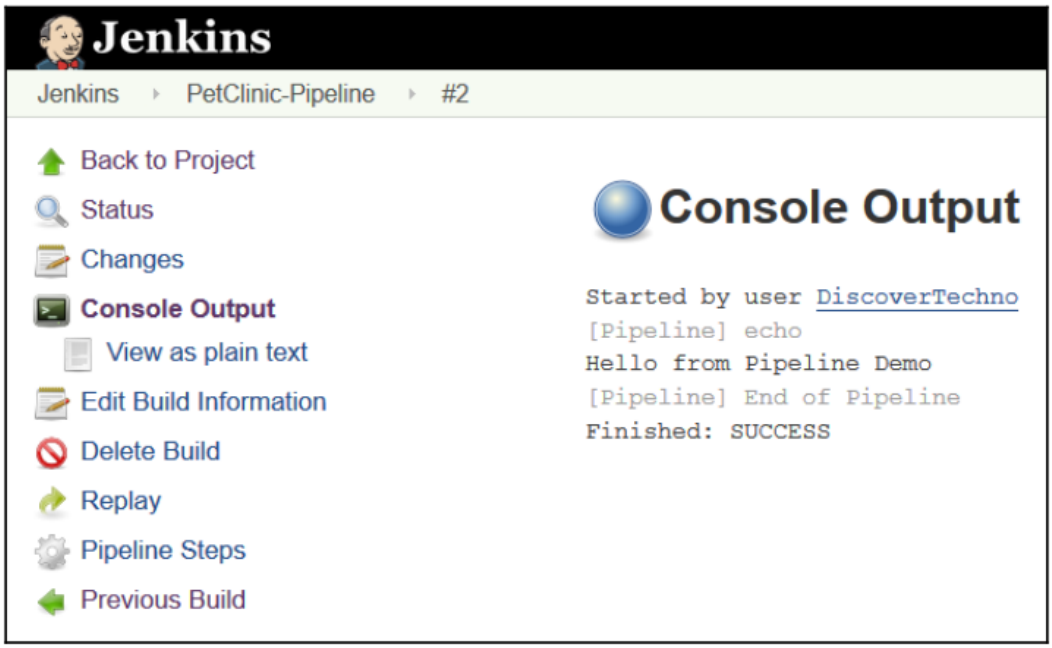

Рисунок 4.4 – Перевірка вмісту консолі

Спробуємо простий сценарій створення конвеєра для компіляції вихідних файлів і виконання модульних тестів.

1. Напишіть наведений нижче сценарій у полі сценаріїв.

echo 'Hello from Pipeline Demo'

stage 'Compile'

build 'PetClinic-Compile'

stage 'Test'

build 'PetClinic-Test'

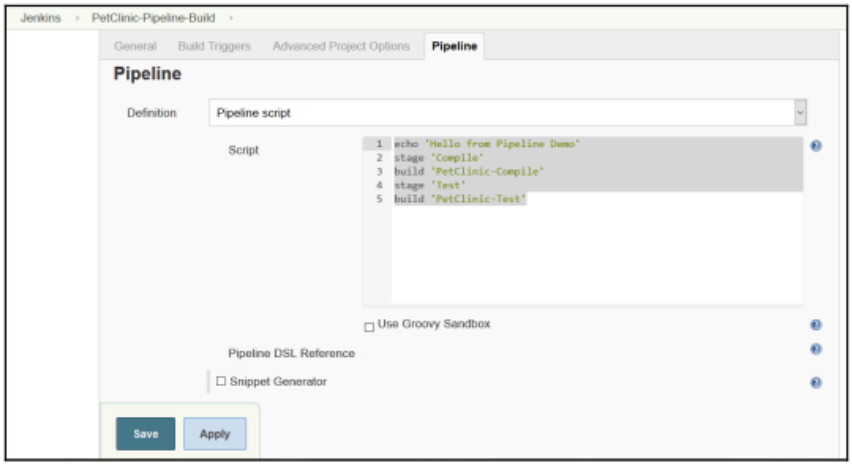

Рисунок 4.5 – Уведення скрипта

2. Натисніть «Побудувати зараз» і перейдіть до виводу консолі, щоб перевірити процес виконання.

