КОНЦЕПЦІЇ ПРОЄКТУВАННЯ MLFLOW У КОНТЕЙНЕРИЗОВАНИХ ТА ХМАРНИХ СИСТЕМАХ

Євген Берщанський, аспірант, Галина Клим, д.т.н., проф.

Національний університет «Львівська політехніка», Україна
e-mails: yevhen.v.bershchanskyi@lpnu.ua,

Анотація

У цій статті представлено основні результати глибокого дослідження концепцій проєктування MLFlow у контейнеризованих та хмарно-орієнтованих системах. Дослідження зосереджується на тому, як MLFlow, як ключова MLOps-платформа, може бути ефективно розгорнута та керована у хмарно-нативному середовищі для забезпечення масштабованих, відтворюваних і безпечних робочих процесів машинного навчання. У роботі проаналізовано архітектурні принципи та шаблони інтеграції компонентів MLFlow Tracking Server, Projects, Models та Registry у розподілених контейнеризованих інфраструктурах. Результати демонструють, що використання хмарно-нативних інструментів і сервісів забезпечує динамічну оркестрацію, покращене використання ресурсів і розширену автоматизацію життєвого циклу моделей. Кейс-дослідження підтверджує, що запропонований підхід покращив масштабованість відстеження експериментів і зменшив складність розгортання, підвищивши стійкість і підтримуваність системи. Ці результати підкреслюють ефективність застосування принципів хмарно-нативного проєктування до MLFlow.

Ключові слова

MLFlow, MLOps, контейнеризація, хмарно-нативні системи, Kubernetes, життєвий цикл моделі, реєстр моделей.

1. Вступ

Операції машинного навчання сформувалися як ключова дисципліна в галузі штучного інтелекту, що дає змогу організаціям оптимізувати розроблення, розгортання та моніторинг моделей машинного навчання у складних середовищах. Із зростанням потреби у масштабованих та автоматизованих AI-системах такі фреймворки, як MLFlow, стають центральними інструментами для управління повним життєвим циклом моделей машинного навчання від відстеження експериментів і навчання моделей до їх розгортання та забезпечення контролю.

Однак традиційні реалізації MLFlow часто стикаються з труднощами у динамічних і розподілених інфраструктурах, де підтримка узгодженості, масштабованості та безпеки стає дедалі складнішою. Поява контейнеризації та хмарно-нативних технологій змінила цей ландшафт, відкривши нові архітектурні можливості для операціоналізації MLFlow у сучасних середовищах [1].

Контейнеризовані та хмарно-нативні системи забезпечують гнучкість і еластичність, необхідні для підтримки масштабних робочих навантажень машинного навчання, надаючи стандартизовані середовища, що гарантують відтворюваність і портативність моделей. У межах таких систем MLFlow виступає як фундаментальний шар, який об'єднує дата-сайєнтистів, інженерів DevOps та продакшн-середовища через єдину платформу для управління життєвим циклом моделей [2].

Завдяки розділенню компонентів, таких як Tracking Server, Model Registry та сховище артефактів, MLFlow підтримує модульні стратегії розгортання, що узгоджуються з принципами мікросервісної та розподіленої архітектури. Така модульність підвищує операційну ефективність, дозволяючи кожному компоненту масштабуватись незалежно та інтегруватись із зовнішніми сервісами такими як сховища даних, бази даних і конвеєри безперервної інтеграції.

Попри значний прогрес, впровадження MLFlow у контейнеризованих і хмарно-нативних екосистемах створює нові виклики, пов'язані з проєктуванням архітектури, оптимізацією продуктивності та управлінням ресурсами. Налаштування MLFlow для розподілених середовищ потребує ретельного врахування комунікацій між компонентами, надійності сховищ та узгодженості даних [3]. Додатково, забезпечення спостережуваності, відмовостійкості та відповідності корпоративним стандартам безпеки залишається ключовим аспектом при масштабному розгортанні MLFlow. Без структурованого підходу до проєктування ці складнощі можуть призвести до операційної неефективності, зниження продуктивності системи та труднощів із підтриманням відтворюваності в різних середовищах.

Організації часто змушені балансувати між використанням локальних ресурсів і хмарно-нативних сервісів, щоб задовольнити регуляторні, продуктивні або фінансові вимоги. У таких умовах досягнення узгодженості у відстеженні експериментів, версіонуванні артефактів і управлінні моделями стає серйозним викликом [4]. Інтеграція MLFlow у подібні розподілені інфраструктури вимагає не лише технологічної адаптації, але й чіткого розуміння архітектурних принципів, що підтримують взаємодію, масштабованість і стійкість системи. Саме тому дослідження концепцій проєктування MLFlow набуває особливого значення, оскільки воно спрямоване на створення систематичного підходу до побудови гнучких, підтримуваних і готових до продакшн-рівня MLOps-рішень у хмарно-нативному контексті.

