

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

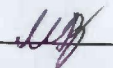
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект системи автоматизованого розгор-
тання конфігурації корпоративної мережі з використан-
ням практик DevNet/DevOps»

Виконав: студент 4-го курсу групи КІ-22-1
спеціальності 123 «Комп'ютерна інженерія»

 Тарас КУЛЬЧИЦЬКИЙ

Керівник: старший викладач кафедри КІ та КБ

 Вадим МИКОЛАЙЧУК

Рецензент: професор кафедри КІ та КБ, д.т.н., доцент

 Володимир ВОРОТНИКОВ

Житомир – 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

2 лютого 2026 р.

ЗАВДАННЯ

на випускн у кваліфікаційну роботу бакалавра

Студент: Кульчицький Тарас Володимирович

Тема роботи: «Проект системи автоматизованого розгортання конфі-гурації ко-
рпоративної мережі з використанням практик DevNet/DevOps»

затверджена Наказом університету від 2 лютого 2026 р. № 57/с

Термін здачі студентом закінченої роботи 30 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

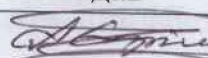
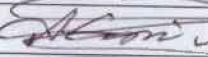
Процес управління конфігурацією корпоративної мережевої інфраструк-
тури з використанням практик DevNet/DevOps та парадигми Network as Code;

Методи та засоби проектування, автоматизації та адміністрування комп'-
ютерних мереж із застосуванням інструментів Ansible та платформи GitLab
CI/CD.

Обладнання та програмне забезпечення:

1. Серверна операційна система Linux Ubuntu Server 26.04 LTS
2. Середовище віртуалізації гіпервізор Oracle VirtualBox
3. Платформа емуляції GNS3 та GNS3 VM
4. Оригінальні образи операційних систем Cisco IOS та Cisco IOU.
5. GitLab CE та GitLab Runner
6. Ansible Core; офіційна колекція модулів cisco.ios; криптографічний ути-
літа Ansible Vault для захисту секретів

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А.А.	16.03.2026 	16.03.2026 
2	Єфіменко А.А.	09.04.2026 	09.04.2026 
3	Єфіменко А.А.	28.05.2026 	28.05.2026 

Студент  Тарас КУЛЬЧИЦЬКИЙ
Керівник  Вадим МИКОЛАЙЧУК

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	06.04.2026	Виконано
2	Пошук та аналіз літературних джерел	10.04.2026	Виконано
3	Проектування середовища GNS3	11.04.2026	Виконано
4	Розгортання DevOps-інфраструктури	10.05.2026	Виконано
5	Реалізація життєвого циклу мережевих змін через GitLab CI/CD	25.05.2026	Виконано
6	Тестування комплексного середовища	19.06.2026	Виконано
7	Формування пояснювальної записки	25.06.2026	Виконано
8	Захист	30.06.2026	Виконано

Студент  Тарас КУЛЬЧИЦЬКИЙ

Керівник  Вадим МИКОЛАЙЧУК

АНОТАЦІЯ

Кваліфікаційна робота на тему «Проект системи автоматизованого розгортання конфігурації корпоративної мережі з використанням практик DevNet/DevOps» присвячена розробці системи, що забезпечує автоматизоване, відтворюване та контрольоване управління конфігурацією мережевої інфраструктури підприємства. Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Текстова частина викладена на 67 сторінках друкованого тексту. Пояснювальна записка містить 20 додатків. Список використаних джерел містить 34 найменування і займає 3 сторінки. В роботі наведено 40 рисунків, 8 таблиць. Загальний обсяг роботи 94 сторінки.

Метою кваліфікаційної роботи є проектування та практична реалізація системи автоматизованого розгортання конфігурації корпоративної мережі з використанням практик DevNet/DevOps на основі парадигми Network as Code. Для досягнення поставленої мети було визначено наступні завдання:

Дослідити теоретичні основи побудови корпоративних мереж за трирівневою ієрархічною моделлю Cisco та методологію NetDevOps;

Спроектувати ієрархічну топологію корпоративної мережі, структурований адресний план і протокольний стек у середовищі емуляції GNS3;

Розробити модульну структуру Ansible-проекту з ролями для автоматизованого розгортання повного стеку мережевих конфігурацій;

Реалізувати конвеєр CI/CD на базі GitLab CE та відпрацювати повний цикл внесення змін через Merge Request;

Верифікувати коректність стану мережі та підтвердити ідемпотентність розробленої системи автоматизації.

В ході роботи було спроектовано повну топологію корпоративної мережі з восьми пристроїв Cisco у середовищі GNS3, розроблено структурований адресний

ККІКБ.КР.Б – 123 – 09 – ПЗ

Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кульчицький Т.В.		29.06	Проект системи автоматизованого розгортання конфігурації корпоративної мережі з використанням практик DevNet/DevOps	Літ.	Арк.	Аркушів
Керівник		Миколайчук В.В.		29.06			4	94
Реценз.		Воротніков В.В.		29.06		ФІКТ, Гр. КІ-22-1		
Н. контр.		Миколайчук В.В.		29.06				
Затверд.		Сфіменко А.А.		29.06				

план із синхронізацією VLAN-ID та третього октету IP-адреси; розгорнуто DevOps-інфраструктуру на базі GitLab CE із Shell-runner та Ansible Core з колекцією cisco.ios; розроблено шість Ansible-ролей (base, vlans, routing, hsrp, dhcp, security), що охоплюють повний стек конфігурації мережевого обладнання; реалізовано чотиристадійний конвеєр CI/CD (lint → dry-run → deploy-staging → deploy-production); практично відпрацьовано повний цикл внесення змін через Merge Request та підтверджено ідемпотентність системи за результатами верифікаційного плейбука verify.yml. Запропонована система є ефективною та відповідає сучасним вимогам до управління конфігурацією корпоративних мереж, а результати роботи мають значний потенціал для впровадження на реальних підприємствах.

КЛЮЧОВІ СЛОВА: КОРПОРАТИВНА МЕРЕЖА, ANSIBLE, DEVOPS, DEVNET, NETWORK AS CODE, GITLAB CI/CD, CISCO IOS, GNS3, OSPF, VLAN, HSRP, АВТОМАТИЗАЦІЯ КОНФІГУРАЦІЙ, INFRASTRUCTURE AS CODE.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ANNOTATION

The qualification work on the topic "Project of an automated configuration deployment system for a corporate network using DevNet/DevOps practices" is dedicated to the development of a system that ensures automated, reproducible and controlled management of enterprise network infrastructure configuration. The explanatory note consists of an introduction, three sections, conclusions, a list of sources used and appendices.

The text part is set out on 67 pages of printed text. The explanatory note contains 20 appendices. The list of sources used contains 34 items and occupies 3 pages. The work contains 40 figures and 8 tables. The total volume of work is 94 pages.

The goal is to design and practically implement an automated configuration deployment system for a corporate network using DevNet/DevOps practices based on the Network as Code paradigm. During the work, a complete eight-device Cisco network topology was designed in the GNS3 environment; a DevOps infrastructure was deployed based on GitLab CE with a Shell runner and Ansible Core with the cisco.ios collection; six Ansible roles covering the full configuration stack were developed; a four-stage CI/CD pipeline was implemented and system idempotency was confirmed by the verify.yml verification playbook.

KEYWORDS: CORPORATE NETWORK, ANSIBLE, DEVOPS, DEVNET, NETWORK AS CODE, GITLAB CI/CD, CISCO IOS, GNS3, OSPF, VLAN, HSRP, CONFIGURATION AUTOMATION, INFRASTRUCTURE AS CODE.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ МЕРЕЖЕВИХ КОНФІГУРАЦІЙ.....	11
1.1 Принципи побудови корпоративних мереж за тривірневою ієрархічною моделлю (Three-Tier Architecture)	11
1.2 Огляд методологій DevNet та DevOps у контексті мережевої інженерії	14
РОЗДІЛ 2 ПРОЄКТУВАННЯ МЕРЕЖЕВОЇ ІНФРАСТРУКТУРИ ТА СИСТЕМИ АВТОМАТИЗАЦІЇ	19
2.1 Проєктування ієрархічної топології корпоративної мережі та вибір платформи емуляції	19
2.2 Аналіз, порівняння та обґрунтування протокольного стеку мережі	23
2.3 Обґрунтування архітектури автоматизації управління конфігураціями	29
2.4 Проєктування конвеєра CI/CD та моделі розгалуження Git	34
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	41
3.1 Розгортання середовища та DevOps-інфраструктури	41
3.2 Програмна реалізація плейбуків та ролей Ansible для обладнання Cisco ..	48
3.3 Налаштування конвеєра CI/CD та практична перевірка сценаріїв роботи ..	55
3.4 Тестування, верифікація та аналіз результатів	62
ВИСНОВКИ	65
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	67
ДОДАТКИ	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система
ПЗ – програмне забезпечення
ACL – Access Control List
CI – Continuous Integration
CD – Continuous Delivery
CLI – Command Line Interface
DHCP – Dynamic Host Configuration Protocol
DNS – Domain Name System
GNS3 – Graphical Network Simulator 3
HSRP – Hot Standby Router Protocol
IaC – Infrastructure as Code
IOS – Internetwork Operating System
MR – Merge Request
NAT – Network Address Translation
NaC – Network as Code
NTP – Network Time Protocol
OSPF – Open Shortest Path First
PoE – Power over Ethernet
SSH – Secure Shell
VLAN – Virtual Local Area Network
VLSM – Variable Length Subnet Masking
YAML – YAML Ain't Markup Language

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Проектування та розробка сучасної, надійної та автоматизованої мережевої інфраструктури для підприємств є надзвичайно актуальним завданням у контексті сучасних технологічних вимог. Зростаюча складність корпоративних мереж, що включають десятки та сотні пристроїв, вимагає створення рішень, здатних забезпечити швидке, відтворюване та захищене розгортання конфігурацій без ризику виникнення людських помилок.

Традиційний підхід до адміністрування мережевої інфраструктури, що базується на ручному введенні команд через інтерфейс командного рядка, на сучасному етапі демонструє критичну неефективність. Основними недоліками такого підходу є висока ймовірність помилок через людський фактор, відсутність системного контролю версій конфігурацій, тривалий час розгортання нових сервісів та складність проведення аудиту змін. Трансформація підходів до побудови IT-інфраструктури зумовила проникнення практик розробки програмного забезпечення у сферу мережевих технологій, що сформувало нову парадигму, відому як NetDevOps. Впровадження принципу Network as Code, за якого конфігурація мережевого обладнання зберігається у вигляді структурованих файлів даних у централізованій системі контролю версій, є актуальним та практично значущим завданням для сучасних підприємств.

Метою кваліфікаційної роботи є розробка та практична реалізація системи автоматизованого розгортання конфігурації корпоративної мережі з використанням практик DevNet/DevOps на основі парадигми Network as Code.

Визначена мета обумовлює наступні завдання:

- дослідити теоретичні основи побудови корпоративних мереж за трирівневою ієрархічною моделлю Cisco та методологію NetDevOps;
- спроектувати ієрархічну топологію корпоративної мережі, структурований адресний план і протокольний стек у середовищі емуляції GNS3;
- розробити модульну структуру Ansible-проекту з ролями для автоматизованого розгортання повного стеку мережевих конфігурацій;

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

- реалізувати конвеєр CI/CD на базі GitLab CE та відпрацювати повний цикл внесення змін через Merge Request;
- верифікувати коректність стану мережі та підтвердити ідемпотентність розробленої системи автоматизації.

Об'єктом дослідження є процес управління конфігурацією корпоративної мережевої інфраструктури.

Предметом дослідження є методи, засоби та технології автоматизованого розгортання мережевих конфігурацій на базі практик DevNet/DevOps, зокрема інструменти Ansible, GitLab CI/CD та парадигма Network as Code.

У роботі проаналізовано технології автоматизації мережевої інфраструктури, розроблено архітектуру корпоративної мережі з підбором необхідного обладнання та схемою адресації, проведено моделювання та тестування системи у середовищі GNS3, підтвердивши її функціональність та ефективність.

У процесі виконання кваліфікаційної роботи було розроблено проєкт системи автоматизованого розгортання конфігурації корпоративної мережі. У межах цього проєкту особлива увага була приділена аналізу технологій DevNet/DevOps, розробці ієрархічної архітектури мережі на базі трирівневої моделі Cisco, побудові модульної структури Ansible-ролей та реалізації повноцінного конвеєра CI/CD на базі GitLab CE. Розроблена система демонструє здатність задовольняти вимоги сучасного підприємства щодо автоматизованого, контрольованого та ідемпотентного управління мережевою інфраструктурою.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ МЕРЕЖЕВИХ КОНФІГУРАЦІЙ

1.1 Принципи побудови корпоративних мереж за трирівневою ієрархічною моделлю (Three-Tier Architecture)

Проектування сучасної мережевої інфраструктури корпоративного рівня вимагає системного підходу до забезпечення масштабованості, відмовостійкості, безпеки та керованості інформаційних потоків. Одним із найбільш ефективних та перевірених практикою підходів до вирішення цих завдань є використання класичної трирівневої ієрархічної моделі корпоративної мережі, яка була розроблена та стандартизована компанією Cisco Systems [1].

Перехід від «пласких» мереж, де всі пристрої функціонують в одному широкомовному домені без чіткого логічного розділення, до ієрархічних структур дозволяє мінімізувати вплив широкомовного шторму, локалізувати технічні збої, спростити процес пошуку несправностей та закласти надійну основу для впровадження сучасних практик автоматизації управління конфігураціями, таких як Infrastructure as Code [1], [2].