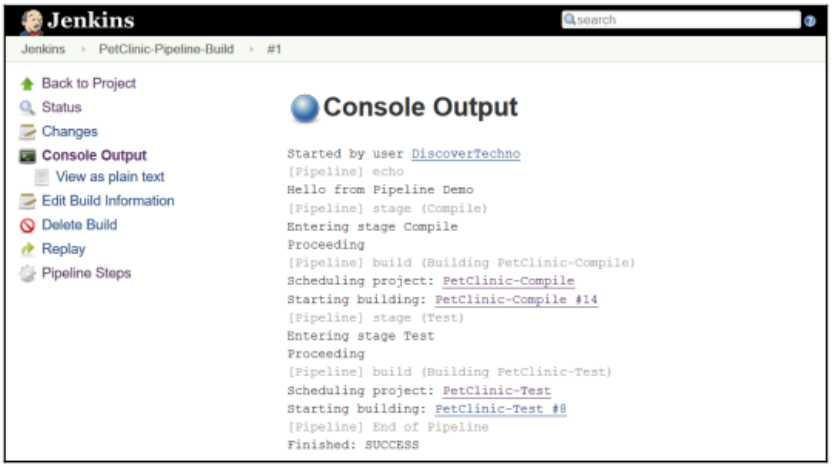

Рисунок 4.6 – Перевірка процесу виконання скрипта у консолі

3. Перейдіть на головну сторінку Build Job. Тут ми можемо побачити вигляд сцени. Пам’ятайте, що ми створили два етапи: один – компіляція, інший – тестування. Перегляд сцени забезпечує миттєву візуалізацію. Він надає такі відомості, як час завершення складання, на якому вузлі було виконано складання, складання було невдалим чи виконане успішно.

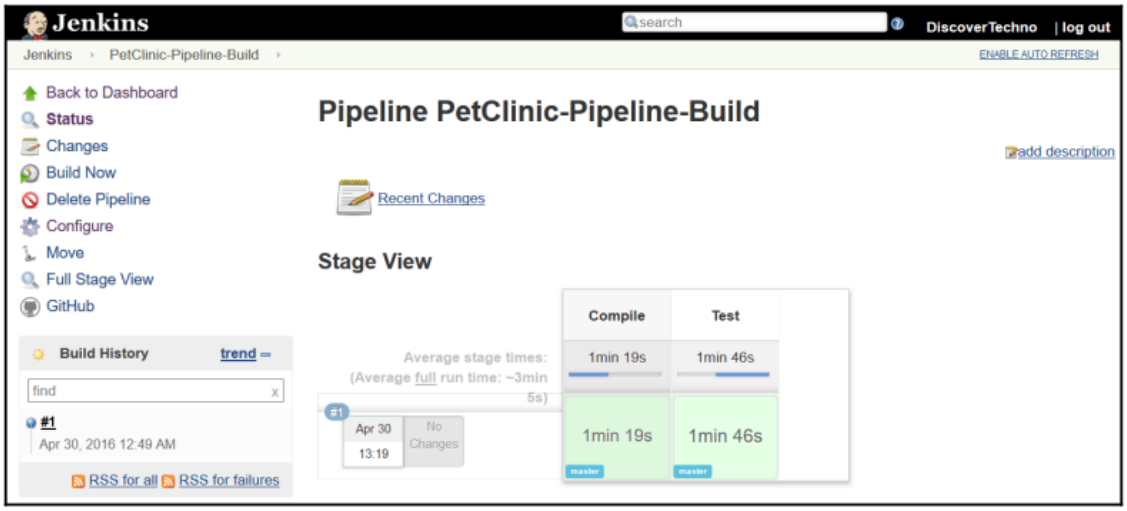

Рисунок 4.7 – Головна сторінка Build Job

4. Виконуючи цей сценарій, перевірити кроки конвеєра.

Рисунок 4.8 – Перевірка кроків конвеєра

5. Натисніть «Повний перегляд», щоб отримати повноекранний режим, як показано на рисунку.

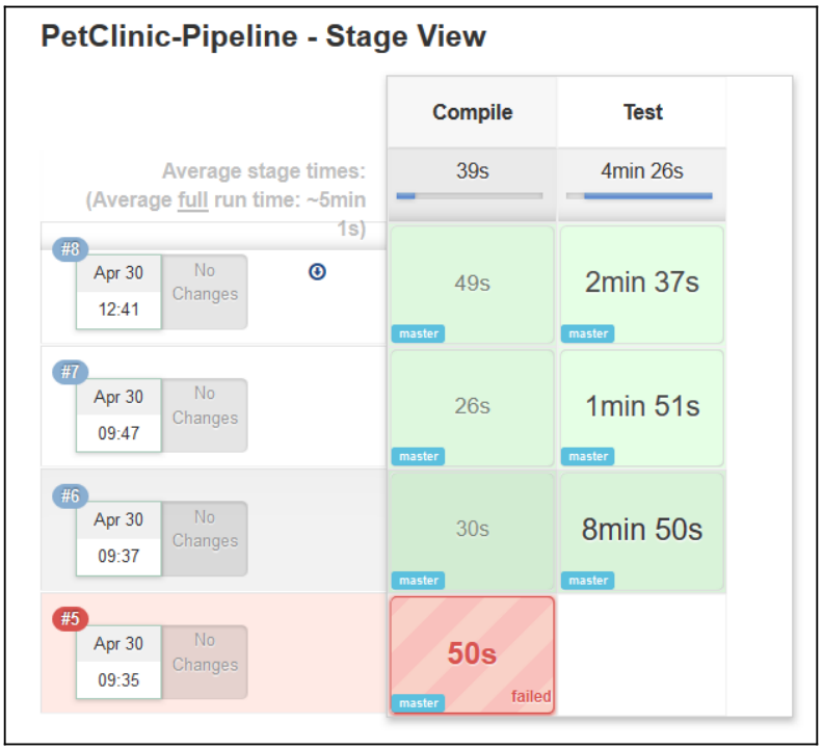

Рисунок 4.9 – Повний перегляд кроків конвеєра

6. Щоб отримати докладні відомості про етап, наведіть курсор миші на певний етап, і він покаже нам статус виконання цього етапу, а також посилання на журнали.

