

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

«ЗАТВЕРДЖУЮ»

Заступник декана з навчальної роботи

_____ Світлана ЄФІМЕНКО

« ____ » _____ 2025 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Мережеві операційні системи

для студентів

галузь знань	F “Інформаційні технології”
спеціальність	F7 “Комп'ютерна інженерія”
рівень вищої освіти	другий
освітньо-наукова програма	Комп'ютерні системи та мережі
вид дисципліни	обов'язкова

Форма навчання	денна
Навчальний рік	2025/2026
Семестр	2
Кількість кредитів ECTS	6
Мова викладання	українська
Форма заключного контролю	іспит

Викладач:

Євген СЛЮСАР, кандидат техн. наук, асистент кафедри комп'ютерної інженерії

Пролонговано: на 20__/20__ н. р. _____ (_____) « ____ » _____ 20__ р.

на 20__/20__ н. р. _____ (_____) « ____ » _____ 20__ р.

Розробник:

Євген СЛЮСАР, кандидат техн. наук, асистент кафедри комп'ютерної інженерії

«ЗАТВЕРДЖЕНО»

Завідувач кафедри комп'ютерної інженерії

_____ **Юрій БОЙКО**

Протокол № ____ від « ____ » _____ 2025 р.

Схвалено науково-методичною комісією факультету радіофізики, електроніки та комп'ютерних систем

Протокол № ____ від « ____ » _____ 2025 р.

Голова науково-методичної комісії

Сергій РАДЧЕНКО

« ____ » _____ 2025 року.

ВСТУП

1. Мета дисципліни — ознайомлення студентів із основними принципами функціонування систем керування конфігураціями, віртуалізації та контейнеризації і застосування їх для розгортання ІТ-інфраструктур як в хмарних середовищах, так і на локальному апаратному забезпеченні.

2. Попередні вимоги до опанування або вибору навчальної дисципліни:

Навчальна дисципліна «Мережеві операційні системи» ґрунтується на знаннях та вміннях базової професійної та практичної підготовки бакалавра галузі знань «Інформаційні технології», а саме: розуміння принципів побудови комп'ютерних систем, знання протоколів та функціонування комп'ютерних мереж та сервісів.

Попередні вимоги:

студент повинен знати: основні принципи побудови та характеристики апаратного та програмного забезпечення ЕОМ, концептуальні моделі та стандартні протоколи комп'ютерних мереж та сервісів, принципи роботи систем віртуалізації на рівні випускника бакалаврату Київського національного університету імені Тараса Шевченка.

студент повинен вміти: здійснювати інсталяцію, адміністрування та обслуговування ОС та прикладного ПЗ, виявляти та усувати завади в їх роботі, користуватись засобами командного рядка UNIX та Windows, створювати Shell-сценарії для автоматизації рутинних задач на рівні випускника бакалаврату Київського національного університету імені Тараса Шевченка.

3. Анотація навчальної дисципліни:

Дисципліна «Мережеві операційні системи» знайомить студентів із сучасними методами та технологіями розгортання ІТ-інфраструктур на базі хмарних сервісів та на власному апаратному забезпеченні. Сучасний підхід DevOps передбачає запровадження принципу «інфраструктура як код» та застосування відповідного інструментарію — систем керування конфігураціями, які застосовують декларативний опис для приведення існуючих об'єктів інфраструктури до бажаного стану незалежно від поточного їх стану. В курсі розглядаються такі системи як Ansible, Puppet, Chef та Terraform.

Динамічне масштабування веб-сервісів потребує створення та видалення великої кількості примірників, для чого часто застосовується технологія контейнерів, яка має окремий інструментарій керуючого ПЗ — менеджери та оркестратори. В курсі розглядаються низькорівневі механізми ядра Linux для організації контейнерів, менеджер Docker та оркестратор Kubernetes.

Побудова хмари на базі власних апаратних ресурсів дозволяє реалізовувати гібридний підхід та керувати ресурсами в публічних хмарах та на місці уніфікованим набором інструментів. Як засіб створення приватної хмари в курсі розглянуто OpenStack.