Ієрархічний підхід передбачає логічний та фізичний розподіл всієї мережевої інфраструктури підприємства на три окремі функціональні блоки: рівень доступу, рівень розподілу або агрегації та рівень ядра [1]. Кожен із цих рівнів виконує суворо визначений набір функцій та висуває специфічні вимоги до обладнання. Приклад трирівневої архітектури на рисунку 1.1.

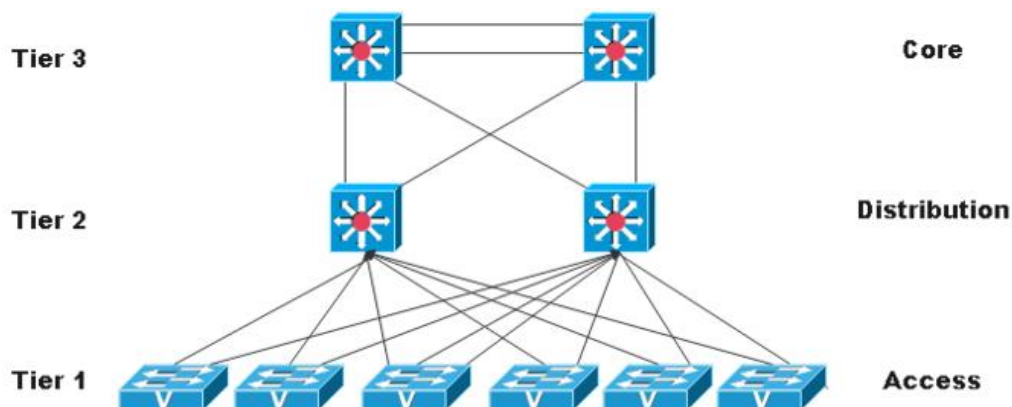


Рисунок 1.1 – Класична трирівнева ієрархічна модель корпоративної мережі

Рівень доступу є безпосередньою точкою входу для кінцевих користувацьких пристроїв у корпоративну мережеву інфраструктуру [1]. До комутаторів цього рівня підключаються робочі станції персоналу, IP-телефони, мережеві принтери, точки доступу Wi-Fi та інші периферійні пристрої. Основним завданням Access-рівня є забезпечення швидкого, надійного та безпечного підключення кінцевих вузлів, а також первинна комутація на другому рівні моделі OSI [1], [3].

У межах автоматизованого розгортання на комутаторах рівня доступу конфігуруються такі технології:

- Сегментація VLAN: логічне розділення користувачів, серверів, IP-телефонії та інфраструктури управління на окремі ізольовані широкомовні домени [2], [4].
- Безпека портів: обмеження кількості MAC-адрес на фізичному інтерфейсі комутатора та активація режимів реагування на порушення, що запобігає несанкціонованому підключенню стороннього обладнання [2], [4].
- Підтримка Spanning Tree Protocol: захист від утворення комутаційних петель на рівні доступу, увімкнення функцій PortFast для миттєвого переходу портів кінцевих пристроїв у стан пересилання та BPDU Guard для захисту від випадкового підключення сторонніх комутаторів [2].

Комутатори рівня доступу зазвичай є найбільш масовим обладнанням у мережі, тому типізація та стандартизація їх конфігурацій за допомогою Ansible-ролей дозволяє звести до нуля ймовірність людської помилки під час ручного налаштування десятків або сотень інтерфейсів [4].

Рівень розподілу/агрегації функціонує як сполучна ланка між масовим рівнем доступу та високошвидкісним ядром мережі [1]. Обладнання цього рівня здійснює агрегацію фізичних каналів, що йдуть від комутаторів Access-рівня, та виконує основний обсяг операцій з логічної маршрутизації трафіку в межах корпоративної мережі підприємства [1], [3].

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Основними функціями рівня розподілу є:

- Маршрутизація між VLAN: комутатори Distribution-рівня виступають в ролі шлюзів за замовчуванням для підмереж рівня доступу та забезпечують маршрутизацію трафіку між ними [2].
- Забезпечення відмовостійкості: впровадження протоколів резервування шлюзів, таких як HSRP або VRRP, для створення віртуального отказостійкого шлюзу на базі двох фізичних комутаторів рівня розподілу [1], [2].
- Агрегація каналів: логічне об'єднання кількох фізичних інтерфейсів в один високошвидкісний магістральний канал для збільшення пропускної здатності та забезпечення надмірності між Access та Distribution [3].

У межах даного проєкту рівень розподілу сфокусований на виконанні завдань чистої маршрутизації та забезпечення внутрішньої відмовостійкості, без задіяння додаткових фільтруючих ACL-матриць, що максимізує його продуктивність при обробці великих обсягів локального трафіку [2].

Рівень ядра представляє собою магістральну основу всієї корпоративної мережі [1]. Його головне і єдине призначення – максимально швидка, безперебійна та ефективна комутація пакетів і маршрутизація трафіку між великими територіальними зонами мережі, серверним сегментом та зовнішніми шлюзами [1], [3]. Обладнання рівня ядра повинно мати найвищу пропускну здатність та мінімальну затримку при обробці пакетів [3].

Важливою архітектурною особливістю даного проєкту є винесення та налаштування списків контролю доступу безпосередньо на рівень Core [2]. Централізоване розмежування доступу між користувацькими сегментами, серверами та MGMT-зоною на рівні ядра має ряд суттєвих переваг:

1. Спрощення автоматизації IaC: Ansible-плейбуки взаємодіють з обмеженою кількістю центральних Core-пристроїв [2], [4]. Оновлення політик безпеки

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

корпоративної мережі відбувається в одній єдиній Ansible-полі security, що суттєво прискорює проходження CI/CD пайплайнів [2], [5].

2. Зняття навантаження з рівнів розподілу: Комутатори розподілу обробляють трафік на апаратній швидкості без додаткових затримок на побайтову перевірку пакетів за допомогою ACL-правил [1], [2].
3. Консолідація точок контролю: Створюється єдина, прозора матриця доступу, що виключає ризик виникнення конфліктів у конфігураціях при розподіленому ручному або напівавтоматизованому налаштуванні ACL на периферійних пристроях мережі [2], [5].

З погляду автоматизованого управління конфігураціями за допомогою DevOps-практик, трирівнева ієрархічна модель є ідеальною архітектурною моделлю. Вона дозволяє розбити всю інфраструктуру на чіткі групи пристроїв в Ansible Inventory: core, distribution, access [2], [4]. Для кожної групи створюються окремі змінні, які описують параметри конкретного рівня [2], [4]. Це забезпечує повну ідемпотентність розгортання, модульність коду та високу швидкість масштабування корпоративної мережі підприємства [2], [5].

1.2 Огляд методологій DevNet та DevOps у контексті мережевої інженерії

Традиційний підхід до адміністрування мережевої інфраструктури, що базується на ручному введенні команд через інтерфейс командного рядка, на сучасному етапі розвитку інформаційних технологій демонструє критичну неефективність. Основними недоліками класичного CLI-керування є висока ймовірність виникнення помилок через людський фактор, відсутність системного контролю версій конфігурацій, тривалий час розгортання нових сервісів та складність проведення операцій аудиту. Трансформація підходів до побудови IT-інфраструктури зумовила проникнення практик розробки програмного забезпечення у сферу мережевих технологій, що сформувало нову парадигму, відому як NetDevOps[6].

Ключовим технологічним фундаментом методології DevOps у мережевій інженерії є концепція Infrastructure as Code, яка стосовно мереж інтерпретується

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

як Network as Code [6], [7]. Цей підхід передбачає, що вся конфігурація мережевого обладнання описується не у вигляді розрізнених текстових інструкцій, а у вигляді структурованих файлів даних форматів YAML або JSON, які зберігаються в централізованій системі контролю версій Git [7].

Впровадження методології NetDevOps докорінно змінює життєвий цикл управління конфігураціями, переводячи його на рейки автоматизованого конвеєра:

1. Source of Truth: Стан мережі повністю визначається кодом у Git-репозиторії. Прямі ручні зміни на пристроях через CLI забороняються.
2. Тестування змін: Будь-яка модифікація параметрів мережі перед реальним застосуванням проходить етап симуляції та синтаксичної перевірки, що мінімізує ризики порушення зв'язності виробничого середовища [9].
3. Автоматизований деплой: Застосування змін виконується спеціалізованими інструментами оркестрації (Ansible) за допомогою CI/CD пайплайнів [6].

Обґрунтування вибору Ansible як засобу автоматизації мережі

Серед сучасних систем управління конфігураціями Ansible, Puppet, Chef, SaltStack у межах даного проєкту було обрано Ansible. Цей вибір є архітектурно обґрунтованим у контексті мережевої інженерії з огляду на такі визначальні фактори [4], [11], [12]:

- Безагентна архітектура: На відміну від Puppet чи Chef, які вимагають встановлення спеціального програмного агента на операційну систему цільового пристрою, Ansible взаємодіє з обладнанням за принципом push. Більшість мережевих пристроїв мають закриту архітектуру та обмежені ресурси, що унеможливлює інсталяцію сторонніх агентів. Ansible вирішує цю проблему, працюючи напряду через стандартні транспортні протоколи [11].
- SSH-транспорт та мережеві модулі: Ansible використовує протокол SSH для автентифікації та передачі конфігураційних команд на комутатори та маршрутизатори [11].

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

- Нативна підтримка колекції cisco.ios: Платформа Ansible має розвинену екосистему офіційних модулів для взаємодії з операційними системами Cisco IOS наприклад, cisco.ios.ios_interfaces, cisco.ios.ios_vlans, cisco.ios.ios_acls. Це дозволяє абстрагуватися від низькорівневого синтаксису CLI і описувати бажаний стан пристрою в декларативній формі [11].

Інтеграція з практиками CI/CD та Git Workflow

CI/CD (Continuous Integration / Continuous Delivery) – це конвеєр пайплайн автоматизації, який гарантує, що зміни у мережі будуть протестовані, безпечні та розгорнуті без втручання людини або з мінімальним ризиком помилок.

CI: Continuous Integration (Безперервна інтеграція)

У NetDevOps етап CI починається тоді, коли мережевий інженер вносить зміну до конфігурації наприклад, додає новий VLAN або оновлює правила Firewall і відправляє commit цей код у репозиторій GitLab, GitHub тощо[6].

Що відбувається автоматично під час CI [7], [8]:

1. Linting (Перевірка синтаксису): Система перевіряє, чи немає помилок форматування у файлах конфігурації наприклад, зайвих пробілів у YAML або синтаксичних помилок у шаблонах Jinja2.
2. Валідація (Testing): Конфігурація тестується на логічні помилки. Використовуються інструменти на кшталт Batfish, які аналізують політики маршрутизації BGP, OSPF та перевіряють, чи не зламає ця зміна доступність мережі.
3. Тестування у віртуальній лабораторії: Зміни можуть розгортатися у цифровому двійнику мережі наприклад, EVE-NG, Cisco CML або GNS3, щоб переконатися, що все працює коректно.

Мета CI – переконатися, що новий код конфігурації є безпечним для застосування.

CD: Continuous Delivery / Deployment (Безперервне розгортання)

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо етап CI пройдено успішно, вмикається етап CD. Це автоматична доставка оновленої конфігурації на реальне мережеве обладнання маршрутизатори, комутатори [6].

Що відбувається під час CD:

1. Розгортання (Deployment): Інструменти автоматизації наприклад, Ansible, Terraform, Nornir або Python-скрипти підключаються до пристроїв через API RESTCONF/NETCONF і застосовують нові налаштування.
2. Перевірка після розгортання (Post-deployment checks): Система автоматично запускає команди наприклад, `show ip bgp summary`, щоб переконатися, що інтерфейси піднялися, а сусіди по маршрутизації встановили зв'язок.
3. Відкат (Rollback): Якщо щось пішло не так на етапі перевірки, CD-пайплайн може автоматично повернути попередню стабільну конфігурацію мережі, мінімізуючи час простою.

Важливою складовою досліджуваної методології є побудова автоматизованого конвеєра розгортання на базі інструменту GitLab CI/CD. Будь-яка зміна конфігурації мережі ініціюється інженером через створення робочої гілки та формування запиту на злиття в основну гілку main [13]. Принципову схему підходу до CI/CD в рамках git продемонстровано на рисунку 1.2.

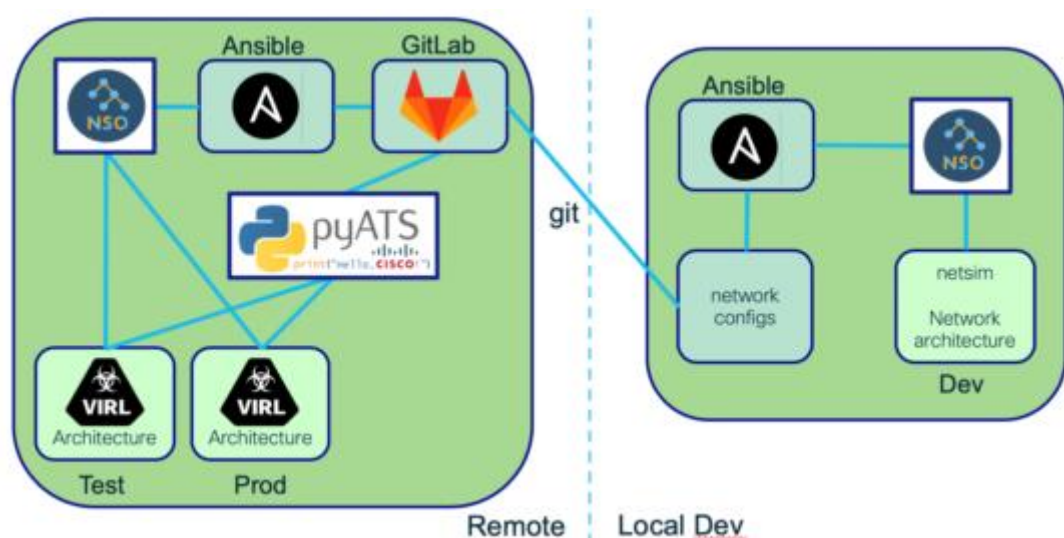


Рисунок 1.2 – Принцип роботи CI/CD в мережевому середовищі [13]

Особлива увага приділяється властивості ідемпотентності Ansible та збиранню логів проходження пайплайнів у GitLab. Повторні запуски плейбуків не призводять до перезапису ідентичних параметрів, що гарантує стабільність та передбачуваність функціонування корпоративної мережі [13].