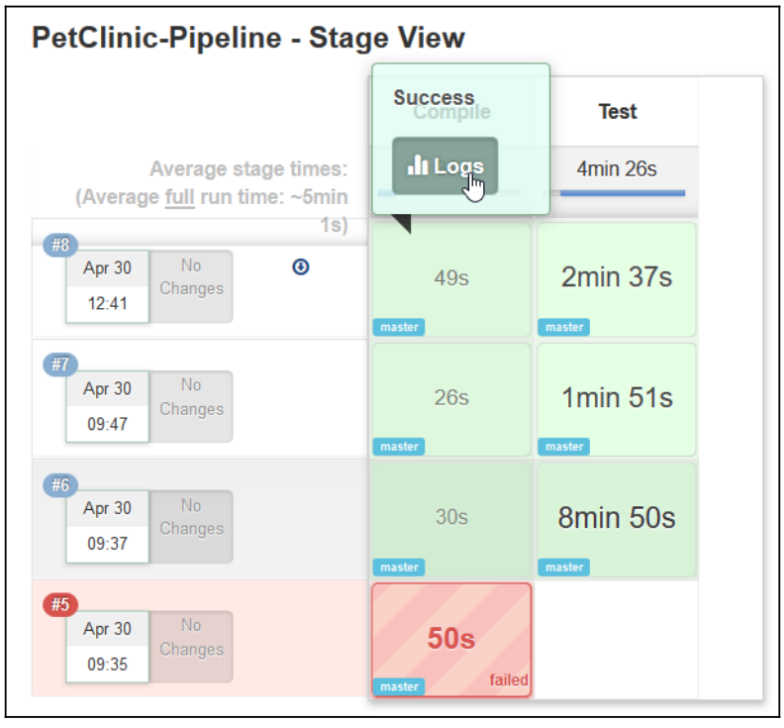

Рисунок 4.10 – Перегляд статусу виконання етапу

7. Натисніть на «Stage Logs», а далі буде надано детальну інформацію про етап. Натисніть спадне меню, щоб отримати докладнішу інформацію про журнали.

Рисунок 4.11 – Отримання журналів виконання

2. Плагін Building Pipeline.

Щоб створити конвеєр складання, установіть плагін Build Pipeline.

На інформаційній панелі Дженкінса клацніть знак плюс, який відкриє сторінку для створення Build Pipeline View. Уведіть назву конвеєра складання та натисніть OK.

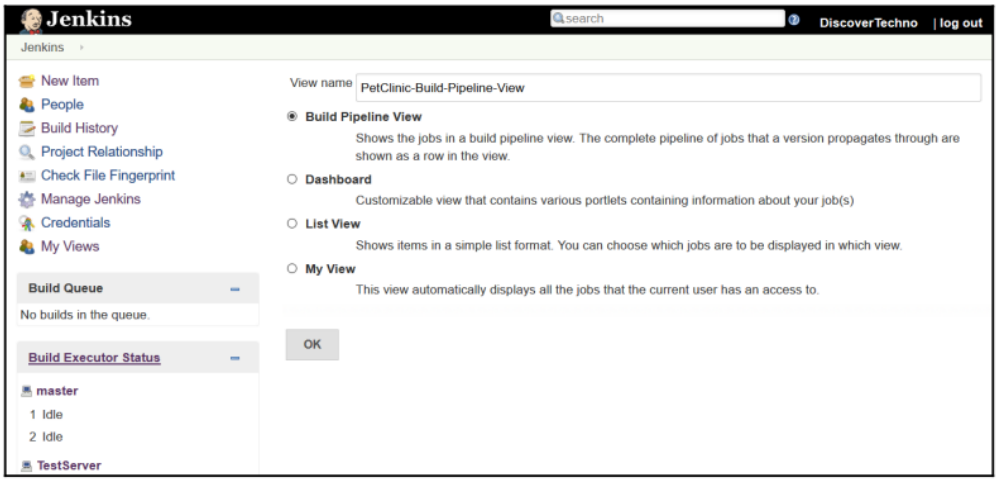

Рисунок 4.12 – Відкриття Build Pipeline View

Ми створили кілька завдань складання для компіляції вихідного коду, перевірки вихідного коду за допомогою Sonar і виконання тестів JUnit.

Ми також визначили порядок, якщо компіляція пройшла успішно, тоді будуть виконані інші два завдання складання. У нашому випадку це PetClinic-Code і PetClinic-Test .

1. Перейдіть на сторінку конфігурації завдання складання PetClinic-Compile.

2. Перейдіть до розділу «Дії після складання» .

3. Введіть ім’я завдань складання в полі Проєкт для створення. Тут можна надати список, розділений комами.

4. Натисніть ЗБЕРЕГТИ, щоб зберегти конфігурацію.

