Детектор Плагиата v. 2808 - Отчёт оригинальности: 18.01.2025 12:37:10

Проанализированный документ: Паршин.pdf Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

Тип поиска: Поиск переписанного Язык: Uk

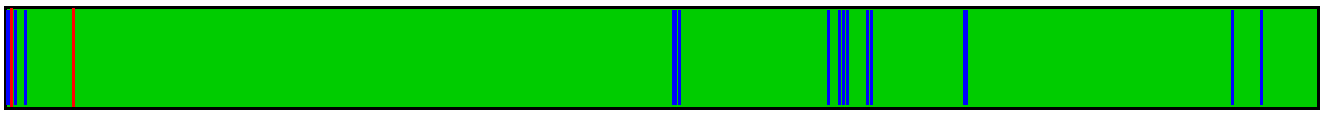
Тип проверки: Интернет

TEE и кодировка: PdfPig

Детальный анализ тела документа:

Диаграмма соотношения частей:

Граф распределения зон:



Источники плагиата: 7

0,1%	19	1. https://lib.iitta.gov.ua/id/eprint/709952/1/Ніколенко_Стаття Нова укр. шк. інтегр..pdf
0,1%	10	2. https://web-crawler.plagiarism-detector.com/get-doc-pt?did=9EiiP4y7-zZBmg
0,1%	10	3. https://ir.kneu.edu.ua/bitstreams/2d953ed5-0fb8-42e6-8087-eff8e6e34db5/download

Детали обработанных ресурсов: 146 - OK / 2 - Ошибок

Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
[не обнаружено]	[не обнаружено]	[не обнаружено]	**Обнаружено сокрытие!**

Античит-отчет UACE:

1. Статус: Анализатор **Включен** Нормализатор **Включен** сходство символов установлено на **100%**
2. Обнаруженный процент загрязнения UniCode: **8,7%** с лимитом: 4%
3. Процент нераспознанных символов после нормализации: **4,7%**
4. Все подозрительные символы будут отмечены фиолетовым цветом: Abcd...
5. Найдены невидимые символы: 0

Рекомендации по оценке:

Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется

передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

Исключённые ресурсы:

URL не найдены

Включённые ресурсы:

URL не найдены

Детальный анализ документа:

Міністерство освіти і науки України Державний заклад

Цитирования: **0,03%** id: **1**

«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут математики та інформаційних технологій Кафедра інформаційних технологій та систем Паршин Сергій Олегович РОЗРОБКА ТА ВПРОВАДЖЕННЯ ІНФРАСТРУКТУРИ ЯК КОДУ (IAC) ДЛЯ LMS MOODLE ІЗ ВИКОРИСТАННЯМ ANSIBLE ТА MOOSH здобувача вищої освіти другого (магістерського) рівня освітньої програми

Цитирования: **0,01%** id: **2**

«Комп’ютерні мережі»

за спеціальністю 123 Комп’ютерна інженерія Особистий підпис _____ Сергій ПАРШИН Науковий керівник _____, Микола СЕМЕНОВ,

Обнаружен Плагиат: **0,11%** https://lib.iitta.gov.ua/id/eprint/70995... id: **3**

кандидат педагогічних наук, доцент кафедри інформаційних технологій та систем Завідувач кафедри _____ Микола СЕМЕНОВ, кандидат педагогічних наук, доцент кафедри

інформаційних технологій та систем Полтава – 2025 АНОТАЦІЯ Тема: Розробка та впровадження інфраструктури як коду (IaC) для LMS Moodle із використанням Ansible та Moosh. Спеціальність: 123

Цитирования: **0,01%** id: **4**

«Комп’ютерна інженерія».

Установа: ЛНУ імені Тараса Шевченка, 2025 р. Магістерська робота містить: 85 с., 12 рис., 8 табл., 41 джерел. Об’єкт дослідження – розгортання та управління інфраструктурою систем управління навчанням (LMS). Предмет дослідження – реалізація інфраструктури як коду (IaC) для Moodle з використанням Ansible і Moosh. Мета роботи – розробка та впровадження інфраструктури як коду (IaC), адаптованої до розгортання та управління Moodle за допомогою Ansible та Moosh. Результати роботи – розроблено та впроваджено інфраструктуру як код (IaC) для LMS Moodle із використанням Ansible і Moosh, яка вирішує проблеми ручного розгортання та керування, автоматизуючи критичні процеси, такі як надання сервера, встановлення програмного забезпечення та конфігурації Moodle. Надано рекомендації з найкращих практик для інтеграції Ansible і Moosh та подолання труднощів впровадження в системах управління навчанням. Ключові слова: ІНФРАСТРУКТУРА ЯК КОД, АВТОМАТИЗАЦІЯ, РОЗГОРТАННЯ, КОНФІГУРАЦІЯ, MOODLE, IAC, ANSIBLE, MOOSH, LMS. 2 ANNOTATION Topic: Development and implementation of Infrastructure as Code (IaC) for the LMS Moodle using Ansible and Moosh. Speciality: 123

Цитирования: **0,01%** id: **5**

"Computer Engineering".

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2025. Master's thesis consists of: 85 p., 12 im., 8 tables, 41 sources. Object of the study – deployment and management of infrastructure for learning management systems (LMS). Subject of the study – implementation of Infrastructure as Code (IaC) for Moodle using Ansible and Moosh. Objective of the study – to develop and implement an efficient Infrastructure as Code (IaC) solution tailored for the deployment and management of Moodle using Ansible and Moosh. Results of the study – an Infrastructure as Code (IaC) solution for the LMS Moodle using Ansible and Moosh was developed and implemented. The solution addresses the challenges of manual deployment and management by automating critical processes such as provisioning servers, installing software, and configuring Moodle. Best practices for integrating Ansible and Moosh and overcoming implementation challenges in learning management systems are provided. Keywords: MOODLE, IAC, ANSIBLE, MOOSH, AUTOMATION, LMS, DEPLOYMENT, CONFIGURATION MANAGEMENT, DIGITAL INFRASTRUCTURE. 3 ЗМІСТ ВСТУП 5 РОЗДІЛ 1. АВТОМАТИЗАЦІЯ ІНФРАСТРУКТУРИ В СИСТЕМАХ УПРАВЛІННЯ НАВЧАННЯМ 9 1.1. LMS Moodle як система цифрового навчання 9 1.2. Інфрастуктура як код (IaC) в управлінні інформаційними системами 13 1.3. Ansible та Moosh як інструменти автоматизації 18 1.4. Теоретичні засади застосування IaC в розгортанні Moodle 25 Висновки до розділу 1 29 РОЗДІЛ 2. МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ ТА

ПРОЄКТУВАННЯ ІНФРАСТРУКТУРИ 30 2.1. Методологія дослідження та аналіз вимог 30 2.2. Проєктування архітектури фреймворку IaC для Moodle 36 2.3. Стратегія впровадження 39 2.4. Методи збору й аналізу даних 47 Висновки до розділу 2 51 РОЗДІЛ 3. ВПРОВАДЖЕННЯ ІНФРАСТУКТУРИ ЯК КОДУ ТА АНАЛІЗ ЇЇ ЕФЕКТИВНОСТІ 52 3.1. Практична реалізація IaC у Moodle та інтеграція Ansible і Moosh 52 3.2. Тестування розробленого фреймворку IaC 61 3.4. Рекомендації із застосування фрейморку та мінімізації ризиків 76 Висновки до розділу 3 78 ВИСНОВКИ 79 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 81 4 ВСТУП У сучасному освітньому середовищі системи управління навчанням (LMS) стали ключовими платформами, що спрощують адміністрування, документування, відстеження та імплементацію цифрових курсів і навчальних програм. LMS Moodle набула поширення завдяки відкритому коду, гнучкості, масштабованості та надійному набору функцій, а її модульна архітектура дозволяє закладам вищої освіти (ЗВО) адаптувати систему до конкретних педагогічних та адміністративних потреб, покращуючи таким чином досвід навчання та ефективність роботи [24]. Оскільки ЗВО все більше покладаються на цифрову інфраструктуру для підтримки своїх навчальних та адміністративних функцій, складність управління такими системами відповідно зростає. Ця проблема вимагає інноваційних підходів до управління інфраструктурою, які забезпечуватимуть узгодженість, масштабованість і операційну ефективність. Інфраструктура як код (IaC) з’явилася як трансформаційна парадигма в цьому контексті, уможливлюючи автоматизоване надання та керування IT-інфраструктурою за допомогою машинозчитуваних файлів конфігурації. IaC втілює принципи розробки програмного забезпечення, такі як контроль версій і автоматизація, щоб підвищити надійність і відтворюваність розгортання інфраструктури [37, 41]. Ansible, інструмент автоматизації з відкритим вихідним кодом, став відомим завдяки своїй безагентній архітектурі (для виконання операцій не потрібно попередньо встановлювати спеціальне ПЗ на цільові машини) та декларативній мові, які спрощують автоматизацію складних IT-завдань [16]. Використовуючи плейбуки на основі YAML, Ansible полегшує координацію системних конфігурацій, розгортання програм і автоматизацію завдань, таким чином спрощуючи керування IT-середовищем. Доповнюючи Ansible, Moosh – інструмент командного рядка, спеціально розроблений для Moodle – надає адміністраторам набір команд для автоматизації рутинних завдань, тим самим підвищуючи ефективність і узгодженість керування платформою [33]. 5 Інтеграція Ansible і Moosh представляє собою підхід до автоматизації розгортання та управління інфраструктурою Moodle. Використовуючи принципи IaC, ця інтеграція спрямована на зменшення проблем, пов’язаних із ручним керуванням інфраструктурою, таких як відхилення конфігурації, людські помилки та обмеження масштабованості. У цій дисертації досліджується розробка та впровадження фреймворку IaC, адаптованого для Moodle, з використанням Ansible і Moosh, з метою підвищення надійності, масштабованості та операційної ефективності розгортання Moodle. Об’єкт дослідження – розгортання та управління інфраструктурою систем управління навчанням (LMS). Предмет дослідження – реалізація інфраструктури як коду (IaC) для Moodle з використанням Ansible і Moosh. Мета роботи – розробка та впровадження узгодженої, масштабованої та операційно ефективної інфраструктури як коду (IaC), адаптованої до розгортання та управління Moodle за допомогою Ansible та Moosh.

Обнаружен Плагиат: **0,06%** https://ir.kneu.edu.ua/bitstreams/2d9… + 3 ресурсов! id: **6**

Для досягнення поставленої мети необхідно вирішити наступні завдання: 1) провести огляд існуючих досліджень та фреймворків, пов’язаних з IaC, для визначення прогалин і можливостей для інновацій в автоматизації інфраструктури Moodle; 2) проаналізувати функціональні та нефункціональні вимоги, щоб визначити потреби та обмеження розгортання Moodle; 3) розробити схему архітектури, що інтегрує Ansible та Moosh для управління Moodle, і встановити стратегії управління конфігурацією та контролю версій; 4) розробити плейбуки Ansible та Moosh-скрипти для автоматизовної конфігурації сервера, інсталяції програмного забезпечення та налаштування Moodle; 5) упровадити IaC фреймворк у середовищі розробки згідно з розробленою архітектурою; 6 6) виконати оцінку ефективності, масштабованості та стабільності розгортання за показниками продуктивності; 7) задокументувати ризики та розробити стратегії їх мінімізації; 8) сформувати рекомендації з впровадження IaC в LMS Moodle для організацій, які впроваджують цифрове навчання. Новизна результатів дослідження полягає в застосуванні нового підходу для створення спеціалізованої інфраструктури як коду шляхом поєднання Ansible і Moosh спеціально для Moodle. Дослідження пропонує нові

сценарії, плейбуки та методики оптимізації розгортання Moodle, підвищення стабільності і масштабованості, які не були широко досліджені в наявній літературі. Методи дослідження. Для досягнення поставлених цілей у дослідженні застосовуються такі методи: аналіз – для огляду існуючих досліджень, пов'язаних з IaC, Moodle, Ansible і Moosh, а також оцінки кількісних даних для вимірювання ефективності архітектури IaC; моделювання – для проєктування фреймворку IaC; експеримент – для тестування реалізації фреймворку IaC у контрольованому середовищі; порівняння – для оцінки ефективності фреймворку IaC шляхом порівняння автоматизованих розгортань із традиційними підходами; вимірювання – для збору кількісних даних про час розгортання, показники помилок, використання системних ресурсів. У першому розділі проведено комплексний огляд літератури для визначення теоретичної основи дослідження, включаючи огляд Moodle як системи управління навчанням, принципи та переваги інфраструктури як коду (IaC), а також детальний аналіз Ansible і Moosh як інструментів для автоматизації. У другому розділі викладено методологію проєктування та розробки структури IaC, включаючи аналіз функціональних і нефункціональних вимог до розгортання Moodle, розробку архітектурної схеми для IaC та створення плейбуків Ansible і сценаріїв Moosh. Обґрунтовано використання обраних інструментів і технологій, розроблено стратегію впровадження та методи 7 тестування й перевірки фреймворку. Розглянуті етичні міркування та обмеження дослідження. У третьому розділі представлено реалізацію розробленої структури IaC та проведена оцінка її ефективності. Описано процес розгортання в контрольованому середовищі та робочі процеси інтеграції Ansible і Moosh з демонстрацією використаних методик тестування. Проведений аналіз результатів за допомогою показників ефективності, виконано порівняння автоматизованого підходу із традиційними ручними методами. Ідентифіковано труднощі, що виникли під час реалізації, та їх вирішення, а також рекомендації для подальшого застосування фреймворку. 8 РОЗДІЛ 1 АВТОМАТИЗАЦІЯ ІНФРАСТРУКТУРИ В СИСТЕМАХ УПРАВЛІННЯ НАВЧАННЯМ 1.1. LMS Moodle як система цифрового навчання Система управління навчанням (LMS) Moodle, що є абревіатурою від Modular Object-Oriented Dynamic Learning Environment, була створена в 2002 році Мартіном Дугіамасом, австралійським педагогом та психологом, який виявляв глибокий інтерес до інтеграції педагогіки й технологій [24]. Його бачення полягало у розробці платформи з відкритим кодом, яка б реалізовувала принципи конструктивістського навчання, сприяла колаборації та створенню середовища, орієнтованого на здобувача освіти. Перший реліз Moodle у 2002 році започаткував новий етап у розвитку систем управління навчанням. Завдяки високій гнучкості та можливостям налаштування платформа швидко набула популярності серед навчальних закладів різних рівнів, від шкіл до університетів, а також у корпоративному секторі [31]. Відкритий код став основою глобальної спільноти розробників, педагогів та адміністраторів, які активно вдосконалюють платформу шляхом розробки плагінів, тем і функціональних інтеграцій. Протягом років Moodle пройшов значну еволюцію як у функціональному, так і в архітектурному аспектах. Порівняно з версією 1.0, що пропонувала базові інструменти для управління курсами, платформа поступово розширювалася, включаючи інноваційні функції, такі як компетентнісне навчання, розширені системи оцінювання та аналітичні інструменти для звітності. Ключовим етапом став реліз Moodle 2.0 у 2010 році, який запровадив модульну архітектуру, яка значно полегшила налаштування системи та інтеграцію з зовнішніми інструментами через API та вебсервіси. Останні версії Moodle враховують сучасні технологічні тенденції, зокрема мобільну сумісність, хмарні розгортання та посилені заходи безпеки для захисту даних. Крім того, платформа відповідає міжнародним стандартам доступності, зокрема WCAG (Web Content Accessibility Guidelines), що 9 забезпечує доступність навчального середовища для всіх категорій користувачів, включно з людьми з інвалідністю [30]. Сталий розвиток Moodle зумовлений його здатністю адаптуватися до змінних потреб освітніх закладів та вимог цифрової епохи. Завдяки цьому платформа залишається важливим інструментом для забезпечення інноваційних та індивідуалізованих навчальних діяльностей. У контексті еволюції освітніх парадигм Moodle продовжує відігравати провідну роль, відповідаючи на виклики цифровізації й забезпечуючи якісну підтримку освітнього процесу [39]. Ключовим компонентом архітектури Moodle є модульний дизайн, що дозволяє адміністраторам налаштовувати платформу шляхом інтеграції різноманітних плагінів і модулів. Така гнучкість сприяє адаптації навчального середовища до конкретних педагогічних завдань та інституційних потреб. Керування цифровими курсами. Одна з центральних функцій Moodle – система керування курсами, яка дозволяє викладачам створювати, організовувати та подавати