Висновки до першого розділу

У першому розділі досліджено теоретичні основи автоматизації мережеских конфігурацій та обґрунтовано методологічну базу проєкту.

Проведений аналіз трирівневої ієрархічної моделі Cisco (Core / Distribution / Access) підтвердив її відповідність вимогам корпоративного середовища щодо масштабованості, відмовостійкості та керованості. Встановлено, що чітке функціональне розмежування рівнів є необхідною передумовою для ефективного застосування декларативної автоматизації: кожен рівень мережі відображається в окрему групу Ansible Inventory, що забезпечує модульність коду і точкове застосування конфігураційних ролей. Обґрунтовано рішення щодо централізованого розміщення ACL на рівні Core, що спрощує управління політиками безпеки та прискорює проходження CI/CD пайплайнів.

Розглянуто методологію NetDevOps як синтез практик розробки програмного забезпечення і мережевої інженерії. Визначено, що парадигма Network as Code де конфігурація мережі зберігається у вигляді структурованих YAML-файлів у системі контролю версій Git є ключовою умовою для реалізації відтворюваного, аудитабельного та ідемпотентного процесу керування інфраструктурою.

Обґрунтовано вибір Ansible як інструменту автоматизації: безагентна SSH-архітектура системи є єдино прийнятним рішенням для мережевого обладнання із закритою операційною системою, а нативна підтримка колекції cisco.ios дозволяє описувати бажаний стан пристроїв декларативно, абстрагуючись від низькорівневого CLI-синтаксису. Чотиристадійний конвеєр GitLab CI/CD визначено як організаційний каркас, що перетворює будь-яку зміну конфігурації на керований, перевірюваний і задокументований процес.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 ПРОЄКТУВАННЯ МЕРЕЖЕВОЇ ІНФРАСТРУКТУРИ ТА СИСТЕМИ АВТОМАТИЗАЦІЇ

2.1 Проєктування ієрархічної топології корпоративної мережі та вибір платформи емуляції

Архітектура трирівневої ієрархічної моделі

Архітектуру корпоративної мережі підприємства спроектовано відповідно до трирівневої ієрархічної моделі Cisco Three-Tier Hierarchical Model. Вибір обумовлений потребою у відмовостійкості, прогнозованій маршрутизації та чіткому розмежуванні функцій обладнання на кожному організаційному рівні. Ця модель є де-факто галузевим стандартом побудови корпоративних мереж середнього та великого масштабу і дозволяє реалізувати принцип «правильне обладнання – на правильному рівні», де кожен функціональний рівень вирішує строго визначений набір завдань [1].

Трирівнева модель має три логічні шари:

- Рівень ядра (Core Layer) – забезпечує максимально швидку та безперебійну комутацію транзитного IP-трафіку між магістральними сегментами мережі. Обладнання цього рівня повинно мати найвищу пропускну здатність та мінімальну затримку при обробці пакетів, а його конфігурація зберігається мінімальною – без фільтруючих ACL, без складної логіки комутації VLAN. Єдине завдання Core – швидко передати пакет далі.
- Рівень розподілу (Distribution Layer) – виступає «точкою політик» корпоративної мережі. Тут зосереджено основну логіку обробки пакетів: агрегація каналів від рівня доступу, маршрутизація між VLAN, резервування шлюзу за замовчуванням через HSRP, функція DHCP-relay, а також, у даному проєкті, – централізоване застосування розширених ACL для міжсегментної фільтрації.
- Рівень доступу (Access Layer) – безпосередня точка входу для кінцевих пристроїв робочі станції, IP-телефони, принтери. Тут реалізовано Port Security та ізоляцію широкомовних доменів через VLAN.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Таке розмежування не лише відповідає структурі корпоративного підприємства, але й є ідеальною основою для побудови автоматизованої системи управління конфігураціями: кожен рівень відображається у групу Ansible Inventory, що дозволяє застосовувати конфігураційні ролі точково – наприклад, роль port-security стосується лише групи access, а роль routing – груп core та distribution [13].

Склад обладнання та ролі пристроїв

Рівень ядра представлений двома L3-комутаторами SW-CORE-1 та SW-CORE-2 на образі i86bi-linux-l2-adventerprisek9-15.2c. Їх основне завдання – швидка комутація транзитного IP-трафіку між магістральними сегментами. SW-CORE-1 налаштовано як первинний корінь Spanning Tree spanning-tree mode rapid-pvst, priority 24576, а з'єднання з граничним маршрутизатором реалізовано через маршрутизовані Point-to-Point інтерфейси /30.

Важливою архітектурною особливістю є розміщення розширених ACL саме на рівні Core, а не Distribution. Це дає три переваги [1], [14]:

- Спрощення автоматизації – Ansible-плейбуки взаємодіють з обмеженою кількістю центральних Core-пристроїв
- Зняття навантаження з рівнів розподілу – комутатори Distribution обробляють трафік на апаратній швидкості без додаткових затримок
- Консолідація точок контролю – єдина прозора матриця доступу, що виключає ризик конфліктів при розподіленому налаштуванні ACL.

Рівень розподілу реалізовано на комутаторах SW-DIST-1 та SW-DIST-2. Тут сконцентровано логіку обробки пакетів: резервування шлюзу за замовчуванням через протокол HSRP, функцію DHCP-relay ip helper-address та міжмережеву маршрутизацію між VLAN. SW-DIST-1 налаштовано як активний шлюз HSRP priority 110, SW-DIST-2 – як резервний priority 100.

Рівень доступу складається з трьох комутаторів SW-ACC-1, SW-ACC-2 та SW-ACC-3, що забезпечують підключення кінцевих пристроїв. На цьому рівні реалізовано Port Security та ізоляцію широкомовних доменів.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Для зв'язку з публічними мережами в топологію інтегровано межовий маршрутизатор ROUTER-1 (образ c7200-adventerprisek9), який виконує функції NAT, DHCP-сервера, NTP-мастера та OSPF ABR.

Перелік обладнання та відповідних IOS-образів зведено у Таблиці 2.1.

Таблиця 2.1

Перелік обладнання проєкту

Пристрій	Рівень	Роль	Образ
ROUTER-1	Edge	NAT, DHCP-сервер, NTP master, OSPF	c7200-adventerprisek9
SW-CORE-1, SW-CORE-2	Core	L3-комутація, OSPF Area 0, STP Root, ACL	i86bi-linux-l2-adventerprisek9-15.2c
SW-DIST-1, SW-DIST-2	Distribution	VLAN-агрегація, HSRP, DHCP-relay	i86bi-linux-l2-adventerprisek9-15.2c
SW-ACC-1/2/3	Access	Port-security, динамічний VLAN на портах	i86bi-linux-l2-adventerprisek9-15.2c
GitLab-Ansible-Server	Management	GitLab CE + Ansible control node	Ubuntu Server (VM)

Адресний план та VLAN-структура

Адресний план розроблено на основі простору 10.0.0.0/8 (RFC 1918) із застосуванням VLSM. Ключовий архітектурний принцип – жорстка синхронізація третього октету IP-адреси з ідентифікатором VLAN. Це підвищує читабельність коду Ansible та спрощує діагностику: побачивши адресу 10.0.30.15, інженер одразу розуміє – це пристрій у VLAN 30 (USERS). Не потрібно звіряти з документацією [2]. VLAN-структура побудована за принципом ізоляції функціональних сегментів:

Адресний розподіл VLAN зведено у таблиці 2.2.

Таблиця 2.2

Перелік обладнання проєкту

VLAN ID	Назва	Підмережа	Призначення
10	MGMT	10.0.10.0/24	Управління пристроями
20	SERVERS	10.0.20.0/24	Серверний сегмент
30	USERS	10.0.30.0/24	Корпоративні робочі станції
40	VOIP	10.0.40.0/24	IP-телефонія
99	NATIVE	-	Trunk native VLAN

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Сегмент VLAN 10 (MGMT) повністю ізольовано від користувацького середовища. У ньому розміщено керуючі інтерфейси всіх комутаторів та сервер GitLab-Ansible-Server (10.0.10.10). Стандартний ACL на лініях VTY дозволяє SSH-доступ до обладнання виключно з цієї адреси, що є важливим елементом безпеки всього рішення.

Для магістральних з'єднань між ROUTER-1 та комутаторами ядра використовуються підмережі /30, що унеможлиблює несанкціоноване підключення. Адреси від .1 до .10 у клієнтських підмережах виключено з пулів DHCP-роздачі і зарезервовано для мережевої інфраструктури [2].

Схему HSRP-резервування для клієнтських VLAN показано у Таблиці 2.3. SW-DIST-1 виконує роль активного шлюзу, SW-DIST-2 – резервного.

Таблиця 2.3

HSRP-розподіл по VLAN

VLAN	SW-DIST-1 (Active, priority 110)	SW-DIST-2 (Standby, priority 100)	Virtual IP (шлюз)
20 (SERVERS)	10.0.20.2	10.0.20.3	10.0.20.1
30 (USERS)	10.0.30.2	10.0.30.3	10.0.30.1
40 (VOIP)	10.0.40.2	10.0.40.3	10.0.40.1

Вибір платформи емуляції

Для розгортання топології розглянуто три платформи: Cisco Packet Tracer, EVE-NG та GNS3. Порівняльний аналіз детально наведено у Додатку Б; нижче – ключові висновки.

Cisco Packet Tracer є симулятором, орієнтованим на освітні цілі. Він не запускає реальні IOS-образи, а лише імітує їх поведінку на програмному рівні. Набір підтримуваних команд суттєво обмежений, неможливо підключити сторонні образи в тому числі Ubuntu VM з GitLab та Ansible, відсутня підтримка складних функцій IOS, таких як RESTCONF/NETCONF або специфічні алгоритми шифрування SSH. Для задач даного проєкту Packet Tracer є принципово непридатним [15].

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

EVE-NG – потужна web-орієнтована платформа з підтримкою мультивендорних образів, технологією UKSM для економії RAM та зручним браузерним інтерфейсом. Проте EVE-NG розгортається поверх операційної системи як окрема ВМ, і всі ресурси споживаються хост-машиною без можливості паралельного запуску інших середовищ. Наявність двох версій Community та Professional обмежує базову функціональність 63 вузлами, а для повноцінного Wireshark-аналізу потрібна платна ліцензія [16].

GNS3 обрано як оптимальний варіант. Ключові причини вибору: підтримка оригінальних IOS-образів Cisco IOS, IOU, клієнт-серверна архітектура з GNS3 ВМ забезпечує стабільну роботу, безпосередня інтеграція з Ubuntu-сервером через мережевий адаптер або Cloud-ноду дозволяє розмістити GitLab-Ansible-Server всередині топології GNS3, вбудована підтримка Wireshark для аналізу трафіку прямо в кабелях емуляції, безкоштовна open-source ліцензія [17].

Принциповою перевагою GNS3 для даного проєкту є саме можливість інтеграції Ubuntu ВМ: GitLab-Ansible-Server отримує два мережеві адаптери – один підключений до VLAN 10 топології GNS3 для доступу до комутаторів по SSH, другий – у режимі NAT для виходу в Інтернет завантаження Ansible-колекцій, синхронізація NTP. Такий гібридний зв'язок неможливо реалізувати в Packet Tracer і значно складніше – в EVE-NG Community Edition.

Порівняння середовищ розгортання подано в Додатку Б.

Логічну топологію спроектованої мережі подано в Додатку А.

2.2 Аналіз, порівняння та обґрунтування протокольного стеку мережі

Вибір протоколів у проєкті здійснювався за трьома критеріями: відповідність трирівневій архітектурі, придатність до автоматизації засобами Ansible та реалізованість у середовищі GNS3 на реальних IOS-образах.

Вибір протоколу динамічної маршрутизації: OSPF

Забезпечення динамічної маршрутизації є фундаментальною вимогою до корпоративної мережі – статична маршрутизація в умовах ієрархічної топології з кількома рівнями комутаторів і резервними шляхами є нежиттєздатною. Розглядалися три протоколи: RIP, EIGRP та OSPF.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Протокол RIP відкинуто через два критичні обмеження. По-перше, максимальний діаметр мережі – 15 хопів значення 16 вважається нескінченністю, мережа вважається недосяжною. По-друге, низька швидкість збіжності: при зміні топології RIP використовує таймери Invalid 180 сек, Hold-down 180 сек, Flush 240 сек, що означає потенційну недоступність мережі на кілька хвилин. В ієрархічній мережі з HSRP-резервуванням це неприйнятно [2], [18].

Протокол EIGRP демонструє відмінні технічні характеристики: алгоритм DUAL забезпечує миттєве переключення на Feasible Successor при відмові, а нерівномірне балансування навантаження є унікальною перевагою. Проте EIGRP є пропрієтарною розробкою Cisco незважаючи на часткове відкриття стандарту через RFC 7868 у 2013 році, що створює ризик прив'язки до вендора. У мультивендорних мережах або при заміні частини обладнання EIGRP перестає бути опцією [19].