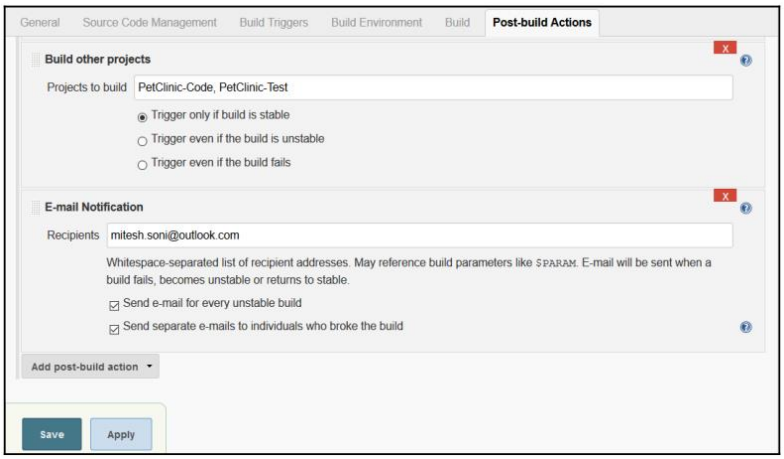

Рисунок 4.13 – Збереження конфігурації

5. Перевірте список проєктів Downstream на головній сторінці Build Job

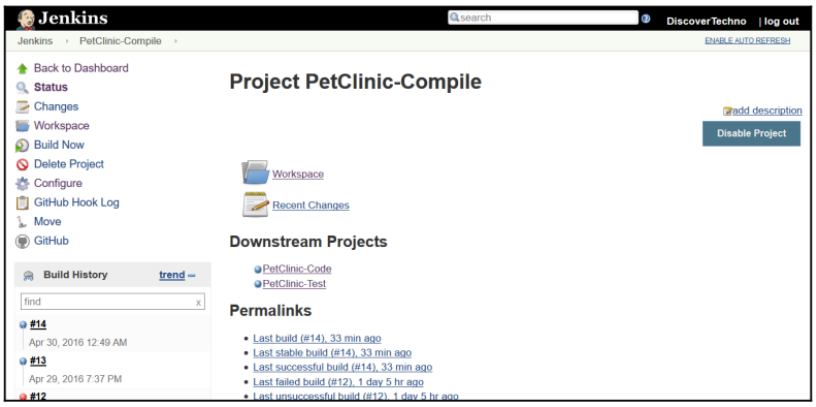

Рисунок 4.14 – Перевірка списку проєктів Downstream

6. Наступним кроком є настройка подання Build Pipeline, яке ми створили раніше.

Таблиця 4.1 – Настройка подання Build Pipeline

Ім’я	**Назва конвеєра збірки**
Опис	Опис відображається на сторінці перегляду конвеєра побудови. Його можна використовувати для відображення деталей, таких як конвеєр, ресурси, мета конвеєра, потік тощо.
Черга складання фільтрів	У черзі відображатимуться лише завдання в цьому конкретному перегляді.
Фільтрувати виконавців складання	Щоб показати виконавців складання, які можуть виконувати завдання в цьому поданні.
Build Pipeline View Назва	Build Pipeline View Title для відображення на інформаційній панелі Jenkins
Макет	За зв’язком «вихідний/нисхідний»: цей режим макета створює структуру конвеєра за зв’язком тригера «вихідний/низхідний» між завданнями.
Виберіть Початкове завдання	Установіть початкове або батьківське завдання в поданні конвеєра складання. Решта робіт зі створення буде розглядатися за зв’язком «вихідний/нисхідний».
Кількість відображених складань	Кількість конвеєрів складання для відображення в поданні.
Обмежити тригери останніми складаннями	Щоб обмежити відображення кнопки тригера лише останніми успішними конвеєрами складання.
Завжди дозволяйте ручний запуск на кроках трубопроводу	Щоб знову виконати успішний крок конвеєра, використовуючи ті самі значення параметрів, якщо складання параметризоване.
Показати заголовки конвеєрних проєктів	Щоб показати заголовок визначення конвеєра в поданні конвеєра.
Показати параметри конвеєра в заголовках проєкту	Для переліку параметрів, які використовуються для виконання останнього успішного завдання, у заголовках проєкту конвеєра.
Показати параметри конвеєра у вікні версії	Для переліку параметрів, які використовуються для запуску першого завдання, у вікні перегляду кожного конвеєра.
Частота оновлення (у секундах)	Укажіть частоту, із якою плагін Build Pipeline оновлює лайтбокс складання за секунди
URL для користувацьких CSS	Спеціальний файл CSS, якщо такий є
Стиль посилання виводу консолі	Лайтбокс, нове вікно, це вікно

7. Ми вибрали завдання складання PetClinic-Compile як початкове завдання, як показано на рисунку нижче.