У цій статті розглядаються концепції проєктування MLFlow у контейнеризованих та хмарно-нативних системах з акцентом на архітектурну інтеграцію, оптимізацію розгортання та найкращі операційні практики. Досліджується, як MLFlow може бути адаптовано для використання переваг хмарно-нативних можливостей, зберігаючи при цьому гнучкість і контроль над робочими процесами машинного навчання. На основі аналізу реальних кейсів оцінюється ефективність, масштабованість і підтримуваність різних моделей розгортання MLFlow, що надає практичні рекомендації для створення надійних і готових до продакшн екосистем MLOps.

2. Недоліки

Хоча MLFlow надає всеосяжний і розширюваний фреймворк для управління повним життєвим циклом моделей машинного навчання, його розгортання та експлуатація в контейнеризованих і хмарно-нативних системах мають низку обмежень, що можуть впливати на ефективність і масштабованість. Одним із ключових викликів є архітектура платформи, спочатку розроблена для відносно статичних і монолітних середовищ. Під час перенесення у розподілені інфраструктури MLFlow часто потребує суттєвого налаштування, щоб забезпечити стабільну взаємодію між компонентами, такими як Tracking Server, бекенд-база даних і сховище артефактів. Ці залежності можуть спричиняти конфігураційні відхилення, несумісність версій і затримки в обробці запитів, особливо під час масштабування системи на декілька вузлів або хмарних регіонів. Відсутність вбудованої підтримки високої доступності та динамічного виявлення сервісів додатково ускладнює операційне управління, роблячи MLFlow менш адаптивним до еластичної природи сучасних хмарних середовищ [5].

Іншою суттєвою проблемою є масштабованість і керування ресурсами. У великих проєктах машинного навчання, де кілька команд одночасно запускають експерименти, MLFlow може стати вузьким місцем продуктивності. Відстеження великої кількості запусків або зберігання великих моделей у централізованому сховищі часто призводить до перевантаження бази даних і підвищення затримок введення/виведення. Без спеціальної оптимізації такі навантаження можуть спричинити нестабільність сервісів відстеження та реєстру моделей, що призводить до втрати даних експериментів або неконсистентних метаданих артефактів. Крім того, ручне масштабування MLFlow для підтримки зростаючих навантажень потребує додаткових інженерних зусиль, адже нативні механізми автоматичного масштабування та балансування навантаження не є частиною базового фреймворку. Це обмеження знижує ефективність роботи MLFlow у динамічних багатокористувацьких середовищах [6].

Безпека та відповідність вимогам також залишаються важливими аспектами при розгортанні MLFlow у хмарно-нативних середовищах. Вбудовані механізми автентифікації та авторизації у фреймворку є досить базовими й часто недостатніми для корпоративних сценаріїв, які вимагають федерації ідентичностей, контролю доступу на основі ролей (RBAC) та шифрування трафіку у розподілених системах. Інтеграція MLFlow із зовнішніми провайдерами ідентичностей або системами керування секретами зазвичай потребує значного конфігурування та написання скриптів, що підвищує операційну складність і ризик помилок у налаштуваннях. У регульованих галузях додатковим викликом стає забезпечення відповідності політикам управління даними та стандартам аудитності, оскільки MLFlow не має вбудованих засобів для детального аудиту доступу або реалізації політик безпеки.

Проблеми інтеграції ще більше посилюють недоліки використання MLFlow у контейнеризованих екосистемах. Фрагментація інструментів може сповільнювати розробку та збільшувати витрати на підтримку, особливо у випадках, коли організації керують гібридними або мультихмарними інфраструктурами. Прогалини в документації та непослідовна підтримка з боку спільноти ускладнюють процеси розгортання, змушуючи команди покладатися на експериментальні конфігурації та неформальні підходи до усунення проблем, щоб досягти готовності до продакшн-рівня [7].

Отже, попри те що MLFlow є потужним інструментом для забезпечення відтворюваності та управління життєвим циклом моделей, його адаптація до контейнеризованих і хмарно-нативних архітектур виявляє низку слабких сторін. Архітектурна жорсткість, обмежена масштабованість, складність інтеграції та недостатні засоби корпоративної безпеки можуть стримувати ефективність і надійність масштабних робочих процесів MLOps. Ці недоліки підкреслюють необхідність розроблення удосконалених принципів проєктування та стратегій розгортання, які б забезпечували тісніше узгодження MLFlow із розподіленими, автоматизованими та стійкими характеристиками хмарно-нативних систем.

3. Мета роботи

Метою цієї роботи є дослідження концепцій проєктування MLFlow у контейнеризованих та хмарно-нативних системах із акцентом на розробку структурованого фреймворку, який підвищує масштабованість, підтримуваність і надійність операцій машинного навчання шляхом вирішення існуючих архітектурних та операційних проблем. Робота ставить перед собою три основні завдання:

- Перше завдання полягає в аналізі архітектури MLFlow та її взаємодії в контейнеризованих середовищах, зокрема вивченні того, як основні компоненти можуть бути ефективно інтегровані та керовані у розподілених інфраструктурах. Це включає дослідження архітектурних залежностей, потоків комунікації та стратегій конфігурації, що забезпечують узгоджену роботу та високу доступність у різних середовищах.