матеріали курсу. Вона підтримує структуризацію за темами чи тижнями, завантаження різних типів ресурсів (документи, відео, інтерактивні медіа) та планування оцінювання через тести, завдання й форуми. Ця функціональність дозволяє реалізувати різні освітні стратегії, включаючи змішане навчання, перевернуті класи та повністю асинхронні курси. Інструменти оцінювання та відстеження виконання. Moodle пропонує розширену систему оцінювання, що підтримує рубрики, шкали й зважені критерії, забезпечуючи гнучкість у виставленні оцінок та наданні зворотного зв'язку. Навчання на основі компетенцій дозволяє відстежувати індивідуальний прогрес здобувачів освіти відповідно до очікуваних результатів навчання (ILO), сприяючи персоналізованому підходу. Інтерактивні інструменти та спільне навчання. Для сприяння колаборативній взаємодії платформа надає широкий набір інтерактивних інструментів, таких як форуми, вікі та семінари, які підтримують активну участь здобувачів освіти та peer-to-peer (P2P) взаємодію. Синхронна й 10 асинхронна комунікація відбувається через вбудовані системи обміну повідомленнями та інструменти відеоконференцій. Адміністрування та звітність. Moodle забезпечує широкі адміністративні можливості: управління користувачами, контроль доступу на основі ролей і створення детальних звітів. Адміністратори можуть керувати обліковими записами, призначати ролі й дозволи та отримувати аналітику щодо активності користувачів, завершення курсів та їхньої успішності [30, 31]. Інтеграція та масштабованість. Інтеграційні можливості Moodle включають підтримку стандартів сумісності, таких як Learning Tools Interoperability (LTI), та API для взаємодії з іншими освітніми системами, зокрема SIS і репозиторіями контенту. Крім того, архітектура Moodle забезпечує масштабованість: платформа підтримує розподілені розгортання, балансування навантаження та кешування для забезпечення стабільної роботи навіть за умов високої відвідуваності. Можливість локалізації та підтримка багатомовності забезпечують доступність Moodle для користувачів у різних країнах, що розширює її використання в глобальному масштабі. Широке впровадження Moodle відображено в його глобальному впливі та різноманітності установ, які використовують цю платформу. Станом на 2024 рік Moodle має понад 240 мільйонів користувачів у всьому світі, охоплюючи мільйони активних курсів у понад 200 країнах [32]. У корпоративному секторі Moodle служить основним інструментом для програм навчання та розвитку. Організації в різних галузях, включаючи технології, охорону здоров'я та фінанси, використовують Moodle для проведення навчання з дотримання вимог, курсів професійного розвитку та програм адаптації. Тенденції упровадження. Кілька ключових тенденцій характеризують нинішній ландшафт впровадження Moodle, що відображає ширші зміни в освітніх технологіях і парадигмах цифрового навчання: 11 1. Фокус на рішеннях з відкритим вихідним кодом. Надання переваги платформам LMS з відкритим кодом, таким як Moodle, продовжує зростати, керуючись прагненням до економічно ефективних, настроюваних і підтримуваних спільнотою рішень. Платформи з відкритим вихідним кодом дозволяють установам змінювати та розширювати функціональні можливості без обмежень власних систем, сприяючи інноваціям та адаптивності. 2. Перехід до гібридних і змішаних моделей навчання. Пандемія COVID-19 прискорила впровадження гібридних і змішаних моделей навчання, де Moodle відіграє вирішальну роль в інтеграції особистої та онлайн-навчальної діяльності. Навчальні заклади все частіше використовують Moodle для створення безперебійного навчання, яке поєднує традиційну взаємодію в класі з цифровим контентом і оцінюванням. 3. Інтеграція з новими технологіями. Здатність Moodle інтегруватися з новими технологіями, такими як штучний інтелект (AI), віртуальна реальність (VR) і аналітика навчання, покращує його функціональність і користувацький досвід. Ці інтеграції забезпечують персоналізовані навчальні шляхи, освітній досвідх з більшим залученням і керовану даними інформацію про ефективність і залученість студентів. 4. Доступність та інклюзивність. Усе більше уваги приділяється тому, щоб зробити цифрові навчальні середовища доступними для всіх користувачів, у тому числі для людей з обмеженими можливостями. Відповідність Moodle стандартам доступності, таким як Рекомендації щодо доступності веб-контенту (WCAG), є суттєвим чинником у його прийнятті установами, які прагнуть інклюзивної освіти [32]. Очікується, що траєкторія впровадження Moodle продовжуватиме висхідну тенденцію, зумовлена постійним розвитком освітніх технологій і зростаючим попитом на гнучкі, масштабовані та економічно ефективні рішення LMS. Прогнози вказують на подальше зростання бази користувачів 12 Moodle, зокрема з тієї причини, що навчальні заклади прагнуть покращити інфраструктуру цифрового навчання у відповідь на зміну освітніх потреб і технологічних інновацій. Інтеграція розширених функцій, таких як персоналізоване навчання на основі штучного інтелекту та покращена сумісність із

мобільними пристроями, зміцнить позицію Moodle як провідної LMS у світовому освітньому та навчальному середовищі [39]. 1.2. Інфраструктура як код (IaC) в управлінні інформаційними системами Інфраструктура як код (IaC) представляє собою фундаментальну зміну парадигми в управлінні IT-інфраструктурою, орієнтуючи її процеси на принципи розробки програмного забезпечення. IaC замінює ручну конфігурацію інфраструктури на автоматизоване управління за допомогою машинозчитуваних файлів конфігурацій [41]. Цей підхід забезпечує автоматизацію, стандартизацію, послідовність і масштабованість, що є ключовими факторами ефективного функціонування сучасних IT-систем. Сутність цієї практики полягає в управлінні й розгортанні (deployment) інфраструктурних ресурсів за допомогою коду та автоматизації. Цей підхід упроваджує принципи контролю версій, рецензування коду та методологій безперервної інтеграції/розгортання (CI/CD), що сприяє узгодженню управління інфраструктурою з гнучкими методами розробки програмного забезпечення [4, 17, 34]. Цей підхід сприяє узгодженню підтримки інфраструктури з практиками Agile, тим самим підвищуючи гнучкість та респонсивність IT-операцій [5]. На підставі цих його властивостей можна виділити кілька основних принципів інфрастуктури як коду. 1. Декларативна конфігурація. IaC типово застосовує декларативний підхід, де користувач визначає бажаний стан інфраструктури, а інструменти автоматизації самостійно виконують дії, необхідні для досягнення цього стану. У порівнянні з імперативним підходом, це покращує читабельність і 13 обслуговування конфігурацій, спрощуючи визначення вимог до інфраструктури. 2. Ідемпотентність. IaC гарантує, що багаторазове застосування однієї і тієї ж конфігурації призводить до однакового результату, без непередбачуваних побічних ефектів. Ідемпотентність зменшує ризик помилок і відхилень у конфігурації, забезпечуючи узгодженість у розгортаннях [41]. 3. Контроль версій. Інтеграція IaC із системами контролю версій (VCS) забезпечує відстеження змін, зручність спільної роботи й можливість відкату до попередніх станів. Це дозволяє застосовувати до інфраструктури найкращі практики розробки програмного забезпечення, підвищуючи прозорість і керованість процесів. 4. Автоматизація та відтворюваність. Автоматизація є центральною складовою IaC, усуваючи ризики, пов’язані з ручною конфігурацією. Завдяки скриптам розгортання інфраструктури IaC усуває невідповідності та помилки, притаманні ручній конфігурації, забезпечуючи точне відтворення середовищ на різних етапах життєвого циклу розробки. 5. Модульність та реюзабіліті. IaC сприяє модульному дизайну, де компоненти інфраструктури визначаються у вигляді незалежних блоків, які можна багаторазово використовувати. Це спрощує управління складною інфраструктурою та сприяє швидшій адаптації до нових вимог [41]. 6. Масштабованість і гнучкість. Автоматизований характер IaC забезпечує динамічне масштабування ресурсів відповідно до потреб організації, і підтримує широкий спектр інфраструктурних компонентів — серверів, баз даних, мережевих пристроїв і хмарних сервісів [34]. Концептуальні основи IaC ґрунтуються на принципах розробки програмного забезпечення та системного адміністрування. IaC проводить паралелі з методологіями розробки програмного забезпечення, наголошуючи на якості коду, тестуванні та постійному вдосконаленні. Застосовуючи орієнтований на програмне забезпечення підхід до управління інфраструктурою, IaC полегшує інтеграцію розробки та операцій (DevOps), 14 сприяючи створенню середовища для співпраці, яке підвищує загальну надійність і продуктивність систем [8, 23]. Крім того, IaC абстрагує базову складність інфраструктури та забезпечує керування на вищому рівні за допомогою коду. Ця абстракція не тільки спрощує створення інфраструктури, але й покращує здатність керувати нею в масштабі, враховуючи динамічні потреби сучасних IT-середовищ. Зазначені особливості свідчать про те, чому еволюція практик управління інфраструктурою значно змінилася з появою концепції інфраструктури як коду. Щоби досконало означити трансформаційний вплив IaC, важливо порівняти його з традиційними методами управління інфраструктурою. Традиційні підходи, які зазвичай характеризуються ручними процесами та розрізненими конфігураціями, різко контрастують із автоматизованою, закодованою природою IaC. Це порівняння дозволяє виявити переваги та обмеження обох підходів, підкреслюючи причини широкого впровадження IaC у сучасних IT-середовищах. Традиційно управління IT-інфраструктурою значною мірою спиралося на ручні процеси. Адміністратори й системні інженери здійснювали налаштування серверів, мережевого обладнання та інших компонентів через пряме взаємодію з апаратними та програмними інтерфейсами. Основні особливості цього підходу: – ручне надання ресурсів при встановленні й налаштуванні серверів та обладнання; – розрізнені конфігурації, а саме застосування налаштувань у кожному конкретному випадку без стандартизованих процедур; – ведення документації

окремо від конфігурацій, що часто призводить до розбіжностей і застарілих даних; – реактивне обслуговування: проблеми вирішуються за фактом їх виникнення, замість попереджувального підходу. 15 Зокрема, Д. Соколовскі зазначає, що традиційні підходи демонструють обмежену ефективність у масштабних середовищах, де зростаюча складність інфраструктури ускладнює забезпечення узгодженості, масштабованості й респонсивності [37], що обґрунтовано в табл. 1.1. Таблиця 1.1 – Порівняння IaC і традиційного управління інфраструктурою Аспект Традиційне управління IaC Розгортання Ручна установка й Автоматизоване – через ресурсів налаштування обладнання сценарії та та програмного конфігураційні файли. забезпечення. Управління Розрізнені, часто Стандартизовані конфігураціями непослідовні декларативні налаштування, виконувані конфігурації, керовані вручну. кодом. Масштабованість Обмежена; масштабування Висока; інфраструктуру потребує ручного можна масштабувати втручання. програмно. Узгодженість і Схильність до помилок Забезпечує узгоджені відтворюваність через людський фактор; розгортання завдяки можливі розбіжності. автоматизації. Контроль версій Зазвичай не інтегровано з Інтегровано, що системами контролю забезпечує відстеження версій. змін і повернення до попередніх версій. Швидкість Повільний цикл через Швидке розгортання розгортання ручні процеси. завдяки автоматизації. Імовірність Висока через втручання Знижена завдяки помилок людини. тестованим сценаріям. Документація Окрема, часто застаріла Самодокументована документація. інфраструктура через код. Обслуговування Реактивне; проблеми Проактивне; вирішуються по мірі їх автоматичне тестування виникнення. й оновлення. Співпраця Обмежена; знання Посилена завдяки зосереджені в окремих спільним репозиторіям і експертів. стандартизованим процесам. 16 Економічність Вищі витрати через ручну Нижчі витрати завдяки працю й потенційну автоматизації й неефективність. оптимізації ресурсів. На підставі порівняльного аналізу, представленого в таблиці, визначено такі переваги IaC над традиційними підходами: – підвищена швидкість і ефективність: автоматизація прискорює розгортання, дозволяючи швидко адаптуватися до змін; – узгодженість і надійність: автоматизовані процеси мінімізують помилки та забезпечують стабільність середовищ; – масштабованість: IaC підтримує динамічне масштабування інфраструктури у відповідь на зміни попиту; – прозорість і відстежуваність: інтеграція з VCS забезпечує контроль змін і спрощує управління; – знижена ймовірність помилок: тестовані сценарії значно зменшують ризики, пов’язані з людським фактором; оптимізація витрат: зменшення потреби в ручній праці та краща оптимізація ресурсів. Попри численні переваги, впровадження Інфраструктури як Коду (IaC) супроводжується низкою викликів. Одним із найбільших бар’єрів є початковий поріг входження. Організаціям і командам потрібно опанувати нові інструменти й методології, а також змінити підхід до управління інфраструктурою, що потребує часу та ресурсів на навчання персоналу [10, 14]. Крім того, зростання масштабу IaC-розгортань призводить до збільшення складності управління. Великі проєкти вимагають створення чітких стандартів, модульних структур і належних процедур, щоб забезпечити підтримуваність і масштабованість конфігурацій [41]. Без впровадження належних практик складність може перешкоджати ефективному використанню IaC. 17 У контексті безпеки існують ризики, що інфраструктура, яка автоматично створюється за допомогою скриптів і конфігураційних файлів, може стати вразливою до атак, якщо ці файли недостатньо захищені. Для мінімізації ризиків необхідно забезпечити надійний контроль доступу, шифрування та регулярний аудит кодової бази [11]. Інші складності можуть виникнути через залежність від інструментів. IaC значною мірою спирається на специфічні платформи й фреймворки, і це спричиняє прив’язку до конкретного постачальника послуг або ускладнити інтеграцію з новими технологіями, якщо зміняться потреби організації. Таким чином, хоча IaC є потужним інструментом для автоматизації та оптимізації інфраструктури, його впровадження потребує стратегічного підходу, врахування викликів і розробки комплексних заходів для їх подолання. 1.3. Ansible та Moosh як інструменти автоматизації Ansible набув поширення як один із провідних інструментів в галузі IT- автоматизації та управління конфігураціями. Ansible базується на безагентній архітектурі, яка вирізняє його серед інших інструментів управління конфігураціями, що зазвичай вимагають встановлення спеціальних агентів на керованих машинах [16]. Ця архітектура використовує стандартні протоколи SSH (Secure Shell) для комунікації між центральним вузлом і хостами, спрощуючи розгортання та знижуючи експлуатаційні витрати [27]. Основні компоненти архітектури Ansible включають: 1. Центральний вузол (Control Node). Це машина, на якій встановлено Ansible і з якої виконуються завдання автоматизації. Центральний (або контрольний) вузол організовує

комунікацію з хостами й підтримує інвентаризацію систем для управління. 2. Хости (Managed Nodes). Цільові системи, які Ansible налаштовує й якими він керує. На хостах не потрібно встановлювати спеціальне 18 програмне забезпечення, окрім доступу через SSH і сумісного інтерпретатора Python. 3. Інвентар (Inventory). Файл або динамічне джерело, що містить перелік хостів і організовує їх у групи. Інвентар може бути статичним (формати INI або YAML) або динамічним, отриманим із хмарних провайдерів чи інших зовнішніх систем. 4. Модулі (Modules). Окремі компоненти коду, що виконують конкретні завдання на хостах, такі як встановлення програмного забезпечення, управління сервісами або налаштування файлів. Ansible включає велику бібліотеку вбудованих модулів, а користувачі можуть розробляти власні модулі для розширення функціональності. 5. Плагіни (Plugins). Розширюють базові можливості Ansible, забезпечуючи інтеграцію з різними системами, покращуючи форматування виводу й додаючи функції, такі як типи підключень і механізми кешування. 6. Плейбуки (Playbooks). Файли у форматі YAML, які визначають послідовність завдань і конфігурацій, що застосовуються до хостів. Плейбуки задають виконання модулів і описують бажаний стан інфраструктури. Окремо слід відзначити модулі – фундаментальні блоки Ansible, що інкапсулюють певні функції, які можна виконувати на хостах. Ansible надає комплексний набір модулів, класифікованих за функціональними можливостями, включаючи такі: – системні модулі: керують конфігураціями системного рівня, такими як облікові записи користувачів, дозволи на файли та встановлення пакетів. – мережеві модулі: налаштування мережевих пристроїв і служб, включаючи маршрутизатори, комутатори та брандмауери. 19 – хмарні модулі: взаємодіють з постачальниками хмарних послуг, такими як AWS, Azure та Google Cloud, полегшуючи надання та керування хмарними ресурсами. – модулі баз даних: керують службами баз даних, включаючи створення баз даних, керування користувачами та виконання резервних копій. – службові модулі: виконують допоміжні завдання, такі як копіювання файлів, керування шаблонами та виконання команд оболонки. Розширюваність бібліотеки модулів Ansible дозволяє користувачам адаптувати завдання автоматизації до конкретних організаційних потреб, сприяючи універсальності та адаптивності в різноманітних IT-середовищах. Утім, ключовою особливістю автоматизаційних можливостей Ansible є плейбуки, написані мовою YAML (YAML Ain't Markup Language) - стандартом серіалізації даних, зрозумілим для людини [40]. Плейбуки визначають бажаний стан інфраструктури, окреслюючи серію сценаріїв, кожен з яких орієнтований на конкретний набір хостів та визначає завдання для виконання Основні характеристики плейбуків включають: 1. Декларативність. Плейбуки описують бажаний кінцевий стан, а не процедурні кроки для його досягнення, що дозволяє Ansible визначати необхідні дії для приведення поточного стану у відповідність з бажаним. 2. Ідемпотентність. Повторне виконання плейбуків не призведе до небажаних побічних ефектів, забезпечуючи стабільність і надійність у розгортаннях. 3. Модульність та реюзабіліті. Плейбуки можуть включати ролі, які є повторно використовуваними колекціями завдань, змінних, шаблонів і обробників, сприяючи модульності та полегшуючи підтримку складних конфігурацій. 4. Змінні та шаблони. Плейбуки підтримують використання змінних і шаблонізації за допомогою Jinja2, що дозволяє впроваджувати 20 динамічні конфігурації на основі різних середовищ або вхідних параметрів. Ansible застосовується в широкому спектрі сценаріїв автоматизації IT та управління інфраструктурою. Одним із ключових напрямків є управління конфігурацією, де Ansible автоматизує налаштування серверів і застосунків, забезпечуючи послідовність виконання між різними середовищами. Це включає встановлення необхідних пакетів, налаштування сервісів та управління конфігураційними файлами, що значно знижує ризик людських помилок і підвищує ефективність операційних процесів [1, 2]. Крім того, Ansible використовується для розгортання застосунків та забезпечення діяльності інфраструктури. Автоматизація процесу розгортання застосунків дозволяє швидко та надійно створювати середовища, розгортати код і управляти залежностями, що суттєво скорочує час виходу продукту на ринок і мінімізує можливі помилки [6]. Таким чином, Ansible автоматизує налаштування ресурсів як локально, так і в хмарних середовищах, включаючи створення віртуальних машин, мереж та систем зберігання даних. Інтеграція Ansible з безперервною інтеграцією та безперервним розгортанням (CI/CD) сприяє автоматизації тестування, розгортання та моніторингу додатків [17]. Також Ansible використовується для координації складних багаторівневих розгортань, забезпечуючи ефективне управління залежностями та мінімізуючи ручне втручання. Завдяки можливостям безпеки та відповідності, Ansible автоматизує впровадження політик безпеки, управління патчами та аудит журналів, що покращує загальний рівень захищеності IT-середовищ [1, 6]. У контексті розгортання та