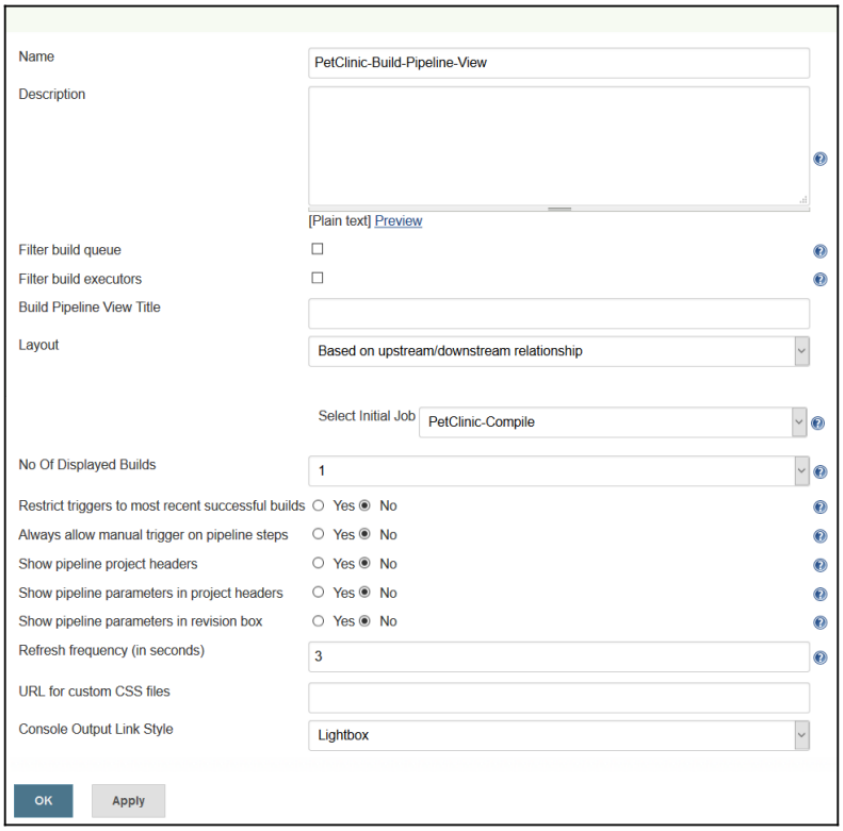

Рисунок 4.15 – Вибір завдання складання PetClinic-Compile

8. На сторінці «Перегляд» можна запустити конвеєр складання, переглянути історію, налаштувати конвеєр, видалити конвеєр тощо. Клацніть «Виконати», щоб уперше виконати Build pipeline.

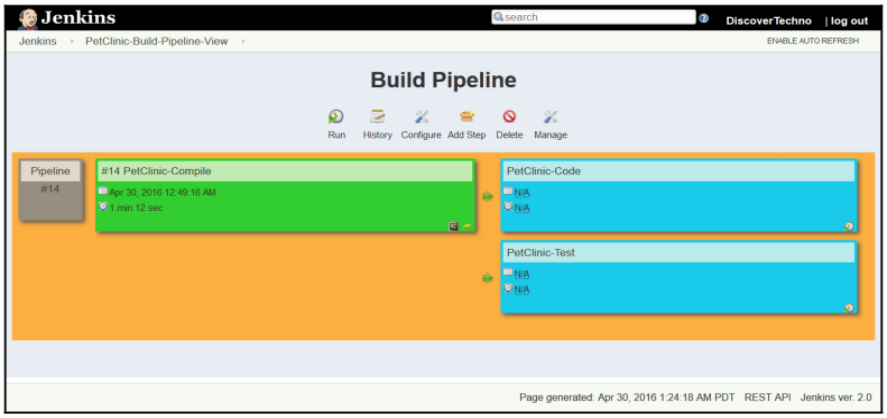

Рисунок 4.16 – Перше виконання Build pipeline

9. Нижче наведено коди кольорів за замовчуванням.

Таблиця 4.2 – Кольорова класифікація результатів складання

Колір	Опис
Червоний	Указує на невдале виконання завдання складання
Зелений	Указує на успішне виконання завдання зі складання
Синій	Указує на завдання складання, яке не було виконано
Жовтий	Указує на виконання завдання складання

10. Тепер просто спостерігайте за виконанням завдання складання в цьому конвеєрі.