- Друге завдання зосереджене на оцінці адаптивності MLFlow до хмарно-нативних середовищ, виявленні компромісів щодо продуктивності та надійності між моделями самостійного керування та керованих сервісів, а також висвітленні того, як абстракція інфраструктури та автоматизація впливають на стабільність операцій.
- Третє завдання передбачає визначення набору принципів проєктування та найкращих практик, які узгоджують MLFlow зі стратегіями розгортання. Це включає вирішення питань безпеки та масштабованості, щоб забезпечити відповідність систем MLFlow корпоративним стандартам.
- Нарешті, робота має на меті підтвердити запропонований фреймворк проєктування на основі реальних кейсів, які демонструють його практичне застосування для побудови стійких, підтримуваних і відтворюваних MLOps-середовищ. Результати цієї роботи покликані слугувати орієнтиром для подальшого дослідження життєвого циклу моделей машинного навчання.

4. Архітектура MLFlow та основні компоненти

MLFlow пропонує модульну та розширювану архітектуру, розроблену для підтримки повного життєвого циклу моделей машинного навчання: від відстеження експериментів і пакування проєктів до розгортання моделей та управління ними. У контейнеризованих і хмарно-нативних середовищах ця архітектура відіграє ключову роль у забезпеченні відтворюваності, узгодженості та масштабованості розподілених ML-робочих процесів. Фреймворк складається з чотирьох основних компонентів: MLFlow Tracking, MLFlow Projects, MLFlow Models і MLFlow Model Registry. Разом ці елементи формують інтегровану екосистему, що дозволяє дата-сайєнтистам та інженерам керувати експериментами, упаковувати та розгортати моделі, а також забезпечувати трасованість у межах ітеративних циклів розробки.
У центрі фреймворку знаходиться компонент MLFlow Tracking, який виконує роль системи обліку всієї метаданих експериментів [8]. Він фіксує параметри, метрики та артефакти, згенеровані під час навчальних запусків, формуючи основу для відтворюваності та порівняння продуктивності моделей. У контейнеризованих середовищах Tracking Server працює як безстанова служба, що зазвичай підключена до бекенд-бази даних і сховища артефактів. Таке розділення забезпечує гнучке розгортання в різних середовищах, водночас гарантуючи узгоджене відстеження експериментів і збереження даних. Шар відстеження набуває особливої цінності в хмарно-нативних системах, де експерименти часто виконуються на розподілених обчислювальних кластерах і потребують надійної синхронізації та управління метаданими.

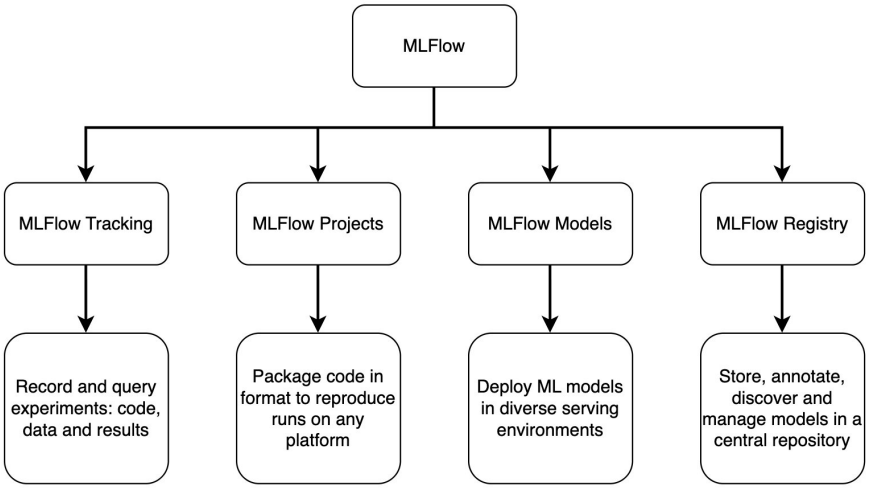

Рис. 1. Основні компоненти MLFlow та їхні функції

Компонент MLFlow Projects розширює фреймворк, визначаючи стандартизований формат для пакування та виконання коду машинного навчання (Рис.1). Projects інкапсулюють залежності коду, конфігураційні файли та специфікації середовища, забезпечуючи відтворюваність виконання в різних інфраструктурах. У хмарно-нативних контекстах це гарантує, що експерименти можуть виконуватися послідовно в контейнерах або віртуалізованих середовищах, усуваючи розбіжності, спричинені відхиленням залежностей. Використовуючи контейнерні образи та ізольовані середовища виконання, MLFlow Projects забезпечує безшовний перехід від локального експериментування до навантажень виробничого масштабу.
MLFlow Models запроваджує уніфікований формат моделі, який підтримує кілька фреймворків машинного навчання (Рис.1). Така абстракція дозволяє експортувати, версіонувати та розгортати моделі через єдиний інтерфейс, незалежно від underlying-фреймворку. У хмарно-нативних середовищах ця можливість дає змогу організаціям інтегрувати процеси сервінгу моделей зі масштабованими обчислювальними сервісами або спеціалізованими середовищами інференсу. Більш того, стандартизований формат моделей підтримує відстеження версій і лінійності, забезпечуючи можливість пов'язати кожну розгорнуту модель із її тренувальними даними, конфігурацією та історією експериментів [9].