управління інфраструктурою Moodle, Ansible пропонує автоматизацію повторюваних завдань, забезпечення узгодженості конфігурацій і сприяння масштабованості розгортань. Наприклад, плейбуки Ansible можуть автоматизувати встановлення та налаштування вебсерверів, баз даних і власне Moodle, що знижує складність і скорочує час, необхідний для налаштування. Крім того, інтеграція Ansible з Moosh – інструментом 21 командного рядка, розробленим спеціально для Moodle – дозволяє координувати завдання, пов’язані з Moodle. Moosh, що є акронімом від Moodle Shell, – це інструмент командного рядка, розроблений для полегшення адміністрування та керування Moodle. Розроблений для усунення обмежень, притаманних типовому вебінтерфейсу адміністрування Moodle, Moosh забезпечує спрощений засіб виконання широкого спектру адміністративних завдань через командний рядок [33]. Використовуючи Moosh, адміністратори можуть автоматизувати регулярні завдання, виконувати масові операції та інтегрувати керування Moodle у ширші робочі процеси автоматизації [15]. Moosh характеризується розширюваністю та повним охопленням адміністративних функцій Moodle, що відповідають різноманітним адміністративним діям, таким як керування користувачами, створення курсів, встановлення плагінів і міграцію даних, забезпечуючи точний і детальний контроль над сайтом Moodle. Це є особливо доцільним у середовищах, де розгортання Moodle відбувається у великому масштабі або потребує частих оновлень і обслуговування. Інструмент також підтримує пакетні операції, дозволяючи адміністраторам виконувати команди для кількох курсів, користувачів або інших об’єктів одночасно, скорочуючи час, необхідний для адміністративних завдань на масштабних платформах. Утім, можливості Moosh не обмежуються базовими адміністративними функціями і включають розширені операції, які оптимізують процес управління Moodle. Серед таких можливостей варто відзначити інтеграцію в сценарії автоматизації та конвеєри, що дає змогу інкорпорувати управління Moodle до фреймворків IaC та процесів безперервної інтеграції/розгортання (CI/CD). Адміністратори можуть створювати власні команди Moosh, щоб відповідати конкретним потребам організації, розширюючи функціонал інструменту та адаптуючи його до унікальних адміністративних процесів. Роль Moosh в управлінні Moodle полягає в доповненні інструментів автоматизації, таких як Ansible. Хоча призначення Ansible полягає в 22 автоматизації загальних завдань IT-інфраструктури, прерогативою Moosh є виконання специфічних операцій Moodle. Він забезпечує спеціалізований рівень автоматизації відповідно до вимог адміністрування Moodle [33]. Поєднання Ansible та Moosh, представлене в цьому дослідженні, здатне створити інтегрований підхід до управління інфраструктурою, де Ansible координує розгортання та конфігурацію основних систем, а Moosh забезпечує управління самою платформою Moodle. Це дозволить адміністраторам створювати комплексні робочі процеси розгортання, що охоплюють як інфраструктурний, так і прикладний рівні. Такий підхід поєднує масштабованість і універсальність Ansible із спеціалізованими функціями Moosh для Moodle, формуючи цілісну й ефективну архітектуру IaC. У табл. 1.2 надано порівняння Ansible і Moosh, де зазначені їхні основні функції та застосування в адмініструванні Moodle. Таблиця 1.2. Компаративний огляд Ansible і Moosh у контексті адміністрування Moodle Аспект Ansible Moosh Основна функція Універсальна Інструмент командного автоматизація IT-процесів рядка (CLI) для і управління адміністрування Moodle. конфігураціями. Обсяг Широкий спектр IT- Адміністративні автоматизації завдань, включаючи завдання Moodle: налаштування серверів, управління мереж і розгортання користувачами, курсами, додатків. плагінами тощо. Управління Використовує плейбуки у Використовує PHP- конфігураціями форматі YAML для команди, адаптовані для визначення бажаних внутрішньої конфігурації станів і автоматизації Moodle. конфігурацій. Зручність для Вимагає знання Інтуїтивно зрозумілий адміністраторів синтаксису Ansible і для адміністраторів Moodle модулів; не є Moodle із досвідом специфічним для Moodle. роботи в CLI. Розширюваність і Висока розширюваність Обмежений налаштування завдяки кастомним функціональністю модулям та ролям; Moodle; 23 підходить для можливість створення різноманітних сценаріїв кастомних команд автоматизації. Moosh. Інтеграція з Інтегрується з конвеєрами Переважно інтегрується іншими CI/CD, хмарними в екосистему Moodle; інструментами сервісами та іншими може бути включений до інструментами ширших скриптів. автоматизації. Продовження табл. 1.2 Спільнота та Широка глобальна спільнота Нішова спільнота, підтримка з великою кількістю орієнтована на Moodle, із ресурсів, модулів і сторонніх спеціалізованою інтеграцій. підтримкою та документацією. Складність Висока, через широту Не є складним для опанування функціоналу та необхідність адміністраторів Moodle; розуміння принципів прості операції через автоматизації. командний рядок. Приклади Автоматизація Масове

створення використання налаштування серверів, користувачів, розгортання багаторівневих автоматизація додатків, інтеграція з налаштування курсів, хмарними встановлення плагінів, інфраструктурами. міграція даних у Moodle. На користь поєднання Ansible та Moosh також свідчить те, що одним із обмежень Moosh є його функціонал, орієнтований виключно на специфічні завдання в середовищі Moodle. Цей інструмент не володіє широкими можливостями автоматизації, необхідними для комплексного управління IT- інфраструктурою. Таким чином, Moosh більше підходить як додаток до Ansible, який може виконувати задачі на рівні всієї інфраструктури. Хоча Moosh дозволяє створювати кастомні команди, це не забезпечує такої гнучкості, як створення кастомних модулів чи ролей в Ansible. Це обмеження може стати проблемою для організацій, які прагнуть інтегрувати Moosh у більш складні або нетипові сценарії автоматизації. 24 Враховуючи ці особливості, Moosh найдоцільніше використовувати у поєднанні з більш універсальними інструментами автоматизації, такими як Ansible. Така комбінація дозволяє компенсувати відсутні у Moosh функції, додаючи ширші можливості автоматизації для комплексного управління інфраструктурою Moodle. 1.4. Теоретичні засади застосування IaC в розгортанні Moodle Упровадження IaC та автоматизації в управлінні IT-інфраструктурою ґрунтується на кількох основоположних теоріях і моделях, які разом підвищують ефективність, надійність і масштабованість технологічних розгортань. Центральними для цих теоретичних основ є принципи DevOps, Agile і системного мислення, кожен з яких робить внесок в еволюцію автоматизованого управління інфраструктурою. DevOps (development & operations) є практикою та методологією, яка втілює культурний і професійний рух, що базується на співпраці, комунікації та інтеграції між розробниками програмного забезпечення та IT- командами [19, 20]. DevOps прагне подолати традиційні підходи, які перешкоджають розробці ПЗ, і сприяє таким практикам, як CI/CD. IaC є критично важливим компонентом DevOps, оскільки він автоматизує надання та керування інфраструктурою, тим самим сприяючи швидкому та надійному випуску програмного забезпечення. Завдяки кодифікації інфраструктури, практики DevOps гарантують, що середовища є послідовними, відтворюваними та масштабованими, що важливо для підтримки гнучкості та оперативності, необхідних для сучасних циклів розробки програмного забезпечення. Agile зміцнює принципи DevOps, впроваджуючи ітераційну розробку, гнучкість і орієнтовані на клієнта підходи [26]. Agile-фреймворки, такі як Scrum і Kanban, надають пріоритет поступовому прогресу та адаптивному плануванню, яке узгоджується з можливостями автоматизації IaC. Здатність швидко створювати та модифікувати інфраструктуру підтримує акцент Agile 25 на швидкій ітерації та безперервному фідбеку, що дозволяє командам розробників оперативно реагувати на зміну вимог і відгуки користувачів. Ця синергія між методами Agile та IaC сприяє створенню середовища, у якому зміни в інфраструктурі можна розгортати разом з оновленнями застосунків, гарантуючи, що як програмне забезпечення, так і його базове середовище розвиваються в тандемі [14]. Системне мислення пропонує цілісну перспективу управління складною IT-інфраструктурою, підкреслюючи взаємозалежності та взаємодії всередині системи [18]. Цей теоретичний підхід виступає за розуміння ширшого контексту, в якому працюють окремі компоненти, просуваючи стратегії, що підвищують стійкість і адаптивність загальної системи. У сфері IaC системне мислення проявляється в розробці автоматизованих робочих процесів, які враховують безліч взаємозв’язків між серверами, мережами, базами даних і програмами. Моделюючи інфраструктуру як взаємопов’язану систему, IaC полегшує виявлення та мінімізацію ризиків, залежностей і точок збою. Зазначені теорії доповнюють практики CI/CD і VCS, які забезпечують структуровані рамки для реалізації автоматизації та забезпечення узгодженості в управлінні інфраструктурою. Конвеєр CI/CD втілює послідовність автоматизованих процесів, які сприяють безперервній інтеграції змін коду, автоматизованому тестуванню та безперебійному розгортанню у виробничих середовищах [4]. IaC інтегрується в цей конвеєр шляхом автоматизації підготовки та налаштування середовищ, необхідних для кожного етапу конвеєра, тим самим зменшуючи ручне втручання та прискорюючи загальний процес доставки. VCS, з іншого боку, дає змогу відстежувати та керувати конфігураціями IaC, гарантуючи, що зміни є документованими, оборотними та доступними для спільного перегляду. Розглянуті теоретичні засади та моделі знаходять практичне застосування в розгортанні та управлінні Moodle. У рамках парадигми DevOps інфраструктура як код дозволяє автоматизувати завдання розгортання Moodle, 26 такі як ініціалізацію сервера, налаштування бази даних і конфігурація платформи. Кодифікувавши ці процеси, адміністратори можуть забезпечити послідовне розгортання, зменшуючи ймовірність

відхилення конфігурації та мінімізуючи час налаштування нових середовищ. Ця автоматизація відповідає цілям DevOps щодо прискорення впровадження програмного забезпечення та покращення співпраці між командами розробників. Так, інтеграція підручників Ansible із командами Moosh дозволяє координувати як загальні завдання інфраструктури, так і специфічні адміністративні дії Moodle, створюючи зв’язаний робочий процес розгортання. Відповідним чином розгортанню Moodle сприяють методології Agile. Оскільки заклади освіти та організації розвивають середовище Moodle для вирішення специфічних педагогічних і адміністративних потреб, IaC сприяє швидкому розгортанню змін інфраструктури без переривання поточних операцій. Можливість швидко надавати додаткові сервери, масштабувати ресурси бази даних або оновлювати конфігурації Moodle підтримує принцип гнучкого реагування на нові вимоги [36]. Використання сценаріїв IaC із керуванням версіями гарантує, що зміни в інфраструктурі можна переглядати, тестувати та розгортати поступово, узгоджуючи з ітераційною природою циклів розробки Agile [14]. Окрім DevOps і Agile, застосування системного мислення до розгортання Moodle передбачає розробку інфраструктури, яка враховує взаємозалежності між різними компонентами, такими як веб-сервери, бази даних, балансувальники навантаження та системи зберігання. IaC дозволяє моделювати ці взаємозв’язки за допомогою автоматизованих сценаріїв, які визначають зв’язки та конфігурації кожного компонента. Наприклад, плейбук Ansible може автоматизувати налаштування кластера Moodle із збалансованим навантаженням, координуючи кілька вебсерверів, налаштовуючи центральну БД і встановлюючи мережеві правила, які забезпечують ефективний розподіл трафіку. 27 Конвеєрна модель CI/CD додатково покращує розгортання Moodle шляхом інтеграції IaC у процес безперервного розгортання [5]. Конвеєри автоматизованого тестування та розгортання можуть включати плейбуки Ansible і команди Moosh для налаштування середовищ Moodle як частини робочого процесу розгортання. Ця інтеграція гарантує, що кожна зміна коду супроводжується відповідними оновленнями інфраструктури, зберігаючи узгодженість між програмою та її базовим середовищем. Системи контролю версій у цьому контексті відіграють роль в управлінні конфігураціями інфраструктури Moodle, відстежуючи зміни в плейбуках Ansible і сценаріях Moosh. На практиці застосування цих теорій до розгортання Moodle призводить до спрощеного, надійного та масштабованого процесу управління інфраструктурою [10]. Наприклад, навчальні заклади можуть використовувати Ansible і Moosh для автоматизації розгортання Moodle у кількох структурних підрозділах, гарантуючи, що для кожного екземпляру Moodle дотримуються стандартизовані конфігурації й протоколи безпеки. Можливість автоматизувати типові адміністративні завдання зменшує навантаження на IT- персонал і дозволяє їм зосередитися на більш стратегічних ініціативах. Масштабованість, яку при цьому забезпечує IaC, дає змогу закладам пристосовуватись до зростаючої кількості користувачів і курсів [35]. Таким чином, інтеграція теоретичних принципів із практичними інструментами автоматизації Ansible і Moosh створює основу для керування впровадженням Moodle. Поєднуючи управління інфраструктурою з DevOps і Agile і системним мисленням, навчальні заклади можуть досягти покращенння операційних можливостей Moodle і надання високоякісних, надійних та адаптованих освітніх програм. 28 Висновки до розділу 1 У розділі закладено основу для контекстуалізації розробки та впровадження інфраструктури як коду (IaC) у LMS Moodle з використанням Ansible і Moosh. Проведено огляд системи управління навчанням Moodle, розглянуто його історію, еволюцію, ключові функції та поточні тенденції впровадження. Розглянуто концепцію інфраструктури як коду, роз’яснено основні принципи та можливості, які IaC пропонує порівняно з традиційними підходами до управління інфраструктурою. Детальне порівняння показало, як IaC покращує узгодженість, масштабованість і ефективність за допомогою автоматизації, контролю версій і декларативних конфігурацій. Представлено детальний огляд архітектури Ansible (модулів, плейбуків та сценаріїв використання) і Moosh у контексті адміністрування Moodle. Порівняльний аналіз підкреслив взаємодоповнюючі ролі Ansible і Moosh та показав їхнє синергетичне використання і його сприяння комплексній структурі IaC для розгортання Moodle. Вдалося дійти висновку, що інтеграція Ansible і Moosh у парадигму IaC має потенціал для підвищення стабільності, масштабованості і ефективності роботи інфраструктури Moodle. Проведено дослідження відповідних теорій і моделей, що лежать в основі IaC та автоматизації, таких як DevOps, методології Agile та системне мислення, а також їх застосування в контексті розгортання Moodle. Ця теоретична основа забезпечила розуміння принципів інтеграції IaC в управління інфраструктурою Moodle та закладає основу для наступних розділів, у яких глибше

розглянуто практичну розробку та стратегії розгортання Moodle за допомогою IaC. 29 РОЗДІЛ 2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ ТА ПРОЄКТУВАННЯ ІНФРАСТРУКТУРИ 2.1. Методологія дослідження та аналіз вимог Впровадження технологічних рішень у комплексних інформаційних середовищах вимагає системного підходу до дослідження і проєктування. У контексті управління LMS Moodle, інтеграція IaC та інструментів автоматизації вимагає структури, здатної врахувати ітераційний розвиток, емпіричну перевірку та адаптацію до виникаючих потреб. За умови використання структурованої методології, де поєднується теоретична точність із практичним застосуванням, розробка інноваційних рішень може мінімізувати ризики, підвищити масштабованість і забезпечити послідовність операцій. У цьому дослідженні застосовується підхід Design Science Research (DSR) для розробки та впровадження IaC для LMS Moodle із використанням Ansible та Moosh. Застосування DSR зумовлено тим, що він зосереджений на створенні та оцінці інноваційних артефактів, спрямованих на вирішення конкретних проблем у певному контексті [21]. У цьому випадку артефактом є фреймворк IaC, адаптований для розгортання Moodle, який використовує Ansible для автоматизації загальної інфраструктури та Moosh для адміністративних завдань, специфічних для Moodle. Обрана методологія включає кілька ключових компонентів, які в поєднані забезпечують систематичний підхід до дослідження. 1. Ідентифікація проблеми та мотивація. Виявлення труднощів, пов’язаних із ручним розгортанням і управлінням інфраструктурою Moodle, таких як непослідовність конфігурацій, обмеження масштабованості й операційна неефективність. 2. Мета вирішення проблеми. Розробка автоматизованого фреймворка IaC, який забезпечить узгоджене, масштабоване та 30 ефективне розгортання Moodle шляхом інтеграції інструментів Ansible і Moosh. 3. Проєктування та розробка. Створення фреймворка IaC шляхом написання плейбуків Ansible та інтеграції команд Moosh для автоматизації завдань, специфічних для Moodle. 4. Демонстрація. Реалізація фреймворка IaC у реальному сценарії розгортання Moodle для демонстрації його функціональності та ефективності. 5. Оцінка. Аналіз продуктивності, надійності та масштабованості фреймворка IaC через емпіричне тестування й порівняльний аналіз із традиційними методами ручного управління. 6. Комунікація. Документування результатів дослідження, методологій і внеску для представлення академічній і професійній спільнотам. Вибір підходу DSR для проведення дослідження ґрунтується на кількох міркуваннях. По-перше, основна мета розробки та впровадження структури IaC вимагає методологічного підходу, який акцентує увагу на створенні артефактів і практичному вирішенні проблем. DSR забезпечує структурований шлях досягнення цього, зосереджуючись на ітеративному проектуванні, розробці та оцінці технологічних рішень [22, 28]. По-друге, DSR враховує динамічну природу IT-середовища, дозволяючи періодично вдосконалювати та адаптувати структуру IaC у відповідь на нові вимоги та проблеми, що виникають під час впровадження. По-третє, акцент на оцінці в рамках DSR забезпечує, що розроблена структура проходить ретельне тестування та валідацію в реальних умовах, тим самим встановлюючи її практичну значущість. Цей емпіричний фокус є вирішальним для обґрунтування переваг IaC у розгортанні Moodle. Завдяки принципам DSR дослідження сприяє як теоретичному, так і практичному аспекту роботи, покращуючи розуміння IaC-застосунків в освітніх технологіях і надаючи значущі відомості для практиків, які прагнуть 31 удосконалити управління інфраструктурою Moodle. Окрім того, DSR підтримує включення якісних і кількісних методів дослідження, сприяючи комплексній оцінці структури IaC. Ця інтегративна здатність підвищує надійність і деталізацію результатів дослідження. Процес дослідження в рамках DSR розгортається через серію циклів ітерації, проілюстрованих на рис. 2.1, кожен з яких містить окремі фази, які сприяють розвитку та вдосконаленню структури IaC у проєкті. Рис. 2.1 – Візуалізація ітеративної моделі Design Science Research 32 Спираючись на обрану методологію дослідження, ефективна розробка та впровадження структури IaC для Moodle потребує розуміння базових вимог, які висуваються до інфраструктури Moodle. Функціональні вимоги. Визначають поведінку, функції та процеси, які має підтримувати інфраструктура Moodle, щоб досягти поставленої мети. Ці вимоги безпосередньо пов’язані із завданнями та операціями, які адміністратори та користувачі виконують у середовищі Moodle. Основні функціональні вимоги до інфраструктури Moodle включають: 1. Автоматичне координування серверів і служб. Інфраструктура повинна підтримувати автоматичне розгортання вебсерверів, серверів баз даних та інших основних служб, необхідних для роботи Moodle. Це включає встановлення та конфігурацію Apache або Nginx для вебобслуговування, MySQL або PostgreSQL для керування базами даних і PHP для серверної обробки Moodle. 2. Управління користувачами