4. Завдання навчальної дисципліни (навчальні цілі):

1. Надати основні відомості з автоматизації розгортання та масштабування ресурсів ІТ-інфраструктури в рамках підходу DevOps та принципу «інфраструктура як код», які складають важливу частину підготовки студента за спеціальністю «Комп'ютерна інженерія».

2. Поглибити знання та навички студента в області технологій віртуалізації, зокрема — контейнерної віртуалізації, та закріпити на практиці навички роботи з інструментарієм збірки образів контейнерів.

3. Навчити застосовувати основні відомості курсу у професійній діяльності, зокрема — застосовувати на практиці програмні засоби інструментарію керування конфігураціями для розгортання типових проєктів та середовищ, таких як веб-сайт на заданому стеку технологій.

4. Навчити застосовувати отримані знання та уміння при проєктуванні, розгортанні, обслуговуванні ІТ-інфраструктур та розробці додатків під створені середовища.

Дисципліна спрямована на формування програмних компетентностей:

ЗК1. Здатність до адаптації та дій в новій ситуації.

СК4. Здатність будувати та досліджувати моделі комп'ютерних систем та мереж.

СК7. Здатність досліджувати, розробляти та обирати технології створення великих і надвеликих систем.

СК8. Здатність забезпечувати якість продуктів і сервісів інформаційних технологій на протязі їх життєвого циклу.

СК11. Здатність обирати ефективні методи розв’язування складних задач комп’ютерної інженерії, критично оцінювати отримані результати та аргументувати прийняті рішення.

5. Результати навчання за дисципліною:

Результат навчання (1, знати; 2, вміти; 3, комунікація; 4, автономність та відповідальність)		Форми (та/або методи і технології) викладання і навчання	Методи оцінювання та пороговий критерій оцінювання (за необхідності)	Відсоток у підсумковій оцінці з дисципліни
Код	Результат навчання			
1	знати:	лекційні заняття	іспит	до 30
1.1	Загальні передумови та засади підходу DevOps до організації робочих процесів в IT-галузі	лекція	іспит	5
1.2	Загальну специфіку декларативного та імперативного підходу до керування станом IT-інфраструктур	лекція	іспит	4
1.3	Загальні підходи до автоматизованого керування конфігураціями IT-інфраструктур	лекція	іспит	5
1.4	Основні види технологій віртуалізації в комп’ютерних системах	лекція	іспит	3
1.5	Загальна специфіка застосування контейнерної віртуалізації в комп’ютерних системах	лекція	іспит	5
1.6	Загальна специфіка застосування хмарних технологій для побудови IT-інфраструктур	лекція	іспит	4
1.7	Механізми оркестрації контейнерних середовищ	лекція	іспит	4
2	вміти:	лекційні демонстрації та лабораторні роботи	захист звітів лабораторних робіт	до 60
2.1	Описувати конфігурацію IT-інфраструктури декларативною мовою під задану систему керування конфігураціями	лабораторні роботи	звіти з лабораторних робіт	12
2.2	Розгортати інфраструктуру віртуальних машин та сервісів в хмарному середовищі	лекційні демонстрації та лабораторні роботи	звіти з лабораторних робіт	12
2.3	Здійснювати розгортання та базові налаштування сервісів систем керування конфігураціями Ansible, Puppet, Chef та Terraform	лекційні демонстрації та лабораторні роботи	звіти з лабораторних робіт	12
2.4	Створювати образи контейнерів прикладних додатків засобами Docker	лабораторні роботи	звіти з лабораторних робіт	12
2.5	Розгортати веб-додатки в середовищі Kubernetes	лабораторні роботи	звіти з лабораторних робіт	12
3	комунікація:	лекційні заняття та лабораторні роботи	захист звітів лабораторних робіт	до 5
3.1	Здатність будувати грамотну науково-технічну комунікацію в усній та письмовій формах, підбирати правильну термінологію	лекція	звіти з лабораторних робіт	3
3.2	Здатність до командної роботи у промислових IT-проєктах	лекція	звіти з лабораторних робіт	2
4	автономність та відповідальність:	лекційні демонстрації та лабораторні роботи	захист звітів лабораторних робіт	до 5
4.1	Здатність до самостійного пошуку технічної літератури або інших джерел інформації для пошуку шляхів усунення помилок студентом під час виконання лабораторних робіт	лекційні демонстрації та лабораторні роботи	звіти з лабораторних робіт	5

