Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інженерії
програмного забезпечення
в енергетиці (ІПЗЕ)

Процеси неперервної інтеграції і деплойменту

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	**Перший (бакалаврський)**
Галузь знань	12 Інформаційні технології
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці
Статус дисципліни	Вибіркова
Форма навчання	очна(денна)
Рік підготовки, семестр	3 курс весняний семестр
Обсяг дисципліни	4 кредитів ЄКТС/120 год (36 год. лекції, 18 год. практичні заняття, 66 год. самостійна робота)
Семестровий контроль/ контрольні заходи	Залік
Розклад занять	Згідно розкладу на весняний семестр поточного навчального року (http://roz.kpi.ua/)
Мова викладання	Українська
Інформація про керівника курсу / викладачів	Лектор: старший викладач, Колумбет В.П., kvplinux@gmail.com, тел. 093-405-35-39 Практичні: старший викладач, Колумбет В.П., kvplinux@gmail.com, тел. 093-405-35-39
Розміщення курсу	Кампус

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

В цій дисципліні детально вивчається концепції, принципи та методи інформаційних технологій. Етапи розробки програмного забезпечення. Міжнародні стандарти програмної інженерії. Моделі та процеси, які впроваджуються в розробленні програмного забезпечення. Вимоги до інформаційної системи. Документування розроблених вимог та моделей для створення інформаційної системи. Управління змінами до вимог протягом циклу розробки програмного забезпечення.

Метою дисципліни є опанування студентами теоретичних та практичних знань і навичок систем тестування, інструментальних засобів розробки програмних систем, розгортання та функціонування програмного забезпечення, розробляти, аналізувати та застосовувати ефективні алгоритми для розв'язання професійних завдань в області інформаційних технологій, та узгодження розробки й постачання програмного забезпечення із його використанням

Предмет дисципліни - вивчення принципів побудови, архітектури, основних функцій, режимів роботи і елементи та комплекси практик процесів неперервної інтеграції і деплойменту та розроблення адаптивних алгоритмів розв'язування задач.

Завдання. В результаті вивчення дисципліни у студентів повинні сформуватися наступні компетентності:

Вивчення дисципліни «Процеси неперервної інтеграції і деплойменту» сприяє формуванню у студентів фахової компетентності (ФК) за освітньою програмою:

ФК01 - Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

ФК05 - Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів ЖИТТЄВОГО ЦИКЛУ.

ФК08 - Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення.

ФК10 - Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя.

Вивчення дисципліни «Процеси неперервної інтеграції і деплойменту» сприяє формуванню у студентів наступних програмних результатів навчання (ПРН) за освітньою програмою:

ПРН01 - Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

ПРН03 - Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.

ПРН05 - Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення.

ПРН07 - Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

ПРН09 - Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення.

ПРН14 - Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення.

ПРН18 - Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних.

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Пререквізити дисципліни. Знання, отримані при вивченні дисциплін: “Функціональне програмування” та “Архітектура та проектування програмного забезпечення”.

Постреквізити дисципліни. Отримані знання при вивченні дисципліни «Архітектура системного програмного забезпечення» формує базові знання для вивчення наступних дисциплін: “Теорія ймовірностей та математична статистика”, “Алгоритми та структури даних”, “Бази даних”, “Моделювання та аналіз програмного забезпечення”, “Основи розробки трансляторів”, “Системи штучного інтелекту”, “Організація баз даних та знань”, “Криптографія та шифрування”. Також викладений матеріал може бути використаний при вивченні дисциплін “Математичне моделювання систем і процесів”, “Дослідження операцій”, які викладаються в наступних семестрах. Компетенції, отримані студентами в процесі вивчення цієї дисципліни, використовуються ними при виконанні дипломної роботи.

3. Зміст навчальної дисципліни

Тема 1. Основи та важливість CI/CD в сучасній розробці ПЗ.
Тема 2. Інструменти та технології в CI/CD.
Тема 3. Автоматизація тестування в CI/CD.
Тема 4. Безпека та команда в процесах CI/CD.
Тема 5. Стратегії розгортання та управління релізами.
Тема 6. Інфраструктура як код (IaC) та контейнеризація.
Тема 7. Майбутнє CI/CD та еволюція практик.