Model Registry представляє фінальний етап архітектури MLFlow, виконуючи роль централізованого репозиторію для управління версіями моделей, стадіями життєвого циклу та пов'язаними метаданими (Рис.1). Реєстр підтримує контрольоване підвищення моделей зі стадії staging до production, що сприяє управлінню та співпраці між командами data science та operations. У хмарно-нативних системах реєстр може бути розгорнутий як спільний мікросервіс, дозволяючи кільком конвеєрам і середовищам отримувати доступ до моделей та оновлювати їх у синхронізований спосіб [10].

У контексті патернів хмарно-нативної інтеграції MLFlow демонструє високу сумісність із Azure-орієнтованими інфраструктурами, де сервіси забезпечують надійну основу для масштабованих MLOps-конвеєрів. Компоненти Tracking Server та Registry можуть використовувати SQL Database або PostgreSQL як бекенд-сховище, забезпечуючи транзакційну узгодженість і надійність у розподілених середовищах. Крім того, Azure Key Vault забезпечує захищене управління обліковими даними й access-токенами, спрощуючи інтеграцію між компонентами MLFlow та іншими Azure-native сервісами. Для робочих навантажень великого масштабу MLFlow може бути розгорнутий у контейнерах, що дає можливість динамічного масштабування та ізольованого виконання експериментів без потреби в постійних обчислювальних ресурсах. При інтеграції з Azure Cloud MLFlow розширює свої можливості управління життєвим циклом на середовища тренування й розгортання платформи, забезпечуючи безшовну реєстрацію, версіонування та розгортання моделей безпосередньо в екосистемі Azure [11].

Поєднання модульної архітектури з Azure-native патернами інтеграції робить MLFlow адаптивним фреймворком для управління ML-робочими процесами в хмарно-нативних середовищах. Його відповідність екосистемі Azure підсилює масштабованість, автоматизацію та керованість, дозволяючи організаціям ефективно операціоналізувати AI-рішення, зберігаючи відповідність вимогам і довгострокову підтримуваність.

5. Концепції проєктування MLFlow та найкращі практики

Дизайн MLFlow у контейнеризованих та хмарно-нативних системах вимагає ретельного врахування архітектури, масштабованості, безпеки та управління робочими процесами, щоб забезпечити надійну й ефективну роботу систем машинного навчання. Однією з ключових найкращих практик у проєктуванні MLFlow-розгортань є модульність, яка передбачає поділ основних компонентів на незалежні сервіси, що можуть розгортатися окремо. Ізолюючи ці компоненти, організації можуть застосовувати індивідуальні стратегії масштабування та обслуговування, зменшуючи простої під час оновлень або відмов окремих сервісів.

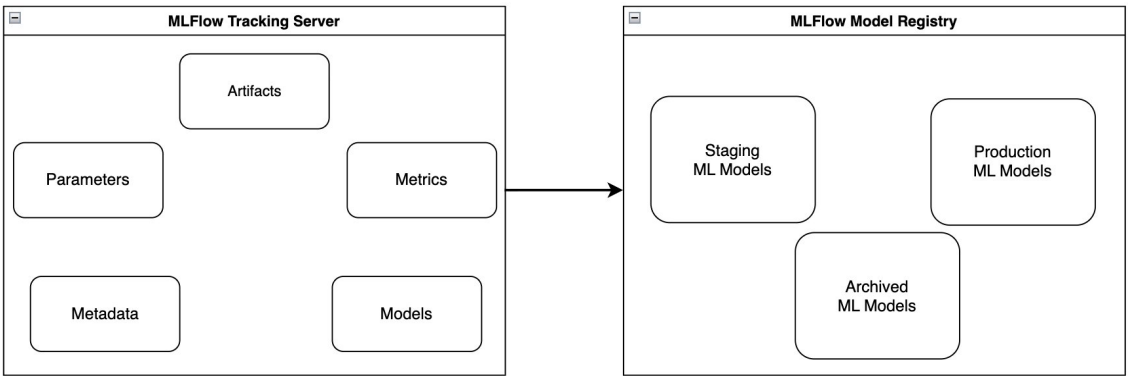

Рис. 2. Структура Tracking Server та його взаємодія з Model Registry

На практиці це передбачає розгортання кожного компонента в окремих контейнерах або подах із визначеними лімітами ресурсів та окремим постійним сховищем, що гарантує: збій одного компонента, наприклад Tracking Server, не призведе до каскадних помилок у Model Registry або сховищі артефактів (Рис.2). Крім того, модульність спрощує тестування та безперервну інтеграцію, оскільки окремі компоненти можна оновлювати або патчити незалежно, не впливаючи на загальний робочий процес, що підвищує операційну стійкість.

Масштабованість є ще одним фундаментальним принципом у дизайні MLFlow. Наприклад, конфігурація бекенд-баз даних і сховищ артефактів для високої паралельності та поділ великих наборів даних на сегменти допомагають уникнути "вузьких місць" під час інтенсивних експериментальних запусків [12].