OSPF обрано з таких причин.

По-перше, сумісність із трирівневою моделлю. OSPF ієрархічно ділиться на зони Areas. Магістральна Area 0 розгорнута між L3-комутаторами ядра та ROUTER-1. Такий поділ локалізує службовий Link-State трафік і зменшує навантаження на CPU: зміна топології в одній зоні не ініціює перерахунок алгоритму SPF на всіх пристроях мережі [20].

По-друге, автоматична верифікація в CI/CD pipeline. Плейбук verify.yml виконує команду `show ip ospf neighbor` і через модуль `ansible.builtin.assert` перевіряє наявність рядка FULL у виводі. Якщо хоча б один сусід не встановив сусідства, pipeline GitLab CI/CD миттєво стає «червоним» і блокує ручне підтвердження деплою у production. Це технічний запобіжник, якого немає при використанні RIP або статичної маршрутизації [20].

По-третє, відкритий стандарт RFC 2328 для OSPFv2. Незалежність від вендора гарантує, що при розширенні мережі обладнанням інших виробників протокол маршрутизації залишиться сумісним [20].

Порівняльну характеристику розглянутих протоколів зведено у Таблиці 2.4.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Порівняння протоколів динамічної маршрутизації

Критерій	RIPv2	OSPFv2	EIGRP
Алгоритм	Беллмана-Форда	Дейкстри (SPF)	DUAL
Тип	Дистанційно-векторний	Стану каналу	Гібридний
Метрика	Кількість хопів	Вартість (Cost)	Складена (BW + Delay)
Макс. хопів	15	Необмежено	Необмежено
Швидкість збіжності	Низька (~3 хв)	Висока	Дуже висока (миттєва)
Ієрархічний поділ	Ні	Так (Areas)	Ні
Стандарт	Відкритий (RFC)	Відкритий (RFC)	Cisco (частково RFC 7868)
Адміністративна відстань	120	110	90 (внутрішні)
Автоматична верифікація через Ansible	Ні	Так (show ip ospf neighbor)	Обмежено

NAT та забезпечення виходу в мережу

На граничному маршрутизаторі ROUTER-1 розгорнуто PAT Port Address Translation – трансляція множини внутрішніх адрес 10.0.0.0/8 на одну публічну IP-адресу, динамічно отриману на зовнішньому інтерфейсі FastEthernet0/0.

FastEthernet1/0 та FastEthernet2/0 у бік ядра позначено як ip nat inside; зовнішній інтерфейс – ip nat outside. Трафік для трансляції фільтрується стандартним ACL NAT-USERS.

Роль NAT у структурі проєкту визначається особливостями архітектури GitLab-Ansible-Server. Сервер має два мережеві адаптери: перший підключений до VLAN 10 (10.0.10.10) для прямого SSH-доступу до обладнання Cisco, другий – через NAT-адаптер VirtualBox для виходу в Інтернет. NAT на ROUTER-1 забезпечує ізольованим пристроєм GNS3 можливість завантаження Ansible-колекцій, оновлень пакетів та синхронізації часу через зовнішні NTP-сервери. Без NAT вся топологія залишається ізольованою, що ускладнює первинне розгортання DevOps-інфраструктури [21].

Резервування першого хопу: порівняльний аналіз HSRP та VRRP

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Без спеціалізованих протоколів резервування кінцеві пристрої рівня доступу не здатні автоматично переключитися на резервний шлях у разі відмови основного шлюзу. Розглядалися два протоколи: HSRP та VRRP.

HSRP – пропрієтарний протокол Cisco. Принцип роботи: кілька фізичних маршрутизаторів об'єднуються у групу зі спільною віртуальною IP-адресою та MAC-адресою. Один пристрій стає Active, інший – Standby. Обмін Hello-пакетами кожні 3 секунди. При відмові Active пристрою Standby перебирає роль через 10 секунд. Конфігурація HSRP повністю описується в Ansible-ролі hsrp, що забезпечує ідемпотентність [22].

VRRP – відкритий стандарт RFC 5798, підтримується обладнанням різних виробників Cisco, Juniper, Huawei, MikroTik. Ролі: Master та Backup. Ключові відмінності від HSRP: оголошення надсилаються щосекунди, швидший failover ~3.6 сек проти ~10 сек у HSRP, дозволяється використовувати реальну IP-адресу фізичного маршрутизатора як віртуальну [23].

Порівняльну характеристику розглянутих протоколів зведено у Таблиці 2.5.

Таблиця 2.5

Порівняння протоколів резервування шляху

Характеристика	HSRP	VRRP
Стандарт	Пропрієтарний (Cisco)	Відкритий (RFC 5798)
Ролі пристроїв	Active / Standby	Master / Backup
Мультикаст адреса	224.0.0.2 (v1), 224.0.0.102 (v2)	224.0.0.18
Таймери за замовчуванням	Hello: 3 сек, Hold: 10 сек	Advertisement: 1 сек, Master Down: ~3.6 сек
Використання реального IP	Не дозволяється	Дозволяється (IP Owner)
Підтримка IPv6	Лише HSRP v2	Так (RFC 5798)
Сумісність	Тільки Cisco	Будь-яке обладнання

Обґрунтування вибору HSRP. Мережа побудована на однорідній екосистемі Cisco, тому перевага VRRP в мултивендорності не є актуальною. HSRP має нативну підтримку DHCP Relay: при відмові SW-DIST-1 резервний SW-DIST-2 автоматично перебирає роль активного шлюзу без ручного втручання, і клієнти

продовжують отримувати адреси від DHCP-сервера. Конфігурація повністю описується кодом Ansible, що гарантує ідемпотентність та відтворюваність.

Агрегація каналів: порівняльний аналіз LACP та PAgP

Для об'єднання фізичних каналів між рівнями Distribution та Access розглядалися два протоколи: LACP (Link Aggregation Control Protocol, IEEE 802.3ad) та PAgP (Port Aggregation Protocol, Cisco).

LACP – відкритий міжнародний стандарт IEEE 802.3ad / IEEE 802.1AX. Підтримується обладнанням будь-яких виробників. Об'єднує до 16 портів у групу 8 активних + 8 у гарячому резерві. Режими роботи: Active ініціативно надсилає LACPDU та Passive відповідає на запити [2], [24].

PAgP – пропрієтарний протокол Cisco. Підтримує до 8 портів. Режими: Desirable активний та Auto пасивний. Принципово обмежений: не запрацює у мережах з обладнанням інших виробників [2].

Порівняльну характеристику розглянутих протоколів зведено у Таблиці 2.6.

Таблиця 2.6

Порівняння LACP та PAgP

Критерій	LACP	PAgP
Стандарт	IEEE 802.3ad (відкритий)	Cisco (пропрієтарний)
Сумісність	Будь-яке обладнання	Лише Cisco
Макс. портів у групі	16 (8 активних + 8 резервних)	8
Активний режим	Active	Desirable
Пасивний режим	Passive	Auto
Захист від петель при автоматизації	Так	Так

Обґрунтування вибору LACP. Відкритий стандарт забезпечує вендорнезалежність інфраструктури. При розширенні мережі серверами або комутаторами інших виробників LACP залишається сумісним. Підтримка 8 портів у гарячому резерві гарантує захист від комутаційних петель при автоматизованому розгортанні через GitLab CI/CD – навіть якщо pipeline помилково надішле конфігурацію агрегації на некоректний порт, LACP відхилить зв'язок при невідповідності параметрів.

Сервіс DHCP

DHCP-сервер розгорнуто на ROUTER-1. Централізований підхід обумовлений архітектурою трирівневої мережі: розміщення окремого DHCP-сервера в кожному VLAN є неефективним та збільшує кількість вузлів, якими потрібно управляти [25].

Оскільки широкомовні DHCP DISCOVER не проходять через маршрутизатори, на SW-DIST-1 та SW-DIST-2 налаштовано DHCP Relay Agent – це усуває необхідність у локальних серверах. При відмові SW-DIST-1 роль DHCP Relay автоматично перебирає SW-DIST-2 завдяки HSRP, що забезпечує безперервну видачу адрес клієнтам.

Адреси від .1 до .10 у клієнтських підмережах виключено з пулів роздачі і зарезервовано для мережевої інфраструктури. Конфігурацію описано декларативно в ролі Ansible `dhcp` і верифіковано після кожного деплою.

Сервіс NTP

Синхронізація часу є критичною умовою для коректної роботи автоматизованої системи.

1. Ansible використовує системний час при виконанні тасок і формуванні логів – розходження часу між керуючим вузлом і пристроями призводить до помилок.
2. Механізми аутентифікації SSH-сертифікати, Ansible Vault чутливі до синхронізації часу.
3. Корельований аналіз логів пристроїв та логів GitLab CI/CD pipeline потребує єдиної точки відліку часу.

ROUTER-1 налаштовано як NTP-мастер stratum 1 для внутрішнього стенду з можливістю синхронізації від зовнішніх джерел через NAT. Усі комутатори та GitLab-Ansible-Server синхронізують час від ROUTER-1. Конфігурація NTP входить до ролі base та застосовується на всіх пристроях [26].

Безпека: Port Security та ACL

Port Security налаштовано на всіх access-портах комутаторів SW-ACC-1/2/3 у режимі Sticky MAC з лімітом maximum 2 та реакцією restrict. Sticky MAC –

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

компромісний підхід: комутатор динамічно вивчає перші підключені пристрої та автоматично записує їх у поточну конфігурацію Running-config. Якщо після цього зберегти конфігурацію, адреси залишаться прив'язаними до порту навіть після перезавантаження [2].

Режим restrict, а не shutdown за замовчуванням обрано з практичних міркувань: при виявленні порушення порт залишається активним для легітимних пристроїв, лічильник порушень збільшується та генерується syslog-запис. Режим shutdown вимагав би ручного відновлення порту адміністратором після кожного інциденту, що суперечить принципу автоматизованого управління [2].

Port Security захищає від двох типів атак рівня L2:

1. MAC Flooding, переповнення таблиці MAC-адрес комутатора шляхом генерації тисяч підроблених MAC
2. Підміни пристроїв, підключення стороннього ноутбука замість легітимної робочої станції.

ACL застосовано на двох рівнях. Розширені Extended ACL розміщено на інтерфейсах рівня Core для захисту MGMT та SERVERS від несанкціонованого доступу з клієнтських VLAN. Централізація ACL на Core-рівні спрощує автоматизацію: одна Ansible-роль `security` керує всіма правилами фільтрації, не розпорошуючи їх по периферійних пристроях [2].

Стандартні ACL на лініях VTY line vty 0 4 обмежують SSH-доступ до Cisco-обладнання виключно для IP-адреси GitLab-Ansible-Server (10.0.10.10). Це гарантує, що навіть якщо злоумисник потрапить до VLAN 10, він не зможе підключитися по SSH до комутаторів з іншої адреси.

2.3 Обґрунтування архітектури автоматизації управління конфігураціями

Вибір системи керування конфігураціями: Ansible vs Puppet vs Chef

На сучасному ринку інструментів автоматизації IT-інфраструктури існує кілька зрілих систем, що набули широкого розповсюдження в корпоративному середовищі. Для обґрунтованого вибору розглянуто три найбільш поширених рішення.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Puppet агентна архітектура, DSL на базі Ruby дозволяє автоматично виявляти та повертати «дрейф конфігурації»: якщо адміністратор вручну змінить налаштування на пристрої, Puppet під час наступного циклу виявить розбіжність і автоматично поверне еталонний стан. Це є головною перевагою Puppet. Однак для мережевих пристроїв Cisco IOS, де неможливо встановити нативний агент, Puppet використовує проксі-агентів, окремі сервери, які транслюють каталог конфігурації у CLI-команди через SSH. Це значно ускладнює розгортання: необхідно керувати SSL-сертифікатами для кожного пристрою, налаштовувати проксі-сервери та імпортувати специфічні провайдери. Для ad-hoc завдань «терміново зміни VLAN на 50 пристроях зараз» Puppet непридатний – він заточений під регулярні перевірки за розкладом [27].

Chef агентна архітектура, мова Ruby пропонує максимальну гнучкість програмування: повна мова Ruby дозволяє реалізувати будь-яку логіку – виконати HTTP-запит до IPAM-системи, отримати вільну підмережу та одразу застосувати її в конфігурації OSPF. Проте Chef є найскладнішим у підтримці: інженер повинен бути не просто мережевим адміністратором, а повноцінним Ruby-розробником. Написання та відлагодження «кулінарних книг» Cookbooks потребує глибоких знань ООП. У великих командах підтримка чужого коду на Ruby стає серйозною проблемою [28].

Ansible обрано за сукупністю переваг.

Безагентна архітектура agentless є принципово важливою для мережевих пристроїв. На відміну від Puppet та Chef, Ansible взаємодіє з обладнанням через стандартний SSH без встановлення жодного додаткового ПЗ на цільовий пристрій. Для Cisco IOS, де закрита архітектура унеможлиблює інсталяцію сторонніх агентів, це єдино правильний підхід [11].

Декларативний YAML-синтаксис знижує поріг входу: мережевий інженер без досвіду розробки може писати та підтримувати Ansible-плейбуки. На відміну від Ruby Chef або Puppet DSL, YAML є простим форматом даних без програмної логіки.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Ідемпотентність – повторний запуск того самого плейбука на вже налаштованій мережі повертає `changed=0` на всіх пристроях. Це підтверджено експериментально у розділі тестування. Ідемпотентність є критично важливою властивістю для CI/CD pipeline: оператор може запустити `site.yml` двічі без ризику пошкодити конфігурацію [11].