4. Навчальні матеріали та ресурси

Основна література

1. Петренко М. Інтеграція та автоматичний деплоймент із GitLab CI/CD. – Львів : Видавництво Львівської політехніки, 2021. – 304 с. ISBN 978-966-936-900-9.
2. Іваненко О. Системи неперервної інтеграції: принципи та практики. – Київ : BHV, 2019. – 320 с. ISBN 978-617-7659-00-1.
3. Ковальчук П. Jenkins у DevOps: неперервна інтеграція та деплоймент. – Харків : КМ-Книги, 2020. – 280 с. ISBN 978-617-7409-80-4.
4. Бондарчук С. Kubernetes CI/CD: побудова pipeline для мікросервісів. – Київ : BHV, 2021. – 288 с. ISBN 978-617-7659-20-9.
5. Смирнов В. Практика Terraform та CI/CD для Infrastructure as Code. – Львів : Грані-Т, 2020. – 240 с. ISBN 978-617-649-234-7..
6. Гнатюк О. Azure DevOps у практиці неперервної інтеграції та деплойменту. – Харків : Фоліо, 2023. – 312 с. ISBN 978-966-03-9900-1.
7. Martin Fowler, "Continuous Integration", martinFowler, may 2006. URL: https://martinfowler.com/articles/continuousIntegration.html
8. Шевченко Л. Spinnaker: оркестрація деплойменту у хмарних середовищах. – Львів : Навчальна книга, 2024. – 360 с. ISBN 978-966-639-400-2.
9. Вербицька Н. Argo CD: від налаштування до впровадження. – Київ : Літопис, 2022. – 264 с. ISBN 978-966-917-650-6.
10. Коваленко І. Drone CI: автоматизація процесів DevOps. – Харків : КМ-Книги, 2023. – 232 с. ISBN 978-617-7409-95-8.
11. Козак С. Інструменти DevSecOps для CI/CD: інтеграція безпеки в pipeline. – Харків : Фоліо, 2025. – 304 с. ISBN 978-966-03-9955-1.

Додаткова література

1. Дмитренко Ю. Ansible у CI/CD: автоматизація розгортання та конфігурації. – Київ : Острів, 2019. – 280 с. ISBN 978-617-679-125-8.
2. Савченко А. GitOps: неперервна інтеграція та деплоймент у Kubernetes. – Київ : BHV, 2022. – 300 с. ISBN 978-617-7659-55-1.
3. Прокопенко Р. Контейнеризація та CI/CD з Docker: практичний підхід. – Харків : Фоліо, 2018. – 256 с. ISBN 978-966-03-8708-5.
4. Назаренко В. Моніторинг та логування в процесах CI/CD. – Львів : Видавництво Львівської політехніки, 2023. – 212 с. ISBN 978-966-936-910-8.
5. Визначення предмету-програмна інженерія.-/Лавріщева К. М.-Проблеми програмування.-Спецвипуск.-2008.-№ 2-3.-с.191-204.
6. Гончаренко П. Безпека в CI/CD процесах: DevSecOps підхід. – Київ : Довіра, 2024. – 276 с. ISBN 978-966-507-916-1.
7. Lotfi Waderni, "HOW TO BUILD YOUR CUSTOM JENKINS DOCKER IMAGE", Yala-Labs, may 2020. URL: https://yallalabs.com/devops/jenkins/howto-build-custom-jenkins-docker-image/.
8. Демченко Л. Архітектура cloud-native: CI/CD у хмарних додатках. – Харків : КМ-Книги, 2025. – 344 с. ISBN 978-617-7409-99-6.
9. Кравець О. Spinnaker & Kubernetes: накладна практика деплойменту. – Одеса : MicroFormat, 2020. – 320 с. ISBN 978-617-7398-50-2.
10. Тимошенко Д. DevOps pipeline: методи та засоби. – Київ : Навчальна книга, 2018. – 296 с. ISBN 978-966-639-320-3.
11. Литвин М. CI/CD у розробці мікросервісів: від теорії до практики. – Київ : Літопис, 2021. – 328 с. ISBN 978-966-917-600-1.