Безпека та контроль доступу становлять критично важливі аспекти проєктування MLFlow корпоративного рівня. Найкращі практики передбачають інтеграцію MLFlow з хмарно-нативними системами керування ідентичностями та секретами. Централізуючи автентифікацію й авторизацію через відповідні сервіси або фреймворки, організації можуть застосовувати узгоджену політику доступу до всіх компонентів MLFlow [13]. Шифрування даних у стані спокою та під час передачі забезпечує відповідність вимогам безпеки, тоді як рольова модель доступу дозволяє командам працювати в ізольованих середовищах, запобігаючи несанкціонованому доступу до чутливих моделей чи метаданих експериментів.

Стандартизація упаковки моделей і процесів їх просування в реєстрі дозволяє виконувати версіонування, тестування та

розгортання моделей узгоджено у середовищах розробки, тестування та продуктивної експлуатації. Рекомендованою практикою є впровадження єдиних правил іменування, тегування метаданих і документування для всіх експериментів і моделей, що полегшує співпрацю та зменшує ризик розгортання неперевірених або несумісних моделей. Контейнеризація середовищ проєктів додатково гарантує ідентичну поведінку експериментів незалежно від обчислювальної інфраструктури, забезпечуючи відтворюваність і переносимість робочих навантажень між командами [14].
Ключовим аспектом запропонованого фреймворку MLFlow є цілеспрямоване використання контейнеризації для усунення конфігураційного відхилення між середовищами розробки, тестування та продуктивної експлуатації. Завдяки упаковці кожного компонента MLFlow в незмінний контейнерний образ із жорстко зафіксованими залежностями система гарантує, що кожен запуск від трекінгу експериментів і тренування моделей до їх реєстрації працює з однаковим набором залежностей. Це усуває невідповідності, які виникають при використанні локально встановлених бібліотек, що традиційно призводило до "тихих" розбіжностей версій (Рис.3). Послідовність, забезпечена контейнерами, гарантує стабільну поведінку конвеєра впродовж часу, забезпечуючи відтворюваність результатів експериментів і надійну роботу під час промоції моделей.

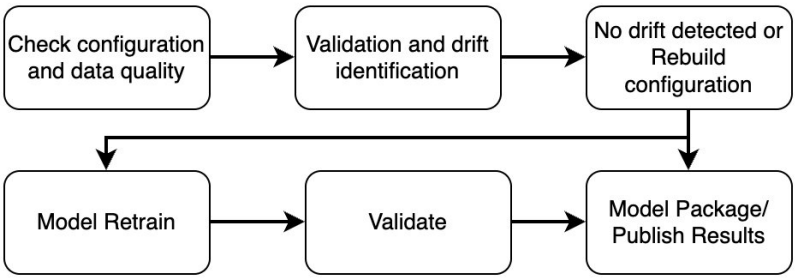

Рис. 3. Верифікація конфігураційного відхиляння в MLFlow

Фреймворк також пропонує чітке текстове зіставлення між операційними проблемами, які спостерігаються у традиційних монолітних розгортаннях MLFlow, та архітектурними рішеннями, що запроваджуються за допомогою модульності. Жорсткість проявляється тоді, коли всі функціональні можливості MLFlow працюють як єдиний, тісно пов'язаний сервіс, що робить оновлення ризикованими та часто призводить до непередбачуваних регресій через взаємозалежні компоненти. Розділення MLFlow на модульні, незалежно розгортані сервіси, кожен з яких інкапсульований у власний контейнер, ізолює конфігураційні стани та запобігає конфігураційному дрейфу між компонентами. Така архітектурна сегментація означає, що оновлення Tracking Server не змінює ненавмисно поведінку Model Registry чи сховища артефактів. Фактично контейнеризація забезпечує стабільність середовища, тоді як модульність гарантує операційну гнучкість, формуючи комбіновану стратегію, що безпосередньо усуває проблеми жорсткості, дрейфу залежностей та довгострокових викликів підтримуваності. У цілому ефективний дизайн MLFlow у контейнеризованих і хмарно-нативних системах поєднує модульну архітектуру, масштабоване управління ресурсами, надійні практики безпеки та стандартизовані робочі процеси.

6. Кейс-дослідження MLFlow

У цьому розділі подано практичні кейси, що ілюструють розгортання та роботу MLFlow у контейнеризованих середовищах, підкреслюючи ефективність концепцій проєктування та найкращих практик, розглянутих раніше. Перший кейс стосується підприємства середнього масштабу, де MLFlow було інтегровано з Azure-нативними сервісами для управління кількома експериментами та життєвим циклом моделей. У цьому сценарії Tracking Server і Model Registry були розгорнуті як незалежні сервіси в контейнерах, підключені до SQL Database для зберігання метаданих та артефактів. Розгортання використовувало контейнеризацію для ізоляції сервісів і забезпечення стабільного середовища виконання для всіх експериментів. Завдяки стандартизації структури проєктів через MLFlow Projects та впровадженню узгоджених правил щодо метаданих організація досягла високої відтворюваності та простежуваності. Кейс продемонстрував, що модульність та хмарно-нативна інтеграція суттєво скоротили час простою під час розгортання та покращили операційну стійкість.