Нативна колекція `cisco.ios` забезпечує декларативне управління всіма аспектами конфігурації Cisco IOS: `ios_vlans`, `ios_ospf_interfaces`, `ios_acls`, `ios_l2_interfaces`, `ios_l3_interfaces` та інші. Замість написання сирих CLI-команд через `ios_config`, ці модулі дозволяють описати бажаний стан у форматі структурованих даних YAML [11].

Порівняльну характеристику розглянутих утиліт зведено у Таблиці 2.7.

Таблиця 2.7

Порівняння LACP та PAgP

Критерій	Ansible	Puppet	Chef
Архітектура	Безагентна (agentless)	Агентна (agent-based)	Агентна (agent-based)
Мова конфігурації	YAML	Puppet DSL (Ruby-подібний)	Ruby
Поріг входу	Низький	Середній	Високий
Підтримка мережевих пристроїв Cisco	Нативна (<code>cisco.ios</code>)	Через проксі-агентів	Через проксі-агентів
Ідемпотентність	Так (гарантована модулями)	Так	Залежить від коду
Виявлення дрейфу конфігурації	Ні (pull-модель)	Так (кожні 30 хв)	Так (кожну годину)
Інтеграція з GitLab CI/CD	Нативна (Shell Runner)	Потребує налаштування	Потребує налаштування
Складність розгортання на Cisco IOS	Мінімальна (тільки SSH)	Висока (SSL, проксі)	Висока (проксі, Ruby)

Детальний порівняльний аналіз Ansible, Puppet та Chef наведено у Додатках В та Г;

Структура Ansible-проєкту

Правильна архітектура Ansible-репозиторію є такою ж важливою, як і вибір самого інструменту. Репозиторій `network-automation` організовано відповідно до

офіційних рекомендацій Ansible Best Practices, приклад структури показано на рисунку 2.1 [30].

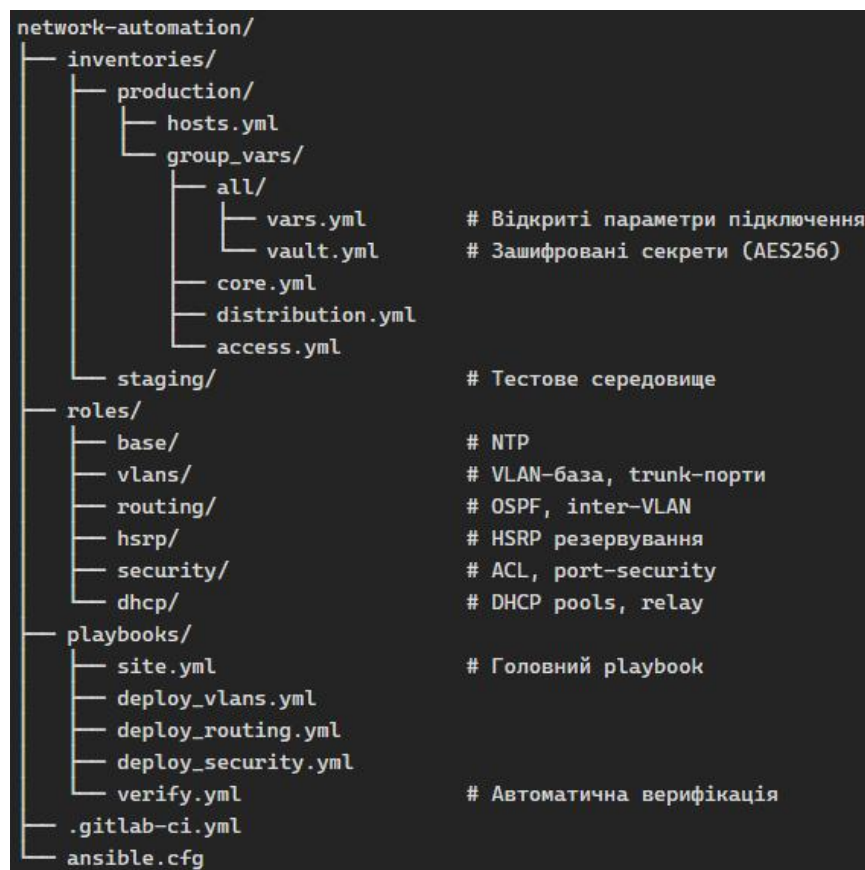


Рис. 2.1 – Приклад Ansible структури

Inventories – найважливіший структурний елемент, що реалізує принцип розмежування середовищ. Існує дві директорії: inventories/staging/ пристрої GNS3 та inventories/production/ реальне обладнання. Один і той самий `site.yml` запускається з різним прапорцем -i, що унеможливило випадковий деплой на production під час тестування. Такий підхід є технічним запобіжником, що виключає людський фактор при роботі з CI/CD pipeline [29].

group_vars поділяється на два файли в директорії all/:

- vars.yml – відкритий файл з архітектурними параметрами та визначенням змінних Ansible тип з'єднання, мережева ОС тощо. Головна особливість: він пов'язує відкриту логіку із зашифрованими секретами через шаблонізатор Jinja2:

ansible_password: "{{ vault_ansible_password }}"

ansible_become_password: "{{ vault_ansible_become_password }}"

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

- vault.yml – файл зашифрований алгоритмом AES256 через Ansible Vault. У Git-репозиторій він потрапляє у зашифрованому стані і виглядає як нечитабельний блок тексту. Містить лише конфіденційні дані з префіксом vault_. Майстер-пароль передається в GitLab CI/CD як Masked & Protected змінна ANSIBLE_VAULT_PASSWORD і матеріалізується у тимчасовий файл .vault_pass.txt лише під час виконання джоба, після чого негайно видаляється.

Roles – основний принцип організації коду автоматизації. Кожна роль відповідає за один аспект конфігурації. Структура ролі vlans для прикладу на рисунку 2.2 [29]:

```
roles/vlans/
├─ tasks/main.yml    # Логіка (Ansible-таски)
└─ vars/main.yml     # Дані (список VLAN)
```

Рис. 2.2 – Приклад структури ролі vlans

Файл vars/main.yml містить лише структуровані дані у форматі YAML без жодної команди Cisco IOS, приклад вмісту показано на рисунку 2.3:

```
vlan_list:
- vlan_id: 10
  name: MGMT
- vlan_id: 20
  name: SERVERS
- vlan_id: 30
  name: USERS
```

Рис. 2.3 – Приклад вмісту даних для ролі vlans

Файл tasks/main.yml містить логіку, яка ітерує через ці дані, приклад команд показано на рисунку 2.4:

```

- name: Створення бази VLAN
  cisco.ios.ios_vlans:
    config:
      - vlan_id: "{{ item.vlan_id }}"
        name: "{{ item.name }}"
    state: merged
    loop: "{{ vlan_list }}"

```

Рис. 2.4 – Приклад вмісту виконавчих команд для полі vlans

Перевага такого підходу: для додавання VLAN 50 (GUEST) достатньо одного рядка у vars/main.yml. Код тасків залишається незмінним. Під час наступного запуску CI/CD pipeline Ansible сам згенерує та застосує нову конфігурацію.

Файл ansible.cfg містить три важливі блоки налаштувань [11], [29]:

Блок [defaults] визначає глобальну поведінку: host_key_checking = False ігнорування SSH fingerprints у динамічних GNS3-середовищах, stdout_callback = debug зручний формат виводу в логах GitLab, timeout = *** достатній час для ініціалізації SSH-сесії на старих IOS-образах.

Блок [persistent_connection] встановлює таймаути 540 секунд – критично важливо для стабільної роботи при пікових навантаженнях GNS3 VM, коли Ansible одночасно надсилає конфігурацію на всі 7 пристроїв.

Блок [ssh_connection] додає підтримку застарілих алгоритмів криптографії:
 ssh_args = -O KexAlgorithms=+diffie-hellman-group1-sha1
 -o HostKeyAlgorithms=+ssh-rsa

Сучасні дистрибутиви Linux заборонили ці алгоритми з міркувань безпеки, однак старі Cisco IOS-образи не підтримують новіші. Без цього параметра Ubuntu відмовляється встановлювати SSH-з'єднання з пристроями.

Повну файлову структуру Ansible-проєкту наведено у Додатку Д.

2.4 Проєктування конвеєра CI/CD та моделі розгалуження Git

Порівняння платформ: GitLab CE та GitHub

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Будь-яка реалізація підходу Infrastructure as Code передбачає наявність централізованої системи контролю версій – єдиного джерела істини щодо поточного стану мережевої інфраструктури. Розглянуто дві платформи: GitLab CE та GitHub (Enterprise Server). GitHub – лідер ринку за кількістю публічних репозиторіїв. Система автоматизації GitHub Actions базується на окремих файлах Workflows `.github/workflows/*.yaml`, що забезпечує гнучкість, але ускладнює підтримку великих pipeline. Основна перевага – величезний Actions Marketplace з готовими блоками. Принципові обмеження для даного проєкту: GitHub є хмарним сервісом за замовчуванням; GitHub Enterprise Server для on-premise розгортання є платним рішенням. Конфіденційні мережеві конфігурації паролі, IP-адреси, схеми маршрутизації неприпустимо зберігати на хмарній платформі [30], [31].

GitLab CE – монолітна платформа «все в одному»: Git-репозиторій, Merge Requests, Code Review та CI/CD в єдиному безкоштовному open-source продукті. Підтримує on-premise розгортання без ліцензійних обмежень [30].

Порівняльну характеристику розглянутих утиліт зведено у Таблиці 2.8.

Таблиця 2.8

Порівняння GitLab CE та GitHub для завдань мережевої автоматизації

Критерій	GitLab CE	GitHub (Enterprise Server)
Модель розгортання	On-premise (безкоштовно)	Хмара або on-premise (платно)
CI/CD двигун	Вбудований (нативний)	GitHub Actions (окремий продукт)
Конфігурація pipeline	Один файл <code>.gitlab-ci.yml</code>	Окремі файли Workflows
Агент виконання	GitLab Runner (Shell mode)	GitHub Actions Runner
Захист секретів	Ansible Vault + Protected Variables	Secrets (аналогічно)
Мінімальні вимоги	4 GB RAM	Більше (для GHES)
Безпека мережевих конфігурацій	100% локальна ізоляція	Ризик при хмарному варіанті
Ціна	Безкоштовно (CE)	Платно (GHES)

Обґрунтування вибору GitLab CE. Концепція «все в одному» виключає необхідність інтегрувати сторонні сервіси. Локальне on-premise розгортання в ізольованому VLAN 10 гарантує 100% захист конфіденційних мережевих даних. Shell Runner виконує Ansible-плейбуки прямо в середовищі хоста Ubuntu,

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

використовуючи встановлені колекції cisco.ios та зашифровані змінні Ansible Vault. Мінімальні системні вимоги 4 GB RAM дозволяють запускати GitLab паралельно з важкою емуляцією GNS3 VM.

Розширене порівняння платформ: GitLab CE та GitHub подано в Додатку Е.

Стадії CI/CD pipeline

Конвеєр реалізовано у файлі .gitlab-ci.yml і складається з чотирьох послідовних стадій [33]. Повний конфігураційний файл наведено у Додатку Х.

Стадія 1: Lint – статичний аналіз коду. Запускається при кожному Push та Merge Request. Виконує ansible-lint перевірка відповідності кращим практикам та ansible-playbook --syntax-check синтаксична перевірка без підключення до пристроїв. Блокує дефектний код на ранньому етапі без навантаження на обладнання.

Стадія 2: Dry-Run – динамічна перевірка змін. Запускається лише при Merge Request. Виконує ansible-playbook --check --diff підключається до реальних пристроїв GNS3, але не вносить змін, а лише генерує звіт про те, що зміниться. Рецензент у Merge Request бачить не абстрактний YAML-код, а конкретні рядки CLI-команд Cisco IOS, які будуть виконані при деплої. Наприклад як на рисунку 2.5:

```
TASK [vlans : Створення бази VLAN] ***
--- before
+++ after
@@ -1,3 +1,5 @@
  vlan 10
  vlan 20
+vlan 50
+ name GUEST
changed: [sw-dist-1]
```

Рис. 2.5 – Приклад CLI-команд Cisco IOS

Це є головною цінністю dry-run стадії: прозора демонстрація практичного наслідку зміни коду ще до його злиття.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Стадія 3: Deploy-Staging – автоматичний деплой. Запускається автоматично після злиття в main. Застосовує site.yml на staging-інвентар пристрої GNS3, після чого одразу запускається verify.yml з assert-перевірками. Якщо будь-яка перевірка провалюється наприклад, OSPF не встановив сусідства FULL, pipeline стає червоним і кнопка ручного запуску deploy-production стає недоступною.

Стадія 4: Deploy-Production – ручне підтвердження. Параметр when: manual вимагає свідомого підтвердження відповідального інженера через GitLab CI/CD → Pipelines → кнопка. Кнопка стає активною лише після 100% успішного проходження staging-тестів (needs: [deploy-staging]). Після деплою знову запускається verify.yml вже для production-інвентаря.

Повна схема стадій та умов запуску на рисунку 2.6:

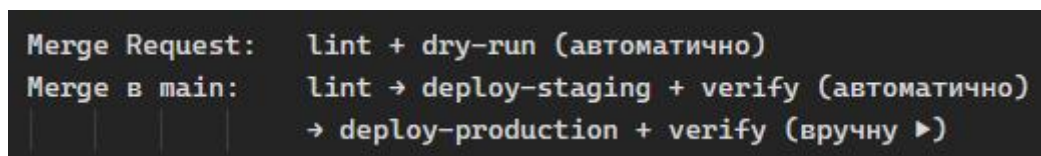


Рис. 2.6 – Схема стадій роботи пайплайнів

Модель розгалуження Git та Merge Request

Використовується проста двогілкова модель: feature/* → main.