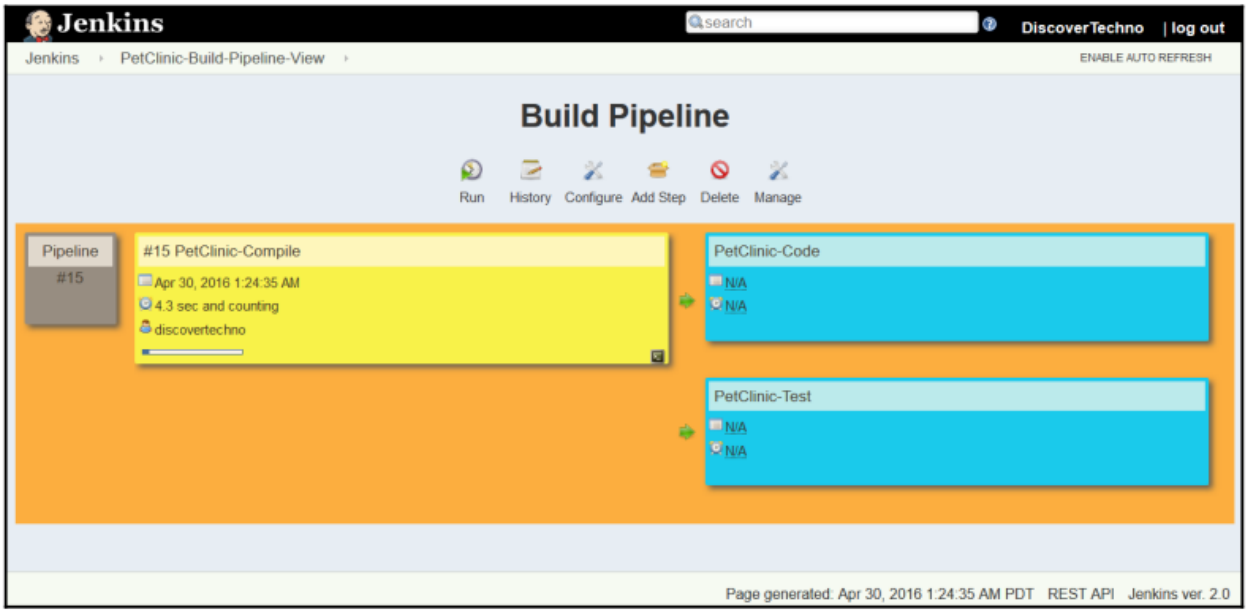

Рисунок 4.17 – Процес складання у контейнері

11. Можна бачити всі завдання зеленим кольором, оскільки всі складання виконано успішно, як показано на зображенні нижче.

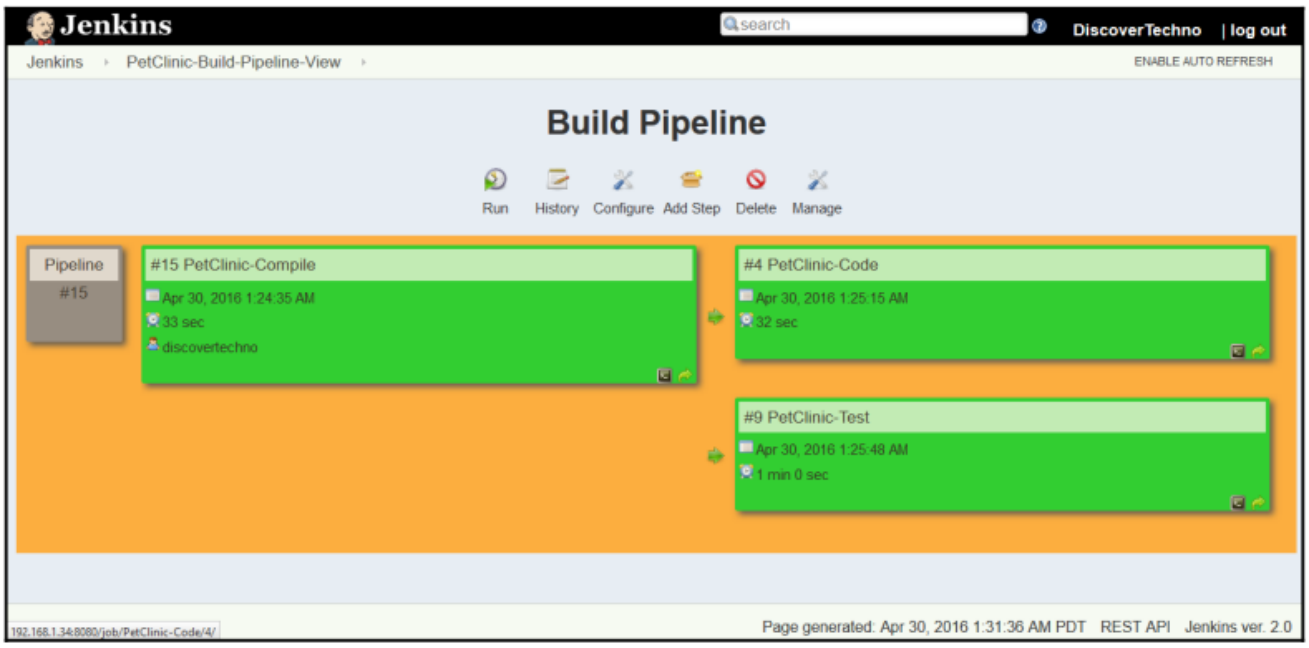

Рисунок 4.18 – Успішне виконання усіх складань

3. Розгортання файлу WAR.

Для інтеграції Maven і Tomcat створимо користувача адміністратора. Ми використовуватимемо облікові дані адміністратора для розгортання програми на сервері Tomcat.

1. Відкрийте apache-tomcat-7.0.68\conf\tomcat-users.xml і додайте в нього наступні оператори.

Тут ми визначаємо такі ролі, як manager-gui, manager-script. Для цього розгортання ми будемо використовувати роль сценарію менеджера.