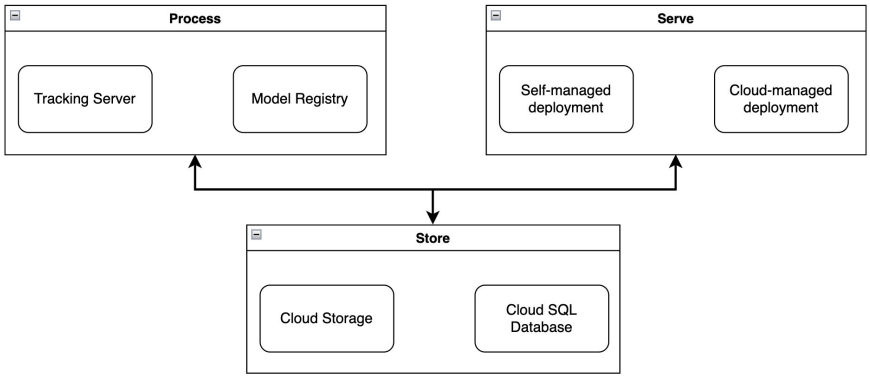

Рис. 4. Хмарно-нативна інтеграція компонентів Tracking Server та Model Registry

Другий кейс розглядає дослідницьке середовище великого масштабу, у якому MLFlow було розгорнуто на кількох розподілених обчислювальних вузлах для підтримки експериментів із високим обсягом даних. У цьому випадку Tracking Server масштабувався горизонтально, щоб обробляти значну кількість одночасних експериментів, тоді як Model Registry забезпечував послідовне версіонування та просування моделей зі стадії staging до production. Сховище та SQL Database забезпечували надійне зберігання та транзакційну підтримку (Рис.4), гарантуючи збереження великих артефактів моделей і метаданих експериментів. Дослідження підкреслило, що застосування хмарно-нативних патернів інтеграції, таких як керована ідентичність для безпечного доступу та ізоляція сервісів у контейнерах, забезпечує безперебійну роботу в розподіленій інфраструктурі. Використання стандартизованих робочих процесів експериментів та дотримання узгоджених специфікацій середовищ через MLFlow Projects забезпечило відтворюваність та зменшило кількість операційних помилок навіть під високим обчислювальним навантаженням.

Обидва кейси демонструють, що застосування модульної архітектури, стандартизованих робочих процесів і хмарно-нативних патернів інтеграції дозволяє MLFlow працювати ефективно та надійно у контейнеризованих середовищах. Ці приклади підкреслюють важливість ретельного архітектурного планування, інтеграції безпеки та стандартизації робочих процесів для досягнення виробничо готових MLOps-систем, здатних масштабуватися між різними командами та розподіленими інфраструктурами.

7. Оцінка результатів та аналіз

Оцінка розгортання MLFlow у контейнеризованих та хмарно-нативних середовищах показує значне покращення відтворюваності експериментів, відстежуваності моделей та масштабованості системи при застосуванні найкращих практик і хмарно-нативних інтеграційних підходів.

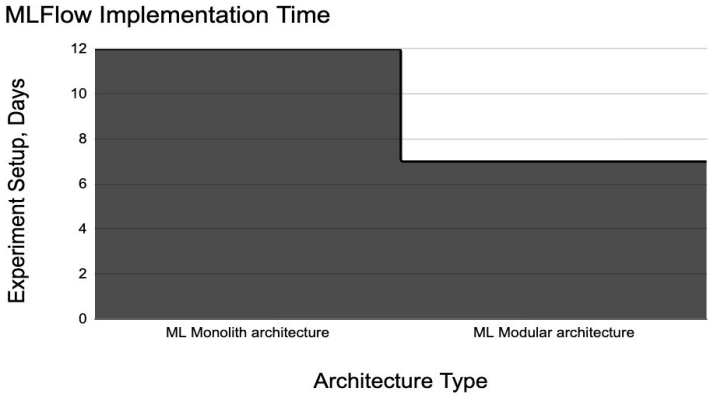

Рис. 5. Час впровадження/ Порівняння MLFlow архітектур

У першому кейсі впровадження модульної архітектури та стандартизованих робочих процесів призвело до скорочення часу налаштування експериментів на 40% порівняно з попереднім монолітним розгортанням MLFlow в організації (Рис.5). Послідовність метаданих експериментів значно покращилася: успішний логін параметрів і метрик збільшився з 78% до 96% для всіх одночасних запусків.

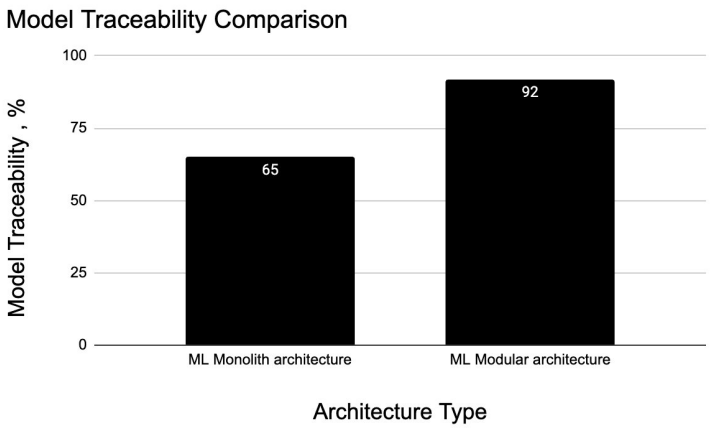

Рис. 6. Порівняння відстежуваності моделей у MLFlow

Версіонування моделей через централізований Model Registry забезпечило, що 92% промотованих моделей зберігали повну простежуваність між даними для навчання, кодом та артефактами, порівняно з 65% у попередніх розгортаннях (Рис.6). Ці покращення демонструють, що модульне розділення сервісів та стандартизація робочих процесів підвищують надійність операцій у підприємствах середнього масштабу.