Гілка main є захищеною protected branch. Вона завжди відображає поточний «бажаний стан» мережі – Source of Truth. Прямий push у `main` заборонений на рівні GitLab. Це технічна гарантія того, що жодна нетестована конфігурація не потрапить на обладнання [33].

Гілки feature/ – робочі гілки для кожної окремої зміни. Назва відображає суть зміни: `feature/add-vlan-50`, `feature/fix-core-acl`. Один MR – одна логічна зміна мережі [33].

Повний цикл внесення змін:

1. Інженер створює гілку: `git checkout -b feature/add-vlan-50`
2. Вносить зміну в `roles/vlans/vars/main.yml`

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

3. Створює Merge Request у GitLab
4. GitLab автоматично запускає lint + dry-run
5. Рецензент перевіряє: diff коду + Ansible-diff у логах dry-run
6. При необхідності – нові коміти в ту саму гілку pipeline перезапущається
7. Після схвалення – натискання кнопки Merge
8. GitLab автоматично запускає deploy-staging + verify
9. Відповідальний інженер перевіряє результат staging і натискає кнопку для production

Кожен merge-commit у main – задокументована, протестована та затверджена зміна конфігурації мережі. Команда git log --oneline main показує повний хронологічний audit trail всіх змін, аналогічно до корпоративного журналу змін, але з автоматизованою перевіркою кожного запису.

Управління секретами через Ansible Vault

Безпека конфіденційних даних є найважливішим аспектом системи Network as Code, оскільки компрометація пароля адміністратора означає втрату контролю над усією корпоративною інфраструктурою.

Ansible Vault шифрує файл vault.yml алгоритмом AES256 на рівні диска. У Git-репозиторій файл потрапляє у зашифрованому вигляді [34]:

```
$ANSIBLE_VAULT;1.1;AES256
386164376464396662663466623962323235623139633531663931383734...
```

В автоматизованому CI/CD pipeline майстер-пароль від Ansible Vault зберігається в GitLab як Protected & Masked змінна середовища ANSIBLE_VAULT_PASSWORD. При запуску джоба деплою Runner на льоту створює тимчасовий файл .vault_pass.txt, передає його плейбуку і негайно видаляє після завершення. Звичайний розробник у feature-гілці фізично не може отримати

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

доступ до production-секретів – Protected Variables стають доступними лише у захищених гілках [30], [34].

Механізм Jinja2 у vars.yml дозволяє зберігати логіку підключення у відкритому файлі, посилаючись на секрети лише через змінні:

```
ansible_password: "{{ vault_{{ ansible_password }}"
```

```
ansible_become_password: "{{ vault_{{ ansible_become_password }}"
```

Таким чином, відкритий код автоматизації та конфіденційні облікові дані ніколи не зберігаються разом, що відповідає принципу розмежування даних та логіки.

Висновки до другого розділу

У другому розділі виконано повне проєктування мережевої інфраструктури та системи її автоматизації.

Спроектовано ієрархічну топологію корпоративної мережі на базі трирівневої моделі у середовищі GNS3, до складу якої увійшли: межовий маршрутизатор ROUTER-1 (NAT, DHCP-сервер, NTP master, OSPF), два комутатори ядра SW-CORE-1/2 (L3-комутація, OSPF Area 0, STP Root, ACL), два комутатори рівня розподілу SW-DIST-1/2 (HSRP, DHCP-relay, inter-VLAN маршрутизація) та три комутатори рівня доступу SW-ACC-1/2/3 (Port-Security, динамічне призначення VLAN). Вибір GNS3 як платформи емуляції обґрунтовано порівняльним аналізом із Cisco Packet Tracer та EVE-NG: лише GNS3 поєднує підтримку оригінальних IOS-образів, можливість безпосередньої інтеграції з Ubuntu-сервером через мережевий адаптер та безкоштовне відкрите програмне забезпечення.

Розроблено структурований адресний план на основі простору 10.0.0.0/8 із жорсткою синхронізацією третього октету IP-адреси з ідентифікатором VLAN (VLAN 10 – 10.0.10.0/24, VLAN 20 – 10.0.20.0/24 тощо), що підвищує читабельність коду автоматизації та спрощує діагностику. Визначено схему HSRP-резервування для VLAN 20, 30 і 40 із фіксованим розподілом ролей

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Active/Standby між SW-DIST-1 і SW-DIST-2 та виділеними транзитними Point-to-Point підмережами /30 для магістральних з'єднань.

Серед всіх конкурентів (Puppet, Chef) обґрунтовано саме структуру Ansible-проєкту з розмежуванням середовищ staging і production через окремі інвентарі та використанням Ansible Vault для захисту облікових даних. Досліджено взаємодію внутрішню структури організації Ansible, а саме: ролі, інвентарі та інші складові. Спроектровано модель розгалуження Git (feature/* → main) та повний конфігураційний файл .gitlab-ci.yml із чотирма стадіями, де деплой у production захищений ручним підтвердженням (when: manual).

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Розгортання середовища та DevOps-інфраструктури

Встановлення та конфігурація сервісів GitLab CE та GitLab Runner

Розгортання DevOps-інфраструктури розпочинається зі встановлення централізованого Git-сервера з вбудованим CI/CD-рушієм. У межах даного проекту GitLab CE та GitLab Runner розгорнуто на єдиній віртуальній машині Ubuntu Server, що забезпечує прямий мережевий доступ runner-а до всіх пристроїв GNS3 без додаткових мережевих налаштувань.

Для розгортання віртуальної машини було обрано Ubuntu Server 26.04, виділено 4096 ОП, 2 процесори та виділено два адаптери один для зв'язку із мережею GNS3 інший для доступу до NAT, що можна підтвердити переглянувши рисунок 3.1.

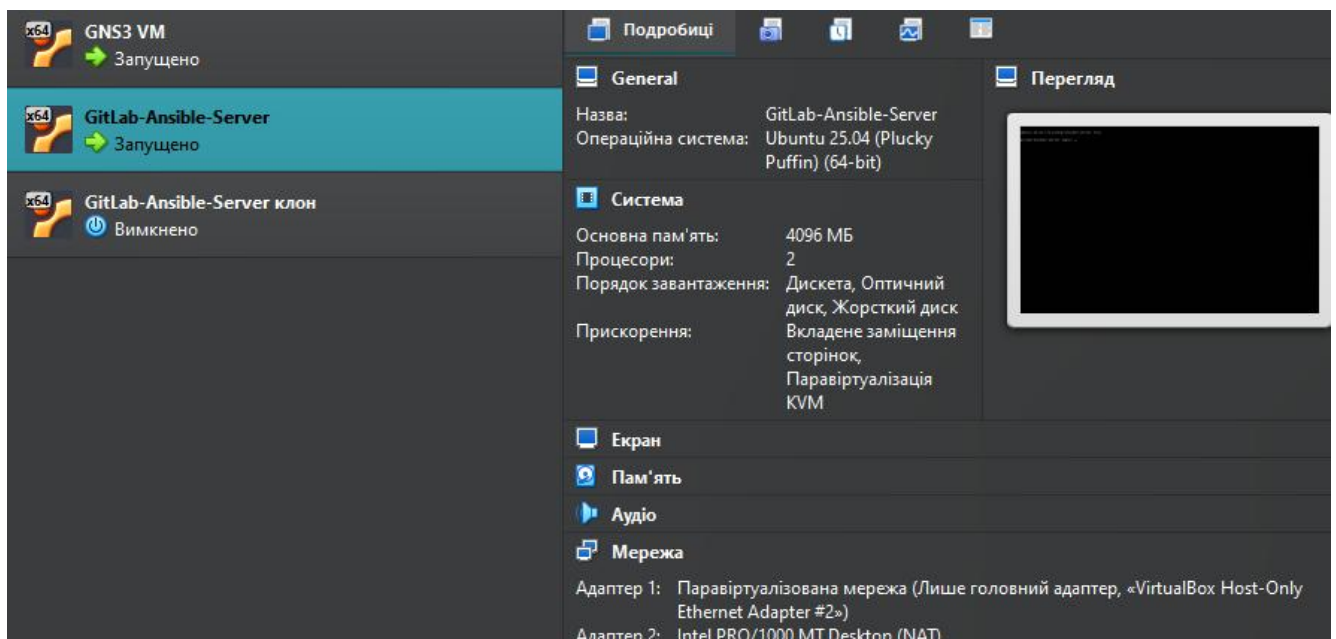


Рисунок 3.1 – Основна віртуальна машина

Перед тим як завантажувати Git-lab необхідно оновити систему:

```
sudo apt update && sudo apt upgrade -y
```

Та завантажити необхідні ключі доступу для SSH

```
sudo apt install -y curl openssh-server ca-certificates tzdata perl
```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Після чого, за бажанням можна увімкнути firewall та налаштувати доступність відповідним портам: 22, 80 та 443. (цей пункт можна пропустити, якщо перед цим брандмауер не був увімкнений) Перезавантажити службу SSH, щоб застосувались раніше завантажені пакети.

```
sudo systemctl enable --now ssh
```

```
sudo ufw allow 22/tcp
```

```
sudo ufw allow 80/tcp
```

```
sudo ufw allow 443/tcp
```

```
sudo ufw enable
```

Після чого можна переходити до завантаження пакунків GitLab CE

```
curl --location "https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh" | sudo bash
```

Якщо система не може завантажити цей пакунок, ймовірно, проблема у версії вашої ОС, якщо вона нова потрібно ввести додаткові команди параметри до рядку, а саме:

```
curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo os=ubuntu dist=jammy bash
```

Оскільки це лабораторний стенд без публічного DNS, простіше використати http:// з внутрішньою IP-адресою VM замість Let's Encrypt, який вимагає публічного домену й порту 80 ззовні:

```
sudo EXTERNAL_URL="http://10.0.10.10" apt install -y gitlab-ce
```

Слід зазначити, що було застосовано мережеву адресу, яка веде до топології мережі, це перший крок для налагодження GitLab Runner-a. Цю команди можна змінити, щоб пізніше не налагоджувати інші параметри, їх можна задати зараз в такому форматі:

```
sudo GITLAB_ROOT_EMAIL="admin@corp.local" \
```

```
GITLAB_ROOT_PASSWORD="StrongPassw0rd!" \
```

```
EXTERNAL_URL="http://10.0.10.10" \
```

```
apt install -y gitlab-ce
```

Ввівши команди ми забезпечимо вхід із раніше заданим паролем та поштою.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

Процес встановлення GirLab CE, було продемонстровано на рисунках 3.2-3.

```
student@gitlab-ansible-server:~/network-automation$ curl -sS https://packages.gitlab.com/install/repositories/gitlab
u dist=jammy bash
В кeмi:1 http://ua.archive.ubuntu.com/ubuntu resolute InRelease
В кeмi:2 http://ua.archive.ubuntu.com/ubuntu resolute-updates InRelease
В кeмi:3 http://ua.archive.ubuntu.com/ubuntu resolute-backports InRelease
В кeмi:4 http://security.ubuntu.com/ubuntu resolute-security InRelease
Отр:5 https://packages.gitlab.com/gitlab/gitlab-ce/ubuntu/jammy jammy InRelease [16,9 kB]
Отр:6 https://packages.gitlab.com/gitlab/gitlab-ce/ubuntu/jammy jammy/main amd64 Packages [32,7 kB]
Отримано 49,6 kB за 2с (29,4 kB/s)
Читаю переліки пакунків... Виконано
Repository configured successfully.
Ready to install packages.
student@gitlab-ansible-server:~/network-automation$ sudo EXTERNAL_URL="http://10.0.10.10" GITLAB_ROOT_PASSWORD="Shu
```

Рисунок 3.2 – Встановлення необхідних пакунків

```
[2026-06-23T21:01:48+00:00] INFO: env_dir[/opt/gitlab/etc/postgres-exporter/env] sending restart action
Recipe: monitoring::postgres-exporter
* runit_service[postgres-exporter] action restart (up to date)
[2026-06-23T21:01:49+00:00] INFO: Cinc Client Run complete in 272.938499091 seconds

Running handlers:
[2026-06-23T21:01:49+00:00] INFO: Running report handlers
Running handlers complete
[2026-06-23T21:01:49+00:00] INFO: Report handlers complete
Infra Phase complete, 632/1735 resources updated in 04 minutes 33 seconds

Notes:
Default admin account has been configured with following details:
Username: root
Password: You didn't opt-in to print initial root password to STDOUT.

NOTE: Because these credentials might be present in your log files in plain text, it is highly recommended
that you change your password. See https://gitlab.com/user/profile/user_passwords/#change-your-password.

gitlab Reconfigured!

      .
    .:O:
  .:ccc:
.:ccccc:
.cccccc:
ccccc:
:ccccc:; ..... 'ccccc'
cccccccccccccccccccccccccccccccccccccccc
:oo1cccccccccccccccccccccccccccccccccccccccc
oooooooo1cccccccccccccccccccccccccccccccccccc
;oooooooo1cccccccccccccccccccccccccccccccccccc
oooooooooooo1cccccccccccccccccccccccccccccccc
.oooooooooooo1cccccccccccccccccccccccccccccccc
cccccccccccccccccccc111cccccccccccccccccccc
 1cccccccccccccccccccccccccccccccccccccccc
.ooo1cccccccccccccccccccccccccccccccccccccccc
.1cccccccccccccccccccccccccccccccccccccccccccc
:kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
:kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
:d

Thank you for installing GitLab!
```

Рисунок 3.3 – Успішно завантажені пакунки

Після завантаження утиліти у нас з'явиться можливість входу, через раніше обрану адресу, в даному випадку 10.0.10.10, результат виводу показано на рисунку 3.4.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