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента)

Тема 1. Основи та важливість CI/CD в сучасній розробці ПЗ.

Лекція 1. Вступ до неперервної інтеграції та неперервного розгортання

Основні поняття та переваги CI/CD. Історія розвитку методологій CI/CD..

Тема 2. Інструменти та технології в CI/CD.

Лекція 2. Інструменти неперервної інтеграції

Огляд популярних інструментів CI: Jenkins, GitLab CI, CircleCI тощо. Вибір інструменту для проекту.

Лекція 3. Конфігурація середовища для CI

Налаштування сервера CI. Автоматизація тестування коду. Створення ефективного CI пайплайну. Управління залежностями та версіями.

Тема 3. Автоматизація тестування в CI/CD.

Лекція 4. Тестування в рамках CI

Автоматизоване тестування: юніт-тести, інтеграційні тести, системні тести. Конфігурація середовища для автоматичного тестування.

Лекція 5. Інтеграція і робота з кодовими репозиторіями

Git workflows: feature branching, Gitflow, trunk-based development. Автоматизація злиття коду та розв'язання конфліктів.

Тема 4. Безпека та команда в процесах CI/CD.

Лекція 6. Безпека в процесах CI/CD

Виявлення вразливостей у коді. Керування доступом і секретами.

Лекція 7. Контейнеризація та CI/CD

Використання Docker у процесах CI/CD. Основи Kubernetes для розгортання.

Тема 5. Стратегії розгортання та управління релізами.

Лекція 8. Неперервний розгортання (CD): основи

Відмінності між CD та CI. Стратегії розгортання: blue-green deployment, canary release. Інструменти та практики моніторингу. Аналіз логів та звітність. Кешування та оптимізація швидкості побудов. Автоматизація та скорочення затрат на обслуговування.

Лекція 9. CI/CD для мікросервісної архітектури

Особливості побудови конвеєра, коли додаток розбитий на численні сервіси – ізольовані збірки, незалежні деплойменти й версіонування.

Лекція 10. Тестування в CI: юніт-, інтеграційні та функціональні

Інтеграція автоматизованих тестів у pipeline, паралельне виконання, звіти про покриття та швидка ідентифікація regres-sion-помилок.

Тема 6. Інфраструктура як код (IaC) та контейнеризація.

Лекція 11. Інфраструктура як код (IaC) у CI/CD

Використання Terraform, Ansible для автоматизації інфраструктури. Переваги IaC у процесах розгортання. Інструменти та підходи до керування конфігурацією.

Лекція 12. Автоматизація налаштувань середовищ.

Використання хмарних платформ для CI/CD. Специфіка розгортання у хмарі. CloudFormation, та автоматизація створення та оновлення середовищ у хмарі й on-premise

Лекція 13. Паралелізація та масштабування конвеєра Розподіл етапів конвеєра між агентами/runner-ами, кешування, шардинг тестів та збірок для прискорення проходження pipeline.

Лекція 14. Мікросервісна архітектура та CI/CD
Особливості розгортання мікросервісів. Управління залежностями та версіями в мікросервісному середовищі. Культура DevOps та її вплив на CI/CD. Роль комунікацій та співпраці.

Лекція 15. Кейс-стаді з реального життя: впровадження CI/CD
Аналіз успішних кейсів впровадження CI/CD. Перешкоди та виклики під час впровадження.

Лекція 16. Моніторинг, логування та оповіщення
Інтеграція Prometheus, Grafana, ELK-стеку для відслідковування здоров'я конвеєра, швидкого реагування на збої та аналізу продуктивності.

Тема 7. Майбутнє CI/CD та еволюція практик.

Лекція 17. GitOps: підхід до керування інфраструктурою
Використання Git як єдиного джерела правди для опису стану кластера, автоматичної синхронізації та рев'ю-процесів.

Лекція 18. Майбутнє CI/CD та останні тренди
Інновації та майбутній розвиток CI/CD. Вплив штучного інтелекту та машинного навчання на CI/CD.