та автентифікація. Система повинна забезпечувати автоматизоване створення користувачів, призначення ролей і процеси автентифікації. Ця вимога включає інтеграцію з існуючими системами керування ідентифікацією, такими як LDAP або OAuth, щоб оптимізувати доступ користувачів і забезпечити безпечні механізми автентифікації. 3. Розгортання цифрових курсів та контенту. Інфраструктура повинна забезпечувати автоматичне створення та налаштування курсів Moodle, включаючи розгортання контенту курсу, ресурсів і діяльностей. Ця функція має вирішальне значення для підтримки узгодженості між кількома курсами та мінімізації ручного втручання. 4. Керування плагінами. Можливість автоматизувати встановлення, налаштування та оновлення плагінів Moodle дозволить середовищу Moodle залишатися розширюваним і доступним для найновіших функцій та патчів безпеки, наданих розробниками плагінів. 33 5. Операції резервного копіювання та відновлення. Автоматичне резервне копіювання бази даних, каталогів даних Moodle і конфігураційних файлів, а також механізми для відновлення цих резервних копій у разі втрати даних або збою системи. 6. Моніторинг і ведення логів. Для забезпечення надійності та цілісності середовища Moodle необхідний автоматичний моніторинг продуктивності системи, стану застосунків і журналів безпеки. Це включає в себе налаштування сповіщень і інформаційних панелей для забезпечення видимості системних показників і потенційних проблем у реальному часі. Нефункціональні вимоги. Визначають якісні атрибути та обмеження, яким має відповідати інфраструктура Moodle для надійної та ефективної роботи системи за різних умов. Основні нефункціональні вимоги до інфраструктури Moodle включають: 1. Масштабованість. Здатність до горизонтального та вертикального масштабування для можливості пристосування до зростаючої кількості користувачів, курсів і одночасних дій. Включає можливість динамічно додавати або видаляти ресурси залежно від навантаження, не порушуючи поточні операції. 2. Продуктивність. Швидкий час відповіді та ефективне використання ресурсів. Це включає оптимізацію конфігурацій сервера, запитів до бази даних і стратегій кешування для ефективної обробки великих обсягів трафіку та великих наборів даних. 3. Безпека. Належна конфігурація серверів і служб, впровадження протоколів шифрування, регулярні оновлення безпеки та дотримання практик щодо контролю доступу й керування вразливими місцями. 4. Підтримуваність. Інфраструктура має бути зручна в обслуговуванні, дозволяючи адміністраторам оновлювати, змінювати та усувати несправності з мінімальними зусиллями. Це передбачає чітку документацію, модульні конфігурації та використання 34 стандартизованих сценаріїв автоматизації для завдань обслуговування. 5. Відповідність стандартам. Інфраструктура Moodle має відповідати відповідним галузевим стандартам і нормативним вимогам, таким як закони про захист даних (наприклад, GDPR) і освітні стандарти. 6. Зручність використання. Хоча цей аспект є більш значимим для кінцевих користувачів, він також впливає на адміністрування. Структура автоматизації має забезпечувати зрозумілі інтерфейси, чітку документацію та всебічну підтримку. 7. Гнучкість і розширюваність. Інфраструктура має адаптуватися до мінливих освітніх потреб і технологічного прогресу. Це включає в себе здатність інтегруватися з новими інструментами та платформами, підтримувати налаштування функціональних можливостей Moodle і враховувати майбутні вдосконалення без значних змін конфігурації. Застосування цих функціональних і нефункціональних вимог до розгортання Moodle включає в себе їх перетворення на завдання в рамках IaC. Так, автоматизоване координування серверів і служб може бути досягнуто за допомогою плейбуків Ansible, управління користувачами та процеси автентифікації – командами Moosh, оптимізація продуктивності – налаштуванням параметрів сервера та впровадженням механізмів кешування, вимоги безпеки – розгортанням брандмауерів і регулярними патчами безпеки, зручність обслуговування – використанням модульних плейбуків і багатоцільових ролей, відповідність стандартам – впровадженням нормативних перевірок і конфігурацій у структуру IaC. Відповідність функціональним і нефункціональним вимогам є підставою вважати структуру IaC такою, що забезпечує комплексне рішення для розгортання та керування інфраструктурами Moodle, які відповідають потребам сучасних закладів вищої освіти [3]. 35 2.2. Проєктування архітектури фреймворку IaC для Moodle Проєктування архітектури IaC для Moodle в цьому дослідженні базується модульному та багаторівневому підході. Основою архітектури є Ansible і його плейбуки, які визначають бажаний стан серверів, баз даних, мережевих компонентів та інших основних служб. Плейбуки структуровано таким чином, щоб забезпечити узгоджене розгортання в різних середовищах, як локальних, так і хмарних. Команди Moosh вбудовані у плейбуки Ansible, що дозволяє безперебійно виконувати такі

завдання, як керування обліковими записами користувачів, створення курсу, встановлення плагінів і міграція даних. Схема архітектури, представлена на рис. 2.2, демонструє, як Ansible обробляє ініціалізацію та конфігурацію компонентів загальної інфраструктури, Moosh відповідає нюансам вимог адміністрування Moodle, створюючи синергічний зв'язок, який підвищує загальну ефективність розгортання. Рис. 2.2 – Архітектура IaC фреймворку для Moodle 36 Як показано на схемі, Ansible виконує координуючу роль, керуючи розгортанням і конфігурацією компонентів базової інфраструктури, таких як вебсервери, бази даних і мережеві ресурси. Після того, як інфраструктура створена, Ansible викликає команди Moosh для виконання специфічних для Moodle завдань. Наприклад, після розгортання екземпляра Moodle, Ansible плейбук може використовувати Moosh для створення курсів, призначення ролей користувачів і встановлення необхідних плагінів. Крім того, ця інтеграція підтримує динамічну та контекстно-залежну автоматизацію, де змінні та шаблони Ansible можна передати командам Moosh для адаптації конфігурацій на основі конкретних сценаріїв розгортання. На рис. 2.3 показана схема, яка ілюструє спосіб, у який відбувається взаємодія Ansible та Moosh в межах спроєктованої архітектури. Рис. 2.3 – Взаємодія Ansible та Moosh через точки інтеграції 37 Інтеграція також поширюється на моніторинг і технічне обслуговування. Плейбуки Ansible можуть планувати та виконувати команди Moosh для рутинних завдань обслуговування, таких як оновлення ядра Moodle, керування обліковими записами користувачів і виконання міграції даних. Управління конфігураціями в даній системі реалізується через плейбуки та ролі Ansible, які визначають бажаний стан серверів, баз даних, мережевих компонентів і Moodle. Контроль версій здійснюється за допомогою Git, який дозволяє відстежувати зміни, проводити рецензування та повертатися до попередніх версій у разі потреби. Централізоване сховище зберігає всі плейбуки Ansible, команди Moosh і конфігураційні файли. Використання стратегій розгалуження, таких як функціональне та релізне розгалуження, ізолює розробку та дозволяє безпечно інтегрувати нові функції чи оновлення. Інтеграція з конвеєрами CI/CD забезпечує автоматичне тестування й перевірку змін, що мінімізує ризик помилок і невідповідностей [7, 12]. Структура дотримується рекомендованих практик документування та автоматизації, спрощуючи обслуговування й покращення. Документація для плейбуків, ролей і команд Moosh забезпечує легке розуміння та модифікацію конфігурацій. Автоматизоване тестування підтримує надійність фреймворка, гарантує коректність змін і зберігає безперервність операцій [9]. У табл. 2.1 представлений перелік стратегій, які використовуються для керування конфігураціями та контролю версій у рамках запропонованої IaC для Moodle. Таблиця 2.1 – Стратегії управління конфігураціями та контролю версій у фреймворку IaC для Moodle Аспект Стратегія Опис Інструмент Плейбуки та Використання плейбуків Ansible для управління ролі Ansible визначення бажаного стану інфраструктури та конфігураціями Moodle. Ролі модулюють конфігурації для повторного використання. Система контролю Git Застосування Git для відстеження змін у версій плейбуках Ansible, командах Moosh та конфігураціях. Забезпечує спільну розробку. 38 Продовження табл. 2.1 Стратегія Гілки функцій і Функціональні гілки — для розробки, розгалуження випусків релізні — для підготовки стабільних версій. Підтримує організовану розробку. Інтеграція Конвеєри CI/CD Інтеграція з CI/CD для автоматичного автоматизації тестування, перевірки та розгортання конфігурацій Moodle й інфраструктури. Документування Коментарі та файли Чітка документація через коментарі в коді README та файли README з інструкціями для налаштування та використання. Тестування Автоматизовані Скрипти для перевірки коректності сценарії тестування плейбуків Ansible та команд Moosh. Перегляд коду Pull-запити та Pull-запити для внесення змін з рецензування обов'язковим переглядом. Механізми відкату Revert у Git та Функції відкату в Git та Ansible для функції Ansible повернення до стабільного стану після помилок. Безпека та Розподіл доступу за Використання RBAC у Git для обмеження контроль доступу ролями (RBAC) доступу до критичних файлів лише авторизованими користувачами. Інтеграція Ansible для автоматизованого керування конфігураціями та Git для контролю версій забезпечує стабільність і узгодженість фреймворку IaC для Moodle. Ansible мінімізує ризик помилок і конфігураційного відхилення, тоді як Git надає інструменти для колаборації, відстеження змін і документування. Додатково автоматизоване тестування, механізми відкату та контроль доступу мають на меті створити основу для управління динамічними потребами Moodle та підвищити стійкість системи до змін, спрощуючи її підтримку та розвиток. 2.3. Стратегія впровадження Реалізація рішення інфраструктури як коду для Moodle за допомогою Ansible включає методичний та ітераційний підхід. Цей підхід гарантує, що кожен компонент інфраструктури точно визначено, контролюється

версіями та ретельно перевіряється перед інтеграцією у виробниче середовище. Наступний план представляє собою структуровану методику створення сценаріїв IaC на основі Ansible, які втілюють архітектурні принципи та вимоги, визначені в попередній главі. 39 1. Перевірка функціональних та нефункціональних вимог. Це включає підтвердження цільової версії Moodle, специфікацій сервера, рішень бази даних і будь-яких залежностей, таких як плагіни чи сторонні інтеграції. На цьому етапі важливо пересвідчитися, що центральний вузол (машина, на якій запущено Ansible) має необхідні залежності, включаючи Python і сам Ansible, і створити локальне середовище розробки або спеціальне середовище тестування, де початкова розробка і перевірка плейбуків може відбуватися без ризику для стабільності. 2. Інвентаризація і визначення змінних. Створюється файл інвентаризації Ansible, який перелічує всі цільові хости (вебсервери, сервери баз даних, балансувальники навантаження) і логічно їх групує. Для розгортання Moodle розглядається можливість групування серверів за їх функціями (web, db, loadbalancer тощо), далі визначаються змінні хоста та групові змінні в окремих файлах YAML, де фіксуються такі деталі, як IP- адреси, облікові дані бази даних, параметри конфігурації Moodle та шляхи до каталогів даних Moodle. 3. Створення ролей та забезпечення модульності. На цьому етапі створюються спеціальні ролі для таких завдань: – webserver_setup; – database_provisioning; – moodle_installation; – moodle_configuration; – moosh_integration. У кожній ролі структуруються завдання, шаблони, обробники та змінні за замовчуванням відповідно до рекомендованих практик Ansible. Модульність забезпечує, що модифікації одного компонента (наприклад, конфігурації бази даних) не впливатимуть на непов’язані компоненти. 40 4. Управління шаблонами й файлами. Полягає у використанні можливостей шаблонів Ansible з Jinja2 для динамічного створення конфігураційних файлів, що забезпечує інкорпорацію унікальних параметрів кожного середовища. На цьому етапі розробляються шаблони для конфігураційних файлів Moodle (наприклад, config.php), конфігурацій віртуального хосту вебсервера та сценаріїв ініціалізації бази даних. Шаблони зберігаються у відповідних ролях, забезпечуючи узгоджену й автоматичну генерацію файлів конфігурації. 5. Розробка плейбуків та команд Moosh. Головним плейбуком є site.yml, який викликає необхідні ролі в належній послідовності, щоби ресурси інфраструктури надавались до початку встановлення Moodle. Команди Moosh інтегруються як завдання в ролі, пов’язані з Moodle, щоб автоматизувати створення курсів, керування плагінами та іншими адміністративними завданнями Moodle. Наприклад, після завершення встановлення Moodle завдання Moosh використовуються, щоб налаштувати початкові курси, параметри сайту за замовчуванням і заповнити облікові записи користувачів. 6. Тестування та валідація в контрольованому середовищі. Перш ніж інтегрувати сценарії IaC у конвеєр CI/CD, виконуються локальні тести, щоб переконатися у належному функціонуванні сценаріїв. Для цього використовується режим ansible-playbook –check, який імітує зміни без їх фактичного застосування, а також виявляє синтаксичні помилки або неправильні конфігурації. Після цього розгортається тестове середовище (наприклад, віртуальна машина або контейнерне налаштування), щоб повністю запустити плейбуки. Після цих кроків необхідно переконатися, що Moodle є доступним, стабільним і правильно налаштованим, і що всі інтегровані команди Moosh виконуються успішно. Для візуалізації цього етапу на рис. 2.4 показано схематичне зображення тестового середовища, що складається з одного вебсервера, одного сервера бази даних, а також контрольного вузла Ansible, на якому запускаються тестові плейбуки. 41 Рис. 2.4 – Схема тестового середовища під управлінням головного хосту Ansible Це тестове середовище демонструє типове налаштування інфраструктури для автоматизованого тестування конфігурації та розгортання. Центральний вузол Ansible служить точкою керування, що забезпечує безперебійну конфігурацію та координування вебсерверів і серверів баз даних. Окрім підготовки до початкового розгортання, ця практика дозволяє тестувати зміни інфраструктури, розгортання додатків і конфігурації системи перед розробкою чи релізом, мінімізуючи людський фактор та прискорюючи життєвий цикл розробки та розгортання. 7. Впровадження безперервної інтеграції та безперервної доставки (CI/CD). Сценарії Ansible для IaC інтегруються в конвеєр CI/CD. Налаштовується CI-конвеєр, який автоматично запускає перевірки синтаксису та інструменти лінтінгу (наприклад, ansible-lint) для кожного коміту. Додатково впроваджуються автоматизовані тести, які перевіряють доступність Moodle, коректну ініціалізацію бази даних і встановлення необхідних плагінів перед внесенням змін. Після успішного завершення тестів 42 автоматично запускається розгортання в середовищі тестування або на стадії продакшну. 8. Документація і передача знань. Забезпечується підтримка повної документації для спрощення поточного