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Результати навчання дисципліни Програмні результати навчання (назва)	Код														
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	2.1	2.2	2.3	2.4	2.5	3.1	3.2	4.1
РН3. Будувати та досліджувати моделі комп'ютерних систем і мереж, оцінювати їх адекватність, визначати межі застосовності.		+	+	+	+	+				+	+	+			
РН6. Аналізувати проблематику, ідентифікувати та формулювати конкретні проблеми, що потребують вирішення, обирати ефективні методи їх вирішення.	+				+			+					+		+
РН11. Приймати ефективні рішення з питань розроблення, впровадження та експлуатації комп'ютерних систем і мереж, аналізувати альтернативи, оцінювати ризики та імовірні наслідки рішень.		+	+				+		+					+	+

7. Схема формування оцінки

7.1. Форми оцінювання

Рівень досягнення всіх запланованих результатів навчання визначається за результатами виконання та захисту звітів лабораторних робіт, а також за результатами підсумкового іспиту. Внесок результатів навчання у підсумкову оцінку, за умови їх опанування на належному рівні:

- результати навчання 1.1 – 1.7 [знання] – до 30%;
- результати навчання 2.1 – 2.5 [вміння] – до 60%;
- результати навчання 3.1 – 3.2 [комунікація] – до 5%;
- результат навчання 4.1 [автономність та відповідальність] – до 5%;

Форми оцінювання:

- **семестрове оцінювання:** Навчальний семестр має 6 лабораторних робіт. Після завершення виконання лабораторних робіт та написання звітів проводяться усні захисти звітів із демонстраціями коректного виконання завдань. Обов'язковим для допуску до іспиту є: захист звіту з кожної лабораторної роботи з кількістю балів не менше 6.
- **підсумкове оцінювання (у формі іспиту):** форма іспиту – письмове електронне тестування. Тест складається з 30 питань із варіантами відповіді, які оцінюються по 1 балу за кожне, та 2 відкритих питань, які оцінюються по 5 балів. Всього за іспит можна отримати від 0 до 40 балів. Умовою досягнення позитивної оцінки за дисципліну є отримання не менш ніж 60 балів, оцінка за іспит не може бути меншою за **24 бали**.
- **умови допуску до підсумкового іспиту:** умовою допуску до іспиту є отримання студентом сумарно не менше, ніж *критично-розрахунковий мінімум* за семестр. Студенти, які протягом семестру сумарно набрали меншу кількість балів, ніж критично-розрахунковий мінімум **36 балів**, для одержання допуску до іспиту обов'язково повинні виконати додаткове завдання за тематикою лабораторних робіт.

У випадку відсутності студента з поважних причин відпрацювання та захист пропущених лабораторних робіт здійснюються у відповідності до «Положення про організацію освітнього процесу у Київському національному університеті»

7.2. Організація оцінювання

Оцінювання за формами контролю:

Семестрова робота	Кількість балів	
	Min. – 6	Max. – 10
Лабораторна робота №1	6	10
Лабораторна робота №2	6	10
Лабораторна робота №3	6	10
Лабораторна робота №4	6	10
Лабораторна робота №5	6	10
Лабораторна робота №6	6	10
Додаткові завдання	–	12

Орієнтований графік оцінювання:

Форма оцінювання	Орієнтовний період для здійснення відповідної форми оцінювання
Лабораторні роботи №1 та №2	березень
Лабораторні роботи №2 та №3	квітень
Лабораторні роботи №5 та №6	травень
Добір балів/додаткові завдання	травень-червень
Іспит	червень

Розрахунок балів, які отримують при успішній здачі іспиту:

Значення	Семестрова робота	Іспит	Підсумкова оцінка
Мінімум	36	24	60
Максимум	60	40	100

7.3. Шкала відповідності оцінок

Оцінка (за національною шкалою) / National grade	Рівень досягнень, % / Marks, %
Зараховано / Passed	60-100%
Не зараховано / Fail	0-59%

8. Структура навчальної дисципліни

8.1. Тематичний план занять

№ з/п	Назва теми	Кількість годин		
		Лекції	Лабораторні роботи	Самостійна робота
1	Вступ до курсу та основні засади та принципи DevOps	2	—	15
2	Принцип «інфраструктура як код» на прикладі Terraform.	2	8	15
3	Безагентні системи керування конфігураціями на прикладі Ansible.	2	8	15
4	Централізовані системи керування конфігураціями. Система Puppet.	2	8	15
5	Універсальні системи керування конфігураціями. Автономний режим на прикладі Chef.	2	8	15
6	Створення контейнерів засобами ОС. Приклад ядра Linux.	2	—	15
7	Збірка та запуск контейнерів засобами Docker. Оркестрація контейнерів на прикладі Kubernetes.	2	12	15
8	Створення приватної хмари на прикладі OpenStack.	2	—	15
Всього		16	44	120

Загальний обсяг **180** год., в тому числі:

Лекції **16** год.

Лабораторні роботи **44** год.

Самостійна робота **120** год.

8.2. Лабораторні заняття

№ з/п	Назва теми	Кількість годин
1	Масове створення віртуальних машин засобами Terraform	8
2	Розгортання веб-додатку в хмарному середовищі засобами Ansible	8
3	Розгортання веб-серверів Linux та Windows засобами Puppet	8
4	Розгортання Ruby-додатка за допомогою системи Chef	8
5	Створення та публікація образу контейнера веб-додатка засобами Docker	6
6	Розгортання набору контейнерів у середовищі minikube	6
Всього		44

9. Рекомендована література:

Основні джерела:

1. Kief Morris. Infrastructure as Code, 2nd Edition. — O'Reilly Media Inc., 2020. — 350 pp.
2. Yevgeniy Brikman. Terraform: Up and Running: Writing Infrastructure As Code, 2nd Edition. — O'Reilly Media Inc., 2019. — 360 pp.
3. Yan Kurniawan. Ansible for AWS. — Packt Publishing, 2016. — 354 pp.
4. Navin Sabharwal, Manak Wadhwa. Automation through Chef Opscode: A Hands-on Approach to Chef. — Apress, 2014. — 264 pp.
5. Gigi Sayfan. Mastering Kubernetes – Third Edition: Level Up Your Container Orchestration Skills with Kubernetes to Build, Run, Secure, and Observe Large-scale Distributed Apps. — Packt Publishing, 2020. — 642 pp.
6. Thomas Uphill, John Arundel, Neependra Khare. DevOps: Puppet, Docker, and Kubernetes. — Packt Publishing, 2017. — 924 pp.

Додаткові і джерела:

1. Shashank Mohan Jain. Linux Containers and Virtualization: A Kernel Perspective. — Apress, 2020. — 148 pp.
2. Gene Kim, Jez Humble, Patrick Debois, John Willis. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. — IT Revolution Press, 2016. — 480 pp.
3. Tom Fifield, Diane Fleming, Anne Gentle et al. OpenStack Operations Guide: Set Up and Manage Your OpenStack Cloud. — O'Reilly Media Inc., 2014. — 330 pp.
4. Jez Humble, David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. — Addison-Wesley, 2010. — 512 pp.
5. Mikolaj Pawlikowski. Chaos Engineering: Site reliability through controlled disruption. — Simon and Schuster, 2021. — 424 pp.