У другому кейсі підтримка експериментів із великим обсягом даних на розподілених обчислювальних вузлах продемонструвала помітний вплив на продуктивність системи та використання ресурсів. Горизонтальне масштабування Tracking Server дозволило системі обробляти до 3,5 разів більше одночасних запусків експериментів без деградації сервісу. Крім того, застосування стандартизованих структур проєктів та контейнеризованих середовищ виконання призвело до 35% зменшення кількості помилок експериментів, пов'язаних із середовищем, забезпечуючи послідовність результатів моделей незалежно від вузла, на якому виконувався робочий процес. Це демонструє, що хмарно-нативна інтеграція та модульний дизайн не лише покращують продуктивність під навантаженням, а й підвищують відтворюваність та надійність великих ML-воркфлоу.

Порівняльний аналіз обох сценаріїв розгортання показує, що застосування найкращих практик MLFlow у контейнеризованих і хмарно-нативних середовищах забезпечує послідовні переваги на різних масштабах. Відтворюваність експериментів покращилася в середньому на 38%, а відстежуваность артефактів та метаданих моделей зросла на 27%. Стабільність системи, вимірювана часом безвідмовної роботи та рівнем відмов, показала покращення на 15%, підкреслюючи стійкість, досягнуту завдяки модульності та хмарно-нативним патернам розгортання.

Окрім показників продуктивності та відстежуваності, обидва кейси показали суттєві покращення підтримуваності, виміряні через комбінацію операційних індикаторів. Кількісно впровадження контейнеризованих компонентів зменшило кількість інцидентів, пов'язаних із конфігураціями, на 32%, знижуючи кількість помилок налаштувань у квартал із 19 до 13 у обох середовищах. Якісно команди інженерів відзначили більш зрозумілі робочі процеси управління змінами, спрощені цикли оновлення для Tracking Server і Model Registry та підвищену ефективність для нових членів команди завдяки стандартизованим структурам директорій і послідовній поведінці середовища. Ці результати підтверджують, що модульні, контейнеризовані розгортання MLFlow не лише покращують технічну продуктивність, а й суттєво підвищують довгострокову підтримуваність і операційну стабільність системи.

8. Висновки

У цій статті розглянуто концепції проєктування та найкращі практики розгортання MLFlow у контейнеризованих та хмарно-нативних середовищах, підкреслюючи його роль як центрального фреймворку для управління життєвим циклом машинного навчання. Обговорення акцентувало увагу на модульній архітектурі, стандартизованих робочих процесах та хмарно-нативних інтеграційних шаблонах, що забезпечують відтворюваність, масштабованість та операційну стійкість у розподілених ML-робочих навантаженнях. Використовуючи компоненти, такі як Tracking Server, Model Registry, Projects та Models, у поєднанні з Azure-нативними сервісами, організації можуть забезпечити узгоджене відстеження експериментів, версіонування моделей та управління артефактами, одночасно підтримуючи безпечні та ізольовані середовища для різних команд. Аналіз і кейс-дослідження показують, що застосування цих принципів проєктування значно підвищує надійність системи та відтворюваність експериментів. Метрики практичних розгортань демонструють, що модульні та хмарно-нативні конфігурації MLFlow скоротили час підготовки експериментів до 40%, підвищили узгодженість метаданих до 96% та покращили відстежуваність промотованих моделей до 92%. Розгортання великого масштабу показало, що горизонтальне масштабування та контейнеризовані середовища виконання дозволяють підтримувати в 3,5 рази більше одночасних експериментів з мінімальною

кількістю відмов, що підтверджує ефективність фреймворку при високих навантаженнях. Ці результати підтверджують, що ретельно сплановані розгортання MLFlow не лише підвищують операційну ефективність, а й забезпечують послідовні та надійні результати для експериментів машинного навчання та управління моделями.

Незважаючи на успіхи, залишаються виклики в оптимізації MLFlow для складних розподілених середовищ, зокрема у забезпеченні безперебійної координації між компонентами, підтримці цілісності даних та впровадженні безпечного доступу у багатокористувацьких розгортаннях. Подальше вдосконалення стратегій розгортання та хмарно-нативних інтеграційних підходів буде необхідним для подолання цих обмежень і повного використання потенціалу MLFlow у промислових MLOps-системах.