обслуговування та впровадження майбутніх удосконалень. Ролі Ansible та плейбуки коментуються вбудованими анотаціями, що пояснюють мету й логіку ключових завдань. У файлах README надається інформація про налаштування та запуск плейбуків, описуються загальні змінні та структура репозиторію. Додатково документується процес додавання нових ролей, інтеграції плагінів Moodle або адаптації конфігурацій для інших LMS-рішень за потреби. 9. Моніторинг, логування та цикли зворотнього зв'язку. Моніторинг системної продуктивності, логів та користувацького відгуку дозволяє ідентифікувати напрямки вдосконалення. У разі виявлення проблем продуктивності, масштабованості або функціональних прогалин, здійснюється коригування ролей або змінних Ansible. Контроль версій у Git забезпечує відстеження всіх змін, полегшує відкат до попередніх версій у разі виникнення проблем і створює можливість аудиту змін конфігурацій. 10. Ітеративне вдосконалення й оновлення. Передбачає адаптацію середовища Moodle до змінних освітніх вимог, що може включати додавання нових функціональностей, плагінів або оптимізацію продуктивності. Модульна структура IaC-скриптів забезпечує можливість внесення змін або додавання нових ролей відповідно до потреб. Модифікації підлягають тестуванню та валідації для забезпечення відповідності вимогам продуктивності, безпеки та нормативних стандартів. Представлений покроковий план встановлює чітку методологію розробки сценаріїв IaC за допомогою Ansible для керування розгортанням Moodle. Він поєднує модульні структури ролей, шаблони динамічної конфігурації, інтегровані команди Moosh і безперервне тестування. Ітеративний характер цього плану, що включає аналіз, розробку, тестування, 43 інтеграцію та вдосконалення, відображає досвід галузі та відповідає принципам DevOps і безперервного вдосконалення. У контексті автоматизації та управління власне Moodle як підсистемою центральне місце займає Moosh, забезпечуючи комплексний підхід до адміністративних задач і спрощуючи рутинні операції. У цьому проєкті Moosh зокрема є засобом заповнення проганлини між адміністративним вебінтерфейсом Moodle і потребою в спрощених, автоматизованих процесах управління. Надаючи повний набір команд, Moosh дозволяє адміністраторам виконувати завдання, які в іншому випадку потребували б повторюваних ручних втручань через графічний інтерфейс Moodle. Після завершення підготовки вебсерверів та серверів бази даних за допомогою Ansible, останній може викликати команди Moosh, який охоплює різноманітний набір команд, що задовольняють різні аспекти адміністрування Moodle. О. Щербина у своїй роботі [15] висвітлив доцільність використання Moosh, зокрема завдяки створенню сценаріїв реєстрації нових користувачів, призначення ролей та керування профілями користувачів без ручного введення даних через інтерфейс Moodle. Приклад команди, яка створює користувача на ім'я

Цитирования: **0,01%** id: **7**

«Дмитро Петренко»

з електронною поштою dmytro.petrenko@gmail.com та ролью

Цитирования: **0,01%** id: **8**

«Студент»:

moosh user-create --firstname=Дмитро --lastname=Петренко –email= dmytro.petrenko@gmail.com –role=student Ще однією можливістю є автоматизація створення та налаштування курсів Moodle. Це включає налаштування категорій курсів, створення нових курсів, реєстрацію користувачів і налаштування параметрів курсу. Приклад команди, яка створює екземпляр курсу

Цитирования: **0,02%** id: **9**

«Вступ до автоматизації»

з короткою назвою

Цитирования: **0,01%** id: **10**

«AUTOM101»

в категорії, що має id=1. moosh course-create --fullname=

Цитирования: **0,02%** id: **11**

"Вступ до автоматизації"

--shortname=

Цитирования: **0,01%** id: **12**

"AUTOM101"

--categoryid=1 44 Подібним чином через Moosh-команди можливе встановлення, оновлення та видалення плагінів Moodle. Приклад команди, яка встановлює модуль, що додає функціональність для створення завдань у цифрових курсах: moosh plugin-install --type=mod --plugin=assignsubmission_file За потреби автоматизації міграції даних між різними сайтами Moodle в межах побудованої інфраструктури або виконання бекапу курсів чи даних користувачів, можуть використовуватися спеціальні команди, які здійснюють копіювання освітнього контенту. Приклад команди, яка виконує операцію резервного копіювання курсу в директорію з id=2, шлях до якої з назвою файлу зазначається в параметрі destination: moosh backup-course --courseid=2 --destination=/backups/course2.bak Зміна параметрів сайту, таких як методи автентифікації, мовні параметри та політики безпеки, здійснюється аналогічним чином. У наведеному далі прикладі команди виконується зміна конфігураційного параметра siteurl для системи Moodle: moosh config-set --name=siteurl --value=https://do.moodle.edu.ua Опанування синтаксису та функціональних можливостей Moosh є критичним завданням для адміністраторів Moodle, що вимагає цілеспрямованого навчання та набуття практичного досвіду. Часто організаціям доводиться виходити за межі стандартного набору команд, розробляючи кастомні рішення для виконання специфічних адміністративних завдань, які не охоплюються базовим функціоналом інструменту. Ключовим аспектом впровадження Moosh є забезпечення повної інтеграції з Ansible, що вимагає ретельного налаштування плейбуків і скриптів, особливої уваги до управління залежностями та послідовності виконання завдань. Додатковим аспектом координування зв’язки Ansible- Moosh є необхідність постійної актуалізації IaC-скриптів, адже еволюція Moodle та оновлення команд Moosh потребують регулярної адаптації 45 інфраструктурних рішень для забезпечення максимальної сумісності та ефективності. Оскільки структура IaC вимагає створення окремих середовищ для сприяння безперебійній розробці, тестуванню та розгортанню інфраструктури Moodle, зміни можуть бути ретельно перевірені в контрольованих налаштуваннях перед впровадженням у продукційне середовище. Це знижує ризик збоїв і забезпечує збереження цілісності робочої системи [23, 25]. Середовище розробки служить основним робочим простором для створення та модифікації плейбуків Ansible і команд Moosh. Він налаштовується таким чином, щоб точно імітувати виробниче середовище, дозволяючи розробникам створювати та вдосконалювати сценарії автоматизації в середовищі, яке відображає реальні умови. Ключові характеристики середовища розробки включають використання віртуальних машин або технологій контейнеризації, таких як Docker, щоб забезпечити ізольовані екземпляри, а також інтеграцію із системами контролю версій для керування змінами коду, полегшення співпраці та збереження історії змін. Середовище тестування, зі свого боку, призначене для перевірки функціональності та надійності сценаріїв IaC перед їх розгортанням. Це середовище реплікує виробничу установку, включалючи специфікації сервера, конфігурації мережі та налаштування Moodle. гарантуючи, що тести дають точні та релевантні результати. Середовище виробництва є власне тим середовищем, де Moodle активно використовується кінцевими користувачами. Забезпечення його стабільності та надійності має найвищий пріоритет, тому воно налаштовується для обробки пікових навантажень і забезпечення безперервної доступності за допомогою балансування навантаження, механізмів відновлення після збоїв і компонентів масштабованої інфраструктури. Захищеність середовища забезпечується шифруванням, контролем доступу та регулярними перевірками безпеки, 46 Щоб підтримувати стабільність і сприяти плавним переходам між середовищами розробки, тестування та виробництва, у структурі IaC використовуються конфігурації, що залежать від середовища. Плейбуки Ansible параметризовані за допомогою змінних і шаблонів, завдяки чому одні й ті самі базові сценарії можна адаптувати до різних середовищ без дублювання. 2.4. Методи збору й аналізу даних Оцінка технологічної реалізації проєкту вимагає системного підходу до збору та аналізу даних для забезпечення її ефективності та визначення областей для оптимізації. Використання точних показників та інструментів моніторингу дозволяє перевірити ефективність цих рішень і сприяти постійному вдосконаленню системи. Для комплексної оцінки впровадження IaC були використані наступні методи збору даних. 1. Моніторинг продуктивності за допомогою Prometheus і Grafana. Prometheus було розгорнуто для збору метрик у реальному часі щодо різних аспектів інфраструктури Moodle, включаючи використання центрального процесора та пам’яті сервера, пропускної спроможності мережі, продуктивності запитів до бази даних і

часу відповіді. Grafana використовувався для візуалізації цих показників за допомогою настроюваних інформаційних панелей, надаючи інтерфейс для моніторингу продуктивності системи з часом. У рамках реалізації моніторингу налаштовуються: – Prometheus – для збирання показників із вебсерверів, серверів баз даних та інших критичних компонентів з 15-секундними інтервалами; – інформаційні панелі Grafana – для відображення ключових показників ефективності (KPI), таких як середній час завантаження сторінки, кількість активних користувачів і рівень використання ресурсів; 47 – увімкнення сповіщень Prometheus, щоб своєчасно повідомляти адміністраторам про аномалії продуктивності. 2. Вимірювання часу розгортання через Jenkins CI/CD Pipeline. Jenkins інтегрований у структуру IaC для автоматизації процесу розгортання. Час розгортання ретельно реєструється для кожного виконання плейбуків Ansible, надаючи кількісні дані щодо підвищення ефективності, досягнутого завдяки автоматизації. Деталі реалізації включають: – створення конвеєру Jenkins для ініціювання виконання плейбуків Ansible після фіксації коду в репозиторії Git; – налаштування конвеєру для реєстрації часу початку та завершення кожного розгортання, обчислення загальної тривалості надання та налаштування середовища Moodle; – зберігання історії даних про розгортання для виявлення тенденція та вимірювання покращення швидкості розгортання протягом послідовних ітерацій. 3. Аналіз помилок і журналу за допомогою стека ELK (Elasticsearch, Logstash, Kibana). Стек ELK використовувався для агрегування, обробки та аналізу журналів, створених програмами Ansible, Moosh і Moodle. Цей комплексний аналіз журналу полегшує виявлення повторюваних помилок, проблем із конфігурацією та областей, які потребують вдосконалення скриптів IaC. Реалізація аналізу помилок включає використання таких інструментів: – Logstash – для збору журналів із плейбуків Ansible, команд Moosh та вебсерверів і серверів баз даних Moodle; – Elasticsearch – для індексації агрегованих журналів, щоб забезпечити ефективний пошук і отримання даних журналів; 48 – інформаційні панелі Kibana – для візуалізації частоти помилок, виявлення поширених точок збою та моніторингу загального стану розгортання Moodle. 4. Виявлення відхилень у конфігурації з Ansible Tower. Цей інструмент надає можливість перевіряти, чи конфігурації залишаються узгодженими з визначеним станом у плейбуках Ansible, висвітлюючи розбіжності, які потребують коригувальних дій. Реалізація цього методу полягає в налаштуванні Ansible Tower на регулярне виконання плейбуків у режимі перевірки, порівнюючи поточний стан інфраструктури з бажаним станом, визначеним у скриптах IaC. Будь-яке виявлене відхилення реєструється та візуалізується в інтерфейсі Ansible Tower, що дозволяє адміністраторам оперативно виправляти невідповідності конфігурації. Звіти про відхилення піддаються аналізу для виявлення закономірностей та вдосконалення скриптів IaC. Для оцінки результатів розгортання IaC було обрано наступні методи системного аналізу: 1. Статистичний аналіз за допомогою Python (Pandas і NumPy). Pandas використовується для організації та обробки зібраних даних, забезпечуючи ефективне очищення та підготовку даних до аналізу. NumPy сприяє числовим обчисленням середніх значень, стандартних відхилень та інших відповідних статистичних даних. Для оцінки тенденцій продуктивності та ефективності розгортання з часом проводиться аналіз часових рядів. 2. Порівняльний аналіз показників до та після розгортання IaC. Проводиться порівняльний аналіз для оцінки продуктивності та надійності інфраструктури Moodle до та після впровадження структури IaC. Ключові показники, такі як час розгортання, частота помилок та час безвідмовної роботи системи, порівнюються на обох 49 етапах розгортання. Візуалізації в Grafana та Kibana полегшують чітке порівняння тенденцій продуктивності та виявлення аномалій. Ці аналітичні методи надають емпіричні докази впливу структури IaC на операційну ефективність та надійність системи. Критерії оцінки ефективності та визначення проблемних ділянок. Для об’єктивної оцінки успішності впровадження IaC та визначення областей для вдосконалення були встановлені такі критерії: 1. Ефективність розгортання. Показник: скорочення часу розгортання від ручних процесів до автоматизованих збірників ігор Ansible. Індикатор успіху: значне зменшення середньої тривалості розгортання, виміряної в хвилинах або годинах. 2. Узгодженість конфігурації. Показник: частота відхилення конфігурації, виявлена за допомогою Ansible Tower. Індикатор успіху: мінімальні або нульові випадки відхилення конфігурації, що вказує на високу узгодженість між розгортаннями. 3. Продуктивність системи та час безвідмовної роботи. Показник: середній час відповіді, використання ресурсів сервера та відсоток безвідмовної роботи системи. Індикатор успіху: покращений час відповіді, оптимальне використання ресурсів і висока доступність системи (наприклад, час безвідмовної роботи 99,9%). 4. Частота виникнення помилок і усунення проблем. Показник: кількість помилок,

пов’язаних із розгортанням, і час, витрачений на їх вирішення. Індикатор успіху: зниження частоти помилок і швидший час вирішення, що вказує на більш надійне розгортання. Систематична оцінка за цими критеріями з використанням зазначених методів збору та аналізу даних забезпечує ретельну оцінку ефективності 50 структури IaC. У разі повного впровадження пропонованої методики, дослідження здатне підтвердити або спростувати переваги автоматизації в розгортанні Moodle, а також визначати практичні рекомендації для постійного вдосконалення, сприяючи тим самим розвитку більш стійкої та ефективної інфраструктури освітніх технологій. 51 Висновки до розділу 2 У розділі було розроблено комплексну методологію для реалізації інфрастуктури як коду (IaC) для Moodle за допомогою Ansible та Moosh, яка включає детальне дослідження процесів її проєктування, налаштування та оцінки. Використано підхід Design Science Research до проектування IaC-фреймворку для розробки й представлення модульної архітектури, що інтегрує загальні можливості автоматизації Ansible зі спеціалізованими функціями управління Moodle від Moosh. Детально розглянуто покрокову стратегію реалізації сценаріїв Ansible, з акцентом на використання ролей, шаблонів та плейбуків з метою забезпечення модульності, повторного використання та зручності технічного обслуговування. Описано налаштування окремих середовищ розробки, тестування та виробництва, кожне з яких служить для підтримки ітераційного розвитку, ретельного тестування та надійних розгортань. Для подальшої оцінки ефективності реалізації IaC в рамках дослідження були встановлені методи збору та аналізу даних, які забезпечуються інструментами Prometheus, Grafana, Jenkins та ELK-стеком для моніторингу показників продуктивності, аналізу логів та вимірювання ефективності розгортання. Були визначені критерії оцінки роботи інфраструктури, включаючи швидкість розгортання та узгодженість конфігурації. 52 РОЗДІЛ 3 ВПРОВАДЖЕННЯ ІНФРАСТУКТУРИ ЯК КОДУ ТА АНАЛІЗ ЇЇ ЕФЕКТИВНОСТІ 3.1. Практична реалізація IaC у Moodle та інтеграція Ansible і Moosh Інфраструктура IaC для Moodle була реалізована за допомогою Ansible шляхом інтеграції зі специфічними для Moodle завданнями управління через Moosh. Етап впровадження зосереджений на перетворенні розробленого фреймворку IaC у робочі скрипти Ansible з інтеграцією команд Moosh для конфігурації, специфічної для Moodle. У цьому розділі представлено детальний опис розроблених скриптів IaC, із акцентом на процесах налаштування серверів, встановлення програмного забезпечення, конфігурації Moodle та автоматизації робочих процесів розгортання. Скрипти Ansible, розроблені для системи, що досліджується, є модульними та організовані за допомогою рольової структури. Кожна роль інкапсулює певний аспект процесу розгортання. Роль Server Setup відповідає за підготовку інфраструктури, необхідної для розгортання Moodle. Це включає такі завдання, як встановлення необхідних системних залежностей, налаштування мережі та забезпечення належної ініціалізації цільових серверів. На рис. 3.1 показано реалізацію завдання, яке дозволяє переконатися, що Python, Git та інші базові інструменти встановлено на цільових серверах, що створює основу для подальшої автоматизації. 53 Рис. 3.1 – Роль налаштування сервера Роль Database Provisioning відповідає за підготовку та налаштування сервера бази даних для Moodle. Залежно від конфігурації, роль встановлює вибраний рушій бази даних (у даному випадку це MySQL). Це може включати встановлення необхідних пакетів, налаштування конфігураційних файлів та запуск служби бази даних. Роль створює саму базу даних, в якій зберігатимуться дані Moodle, а також користувача бази даних з необхідними правами доступу до бази даних Moodle. Це включає встановлення пароля для цього користувача та надання йому прав на виконання операцій CRUD. На рис. 3.2 показано завдання створення бази даних, в якому в рамках ролі Database Provisioning динамічно налаштовуються облікові дані та засоби контролю доступу, необхідні Moodle для безпечного підключення до бази даних. З міркувань конфіденційності реальний пароль на рисунку було замінено шаблоном. Рис. 3.2 – Створення бази даних Moodle Роль Moodle Installation призначена для автоматизації процесу встановлення Moodle. Вона забезпечує завантаження вихідного коду Moodle, підготовку файлових структур, а також базове налаштування середовища для подальшого використання платформи. Основним етапом є отримання останньої стабільної версії вихідного коду Moodle з офіційного сайту. Після завантаження архів із вихідними файлами розпаковується у вказаний цільовий каталог. В результаті серверна частина платформи підготовлюється до подальших етапів конфігурації. Типовим завданням у межах цієї ролі є використання модуля get_url для завантаження архіву з вихідним кодом Moodle. На рис. 3.3 показано завдання, 54 яке виконує завантаження файлу moodle-latest-401.tgz та зберігає його у каталозі /var/www/moodle.tgz на сервері. Рис. 3.3 – Встановлення Moodle з використанням

ролі Moodle Installation Після виконання цього завдання файли Moodle готові до розпакування та розміщення у серверному середовищі, що є ключовим етапом підготовки до розгортання системи. Роль Web Server Configuration відповідає за встановлення та конфігурацію вебсервера для обслуговування Moodle. Це включає створення конфігураційних файлів, налаштування віртуальних хостів та забезпечення продуктивності вебсервера. Для динамічного генерування конфігураційних файлів використовується шаблон Jinja2. Наведений на рис. 3.4 фрагмент коду включає налаштування віртуального хоста Apache за допомогою шаблону moodle.conf.j2, який копіюється до директорії /etc/apache2/sites- available/moodle.conf. Рис. 3.4 – Конфігурація вебсервера за допомогою ролі Web Server Configuration Коли інфраструктуру налаштовано, для автоматизації адміністративних завдань Moodle використовуються команди Moosh. 1. Керування користувачами. Під час виконання цієї команди, Moosh створює нового користувача з іменем

Цитирования: **0,01%** id: **13**

«Admin User»

та електронною адресою

Цитирования: **0,01%** id: **14**

«admin@mydomain.com»,

а також призначає йому роль адміністратора. Такий користувач зможе керувати 55 всіма аспектами сайту Moodle, включаючи створення нових користувачів, призначення ролей, управління курсами та ін. moosh user-create --firstname=Admin --lastname=User --email=admin@mydomain.com --role=admin Крім того, для Moosh було реалізовано інші команди для управління користувачами, такі як: – moosh user-delete – видаляє користувача з сайту Moodle; – moosh user-update – оновлює інформацію про користувача; – moosh user-list – відображає список усіх користувачів на сайті Moodle. 2. Створення курсу. Автоматизація створення курсів забезпечує викладачам доступ до готових шаблонів, які вони можуть використовувати для швидкого запуску навчальних програм. Це дозволяє зосередитися на розробці змісту, а не на технічних аспектах. Завдяки автоматизації, викладачі можуть створювати курси з урахуванням стандартних вимог, таких як структура модулів, налаштування оцінювання та інші параметри. moosh course-create --fullname=

Цитирования: **0,02%** id: **15**

"Вступ до програмування"

-- shortname=

Цитирования: **0,01%** id: **16**

"CS101"

--categoryid=1 --summary=

Цитирования: **0,04%** id: **17**

"Цей курс присвячений основам програмування та алгоритмів."