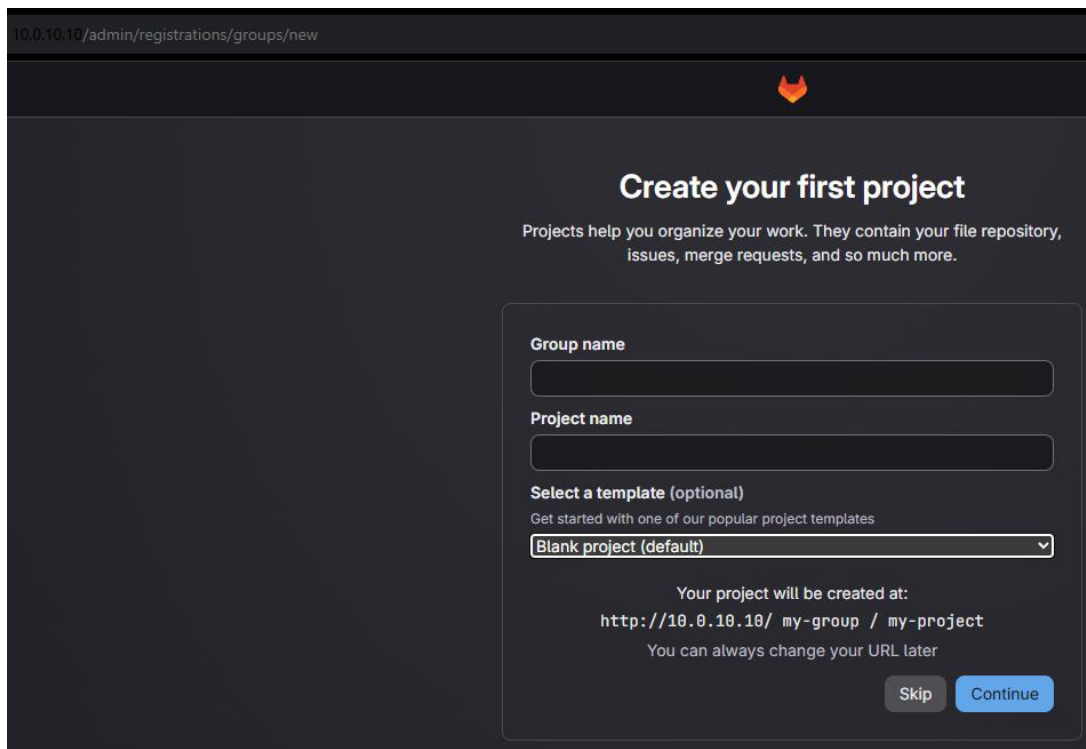


Рисунок 3.4 – Успішний вхід на початкову сторінку Gitlab

Щоб продовжувати роботу далі і прив'язати нашу майбутню або вже наявну структуру ansible, потрібно створити проєкт із назвою папки ansible, в якій є або буде вся структура, процес створення показано на рисунку 3.5-6.

Рисунок 3.5 – Створення проєкту на GitLab

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

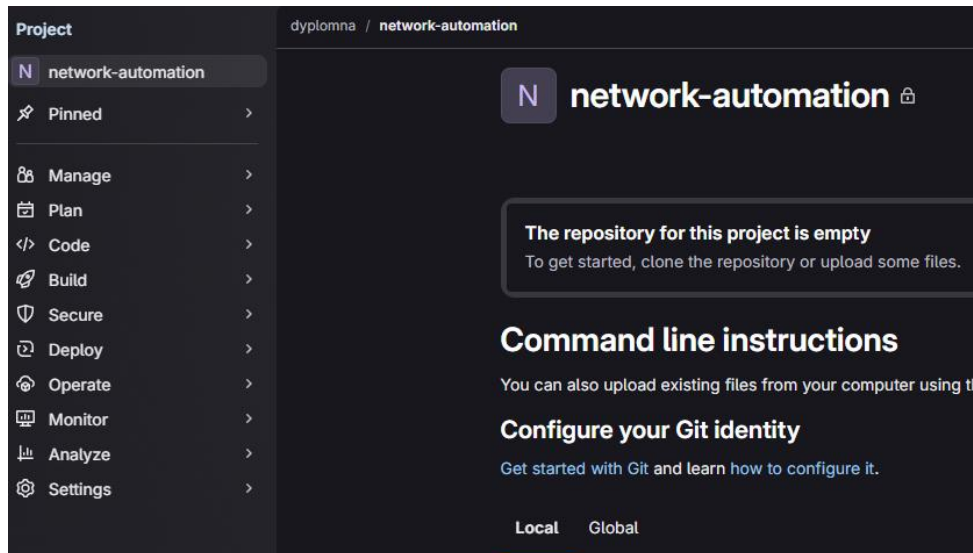


Рисунок 3.6 – Успішний вхід та створена відповідна група

Тепер опишемо як налагодити GitLab Runner, перейдемо в командний рядок нашого сервера, зараз нам необхідно завантажити пакунка для встановлення GitLab Runner, щоб це реалізувати вводимо команду нижче:

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash
```

Після встановлення потрібно налагодити зв'язок, для цього нам необхідно створити токен за шляхом Project → Settings → CI/CD → Runners → Project runners → Create project runner, прикладом успішного створення токена є рисунок 3.7.

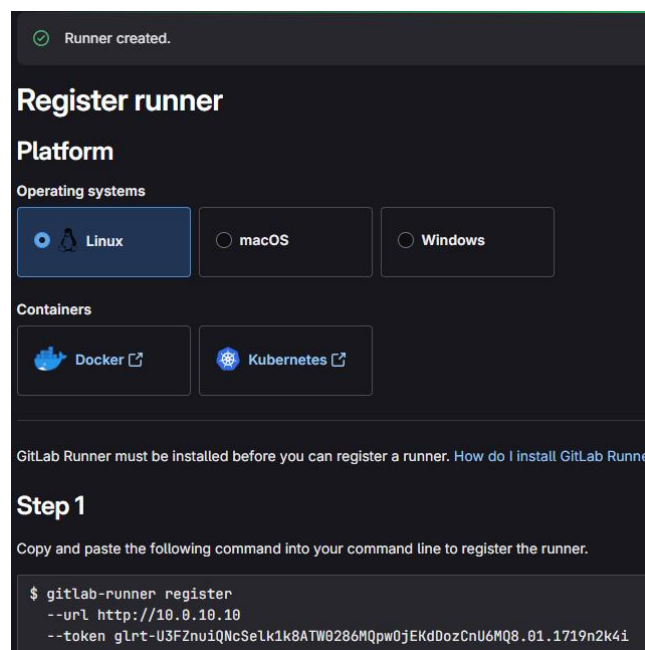


Рисунок 3.7 – Створений токен для під'єднання Runner-а

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Після токена необхідно вже приєднати runner для цього вводимо команду нижче, вводимо необхідні поля, результат приєднання показано на рисунку 3.8.

sudo gitlab-runner register

```
student@gitlab-ansible-server:~/network-automation$ sudo gitlab-runner register
[sudo: authenticate] Password:
Runtime platform                                arch=amd64 os=linux pid=52213 rev=
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://10.0.10.10
Enter the registration token:
glrt-U3FZnuiQMcSelklk8ATW0286MQpwOjEKdDozCnU6MQ8.01.1719n2k4i
Verifying runner... is valid                    correlation_id=01KVV9BCFV4NWWRCWHV
Enter a name for the runner. This is stored only in the local config.toml file:
[gitlab-ansible-server]: Network Automation Runner
Enter an executor: parallels, custom, instance, docker-windows, docker-autoscaler, docker
shell
Runner registered successfully. Feel free to start it, but if it's running already the
Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.
student@gitlab-ansible-server:~/network-automation$
```

Рисунок 3.8 – Створений токен для під'єднання Runner-а

Щоб перевірити чи справді все працює, можемо переглянути статуси із самого сервера або ж перейти по вкладкам Settings → CI/CD → Runners та подивитися, які runner-и працюють, також можна переглянути версію Ansible, щоб перевірити/затвердити його наявність, результат пошуку стану можна переглянути на рисунках 3.9-10.

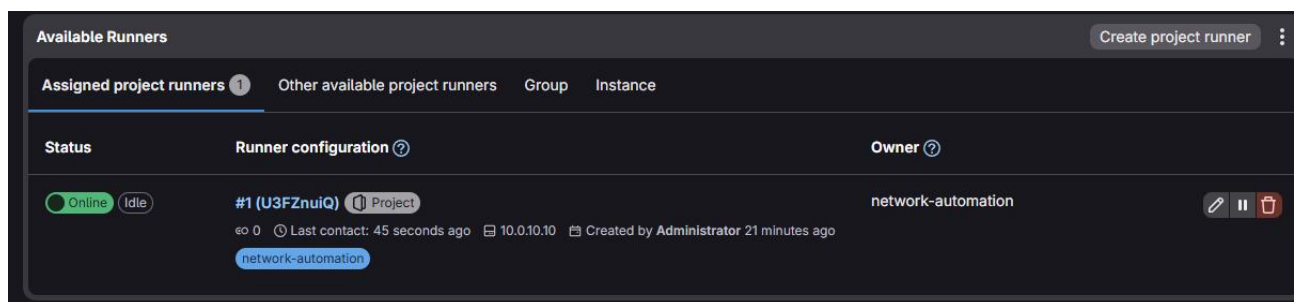


Рисунок 3.9 – Стан Runner-а за сайту

```
student@gitlab-ansible-server:~/network-automation$ sudo -u gitlab-runner ansible --version
ansible [core 2.20.1]
  config file = None
  configured module search path = ['/home/gitlab-runner/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/gitlab-runner/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.14.4 (main, Apr  8 2026, 04:02:31) [GCC 15.2.0] (/usr/bin/python3)
  jinja version = 3.1.6
  pyyaml version = 6.0.3 (with libyaml v0.2.5)
```

Рисунок 3.10 – Під'єднана тека Runner-а до ansible

Ініціалізація та налаштування базової топології в середовищі GNS3

Після розгортання DevOps-інфраструктури наступним кроком є підготовка мережевого середовища – початкове ручне налаштування SSH-доступу на кожному пристрої як передумови для роботи Ansible, та ініціалізація Git-репозиторію.

Щоб далі працювати із мережею потрібно виконати мінімальні налаштування, завдяки яким можна буде продовжувати роботу, зазвичай налаштовується адреса чи то віртуальна, чи то фізична та генерується rsa ключ для SSH та вмикається під'єднання по лініям VTY. Але конкретно в нашому проєкті, до базових налагодження було додано протокол STP, щоб не налаштовувати один пристрій за одним та була доступна вся мережа відразу. Приклади застосування команд показано у додатках Ж,И,К та Л.

Після налагодження базових команд потрібно перевірити мережу на доступність, використаємо команду ping до всіх пристроїв в мережі, результат показано на рисунку 3.11.

```
student@gitlab-ansible-server:~$ for ip in 10.0.10.1 10.0.10.2 10.0.10.11 10.0.10.12 10.0.10.21 10.0.10.22 10.0.10.23; do
  ping -c 2 -W 2 $ip | grep -E "rtt|100%|packets" | tail -1
  echo "  -> $ip"
done
rtt min/avg/max/mdev = 0.878/1.402/1.926/0.524 ms
  -> 10.0.10.1
rtt min/avg/max/mdev = 0.782/1.247/1.713/0.465 ms
  -> 10.0.10.2
rtt min/avg/max/mdev = 1.053/1.157/1.261/0.104 ms
  -> 10.0.10.11
rtt min/avg/max/mdev = 1.002/1.002/1.002/0.000 ms
  -> 10.0.10.12
rtt min/avg/max/mdev = 1.131/1.131/1.131/0.000 ms
  -> 10.0.10.21
rtt min/avg/max/mdev = 0.997/1.065/1.133/0.068 ms
  -> 10.0.10.22
rtt min/avg/max/mdev = 1.274/1.274/1.274/0.000 ms
  -> 10.0.10.23
student@gitlab-ansible-server:~$
```

Рисунок 3.11 – Перевірка зв'язку мережі

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Тепер ми маємо, майже, готове середовище для роботи із мережею, навіть не заходячи на їх пристрої, тепер нам потрібно під'єднати раніше створений проєкт до вже раніше написаної структури ansible, опис всіх ролей та тасків буде згодом, а зараз можна переглянути додаток М.

Те що ми під'єднали репозиторій до нашого ansible це, майже, повне розгортання гітлаба, нам залишилося додати дві змінні, які будуть відповідати за пароль Vault (ANSIBLE_VAULT_PASSWORD), в якому зберігається пароль від користувача SSH та додати змінну SSH_KNOWN_HOSTS_SKIP. Рисунок 3.12 є підтвердженням дій.

Key ↑	Value	Environments
ANSIBLE_VAULT_PASSWORD		All (default)
SSH_KNOWN_HOSTS_SKIP		All (default)

Рисунок 3.12 – Налагоджені змінні на GitLab CE

На цьому ми повністю завершили базове під'єднання мережі до нашого Ubuntu Server.

3.2 Програмна реалізація плейбуків та ролей Ansible для обладнання Cisco

Розробка шаблонів Jinja2 та реалізація ролей базового налаштування, VLAN та LACP

Основу системи автоматизації складають Ansible-ролі – модульні, повторно використовувані одиниці коду, кожна з яких відповідає за конкретний функціональний аспект конфігурації мережі. Реалізація ролей спирається на декларативний підхід: замість послідовності CLI-команд інженер описує бажаний стан пристрою, а Ansible самостійно визначає необхідні дії для його досягнення.

Щоб переглянути кінцевий результат файлової структури зверніться до додатку Д. Ми пройдемося по кожній ролі, яка є в цьому ansible, а почнемо ми з ролі base, у ній налаштовується лише NTP, як одна із базових потреб мережі, на рисунку чітко видно всі команди.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

GNU nano 8.7.1 roles/base/tasks/main.yml
--
- name: Налаштування NTP Master на маршрутизаторі
  cisco.ios.ios_config:
    lines:
      - ntp master 3
    when: inventory_hostname == 'router-1'

- name: Налаштування NTP клієнтів на комутаторах
  cisco.ios.ios_config:
    lines:
      - ntp server 10.0.0.1
    when: "'routers' not in group_names"

```

Рисунок 3.13 – Команди для налагодження часу в ролі base

Task: Налаштування NTP Master. Застосовує команду ntp master 3.