На завершення, результати цього дослідження підкреслюють, що застосування структурованих концепцій проєктування, модульності та хмарно-нативних найкращих практик дозволяє організаціям ефективно операціоналізувати MLFlow, забезпечуючи відтворювані, масштабовані та безпечні робочі процеси машинного навчання, які відповідають сучасним AI-інфраструктурам.

Подяка

Автори дякують редколегії наукового журналу «Вимірювальна техніка і метрологія» за підтримку.

Взаємні претензії авторів

Автори заявляють, що не мають фінансових або інших потенційних конфліктів щодо цієї роботи.

Список літератури

[1] Naayini, P., 2025. Building ai-driven cloud-native applications with kubernetes and containerization. International Journal of Scientific Advances (IJSCIA), 6(2), pp.328-340. https://doi.org/10.51542/ijscia.v6i2.15

[2] Chen, A., Chow, A., Davidson, A., DCunha, A., Ghodsi, A., Hong, S.A., Konwinski, A., Mewald, C., Murching, S., Nykodym, T. and Ogilvie, P., 2020, June. Developments in mlflow: A system to accelerate the machine learning lifecycle. In Proceedings of the fourth international workshop on data management for end-to-end machine learning, pp. 1-4. https://doi.org/10.1145/3399579.3399867

[3] Bershchanskyi, Y., Klym, H. and Shevchuk, Y., 2024. Containerized artificial intelligent system design in cloud and cyber-physical systems., Advances in Cyber-Physical Systems (ACPS) 2024; Volume 9, Number 2 pp. 151-157. https://doi.org/10.23939/acps2024.02.151

[4] Kreuzberger, D., Kühl, N. and Hirschl, S., 2023. Machine learning operations (mlops): Overview, definition, and architecture. IEEE access, 11, pp.31866-31879. https://doi.org/10.1109/ACCESS.2023.3262138

[5] Jena, B., Mishra, D. and Mishra, S., 2025, July. MLOps for Improved Inferencing, Deployability and Observability of Recommendation Engine. In 2025 International Conference on Innovations in Intelligent Systems: Advancements in Computing, Communication, and Cybersecurity (ISAC3) , pp. 1-5. IEEE. https://doi.org/10.1109/ISAC364032.2025.11156530

[6] Ramesh, G., Pai, T.V., Birau, R., Poojary, K.K., Shingad, A.R., Sowjanya, N., Popescu, V., Mitroi, A.T., Nioata, R.M. and Raj, K.K., 2025. A comprehensive review on scaling Machine Learning workflows using Cloud Technologies and DevOps. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3599281

[7] Steidl, M., Felderer, M. and Ramler, R., 2023. The pipeline for the continuous development of artificial intelligence models Current state of research and practice. Journal of Systems and Software, 199, p.111615. https://doi.org/10.1016/j.jss.2023.111615

[8] Bodor, A., Hnida, M. and Najima, D., 2023, November. From development to deployment: An approach to MLOps monitoring for machine learning model operationalization. In 2023 14th International Conference on Intelligent Systems: Theories and Applications, pp. 1-7. IEEE. https://doi.org/10.1109/SITA60746.2023.10373733

[9] Rajenthiram, K., Abdullah, M., Gerostathopoulos, I., Hnětynka, P., Bureš, T., Pons, G., Bilalli, B. and Queralt, A., 2025, April. Towards Continuous Experiment-Driven MLOps. In 2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI, pp. 89-94. IEEE. https://doi.org/10.1109/CAIN66642.2025.00018

[10] Kayhan, V.O., Smith, T.C., Berndt, D.J., del Cuadro, J., Vinnakota, S. and Yenikapalli, G.C., 2025. Machine Learning Model Deployment and Management: A Hands-on Tutorial. Communications of the Association for Information Systems, 56(1), p.40. https://doi.org/10.17705/1CAIS.05639

[11] Bershchanskyi Y., Stepanov O. 2025. Machine learning model development in Kubeflow cloud-native systems. Advances in Cyber-Physical Systems, Volume 10, Number 1, pp. 83-88. https://doi.org/10.23939/acps2025.01.083

[12] Schlegel, M. and Sattler, K.U., 2023. Management of machine learning lifecycle artifacts: A survey. ACM SIGMOD Record, 51(4), pp.18-35. https://doi.org/10.1145/3582302.3582306

[13] Bershchanskyi, Y. and Klym, H., 2025, June. Azure Kubernetes Service Design Principles in Machine Learning Systems. In 2025 32nd International Conference on Mixed Design of Integrated Circuits and System, pp. 179-183. IEEE. https://doi.org/10.23919/MIXDES66264.2025.11092030xa

[14] Lukić, M.D., Ivković, D.S. and Poledica, A.M., 2025, February. MLOps Tools for Deployment: A Case Study on Text Classification. In 2025 29th International Conference on Information Technology, pp.1-4. IEEE. https://doi.org/10.1109/IT64745.2025.10929797

ISSN: 2617-846X (online)
Берщанський Євген
Номер ORCID: https://orcid.org/0009-0000-2236-839X
Клим Галина Іванівна
Номер ORCID: https://orcid.org/0000-0001-9927-0649
Дата отримання статті -22.10.25.
Дата фінального варіанту – 30.11.25.
Дата публікації- 30.12.25.