Детальний опис параметрів: – fullname: повна назва курсу, яка відображатиметься у системі; – shortname: коротка назва курсу, яка використовується для ідентифікації; – categoryid: ідентифікатор категорії, до якої належить курс; – summary: короткий опис курсу, який допомагає здобувачам освіти зрозуміти його зміст. 56 Таким чином, представлена команда ініціює створення курсу з назвою

Цитирования: **0,02%** id: **18**

«Вступ до програмування»

та короткою назвою

Цитирования: **0,01%** id: **19**

«CS101»,

розміщуючи його в категорії

Цитирования: **0,01%** id: **20**

«Комп'ютерні науки»,

що має ідентифікаційний номер 1. 3. Інсталяція плагінів. Керування плагінами прискорюється завдяки автоматизації їх встановлення та налаштування. Це дозволяє

уникнути надлишкових ручних дій і зменшує ймовірність виникнення помилок під час роботи з середовищем. Представлена команда вказує на встановлення плагіна assignsubmission_file (

Цитирования: **0,01%** id: **21**

«Завдання»

) як модуля (--type=mod). Замість ручного пошуку плагіна, завантаження файлів та виконання ручних кроків налаштування, ця єдина команда запускає автоматизований процес. moosh plugin-install --type=mod --plugin=assignsubmission_file У рамках проєкту було встановлено такі плагіни для розширення функціональності Moodle та відповідності цілям розгортання: – Completion Progress – забезпечує візуальний індикатор завершення курсу для здобувачів освіти; – Attendance – полегшує облік відвідуваності цифрових курсів у Moodle; – Quiz Analytics – розширює можливості звітності та аналізу Moodle для тестів, надаючи детальну інформацію про результати здобувачів освіти. – Group Choice – дозволяє здобувачам освіту самостійно записуватись у групи в межах курсу; – PDF Feedback – дає можливість викладачам анотувати та надавати зворотний зв'язок безпосередньо на завантажених PDF-файлах завдань; – Checklist – допомагає здобувачам освіти відстежувати завдання та цілі курсу у форматі чек-листа; 57 – H5P – інтегрує інтерактивний контент, такий як відео, тести та презентації, у курси Moodle; – Questionnaire – додає інструмент для збору зворотного зв'язку або проведення опитувань серед учасників курсу; – Custom Certificate – дозволяє адміністраторам створювати та видавати персоналізовані сертифікати про завершення курсу; – BigBlueButtonBN – інтегрує BigBlueButton, систему для вебконференцій, у Moodle для проведення онлайн-зустрічей і занять. Використаний підхід дозволяє усім необхідним плагінам бути встановленими однаково на різних розгортаннях, серверах чи середовищах. Це має значущість при збільшенні складності інфраструктури, де потрібна консистентність налаштувань, та сприяє створенню більш передбачуваного робочого середовища. 4. Оновлення конфігурацій. Для забезпечення однорідності та узгодженості на всьому сайті, налаштування конфігурації виконуються автоматично. Це дозволяє підтримувати стандартизовані параметри та спрощує управління великою інфраструктурою. Для налаштування конфігураційних параметрів сайту, зокрема для зміни його повної назви, використовується командний рядок. Так, наступна команда, використовуючи moosh, встановлює нову назву сайту: moosh config-set --name=sitefullname --value=

Цитирования: **0,02%** id: **22**

"My Moodle Site"

Зокрема, moosh config-set викликає функцію для встановлення значення конфігурації. При цьому вказується параметр конфігураційної змінної, яку потрібно змінити, в даному випадку це sitefullname, яка відповідає за повну назву сайту. Відповідно, параметр --value=

Цитирования: **0,02%** id: **23**

"My Moodle Site"

визначає нове значення, яке буде встановлено для зазначеної конфігураційної змінної. Процес розгортання об'єднує налаштування сервера, установку Moodle і конфігурацію в єдиний робочий процес. На рис. 3.5 показано вміст основного 58 плейбуку (site.yml), який координує виконання ролей і завдань у коректній послідовності. Рис. 3.5 – Плейбук site.xml Відповідно до рисунку, виконання робочого процесу з ініціалізації сервера. На цьому етапі плейбук ініціалізує цільові сервери, встановлює залежності та готує середовище для встановлення Moodle. Наступним кроком є налаштування бази даних. Роль Database Provisioning забезпечує готовність бази даних Moodle до використання, включаючи створення необхідних користувачів і дозволів. Далі, на етапі розгортання Moodle, вихідні файли Moodle завантажуються та налаштовуються, а вебсервер готується до обслуговування програми. Коли все готово, відбувається конфігурація Moodle через Moosh. Команди Moosh виконуються як завдання після розгортання, автоматизуючи специфічні налаштування та налаштування Moodle. Взаємодія між Ansible і Moosh утворює основу робочих процесів автоматизації в проєкті. У той час як Ansible виконує загальні завдання з налаштування і координування інфраструктури, Moosh доповнює це виконанням адміністративних команд Moodle. Ця взаємодія спрощується завдяки виконанню команд Moosh як плейбуків Ansible. Ці завдання виконуються на цільових вузлах, де встановлено Moodle, забезпечуючи застосування конфігурацій Moodle безпосередньо до робочого

середовища. Змінні Ansible передаються до команд Moosh динамічно, що дозволяє виконувати контекстно-залежне виконання. Так, налаштування конфігурації Moodle, відомості про користувача та параметри курсу, визначені у файлах інвентаризації Ansible, вводяться в команди Moosh під час виконання. 59 Ідемпотентна природа Ansible забезпечує, що команди Moosh виконуються лише за необхідності. Наприклад, такі завдання, як створення курсу або встановлення плагіна, пропускаються, якщо зазначений курс або плагін уже існує, уникаючи зайвих дій. Вбудовані механізми обробки помилок Ansible фіксують і реєструють проблеми, що виникають під час виконання команди Moosh. Координація завдань між Ansible і Moosh досягається за допомогою структурованих плейбуків, які визначають послідовність і залежності кожної операції. У даному проєкті застосовано такі ключові механізми координації: 1. Залежності завдань. Плейбуки Ansible визначають залежності між завданнями, щоб гарантувати, що компоненти інфраструктури підготовлені перед застосуванням специфічних конфігурацій Moodle. наприклад: – базу даних необхідно налаштувати перед встановленням Moodle; – перед виконанням команд Moosh необхідно встановити Moodle. 2. Паралельне виконання. У системі задіяні можливості паралелізму Ansible, які дозволяють одночасне виконання завдань на кількох серверах. 3. Модуляція на основі ролей. Ролі Ansible інкапсулюють пов’язані завдання, такі як налаштування сервера, надання бази даних і налаштування Moodle. Команди Moosh включені в ролі, пов’язані з Moodle, щоб завдання Moodle виконувалися у відповідному контексті. 4. Динамічне управління інвентарем. Файли інвентаризації Ansible містять відомості про цільові сервери та конфігурації Moodle. Ці деталі динамічно передаються до команд Moosh, що дозволяє налаштовувати виконання на основі середовища розгортання. 5. Ведення журналів і звітність. Ansible фіксує детальні журнали виконання команд Moosh, забезпечуючи видимість процесу автоматизації. Адміністратори можуть переглядати ці журнали, щоб виявити помилки, відстежувати прогрес і підтверджувати успішне завершення. 60 Після завершення процесу розгортання проводиться комплексна валідація, метою якої є підтвердження працездатності Moodle та відповідність всім визначеним вимогам. Валідація відбувається на трьох рівнях. 1. Тести на рівні програми. Використовуються запити Moosh та API, щоб перевірити існування та функціональність ключових компонентів Moodle, таких як курси, користувачі та плагіни. 2. Послідовність конфігурації. Порівнюються розгорнуті конфігурації з бажаним станом, визначеним у плейбуках Ansible за допомогою ansible-playbook --check та можливостей виявлення дрейфу Ansible Tower. 3. Тести вебдоступності. Виконуються автоматичні тести HTTP, щоб переконатися, що сайт Moodle доступний (див. рис. 3.6), використовуючи такі інструменти, як curl та Selenium для перевірок на основі браузера. Рис. 3.6 – Перевірка доступності сайту Moodle Механізм валідації інтегровано в автоматизований робочий процес фреймворку IaC, що забезпечує виконання завдань валідації паралельно з операціями розгортання. Такий підхід надає безперервний зворотний зв’язок і запобігає поширенню некоректних конфігурацій у системі. У кожному плейбуці Ansible реалізовано завдання з валідації, які виконуються після ключових операцій для перевірки коректності виконання. Валідація також інтегрована в конвеєри CI/CD. Для кожного коміту Jenkins 61 запускає виконання плейбуків у режимі --check, що дозволяє симулювати зміни та перевіряти конфігурації без реальних модифікацій. У разі помилок валідації результати реєструються в Ansible і передаються до централізованих інструменту логування Elasticsearch (через Logstash). Аналіз цих логів виконується через Kibana-дашборди для діагностики та усунення проблем. Результати валідації відображаються в реальному часі через консольний вивід Ansible та CI/CD-дашборди. Такий механізм зворотного зв’язку дозволяє виявляти й усувати проблеми ще до їхнього впливу на продукційне середовище. 3.2. Тестування розробленого фреймворку IaC Реалізація IaC для Moodle вимагає застосування методологій тестування для перевірки показників продуктивності, безпеки автоматизованого процесу розгортання та забезпечення відповідності розробленого фреймворку IaC функціональним і нефункціональним вимогам, викладеним у попередніх розділах. Тестування у фреймворку IaC охоплює перевірку налаштувань інфраструктури, конфігурацій, специфічних для Moodle, а також загальної продуктивності системи [29, 38]. Для забезпечення всебічного охоплення були застосовані такі методології: 1. Модульне тестування. Забезпечує коректність виконання окремих завдань у плейбуках і командах Moosh. Кожне завдання Ansible, наприклад встановлення пакета чи налаштування служби, виконується ізольовано за допомогою ansible-playbook --check. Команди Moosh тестуються на автономному екземплярі Moodle, щоб підтвердити їх функціональність, наприклад створення користувачів або встановлення плагінів. Використані інструменти: – режим Ansible --check

для імітації виконання завдання; 62 – Moosh CLI для тестування команд Moodle. На рис. 3.7 зокрема показано впровадження перевірки, чи запущений сервіс Apache. 2. Інтеграційне тестування. Підтверджує взаємодію між компонентами інфраструктури та специфічними конфігураціями Moodle. Рис. 3.7 – Валідація запуску Apache після інсталяції Відповідно до інтеграційного тестування, плейбуки Ansible виконуються наскрізно для перевірки підключення та функціональності залежних компонентів. Команди Moosh виконуються після створення інфраструктури для забезпечення безперебійної інтеграції. На рис. 3.8. показано приклад перевірки того, чи Moodle вдалося підключитися до БД. Рис. 3.8 – Перевірка підключення Moodle до бази даних 3. Системне тестування. Оцінює всю інфраструктуру IaC, щоб переконатися, що інфраструктура Moodle функціонує коректно як єдине ціле. Першим кроком системного тестування Moodle, розгорнутої за допомогою IaC, є розгортання повної інфраструктури Moodle у контрольованому тестовому середовищі. Це середовище максимально точно відтворює продуктивне середовище, але ізольоване від нього, щоб уникнути впливу на реальних користувачів. Розгортання здійснюється з використанням визначених IaC конфігурацій та скриптів. 63 Наступним етапом є валідація основних функціональних можливостей Moodle. Після розгортання інфраструктури перевіряється робота всіх ключових функцій Moodle. Це включає, але не обмежується: – аутентифікація та авторизація користувачів: перевірка можливості успішного входу в систему для різних типів користувачів (здобувачі освіти, викладачі, адміністратори) з відповідними правами доступу; – доступність курсів та навчальних матеріалів: переконатися, що користувачі можуть знаходити та отримувати доступ до призначених їм курсів, а також до всіх пов'язаних з ними матеріалів (лекцій, завдань, форумів тощо); – функціональність плагінів: якщо використовуються додаткові плагіни Moodle, необхідно перевірити їхню коректну роботу та інтеграцію з основною системою, це включає тестування функціональності специфічних плагінів, таких як плагіни для оцінювання, проведення опитувань, інтеграції з зовнішніми сервісами тощо; – інші функції: робота системи сповіщень, календарів, інструментів комунікації. Проведення наскрізних тестів розгортання проводиться з використанням Jenkins для автоматизації виконання плейбуків Ansible та завдань перевірки. Цей етап включає автоматизоване виконання всього процесу розгортання інфраструктури з нуля; Jenkins запускає Ansible playbooks для конфігурації серверів та розгортання Moodle. Після розгортання автоматично запускаються скрипти та завдання для перевірки працездатності системи. Це дозволяє гарантувати, що процес розгортання є відтворюваним та надійним. Приклад тестового сценарію: перевірка аутентифікації та доступу до курсу. 64 Мета: підтвердити, що користувачі різних ролей (наприклад, здобувач вищої освіти) можуть успішно увійти в систему Moodle та отримати доступ до призначеного їм курсу. Кроки: 1) запуск автоматизованого сценарію Selenium: сценарій відкриває браузер та переходить на сторінку входу Moodle; 2) введення облікових даних користувача: сценарій вводить логін та пароль тестового користувача; 3) натискання кнопки

Цитирования: **0,01%** id: **24**

«Увійти»

(сценарій імітує натискання кнопки входу); 4) перевірка успішної аутентифікації: сценарій перевіряє, чи відбувся перехід на головну сторінку користувача або іншу сторінку, яка свідчить про успішний вхід; 5) навігація до списку курсів: сценарій переходить до розділу

Цитирования: **0,01%** id: **25**

«Мої курси»

та аналогічних; 6) перевірка наявності призначеного курсу: сценарій перевіряє, чи відображається у списку курс, призначений тестовому користувачу; 7) перехід до курсу: сценарій імітує клік по назві курсу для переходу на його сторінку; 8) перевірка доступності вмісту курсу: перевіряється наявність та доступність основних елементів курсу (розділів, матеріалів, завдань); 9) завершення сеансу (вихід із системи): імітація виходу користувача з системи; 10) звіт про результати: Selenium генерує звіт про проходження тесту, вказуючи на успішні та неуспішні кроки. Цей автоматизований тестовий сценарій дозволяє перевірити весь процес входу користувача в систему Moodle, його навігацію до призначеного курсу та доступ до вмісту курсу. Завдяки використанню Selenium імітуються дії реального користувача, такі як введення облікових даних, натискання кнопок та перехід між сторінками. Це дає змогу переконатися, що система 65 працює належним чином і користувачі можуть успішно отримати доступ до потрібних їм ресурсів.