6. Практичні заняття

1. *Налаштування Jenkins Pipeline*

Завдання: Встановити Jenkins, створити Freestyle-job та програми pipelined-job для збірки простого Java-проекту з використанням Maven.
Мета: Освоїти базову архітектуру Jenkins, роботу з джобами та засади «pipeline as code».

2. *CI з GitLab CI/CD*

Завдання: Написати .gitlab-ci.yml для Node.js-додатка, що виконує інсталяцію залежностей, запуск юніт-тестів та публікацію артефактів.
Мета: Зрозуміти синтаксис GitLab CI/CD, стадії (stages) та кешування між завданнями.

3. *Автоматизований збір Docker-образу і пуш у Registry*

Завдання: Додати в CI-pipeline кроки для побудови Docker-образу, тегування за номером коміту та пушу в Docker Hub або приватний Registry.
Мета: Навчитись інтегрувати основу Docker-workflow у процес CI/CD та керувати версіями образів.

4. *CI/CD з GitHub Actions*

Завдання: Створити GitHub Actions workflow, який на кожен push на main запускає тести, будує фронтенд (наприклад, React) і публікує результат у GitHub Pages.
Мета: Освоїти структуру .github/workflows/*.yml, тригери та роботу з артефактами у GitHub Actions.

5. *Інтеграція статичного аналізу коду*

Завдання: Додати в CI-pipeline крок запуску SonarQube або ESLint/flake8 і зупинку конвеєра у разі порушення порогів якості.
Мета: Показати, як підвищити якість коду через автоматичні перевірки та забезпечити fail-fast стратегія.

6. *Деплой на тестовий Kubernetes-кластер*

Завдання: Розгорнути Helm-чарт з вашим додатком у тестовому Minikube або Kind-кластері після успішного проходження CI.

Мета: З’єднати CI-процес із CD, налаштувати доступ до kubeconfig і автоматизувати реліз у Kubernetes.

7. *Налаштування середовища та секретів*

Завдання: Використати HashiCorp Vault або Kubernetes Secret для безпечного підключення бази даних під час деплойменту у CI/CD pipeline.

Мета: Вивчити методи захисту чутливих даних і ін’єкції секретів у build- і release-етапи.

8. *Паралельне виконання тестів та збірок*

Завдання: Переписати pipeline таким чином, щоб юніт-тести, інтеграційні тести та побудова Docker-образу виконувалися одночасно в окремих агентах або runners.

Мета: Оптимізувати час проходження конвеєра через розподілену обробку завдань.

9. *Оповіщення та моніторинг CI/CD*

Завдання: Додати в pipeline кроки для відправки повідомлень у Slack або електронною поштою про успіх чи помилку збірки, а також інтегрувати збір метрик через Prometheus.

Мета: Забезпечити прозорість процесу CI/CD, вчасно інформувати команду про стан конвеєра та аналізувати його ефективність.

7. Самостійна робота студента

Тема 1. Аналіз і порівняння інструментів CI/CD

Дослідіть та порівняйте щонайменше три різні інструменти CI/CD (наприклад, Jenkins, GitLab CI/CD, CircleCI), зосередившись на їх основних характеристиках, перевагах та недоліках.

Тема 2. Встановлення та налаштування Jenkins на власному комп'ютері

Встановіть Jenkins, створіть простий проект і налаштуйте автоматичну збірку проекту з GitHub кожного разу, коли в репозиторій здійснюється push.

Тема 3. Створення Dockerfile для власного проекту

Напишіть Dockerfile для контейнеризації будь-якого простого веб-додатку або скрипта. Побудуйте образ і запустіть його локально.

Тема 4. Розгортання простого застосунку використовуючи Heroku

Використовуючи платформу Heroku, розгорніть простий веб-додаток і налаштуйте автоматичне розгортання з GitHub репозиторію.

Тема 5. Автоматизація тестування з використанням GitLab CI/CD

Створіть простий проект з юніт-тестами. Налаштуйте .gitlab-ci.yml файл для автоматичного запуску тестів при кожному push в репозиторій.