ДОДАТОК 1

Основні тези лекційних занять

Лекція 1. Вступ до курсу та основні засади та принципи DevOps.

- Знайомство, структура курсу та система оцінювання.
- Історичний розвиток підходів до організації життєвого циклу ПЗ.
- Основні функції ролей розробника та адміністратора/операціоніста.
- Сучасні вимоги ІТ-бізнесу як чинник виникнення DevOps-підходу.
- Мотивація до автоматизації життєвого циклу ПЗ.
- Замкнений цикл виробництва DevOps, його застосовність та переваги.

Лекція 2. Принцип «інфраструктура як код» на прикладі Terraform.

- Огляд складових частин інфраструктури ІТ-проєкту.
- Імперативний підхід до керування інфраструктурою та його недоліки
- Формалізація опису стану інфраструктури, поточний та бажаний стан.
- Декларативний підхід до керування інфраструктурою.
- Приклад системи Terraform: ресурси, провайдери і стан.
- Демонстрація роботи системи Terraform.

Лекція 3. Безагентні системи керування конфігураціями на прикладі Ansible.

- Загальна характеристика системи Ansible.
- Складові частини конфігурації: плейбуки, ролі, хендлери.
- Інтеграція з хмарою: динамічний інвентар, модулі.
- Редагування конфігураційних файлів. Інструмент Augeas.
- Демонстрація на прикладі розгортання обчислювального кластера.

Лекція 4. Централізовані системи керування конфігураціями. Система Puppet.

- Загальна характеристика системи Puppet.
- Компоненти Puppet: сервер та агент.
- Доставка конфігурації – схеми push та pull.
- Складові частини конфігурації: ресурси, каталог.
- Процедура розгортання Puppet. Генерація ключів.
- Демонстрація.

Лекція 5. Універсальні системи керування конфігураціями. Автономний режим на прикладі Chef.

- Загальна характеристика системи Chef.
- Компоненти Chef: сервер, робоча станція, агент, «супермаркет».
- Складові частини конфігурації: ресурс, рецепт, книга рецептів.
- Розробка конфігурації для Chef. Утиліта knife.
- Соло-режим на прикладі Omnibus GitLab. Демонстрація.

Лекція 6. Створення контейнерів засобами ОС. Приклад ядра Linux.

- Визначення контейнера. Порівняння із віртуальною машиною.
- Обмеження та облік споживання ресурсів: групи контролю в ядрі Linux.
- Ізоляція груп процесів та створення різних представлень ресурсів системи для них: простори імен в ядрі Linux.
- Метод нашарування файлових систем: спадкування та відділення тимчасового стану від незмінної основи.
- Демонстрація роботи засобами unshare та systemd.

Лекція 7.1. Збірка та запуск контейнерів засобами Docker.

- Контейнери для розробників додатків. Основні ідеї Docker.
- Архітектура системи Docker та основні сценарії використання.
- Взаємодія з реєстром образів контейнерів.
- Збірка образів контейнерів засобами Docker. Демонстрація.

Лекція 7.2. Оркестрація контейнерів на прикладі Kubernetes.

- Необхідність оркестрації для масштабування та розгортання складних додатків.
- Контейнерний оркестратор Kubernetes: історія та функції.
- Керування ресурсами: поди, контейнери та оператори.
- Розгортання додатку в Kubernetes: інструмент Helm. Демонстрація.
- Одержання вхідного веб-трафіку: вхідний контролер.

Лекція 8. Створення приватної хмари на прикладі OpenStack.

- Базові складові хмарної інфраструктури з точки зору провайдера.
- Історія виникнення та розвиток OpenStack.
- Основні сервіси OpenStack-інфраструктури та їх функції.
 - Служби обчислень – віртуальні машини та контейнери
 - Служби сховищ – файлові, блокові та об'єктні
 - Мережева інфраструктура
 - Автентифікація та авторизація
 - Оркестрація та телеметрія
- Розробка під OpenStack. Середовище DevStack.