Генерований звіт з чітким зазначенням успішних та неуспішних кроків надає інформацію для оперативного виявлення та усунення потенційних точок збою. 4. Тестування безпеки. Включає в себе ряд етапів, спрямованих на всебічну оцінку захищеності системи. Для автоматизованого виявлення потенційних проблем безпеки в інфраструктурі використовується OpenVAS, який перевіряє систему на наявність відомих вразливостей, слабких місць у конфігурації та інших потенційних ризиків. Важливим етапом є перевірка безпечних конфігурацій усіх компонентів інфраструктури на відповідність встановленим політикам безпеки та найкращим практикам [13]. Це охоплює такі аспекти: – примусове використання HTTPS – забезпечує шифрування даних, що передаються між користувачем та сервером, захищаючи їх від перехоплення; – захист доступу до бази даних – включає налаштування надійних паролів, обмеження доступу до бази даних лише з авторизованих IP- адрес, використання шифрування та регулярне резервне копіювання; – застосування принципу найменших привілеїв – користувачам та сервісам надаються лише ті права доступу, які є абсолютно необхідними для виконання їхніх функцій, що мінімізує потенційну шкоду у разі компрометації облікового запису; – налаштування брандмауера (firewall) – дозволяє контролювати вхідний та вихідний трафік, блокуючи небажані з'єднання. Ansible плейбуки в цьому контексті використовуються для автоматизації процесу налаштування та підтримки безпечних конфігурацій інфраструктури. За допомогою Ansible визначається бажаний стан системи та автоматично застосовуються необхідні зміни. 66 Приклад тестового сценарію: перевірка доступності сайту Moodle лише через HTTPS. Мета: переконатися, що доступ до сайту Moodle здійснюється виключно через захищений протокол HTTPS, а будь-які спроби з'єднання через HTTP автоматично перенаправляються на HTTPS. Кроки: 1) ввести адресу сайту Moodle у веббраузері, використовуючи протокол HTTP; 2) перевірити, чи відбувається автоматичне перенаправлення на HTTPS; 3) переконатися, що у рядку адреси браузера відображається іконка замка, що свідчить про безпечне з'єднання; 4) спробувати отримати доступ до будь-якої сторінки сайту Moodle, безпосередньо вказавши протокол HTTP у URL; 5) перевірити, чи знову відбувається автоматичне перенаправлення на HTTPS; Очікуваним результатом тесту є автоматичне перенаправлення всіх запитів, зроблених через HTTP, на відповідну сторінку через HTTPS. При цьому сайт Moodle повинен бути недоступний безпосередньо через HTTP, а браузер повинен відображати інформацію про безпечне HTTPS з'єднання. Результат проходження тесту: – пройдено – якщо всі очікувані результати виконуються. – не пройдено – якщо сайт доступний через HTTP без перенаправлення на HTTPS або якщо перенаправлення налаштовано некоректно. 5. Регресивне тестування. Тестування на регресію перевіряє, що оновлення скриптів інфраструктури як коду (IaC) не порушують існуючі функціональні можливості. Методологія регресивного тестування в цьому проєкті полягає в повторному виконанні всіх кроків перевірки та тестування після внесення змін до плейбуків Ansible або команд Moosh (дозволяє переконатися, що нові зміни не вплинули на коректність роботи системи). Далі 67 відбувається інтеграція тестів на регресію в Jenkins CI/CD конвеєр для забезпечення безперервної перевірки і, відповідно, оперативному виявленню точок збою та підвищенню надійності інфраструктури. Використані інструменти: 1. Jenkins – дозволяє налаштувати регулярне запускання тестів на регресію, що забезпечує постійний контроль за станом системи. 2. Git – забезпечує відстеження історії змін у коді, а також можливість швидко повертатися до попередніх версій. Методології тестування, які використовуються в цій реалізації IaC, забезпечують належну оцінку надійності та безпеки інфраструктури. Поєднуючи модулі, інтеграцію, систему, безпеку та регресійне тестування, структура розглядає кожен критичний аспект розгортання Moodle. Ці методології забезпечують основу для виявлення та вирішення потенційних проблем. Результати цих спроб тестування аналізуються в наступній главі, щоби підвести підсумки розробки проєкту та виділити області для подальшого вдосконалення. 3.3. Оцінка продуктивності системи Оцінка ефективності впровадження інфраструктури як коду (IaC) має вирішальне значення для розуміння її впливу на операційну ефективність, надійність і масштабованість. Для комплексної оцінки було застосовано систематичний підхід, що поєднує моніторинг у реальному часі, аналіз часу розгортання та відстеження помилок. Цей підхід дозволяє отримати емпіричні дані, які підтверджують ефективність обраної моделі автоматизації, а також виявити потенційні області для подальшої оптимізації. Методологія оцінювання базується на використанні точних метрик і показників, зокрема часу розгортання, кількості помилок, що виникли під час виконання сценаріїв, та продуктивності системи під навантаженням. 68 На основі обраних методів збору й аналізу даних для систематичного представлення результатів і висновків було проведено оцінку продуктивності системи за

низкою показників. У табл. 3.1 наведено детальний аналіз часу розгортання, зафіксованого під час послідовних ітерацій структури Moodle IaC. Процес розгортання включає налаштування сервера, установку Moodle, конфігурацію за допомогою плейбуків Ansible і специфічні для Moodle завдання, що виконуються за допомогою команд Moosh. Таблиця 3.1 – Аналіз часу розгортання інфраструктури як коду Номер Час Час Загальний Покращення Спостереження ітерації початку закінчення час порівняно з розгортання, попередньою хв ітерацією, % 1 10:00 10:45 45 N/A Початкове розгортання; для вирішення помилок скриптів були потрібні ручні втручання. 2 11:00 11:38 38 15.56% Покращена логіка сценарію зменшила надмірність завдань; незначні затримки в установці плагінів. 3 12:00 12:33 33 13.16% Автоматизовані кроки валідації оптимізували розгортання; виправлено проблеми з ініціалізацією бази даних. 4 13:00 13:29 29 12.12% Оптимізований рендеринг шаблонів та послідовність завдань; ручні втручання не знадобилися. 5 14:00 14:25 25 13.79% Остаточна ітерація досягла майже оптимальної продуктивності; 69 розгортання повністю автоматизоване. За підсумками аналізу вдалося встановити, як ітеративний підхід до оптимізації IaC скриптів може покращити час розгортання та загальну ефективність процесу. Так, перше розгортання зайняло 45 хвилин, що відображає меншу ефективність початкових скриптів IaC. Проблеми включали зайві завдання та помилки в конфігураційних файлах, які вимагали ручного виправлення. Удосконалення скриптів Ansible на другому етапі, а саме усунення зайвих завдань та вирішення логічних помилок, скоротили час розгортання до 38 хвилин — на 15,56% швидше. Досягненню цього покращення також сприяло додавання автоматичного виявлення помилок. Третій та четвертий етапи показали стале покращення часу розгортання, досягнувши відповідно 33 та 29 хвилин. Ключові оптимізації включали уточнення послідовності завдань, поліпшення завдань ініціалізації бази даних та автоматизацію валідації команд Moosh. Остаточне розгортання досягло загального часу 25 хвилин, що становить кумулятивне покращення на 44,44% порівняно з початковим етапом. Автоматизація була повністю впроваджена, і всі ручні втручання були усунені, що демонструє ефективність IaC фреймворку. Під час впровадження інфраструктури як коду важливим етапом є перевірка узгодженості конфігурації системи з бажаним станом. У наведеній нижче таблиці підсумовано показники узгодженості конфігурації, записані на етапі тестування реалізації IaC. Ці показники відстежують частоту та типи зміщень конфігурації, виявлених Ansible Tower під час автоматизованих перевірок, а також час вирішення та спостережувані шаблони. Зміщення конфігурації стосується розбіжностей між фактичним станом інфраструктури та бажаним станом, визначеним у сценаріях IaC. Таблиця 3.2 – Метрики зміщення конфігурації 70 Номер Кількість Компонент Час Частота Спостереження ітерації виявлених під питанням вирішення повторних зміщень (хвилини) зміщень 1 5 Конфігурація 20 Висока Початкові скрипти вебсервера мали неточності; (Apache), знадобилися ручні облікові дані виправлення. бази даних 2 3 Облікові дані 15 Помірна Проблему дрейфу бази даних, бази даних налаштування вирішено; Moodle виявлено незначну невідповідність у config.php Moodle. Продовження табл. 3.2 Номер Кількість Компонент Час Частота Спостереження ітерації виявлених під питанням вирішення повторних зміщень (хвилини) зміщень 3 2 Налаштування 10 Низька Зміщення Moodle спричинене неправильним налаштуванням плагіна; впроваджено автоматизоване рішення. 4 1 Конфігурація 5 Дуже Покращена логіка вебсервера низька шаблонів мінімізувала дрейфи; вирішено без ручного втручання. 5 0 Відсутні 0 Відсутня Зміщення не виявлені; скрипти IaC повністю відповідають очікуваному стану. Під час вимірювання метрик було здійснено кілька ітерацій перевірки та вдосконалення конфігурації системи з метою досягнення стабільної та передбачуваної конфігурації системи шляхом зменшення зміщення (дрейфу) конфігурації. Перша ітерація виявила значне зміщення конфігурації, із п'ятьма розбіжностями, що головним чином вплинули на конфігурацію вебсервера та бази даних. Ці зміщення були спричинені неповною або несистематичною 71 логікою шаблонів у книгах Ansible. Вирішення цих питань вимагало ручних виправлень, що призвело до середнього часу вирішення проблеми в 20 хвилин. Завдяки ітеративним вдосконаленням частота виявлених зміщень поступово зменшувалася. До другої ітерації їхня кількість зменшилася до трьох, із покращенням конфігурації бази даних, що зменшило повторювані проблеми. Налаштування Moodle, такі як неправильні URL-адреси сайтів у файлі config.php, становили решту розбіжностей. До третьої ітерації механізми автоматичного вирішення, такі як повторне застосування ролей Ansible та перевірка виходів команд Moosh, ефективно зменшили дрейфи до двох. Час вирішення проблеми був зменшений до 10 хвилин у середньому, що відображає зростаючу ефективність системи. Четверта ітерація виявила лише одне зміщення, спричинене

невеликим розходженням у конфігурації вебсервера. Це було вирішено протягом п'яти хвилин шляхом автоматичних виправлень. Врешті-решт, під час останньої ітерації не було виявлено жодного зміщення конфігурації. Цей етап відображає зрілість та точність системи IaC, що вказує на те, що розгорнуті конфігурації послідовно відповідали бажаному стану. Таким чином, ітераційне зменшення зміщення конфігурації демонструє ефективність ітеративного тестування та вдосконалення сценаріїв. Введення механізмів автоматичного вирішення значно зменшило як частоту, так і вплив зміщень. Хоча під час останньої ітерації не було виявлено жодного дрейфу, постійний моніторинг є необхідним для підтримки стабільності системи. Після конфігурації інфраструктури увага була приділена аналізу продуктивності середовища Moodle під різним навантаженням. Ключові показники відстежувалися за допомогою Prometheus та візуалізувалися на дашбордах Grafana. У табл. 3.3 представлено результати вимірювання цих показників, які включають середній час відповіді, максимальне використання процесора та пам’яті, а також частоту помилок, що спостерігаються під час навантажувальних тестів. Кожен сценарій навантаження імітує одночасну 72 діяльність користувача, починаючи від низького до інтенсивного використання. Таблиця 3.3 – Продуктивність системи за різних умов навантаження Кількість Середній Максимальне Максимальне Рівень одночасних час завантаження завантаження помилок користувачів відгуку ЦП пам’яті 50 120 мс 25% 40% 0% 100 180 мс 40% 60% 0% 200 250 мс 65% 75% 2% 500 400 мс 85% 90% 5% 1000 650 мс 95% 95% 10% Під час аналізу даних було виявлено, що при низькому навантаженні, а саме до 50 користувачів, система демонструвала оптимальну продуктивність. Середній час відповіді становив 120 мс, а використання ресурсів було мінімальним. Усі завдання, включаючи запити до бази даних та доставку контенту, виконувалися безперебійно та без жодних помилок. Збільшення навантаження до 100 одночасних користувачів призвело до незначного зростання часу відповіді до 180 мс. При цьому використання процесора та пам'яті залишалося в межах допустимих значень. Коли кількість користувачів сягнула 200, час відповіді збільшився до 250 мс, а також спостерігалося незначне зростання частоти помилок до 2%. Ці помилки були пов'язані з окремими плагінами і були вирішені шляхом подальшої оптимізації скриптів. При високому навантаженні в 500 користувачів система почала демонструвати ознаки обмеження ресурсів. Час відповіді значно зріс до 400 мс, а використання процесора досягло пікового значення у 85%. Частота помилок, в основному пов'язаних з інтенсивними операціями з базою даних, зросла до 5%. Цей сценарій вказав на необхідність покращеної оптимізації бази даних та потенційного масштабування ресурсів. В умовах екстремального навантаження, коли кількість користувачів досягла 1000, система вийшла на межі своїх можливостей. Середній час відповіді становив 650 мс, а використання процесора та пам'яті наближалося 73 до максимальної потужності. Частота помилок зросла до 10%, що свідчило про труднощі у підтримці стабільності при інтенсивному використанні. Цей сценарій підкреслив необхідність впровадження механізмів автоматичного масштабування та подальшої оптимізації обробки запитів. Основні висновки з проведеного аналізу полягають у тому, що система ефективно масштабується до 200 одночасних користувачів з мінімальним впливом на продуктивність. Однак при більшому навантаженні виявляються вузькі місця, зокрема в операціях з базою даних та розподілі ресурсів, що потребує цілеспрямованої оптимізації. Пікове використання процесора та пам'яті при високому та екстремальному навантаженнях підкреслює важливість моніторингу ресурсів та потенційні переваги динамічного масштабування інфраструктури для задоволення попиту. Для обробки екстремальних навантажень у майбутньому будуть пріоритетними конфігурації автоматичного масштабування та оптимізація запитів до бази даних. Крім того, буде переглянута продуктивність плагінів, щоб зменшити кількість помилок за умов високого навантаження. Оцінка продуктивності за різних умов навантаження дала розуміння масштабованості системи та висвітлила сценарії, у яких виникали вузькі місця та помилки. Ці висновки підкреслюють важливість розуміння помилок, пов’язаних із розгортанням, їхніх основних причин і застосованих стратегій усунення. Систематично відстежуючи та аналізуючи ці помилки, можна вдосконалити структуру IaC для підвищення надійності та мінімізації збоїв під час майбутніх розгортань. У таблиці 3.4 підсумовано помилки, пов’язані з розгортанням, які виникли під час різних ітерацій, та класифіковано проблеми з визначенням їх основних причин та документацією часу, витраченого на їх вирішення. Таблиця 3.4 – Найчастіші помилки розгортання та їх усунення Тип помилки Компонент Основна Час Підхід до причина розв’язання вирішення Синтаксична Плейбук Неправильний 10 хв Виправлено помилка Ansible відступ YAML синтаксис і скрипта повторно 74 запущено плейбук Помилка

Плагін Moodle Відсутність 20 хв Встановлено встановлення залежності під час залежність і плагіна встановлення повторно плагіна запущено команду Moosh Помилка Конфігурація Неправильні 15 хв Оновлено з’єднання з бази даних облікові дані у облікові дані у базою даних файлі змінних файлі змінних Ansible Проблеми з Каталог даних Неправильні 12 хв Змінено права правами Moodle права власності та доступу через доступу доступу в завдання Ansible плейбуці Продовження таблиці 3.4 Помилка Команда Невірно вказаний 10 хв Виправлено ID створення Moosh ID категорії курсу у категорії у курсу команді змінних плейбука Ansible Таймаут Завдання Надмірний час 15 хв Оптимізовано скрипта валідації виконання через логіку скрипта валідації розгортання неефективну логіку валідації На початку розробки виникали помилки, пов'язані переважно з синтаксисом скриптів та відсутніми залежностями. Ці помилки були характерними для початкового етапу розробки та усунені шляхом ручних виправлень і вдосконалення скриптів. Подальша робота повністю виключила такі помилки. Проблема з підключенням до бази даних виявила недогляд у керуванні змінними. Це питання було вирішене оновленням файлу змінних Ansible, і більше помилок, пов’язаних з базою даних, не виникало. Помилки з правами доступу до файлів у каталозі даних Moodle підкреслили необхідність більш суворого керування правами. Запровадження завдання Ansible для забезпечення коректних прав доступу вирішило цю проблему і запобігло її повторенню. Команди Moosh час від часу стикалися з проблемами, такими як недійсні параметри (наприклад, некоректні ідентифікатори категорій курсів). Ці 75 помилки були усунені шляхом вдосконалення плейбуків Ansible для перевірки змінних перед виконанням команд. Тайм-аут скрипта валідації підкреслив важливість ефективних процесів перевірки. Оптимізація логіки скрипта скоротила час перевірки та запобігла виникненню подібних проблем у майбутніх розгортаннях. На ранніх етапах помилки були переважно пов’язані з розробкою скриптів та керуванням змінними, тоді як у подальшому виявилися можливості для оптимізації у валідації та виконанні завдань. Час вирішення помилок значно скоротився, оскільки були виявлені закономірності помилок та вжиті заходи з цільового вдосконалення скриптів та покращення автоматизації. Для подальшої кількісної оцінки впливу фреймворку IaC було проведено порівняльний аналіз ключових показників ефективності (KPI) до та після її впровадження. Це порівняння оцінює важливі показники, такі як час розгортання, частота помилок та час безвідмовної роботи системи. У таблиці 3.5 представлено паралельне порівняння ключових показників, щоб наочно показати покращення, досягнуті шляхом переходу від ручного керування інфраструктурою до автоматизованого за допомогою Ansible і Moosh. Таблиця 3.5 – Порівняння ключових показників ефективності (KPI) до та після впровадження IaC Метрика Значення Значення Покращення Спостереження до IaC після IaC Середній час 120 хвилин 25 хвилин 79.17% Значне скорочення часу розгортання розгортання завдяки автоматизованим плейбукам і послідовності завдань. Частота 6 помилок 1 помилка 83.33% Автоматизація мінімізувала помилок за людські помилки; більшість цикл проблем виникли під час початкового налаштування. Рівень 25% 0% 25% Відхилення усунуто зміщення розгортань завдяки постійній валідації конфігурації та автоматизованим корекціям. 76 Час 98% 99.9% 1.93% Покращена стабільність безвідмовної завдяки узгодженим роботи розгортанням і механізмам системи проактивного виявлення помилок. Порівняння підкреслює значні покращення в ефективності та відмовостійкості, досягнуті завдяки фреймворку IaC. У довгостроковій перспективі, усунення розбіжностей у конфігурації та зменшення частоти помилок свідчать про те, що фреймворк IaC забезпечує основу для сталого, масштабованого та надійного управління інфраструктурою. Хоча покращення є суттєвими, існують можливості для подальшого розвитку, а саме оптимізація процесів валідації та вивчення розширених механізмів масштабування. 3.4. Рекомендації із застосування фрейморку та мінімізації ризиків Впровадження інфраструктури як коду для розгортання та керування Moodle представляє низку переваг, утім організації, які планують впровадження IaC для Moodle, повинні почати з чіткого визначення цілей і вимог. Розуміння конкретних потреб їх розгортання LMS, таких як масштаб активності користувачів, інтеграція із зовнішніми системами та рівні налаштування, є критичним для розробки ефективної структури IaC. Ранні інвестиції в надійне планування та документацію спростять розробку та впровадження. Одним із найкритичніших кроків є вибір інструментів і технологій, а також побудова архітектури проєкту. Модульна конструкція сценаріїв IaC є важливою для зручності обслуговування та масштабованості. Розбиття завдань на повторно використовувані ролі та шаблони, як показано в цьому проєкті, надає можливість послідовного застосування змін із мінімальними збоями. Організаціям слід надати пріоритет створенню ретельно структурованих посібників і дотримання найліпших