Створіть користувача з іменем admin і призначте пароль і ролі, як показано нижче:

<роль rolename="manager-gui"/>

<роль rolename="manager-script"/>

<user username="admin" password="cloud@123" roles="manager-script" />

2. Тепер потрібно додати користувача адміністратора Tomcat, якого ми створили у файлі налаштувань Maven.

<сервери>

<сервер>

<id>сервер-розробки tomcat</id>

<ім'я користувача>адміністратор</ім'я користувача>

<password>пароль</password>

</server>

</servers>

3. Тепер відредагуємо файл pom.xml. Знайдіть блок Tomcat Plugin у Pom.xml і додайте наступні відомості. Переконайтеся, що ім’я сервера таке, яке ми вказали в settings.xml Maven як ідентифікатор.

<плагін>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<конфігурація>

<server>tomcat-development-server</server>

<url> http://192.168.1.35:9999/manager/text</url >

<warFile>target\petclinic.war</warFile>

<path>/petclinic</path>

</configuration>

</plugin>

4. Можна перевірити виконання з командного рядка за допомогою команди mvn tomcat7:deploy. Maven розгортає файл WAR у Tomcat 7 за допомогою програми Manager http://localhost:8080/manager/text , на шляху /petclinic .

5. У разі будь-яких збоїв через уже наявний WAR-файл у папці Tomcat webapps використовуйте tomcat7:redeploy .

Створимо завдання складання в Jenkins і додамо крок складання для виклику цілей Maven верхнього рівня.

1. Використовуйте tomcat7:redeploy як цілі. Збережіть конфігурацію.

2. Виконайте складання, натиснувши **«Створити зараз».** Перевірте процес розгортання у **вихідних даних консолі**.

[INFO] maven-war-plugin:2.3:war (default-war) @ spring-petclinic

[INFO] Веб-програма для упаковки

[INFO] Складання веб-додатка [spring-petclinic] у [d:\jenkins\workspace\PetClinic-Deploy\target\spring- petclinic-4 . 2 . 5-ЗНІМОК]

[ІНФО] Обробка військового проєкту

[INFO] Копіювання ресурсів webapp [d:\jenkins\workspace\PetClinic-Deploy\src\main\webapp]

[INFO] Веб-додаток складено за [969 мс]

[INFO] Будівельна війна: d: \jenkins \workspace \PetClinic-Deploy \target \spring-petclinic-4.2 .5-SNAPSH0T.war

[INFO]

[INFO] «< tomcat7-maven-plugin:2.2:redeploy (default-cli) < package @ spring-petclinic «<

[INFO]

[INFO] tomcat7-maven-plugin:2.2:redeploy (default-cli> @ spring-petclinic

[ІНФО] Розгортання війни на http://192.168.1.35:9999/pecclinic

Завантаження: http://192.168.1.35:9999/manager/text/deploy2path=%2Fpetclinic1/8update=true

40002/40946 Кб

40004/40946 Кб

40006/40946 Кб

40008/40946 Кб

40010/40946 Кб

40012/40946 Кб

Після успішного завантаження WAR-файлу завдання складання буде успішно завершено.

40940/40946 Кб

40942/40946 Кб

40944/40946 Кб

40946/40946 Кб

Завантажено:

http://192.168.1.35:9999/manager/text/deploy7path=%2Fpetclinic&update=true (40946 КБ на

9024,8 КБ/с)

[INFO] код статусу tomcatManager : 200, ReasonPhrase: OK

[ІНФОРМАЦІЯ] Добре – програму розгорнуто в контекстному шляху /petclinic

[ІНФО] Загальний час: 58,469 с

[ІНФО] Завершено о: 2016-05-07T23:41:13+05:30

[ІНФО] Остаточна пам`ять: 38M/263M

Завершено: УСПІХ

Коли ми використовуємо tomcat7:deploy або tomcat7:redeploy, це включає життєвий цикл пакета у виконання. Якщо ми хочемо розгорнути лише файл WAR, тоді ми можемо використати tomcat7:deploy -only, як показано у вихідних даних консолі нижче

Завдання для самостійного виконання

Необхідно нижче ознайомитися з додатковою інформацією про команди. Також потрібно виконати такі завдання.

1. Створити простий вбудований конвеєр доставки з вибраної вами теми.

2. Установіть плагін Build Pipeline.

3. Перейти на сторінку конфігурації завдання складання PetClinic-Compile.

4. Перейти до розділу «Дії після складання» .

5. Ввести ім’я завдань складання в полі Проект для створення .

6. Завантажити файл README.md зі змінами та перевірити його в репозиторії.

7. Розгорнути файл WAR.

Зміст звіту

1. Номер, тема та мета лабораторної роботи.

2. Відповіді на контрольні питання.

3. Порядок і результати розв’язання самостійного завдання (скріншоти та текстовий опис кроків розв’язання завдань).

4. Висновки до роботи.

Контрольні питання

1. Як створюються контейнери доставки на базі Jenkins?

2. Яке призначення плагіна Jenkins build pipeline?

3. Для чого потрібні PetClinic-Code і PetClinic-Test?