Тема 6. Практика з інфраструктурою як код (IaC) за допомогою Terraform

Використовуючи Terraform, напишіть конфігурацію для створення віртуальної машини в AWS або іншому хмарному провайдері.

Тема 7. Інтеграція та використання ELK Stack для моніторингу логів

Розгорніть Elasticsearch, Logstash та Kibana (ELK Stack) для збору та аналізу логів з вашого проекту або додатку.

Тема 8. Виконання blue-green розгортання за допомогою Kubernetes

Використовуючи мінімальний Kubernetes кластер (наприклад, Minikube), реалізуйте стратегію blue-green розгортання для простого веб-додатку..

Політика та контроль

8. Політика навчальної дисципліни (освітнього компонента)

Відвідування лекційних та лабораторних занять є обов'язковим за винятком поважних причин (хвороби, форс-мажорних обставин).

В разі пропущення занять з поважних причин викладач надає можливість студенту виконати усі або деякі лабораторні завдання (винятком є виконання деяких завдань у зв'язку із закінченням навчального процесу).

В разі пропущення занять без поважних причин, а також через порушення граничного терміну виконання завдання (deadline) студент може отримати 80% від максимальної оцінки відповідне завдання.

Протягом семестру студенти:

- виконують та захищають лабораторні роботи у відповідні терміни (на кожну лабораторну роботу відводиться два тижні для здачі),
- пишуть модульну контрольну роботу,
- повинні позитивно закрити дві атестації (в кінці березня та в середині травня),
- по закінченні навчального процесу складають екзамен.

9. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Система рейтингових (вагових) балів та критерії оцінювання

1) Робота на лекціях

На лекціях може бути проведено бліцопитування студентів. Такі опитування проводяться на довільних лекціях 5 разів протягом семестру, наприкінці лекції. Ваговий бал за вірну відповідь - 1. Максимальна кількість балів, що може отримати кожен студент за семестр - 5.

2) Лабораторні практикуми

Максимальна кількість балів за усі виконані комп'ютерні практикуми дорівнює 60 балів. Розподіл балів серед лабораторних практикумів наступний:

№ з/п	Назва лабораторного практикуму	Кількість балів
1	Робота з Git та реалізація Git Hooks	12
2	Контейнеризація за допомогою Docker	12
3	Використання Docker Compose для оркестрації контейнерів	12
4	Впровадження стратегії неперервного розгортання за допомогою GitLab CI/CD	12
5	Моніторинг та логування в CI/CD процесах	12
Всього:		60

Критерії оцінювання:

Виконання лабораторного практикуму:

- виконаний своєчасно (протягом двох тижнів з моменту видачі), у повному обсязі – відповідний бал згідно номеру комп'ютерного практикуму;
- виконаний із запізненням – знімається 10 – 30% від максимальної кількості балів в залежності від терміну запізнення;
- виконаний не самостійно, із запізненням – знімається 50% від максимальної кількості балів;
- невиконани1 протягом відведеного часу – 0 балів.

3) Модульна контрольна робота

Максимальна кількість балів за модульну контрольну роботу дорівнює 10 балів.

Якість виконання роботи:

- усі відповіді вірні та повні –10 балів,
- у відповідях допущені несуттєві неточності – 8 балів,
- половина відповідей вірна – 5 балів,

– відповіді з суттєвими неточностями, але без критичних помилок – 2 бали,
– менше половини відповідей вірна – 0 балів.

Штрафні та заохочувальні бали за:

- активність на комп'ютерних практикумах + 2 бали
- виконання комп'ютерного практикуму з використанням власного оптимального алгоритму + 1 бали
- відсутність на занятті без поважної причини – 2 бали
- несвоєчасна здача комп'ютерного практикуму (пізніше ніж за тиждень) – 0,5 балів;

4) Складання заліку

Максимальний ваговий бал $r_{icп}$=40

На іспиті студент виконує письмову контрольну роботу, яка містить два теоретичних питання і одне практичне питання. Теоретичні питання оцінюються максимально по 10 балів, практичне – 20 балів.