практик, таких як послідовне керування змінними та надійна обробка помилок. 77 Завдяки інтеграції механізмів перевірки безпосередньо в робочі процеси розгортання організації можуть завчасно виявляти та усувати помилки, зменшуючи ризик збоїв у виробничих середовищах. Автоматизовані конвеєри тестування, такі як ті, що реалізовані за допомогою Jenkins, слід розглядати як можливість надання зворотного зв'язку в реальному часі під час циклів розробки та розгортання. Організаціям, які прагнуть масштабувати свої середовища Moodle, рекомендується впровадження інструментів моніторингу та оцінки ефективності, таких як Prometheus і Grafana. Ці інструменти надають практичну інформацію про продуктивність системи за різних навантажень, дозволяючи адміністраторам завчасно вирішувати вузькі місця ресурсів і оптимізувати конфігурації. Незважаючи на переваги, розробка та впровадження IaC для Moodle не позбавлені ризиків. Одним з головних викликів є початкова крива навчання, пов'язана з Ansible і Moosh. Розвиток навичок роботи з цими інструментами вимагає виділення часу на тренування співробітників, особливо для завдань із розширеними функціями, такими як динамічні шаблони та власні сценарії перевірки. Зміщення конфігурації та невідповідності були повторюваними проблемами під час ранніх ітерацій. Хоча ці ризики врешті-решт було мінімізовано шляхом ретельного тестування та автоматизованих виправлень, вони підкреслили важливість впровадження надійних механізмів перевірки на початкових етапах розробки. Крім того, неправильне керування змінними та невідповідність параметрів у початкових плейбуках часто призводило до помилок, які вимагали ручного налагодження і, відповідно, збільшувало часові рамки розробки. Іншим викликом може стати оптимізація взаємодії між Ansible і Moosh. Хоча інтеграція є концептуально простою, практична реалізація виявила обмеження в обробці складних скриптів, таких як забезпечення 78 ідемпотентності для певних команд Moosh. Ці обмеження вимагали обхідних шляхів і додаткових сценаріїв, що додало складності системі автоматизації. Утім, прийняття IaC для Moodle та подібних платформ LMS пропонує трансформаційний потенціал, переваг його можливо досягти за умови ретельного планування, ітераційної розробки та безперервного вдосконалення. Дотримуючись рекомендацій, вище, і активно вирішуючи проблеми, що виникають під час цього проекту, організації можуть максимізувати переваги IaC і створити міцну базу для модерного управління інфраструктурою. 79 Висновки до розділу 3 У розділі описано впровадження інфраструктури як коду для Moodle із застосуванням інструментів автоматизації Ansible і Moosh. Покроковий опис розроблених IaC-скриптів висвітлив модульну структуру та логічний потік завдань, демонструючи здатність фреймворку обробляти налаштування серверів, встановлення програмного забезпечення та специфічні для Moodle конфігурації. Особливий акцент було зроблено на безшовній взаємодії між Ansible та Moosh, де специфічні для Moodle команди динамічно виконувалися як частина ширшого робочого процесу автоматизації. Фаза оцінки охоплювала методології тестування, включаючи модульне, інтеграційне, системне, продуктивне та безпекове тестування. Метрики продуктивності, зібрані за допомогою Prometheus та візуалізовані в Grafana, надали інформацію про масштабованість системи та її чутливість за різних умов навантаження. Помилки, пов'язані з розгортанням, та відхилення конфігурації систематично відстежувалися та усувалися, що значно покращило швидкість розгортання, зменшило рівень помилок та підвищило узгодженість конфігурації. Порівняльний аналіз ключових показників ефективності до та після впровадження IaC підкреслив вплив фреймворку зі значними вимірюваними досягненнями в ефективності та часі безвідмовної роботи системи. Запропоновано практичні рекомендації та стратегії подолання ризиків для організацій, які прагнуть впровадити IaC для Moodle. Наголошено на важливості модульного дизайну скриптів, тестування, постійної перевірки та організаційної готовності. 80 ВИСНОВКИ Інтеграція освітніх технологій та управління IT-інфраструктурою набуває значущості в сучасному навчальному середовищі. Системи управління навчанням (LMS) слугують основою цих середовищ, і управління базовою інфраструктурою цих платформ створює унікальні виклики, особливо щодо забезпечення послідовних розгортань, мінімізації простоїв та масштабування ресурсів для задоволення динамічних потреб. Автоматизація за допомогою інфраструктури як коду (IaC) пропонує трансформаційний підхід, оптимізуючи управління інфраструктурою та одночасно підвищуючи надійність та адаптивність до мінливих освітніх потреб. У рамках цієї дисертації було здійснено розробку, впровадження та аналіз ефективності інфраструктури як коду для системи управління навчанням Moodle з використанням Ansible і Moosh. Цілі дослідження досягалися систематично шляхом ретельного огляду літератури, розробки

структурованої методології та детального процесу впровадження. Завдяки інтеграції автоматизації загального призначення з адміністративними інструментами Moodle було наочно показано, як автоматизація може оптимізувати процеси розгортання, підвищити узгодженість і зменшити ручне втручання. Огляд літератури підкреслив зростаючу залежність від автоматизації в управлінні IT-інфраструктурою та її потенціал для вирішення проблем у розгортанні LMS. Методологія представила модульні сценарії IaC для багаторазового використання, які інтегрували надання інфраструктури та специфічні конфігурації Moodle, перевірені за допомогою комплексних методологій тестування. На етапі впровадження вдалося виявити значні покращення в часі розгортання, частоті помилок і узгодженості конфігурації, що підтверджено оцінками продуктивності за допомогою Prometheus і Grafana. Таким чином, дисертація робить внесок у сферу освіти та галузь управління IT-інфраструктурою, зокрема в таких ключових аспектах: 81 1. Покращення розуміння того, як методології IaC можуть сприяти розгортанню та керуванню платформами LMS. 2. Надання повної структури для інтеграції інструментів автоматизації загального призначення, таких як Ansible, з інструментами для специфічної області, такими як Moosh. 3. Надання емпіричних даних про ефективність IaC у покращенні ефективності розгортання та продуктивності системи. 4. Перевірена та підтверджена структура IaC, придатна для впровадження в установах, що прагнуть автоматизувати свої розгортання Moodle. 5. Надання практичних рекомендацій з переходу від ручного керування інфраструктурою до автоматизованого. 6. Демонстрація масштабованості й адаптивності фреймворку для інших платформ LMS. Завдяки автоматизації надання інфраструктури, встановлення програмного забезпечення та специфічних для Moodle конфігурацій, установи можуть скоротити час розгортання, мінімізувати помилки конфігурації та підтримувати узгодженість операцій у різних середовищах. Крім того, принципи та методології, застосовані в цій роботі, можна адаптувати до інших навчальних інструментів, таких як Blackboard, Canvas або Sakai, що забезпечує ширше застосування в освітніх технологіях. 82 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 1. Антоненко А., Жаріков Е. Практичне використання Ansible як інструменту реалізації підходу

Цитирования: **0,02%** id: **26**

«Інфраструктура як код».

2022. URL: https://ela.kpi.ua/server/api/core/bitstreams/aecbafd4-1df5-4ea6-84ad-613659459313/content (дата звернення: 01.04.2024). 2. Балієва А. Архітектура та особливості інструменту Ansible. IT Education Center Blog. URL: https://itedu.center/ua/blog/guides/ansible-capabilities- features-processes (дата звернення: 31.08.2024). 3. Безугла Г. Є., Хряпкін О. В. Розробка інфраструктури Web-компонентів інформаційної системи. The I International Science Conference on Multidisciplinary Research : Abstracts of I International Scientific and Practical Conference, м. Berlin, 19–21 січ. 2021 р. С. 1007–1012. 4. Берко А., Коблик І. Система керування процесом безперервної доставки програмного забезпечення. Information Systems and Networks. 2024. № 15. С. 238–251. URL: https://science.lpnu.ua/sites/default/files/journal- paper/2024/aug/35674/maket2402951-242-255_0.pdf (дата звернення: 06.06.2024). 5. Дослідження ефективності процесів CI/CD в гнучких технологіях розробки програмного забезпечення / А. Вивюрка та ін. Актуальні задачі сучасних технологій : матеріали XII Міжнар. науково-практ. конф. молодих уч. та студентів, м. Тернопіль, 6–7 груд. 2023 р. URL: https://elartu.tntu.edu.ua/bitstream/lib/43863/2/MNPK_2023_Vyviurka_A-Research_of_the_efficiency_402-403.pdf (дата звернення: 07.06.2024). 6. Киричек Г. Г., Щетінін М. О. Управління конфігурацією серверів на основі Ansible. Інформатика, обчислювальна техніка та автоматизація. 2022. С. 109–114. URL: https://doi.org/10.32838/2663- 5941/2022.1/18 (дата звернення: 03.05.2024). 7. Козачок В. О. Метод підвищення захищеності Docker-контейнерів : автореф. дис. д-ра філософії. Вінниця, 2019. URL: https://ir.lib.vntu.edu.ua 83 /bitstream/handle/123456789/38996/138881.pdf (дата звернення: 12.07.2024). 8. Кравченко С. М., Марчук Г. В. Використання методології DevOps в управлінні IT проектами. Вчені записки Таврійського національного університету імені ВІ Вернадського. Серія: Технічні науки. 2018. Т. 29 (68), № 4 (1). С. 175–179. 9. Мартинова О. В. Автоматизація CI/CD з Jenkins : автореф. дис. д-ра філософії. Вінниця, 2023. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/ 123456789/38234/16985.pdf (дата звернення: 03.07.2024). 10. Орлов М., Дмитрів Ю. Аналіз програмних інструментів для автоматизації функцій конфігурування і управління в IT інфраструктурах. Information Systems and Networks. 2024. № 15. С. 370– 388. URL: https://doi.org/10.23939/sisn2024.15.370 (дата

звернення: 01.08.2024). 11. Прокопенко М. Використання підходу інфраструктури як код. Доцільність підходу. Приклади застосування. Ефективність. Молодь: наука та інновації : матеріали 10-ої всеукр. наук.-техн. конф. студентів, аспірантів і молодих уч., м. Дніпро, 23–25 листоп. 2022 р. Дніпро, 2022. С. 368–369. URL: http://ir.nmu.org.ua/handle/123456789/162746 (дата звернення: 05.04.2024). 12. Радченко А. CI vs CD vs CD: різниця підходів і чому вони важливі. Спільнота програмістів | DOU. URL: https://dou.ua/forums/topic/47081/ (дата звернення: 03.07.2024). 13. Рудьковський О.Р., Киричек Г.Г. Програмний комплекс з підтримки розподіленої взаємодії мережевих пристроїв та додатків. Вчені записки ТНУ імені В.І. Вернадського. Серія

Цитирования: **0,01%** id: **27**

«Технічні науки»

. 2021. Вип. 32 (71). № 2. С. 229–234. 14. Сенківський В. І. Проблеми масштабування Agile-методів розробки програмних продуктів для великих організацій та проєктів : магістерська 84 дисертація. Тернопіль, 2022. URL: https://elartu.tntu.edu.ua/handle/ lib/44859 (дата звернення: 02.07.2024). 15. Щербина О. А. Автоматизація створення, наповнення і адміністрування категорій курсів сайту Moodle. Information Technologies and Learning Tools. 2023. Т. 93, № 1. С. 178–198. URL: https://doi.org/10.33407/itlt.v93i1.5117 (дата звернення: 06.03.2024). 16. Ansible Documentation. URL: https://docs.ansible.com/ (date of access: 02.03.2024). 17. Arachchi S., Perera I. Continuous integration and continuous delivery pipeline automation for agile software project management. 2018 Moratuwa Engineering Research Conference (MERCon), Moratuwa, 30 May – 1 June 2018. 2018. P. 156–161. 18. Arnold R. D., Wade J. P. A Definition of Systems Thinking: A Systems Approach. Procedia Computer Science. 2015. Vol. 44. P. 669–678. URL: https://doi.org/10.1016/j.procs.2015.03.050 (date of access: 04.05.2024). 19. Bass L., Weber I., Zhu L. DevOps: A Software Architect's Perspective. Addison-Wesley Longman, Incorporated, 2015. 352 p. 20. Bigelow S. J., Courtemanche M., Gillis A. S. What Is DevOps? Meaning, Methodology and Guide. Search IT Operations. URL: https://www.techtarget.com/searchitoperations/definition/DevOps (date of access: 04.04.2024). 21. Design Science in Information Systems Research / Hevner et al. MIS Quarterly. 2004. Vol. 28, no. 1. P. 75. URL: https://doi.org/10.2307/25148625 (date of access: 04.07.2024). 22. Design Science Research Evaluation / K. Peffers et al. Lecture Notes in Computer Science. Berlin, Heidelberg, 2012. P. 398–410. URL: https://doi.org/10.1007/978-3-642-29863-9_29 (date of access: 17.06.2024). 23. DevOps Handbook, Second Edition: How to Create World-Class Agility, Reliability, and Security in Technology Organizations / P. Debois et al. IT Revolution Press, 2021. 85 24. Dougiamas M., Taylor P. Moodle: Using learning communities to create an open source course management system. EdMedia+ innovate learning. 2003. P. 171–178. URL: https://www.learntechlib.org/primary/d/13739 (date of access: 03.03.2024). 25. Experiences with Hundreds of Similar and Customized Sites with DevOps / C. Rodríguez et al. 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, 15–17 December 2021. P. 1031–1036. URL: https://doi.org/10.1109/CSCI54926.2021.00014 (date of access: 13.06.2024). 26. Fowler M., Highsmith J. The agile manifesto. Software development. 2001. Vol. 9, no. 8. P. 28–35. 27. Gasser O., Holz R., Carle G. A deeper understanding of SSH: Results from Internet-wide scans. NOMS 2014 – 2014 IEEE/IFIP Network Operations and Management Symposium, Krakow, Poland, 5–9 May 2014. 2014. URL: https://doi.org/10.1109/noms.2014.6838249 (date of access: 05.05.2024). 28. Gregor S., Hevner A. R. Positioning and Presenting Design Science Research for Maximum Impact. MIS Quarterly. 2013. Vol. 37, no. 2. P. 337–355. URL: https://doi.org/10.25300/misq/2013/37.2.01 (date of access: 31.12.2024). 29. Hasan M. M., Bhuiyan F. A., Rahman A. Testing practices for infrastructure as code. ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual USA. New York, NY, USA, 2020. URL: https://doi.org/10.1145/3416504.3424334 (date of access: 04.05.2024). 30. Moodle 4 Administration: An Administrator's Guide to Configuring, Securing, Customizing, and Extending Moodle, 4th Edition. Packt Publishing, Limited, 2022. 640 p. 31. MoodleDocs. URL: https://docs.moodle.org/405/en/Main_page (date of access: 02.03.2024). 32. Moodle.org. Moodle. URL: https://moodle.org/ (date of access: 29.02.2024). 33. Moosh. URL: https://moosh-online.com/ (date of access: 04.03.2024). 86 34. Pittet S. Continuous integration vs. delivery vs. deployment. Atlassian. URL: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment (date of access: 05.05.2024). 35. Rastenis M. A study of bugs found in the Ansible configuration management system : Dissertation. 2022. URL: https://repository.tudelft.nl/file/ File_29ed7643-5d10-479e-8a90-54a114dbd7f9?preview=1 (date

of access: 30.06.2024). 36. Santoso B. J., Ijtihadie R. M., Millah Z. A Docker Container-Based Solution for Course Archival on Moodle: Implementation and Evaluation. 2023 8th International Conference on Electrical, Electronics and Information Engineering (ICEEIE), Malang City, 28–29 September 2023. 2023. P. 1–6. 37. Sokolowski D. Infrastructure as code for dynamic deployments. ESEC/FSE '22: 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering , Singapore Singapore. New York, NY, USA, 2022. URL: https://doi.org/10.1145/3540250.3558912 (date of access: 06.04.2024). 38. Sokolowski D., Spielmann D., Salvaneschi G. Automated Infrastructure as Code Program Testing. IEEE Transactions on Software Engineering. 2024. P. 1–15. URL: https://doi.org/10.1109/tse.2024.3393070 (date of access: 04.06.2024). 39. Strutynska O. V., Umryk M. A. Modern educational trends under the conditions of digital society development. Innovate Pedagogy. 2020. No. 26. P. 201–205. URL: https://doi.org/10.32843/2663-6085/2020/26.40 (date of access: 05.03.2024). 40. The Official YAML Web Site. URL: https://yaml.org/ (date of access: 01.04.2024). 41. What is infrastructure as code and why is it important? HashiCorp | The Infrastructure Cloud Company. URL: https://www.hashicorp.com/resources/ what-is-infrastructure-as-code (date of access: 22.03.2024). 87

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу.
Пожалуйста, следуйте инструкциям: Рекомендации по оценке

Детектор Плагиата - Ваше право на оригинальность! □ SkyLine LLC