4. За допомогою якої команди можна перевірити виконання команди розгортання файлу WAR?

Література: [2, 5, 6, 11, 13].

2 КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

Розподіл балів, які отримують студенти

Вид роботи	Зміст	Бали
Робота на лекціях	Активна участь у дискусіях, вирішення практичних прикладів тощо	15
Завдання з ЛР	Взаємодія та специфіка роботи з GitHub. Створення першого репозиторію	5
	AWS, розгортання типової серверної архітектури	5
	Огляд контейнерів Docker. Встановлення та налаштування Docker	5
	Створення коду та налаштування конвеєра складання	5
	Встановлення Chef Workstation та сервера на Ubuntu. Налаштування сервера Chef за замовченням	5
	Безпека та моніторинг. Безпека в Jenkins і VSTS. Керування користувачами в Jenkins. Моніторинг Jenkins і Microsoft Azure	5
	Наскрізна автоматизація	5
Поточний контроль	Індивідуальні завдання 1, 2	30
Семестровий контроль	Контрольні роботи 1, 2 (тести)	20

	Усього балів	**100**

СПИСОК ЛІТЕРАТУРИ

Основна

1. Armstrong S. DevOps for Networking. Birmingham, England: Packt Publishing, 2023. P. 185. ISBN: 9781786464859.

2. Duffy M. DevOps Automation Cookbook. Birmingham, England: Packt Publishing, 2015. P. 334. ISBN: 9781784392826.

3. Giridhar C. Automate it! – Recipes to upskill your business. Birmingham, England: Packt Publishing, 2017. P. 383. ISBN: 9781786464859.

4. Soni M. Jenkins Essentials. Birmingham, England: Packt Publishing, 2015. P. 186. ISBN: 9781783553471.

5. Soni M. DevOps Bootcamp. Birmingham, England: Packt Publishing, 2023. P. 309. ISBN: 9781787285965.

6. Soni M. DevOps for Web Development. Birmingham, England: Packt Publishing, 2023. P. 301. ISBN: 9781786465702.

Додаткова

7. Sydorenko V., Perekrest A., Shendryk V., Shendryk S. Machine Learning Optimization of Air Heating Time in the Heating Control System of a Smart House. *New Technologies, Development and Application. Lecture Notes in Networks and Systems*. 2023, vol. 707, P. 3644. (Scopus, Q4).

8. Коростельов А. С., Гученко, М. І., Перекрест А. Л., Самойлов А. М., Вадурін К. О. Аналітичні розрахунки корпоративної мережі базованої на технологіях інтернету речей підприємства з екологічних досліджень. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. Том 34 (73) № 5 2023, С. 140–148. (фахове видання).

9. Korostelov A., Guchenko M., Perekrest A., Nikitina A., & Vadurin K. Модель корпоративної мережі базованої на технологіях інтернету речей підприємства з екологічних досліджень. *Системи управління, навігації та*

зв'язку: збірник наукових праць. Полтава: ПНТУ, Т. 3 (73), 2023. С. 111–114. (фахове видання).

10. Перекрест А. Л., Бахарєв В. С., Вадурін К. О., Дерієнко А. І., Іващенко А. В., Шкарупа С. А. Розробка бази даних для зберігання показників стану атмосферного повітря з дослідних станцій комунального підприємства. *Проблеми інформатизації та управління.* Київ: НАУ. Випуск 3, № 75, 2023. С. 68–86. (фахове видання)

11. Перекрест А. Л., Вадурін К. О., Юдіна А. Л. Цифровий автомат як інструмент моделювання стану суспільства. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. Кременчук: КрНУ, 2023. Випуск 3 (140). С. 52–61. (фахове видання).

12. Перекрест А. Л., Дружиніна В. В., Морозов Ю. О., Ноженко В. Ю. Використання технологій доповненої та віртуальної реальності під час створення інноваційної екосистеми громад. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. Кременчук: КрНУ, 2023. Випуск 3 (140). С. 62–73. (фахове видання).

13. Chencheva O., Lashko Y., Rieznik, D., Perekrest A., Bozhyk M. Development and research of the functional possibilities of the automated fuzzy indoor air quality management system of production premises. *Labour Protection Problems in Ukraine*. 2023, 39 (3–4), P. 36–42.

Інформаційні ресурси

14. DevOps – The Web's Largest Collection of DevOps Content. DevOps.com. URL: https://devops.com/ (дата звернення: 26.03.2023).

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Технології DevOps» для студентів денної форми навчання зі спеціальностей: 122 – «Комп'ютерні науки», 123 – «Комп'ютерна інженерія» освітньо-професійних програм: «Комп'ютерні науки», «Комп'ютерна інженерія» освітнього ступеня «Бакалавр» (частина 1)

Укладачі: д. т. н., проф. А. Л. Перекрест;
асист. К. О. Вадурін

Відповідальний за випуск д. т. н., проф. М. І. Гученко

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. _____. Наклад _____ прим. Зам. № _____. Безкоштовно.

Редакційно-видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Університетська 20, м. Кременчук, 39600