Умови позитивної проміжної атестації

Для отримання „зараховано" з першої проміжної атестації студент матиме не менше ніж 30 балів (за умови, що за 8 тижнів згідно з календарним планом контрольних заходів „ідеальний" студент має отримати 12 +12 +12 = 36 бал).

Для отримання „зараховано" з другої проміжної атестації студент матиме не менше ніж 50 балів (за умови, що за 14 тижнів згідно з календарним планом контрольних заходів „ідеальний" студент має отримати 36 +12 +12 =60 балів).

Умови допуску до *залікуу*

Необхідною умовою допуску до заліку є зарахування усіх комп'ютерних практикумів та виконання модульної контрольної роботи, а також стартовий рейтинг (R_c) не менше 40 балів.

Розрахунок шкали (R) рейтингу:

Сума вагових балів контрольних заходів протягом семестру (шкала рейтингу) складає:

$$R = r_{лек} + r_{практ} + r_{мод} + r_{icп} = 5+45 + 10 + 40= 100 \text{ балів.}$$

Максимальний стартовий рейтинг становить $R_c = r_{лек} + r_{практ} + r_{мод} = 60$ балів.

Рейтинг заліку дорівнює 40 балів. Мінімальний рейтинг допуску до заліку становить 40 балів.

Таким чином, рейтингова шкала з кредитного модуля складає

$$R = 60+40=100 \text{ балів.}$$

Для отримання студентом відповідних оцінок рейтингова оцінка студента переводиться згідно таблиці:

Бали	**Оцінка**
95 - 100	Відмінно
85 - 94	Дуже добре
75 - 84	Добре
65 - 74	Задовільно
60 - 64	Достатньо
Менше 60	Незадовільно
R < 40 є незараховані роботи комп'ютерного практикуму або не виконані інші умови допуску до екзамену	Не допущено

1. Додаткова інформація з дисципліни (освітнього компонента)

Перелік питань, які виносяться на семестровий контроль:

1. Що таке неперервна інтеграція (CI)? Опишіть основні принципи.
2. Що таке неперервне розгортання (CD)? Як воно відрізняється від неперервної доставки?
3. Які переваги впровадження CI/CD для проекту з розробки програмного забезпечення?
4. Назвіть та описіть функціонал трьох інструментів для реалізації CI/CD.
5. Які основні компоненти CI/CD пайплайну?
6. Що таке "pipeline as code"? Наведіть приклад.
7. Опишіть процес налаштування автоматичних тестів у CI/CD пайплайні.
8. Як контейнеризація впливає на процеси CI/CD?
9. Опишіть стратегію blue-green розгортання. Які її переваги?
10. Які виклики можуть виникати під час впровадження CI/CD?
11. Які методи забезпечення безпеки ви знаєте для CI/CD пайплайну?
12. Що таке IaC (Infrastructure as Code) і яку роль він відіграє в CI/CD?
13. Назвіть приклади інструментів для IaC та описіть їх основні функції.
14. Як здійснюється моніторинг і логування в CI/CD?
15. Опишіть процес ручного vs автоматичного розгортання. Коли варто використовувати кожен з них?
16. Як використання мікросервісної архітектури впливає на CI/CD пайплайн?
17. Як забезпечити високу доступність та масштабованість CI/CD інфраструктури?
18. Що таке Canary releases? Як вони впроваджуються у CI/CD?
19. Як управління конфігурацією інтегрується з CI/CD?
20. Як вибрати між самостійним розгортанням CI/CD інструментів проти використання хмарних сервісів?
21. Які критерії ви б використали для оцінки ефективності CI/CD пайплайну?
22. Які практики DevOps можуть підтримувати успішне впровадження CI/CD?
23. Що таке GitOps і як воно пов'язане з CI/CD?
24. Як зміни в коді просуваються через CI/CD пайплайн від розробки до продакшну?.

Складено старшим викладачем ІПЗЕ, Колумбетом Вадимом Петровичем

Ухвалено кафедрою ІПЗЕ (протокол №42 від 18.06.2025 р.)

Погоджено Методичною комісією факультету (протокол № 09 від 27.06.2025 р.)