Task: Налаштування NTP клієнтів. Застосовує команду ntp server 10.0.0.1.

Тепер переглянемо роль vlans, яка розділяється на tasks/mail.yml та vars/mail.yml, рисунки 3.14-15 для демонстрації.

```

GNU nano 8.7.1
--
- name: Створення бази VLAN
  cisco.ios.ios_vlans:
    config:
      - vlan_id: "{{ item.vlan_id }}"
        name: "{{ item.name }}"
        state: merged
    loop: "{{ vlan_list }}"

- name: Дозвіл клієнтських VLAN на Trunk-інтерфейсах
  cisco.ios.ios_l2_interfaces:
    config:
      - name: "{{ item }}"
        trunk:
          allowed_vlans: 1,10,20,30,40,99
        state: merged
    loop: "{{ trunk_interfaces }}"
    when: trunk_interfaces is defined

```

Рисунок 3.14 – Команди для налагодження вланів в ролі vlans

```

GNU nano 8.7.1
--
vlan_list:
  - vlan_id: 10
    name: MGMT
  - vlan_id: 20
    name: SERVERS
  - vlan_id: 30
    name: USERS
  - vlan_id: 40
    name: VOIP
  - vlan_id: 99
    name: NATIVE

```

Рисунок 3.15 – Допоміжні команди для налагодження вланів в ролі vlans

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

vlan_list – це структурований список, де для кожного VLAN чітко задано два параметри: його номер та текстова назва.

Task: Створення бази VLAN. За один запуск таска на комутаторі створюються абсолютно всі VLAN 10, 20, 30, 40, 99 з відповідними іменами.

Task: Дозвіл VLAN на Trunk-інтерфейсах. Переводить кожен порт із циклу в режим trunk і жорстко прописує команду switchport trunk allowed vlan 1,10,20,30,40,99.

Реалізація ролей маршрутизації (OSPF), резервування (HSRP) та безпеки (Port-Security, Core ACL)

Після реалізації базових ролей налаштування переходить до функціональних ролей, що відповідають за протокольний стек мережі та політики безпеки. Кожна роль реалізує конкретну технологію, описану у проєктному рішенні другого розділу.

В цьому підрозділі ми розберемо всі інші ролі по черзі. Першою розберемо маршрутизацію на рисунку

```
GNU nano 8.7.1
--
- name: Перевірка стану маршрутизації (наявність 'no ip routing')
  cisco.ios.ios_command:
    commands: show run | include no ip routing
  register: routing_state
  when: "'routers' not in group_names"
  changed_when: false

- name: Увімкнення IP-маршрутизації (тільки для L3-комутаторів)
  cisco.ios.ios_config:
    lines:
      - ip routing
    when:
      - "'routers' not in group_names"
      - "'no ip routing' in routing_state.stdout[0]"

- name: Базове налаштування процесу OSPF
  cisco.ios.ios_config:
    parents: router ospf 1
    lines:
      - network 10.0.0.0 0.255.255.255 area 0
```

Рисунок 3.16 – Командний вміст ролі routing

Task: Перевірка стану маршрутизації. Виконує команду `show run | include no ip routing`, щоб дізнатися, чи вимкнено маршрутизацію.

Task: Увімкнення IP-маршрутизації. Надсилає команду `ip routing` для активації функцій Layer 3 на комутаторі.

Task: Базове налаштування процесу OSPF. Всередині цього процесу виконує команду `network 10.0.0.0 0.255.255.255 area 0`, яка вмикає OSPF на всіх інтерфейсах, чії IP-адреси починаються з 10., та відносить їх до магістральної зони.

Далі розглянемо роль HSRP, її вміст показано на рисунку 3.17

```
--
- name: Налаштування HSRP на SW-DIST-1 (Active)
  cisco.ios.ios_config:
    parents: interface Vlan{{ item }}
    lines:
      - ip address 10.0.{{ item }}.2 255.255.255.0
      - standby {{ item }} ip 10.0.{{ item }}.1
      - standby {{ item }} priority 110
      - standby {{ item }} preempt
    loop:
      - 20
      - 30
      - 40
    when: inventory_hostname == 'sw-dist-1'

- name: Налаштування HSRP на SW-DIST-2 (Standby)
  cisco.ios.ios_config:
    parents: interface Vlan{{ item }}
    lines:
      - ip address 10.0.{{ item }}.3 255.255.255.0
      - standby {{ item }} ip 10.0.{{ item }}.1
      - standby {{ item }} preempt
    loop:
      - 20
      - 30
      - 40
    when: inventory_hostname == 'sw-dist-2'
```

Рисунок 3.17 – Командний вміст ролі HSRP

Task: Налаштування HSRP на SW-DIST-1 (Active). Налаштовує протокол резервування шлюзу HSRP на першому комутаторі розподілу sw-dist-1, роблячи його основним Active. За допомогою циклу `loop: 20, 30, 40` завдання проходить по віртуальних інтерфейсах VLAN і налаштовує для кожного.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Task: Налаштування HSRP на SW-DIST-2 (Standby). Налаштовує резервний Standby комутатор розподілу sw-dist-2 для тих самих VLAN 20, 30 та 40. У цьому блоці налаштовується:

Розглянемо структуру ролі DHCP, в якій реалізовано виключені адреси, пули SERVERS/USERS/VOIP та DHCP-relay на Distribution. Її вміст можна переглянути в додатку Н.

Task: Виключення IP-адрес шлюзів та свічів із роздачі DHCP. Налаштовує центральний маршрутизатор router-1, щоб він не видавав клієнтам перші 10 IP-адрес від .1 до .10 у трьох різних підмережах 10.0.20.0, 10.0.30.0, 10.0.40.0. Ці адреси резервуються exclude для статичного призначення наприклад, для віртуальних шлюзів HSRP та самих комутаторів.

Task: Створення DHCP-пулів для клієнтських мереж. Створює на тому ж маршрутизаторі router-1 три пули адрес для динамічної роздачі. За допомогою циклу генеруються пули з іменами SERVERS, USERS та VOIP. Для кожного пулу автоматично прописується підмережа, адреса шлюзу за замовчуванням(default-router, що закінчується на .1 та публічний DNS-сервер від Google.

Task: Налаштування DHCP Relay (ip helper-address) на Distribution. Виконується на комутаторах рівня розподілу. Оскільки клієнтські пристрої знаходяться у своїх VLAN 20, 30, 40, а DHCP-сервер знаходиться за їхніми межами на маршрутизаторі, комутатори отримують команду ip helper-address 10.0.0.1 на своїх інтерфейсах VLAN. Це дозволяє комутаторам перехоплювати ширококомовні запити на отримання IP від клієнтів і пересилати їх на пряму до router-1.

Task: Гарантуємо, що клієнтські VLAN інтерфейси увімкнені. Використовує декларативний модуль cisco.ios.ios_interfaces для перевірки стану віртуальних інтерфейсів Vlan20, Vlan30, Vlan40 на комутаторах розподілу. Параметр enabled: true гарантує, що ці інтерфейси адміністративно увімкнені.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Перейдемо до ролі security в ній налагоджено Port-Security на портах доступу та динамічне призначення VLAN. Переглянути вміст цієї ролі можна в додатку П.

Task: Адміністративне увімкнення клієнтських портів. Проходить по списку портів від Ethernet2/0 до Ethernet2/3 і переводить їх в активний стан. Виконується виключно на пристроях групи access.

Task: Налаштування Access-портів (Динамічний VLAN). Використовує динамічний словник змінних access_ports для налаштування портів. Переводить їх у режим доступу switchport mode access, призначає кожному свій VLAN згідно зі змінними та додає текстовий опис інтерфейсу.

Task: Налаштування Port-Security для USERS. Активує захист портів на рівні доступу. Команди обмежують кількість дозволених MAC-адрес до двох, вмикають їхнє динамічне збереження в конфігурацію та встановлюють політику violation restrict, за якої трафік від невідомих MAC-адрес відкидається, але сам порт не вимикається.

Task: Налаштування списків доступу (ACL) безпеки. За допомогою модуля cisco.ios.ios_acls створює два правила фільтрації на комутаторах ядра (core):

- Розширений ACL PROTECT-SERVERS: дозволяє TCP-трафік з клієнтської підмережі 10.0.30.0 до серверної 10.0.20.0 тільки по портах 80 та 443, а весь інший IP-трафік блокує.
- Стандартний ACL RESTRICT-SSH: дозволяє підключення лише з однієї конкретної IP-адреси 10.0.10.10, блокуючи всі інші.

Task: Застосування ACL PROTECT-SERVERS до інтерфейсу серверів. Навіщує створений раніше список доступу PROTECT-SERVERS на віртуальний інтерфейс Vlan20 у вихідному напрямку. Це фізично застосовує правило фільтрації до трафіку, що йде до серверів.

Task: Застосування ACL RESTRICT-SSH на лінії VTY для захисту SSH. Застосовує список RESTRICT-SSH до віртуальних ліній керування

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

маршрутизатором за допомогою команди `access-class` у вхідному напрямку. Це гарантує, що керувати обладнанням по SSH можна лише з дозволеного сервера.

Також переглянемо змінні інвентарів, а почнемо саме з `group_vars/all/vars.yml`. Вміст цього елементу можна переглянути в додатку Р.

Вміст суто технічний, всім ролям, яким необхідно взяти, якусь інформацію будуть отримувати дані саме з цього файлу.

Останнім елементом перед плейбуком, який ми будемо розглядати це `group_vars/all/vaults.yml`, його вміст зображено на рисунку



```
ansible_password: cisc0123
ansible_become_password: enable123
```

Рисунок 3.18 – Розшифрований вміст `vaults.yml`

Тут лише паролі від SSH та для оболонки `enable`.

Завершемо цей розділ це буде плейбуком `site.yml`, у ньому розгортаються всі раніше перераховані ролі, його вміст можна переглянути в додатку С.

Плейбук майже не містить окремих завдань. Замість цього він імпортує Ansible-ролі `base`, `vlsns`, `routing`, `hsrp`, `dhcp`, `security`.

Розбір етапів (плеїв)

Етап: Перевірка зв'язку. Застосовується на всіх пристроях (`hosts: all`). Збирає мінімальну інформацію. Це гарантує, що Ansible має SSH-доступ до кожного пристрою, перш ніж почати вносити зміни.

Етап: Базові налаштування інфраструктури. Застосовується на всіх пристроях (`hosts: all`). Запускає роль `base`.

Етап: Налаштування L2-рівня (VLAN). Застосовується тільки на комутаторах (`hosts: core, distribution, access`). Запускає роль `vlsns`, яка створює базу VLAN і налаштовує транкові з'єднання між свічами.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Етап: Налаштування L3-рівня (OSPF) . Застосовується на маршрутизаторі і комутаторах верхніх рівнів (hosts: routers, core, distribution). Піднімає динамічну маршрутизацію, щоб різні сегменти мережі могли бачити один одного.

Етап: Налаштування резервування (HSRP) . Застосовується тільки на рівні розподілу (hosts: distribution). Створює віртуальні IP-шлюзи для клієнтських мереж, забезпечуючи відмовостійкість, якщо один з комутаторів впаде.

Етап: Налаштування сервісів (DHCP) . Застосовується на маршрутизаторі та рівні розподілу (hosts: routers, distribution). Налаштовує DHCP-сервер на роутері та DHCP Relay на комутаторах, щоб клієнти могли автоматично отримувати IP-адреси.

Етап: Безпека портів (Port-Security) . Застосовується тільки на комутаторах доступу (hosts: access). Запускає роль security, яка робить налагодження на кінцевих портах: обмеження MAC-адрес, захист від несанкціонованих підключень.

3.3 Налаштування конвеєра CI/CD та практична перевірка сценаріїв роботи

Автоматизоване розгортання початкової конфігурації мережі з нуля

Першим практичним сценарієм тестування системи є розгортання повної конфігурації мережі з нульового стану – коли пристрої містять лише мінімальне початкове налаштування SSH. Цей сценарій демонструє здатність системи відтворити повний стек конфігурації за один запуск головного плейбука.

Перший запуск головного плейбука (site.yml) – це фактично перевірка готовності всієї вашої автоматизації. Його головна мета – переконатися, що Ansible може розшифрувати секрети, підключитися по SSH до кожного пристрою та зібрати базові факти, не вносячи жодних деструктивних змін.

Тепер розберемо кожну виконану подію, першою була звичайна перевірка зв'язку, як бачимо на рисунку, помилок не виникнуло.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```
PLAY [Перевірка зв'язку з пристроями Cisco]

TASK [Збір базової інформації (IOS Facts)] *
ok: [router-1]
ok: [sw-core-1]
ok: [sw-core-2]
ok: [sw-dist-1]
ok: [sw-dist-2]
ok: [sw-acc-1]
ok: [sw-acc-2]
ok: [sw-acc-3]
```

Рисунок 3.19 – Перший тестовий таск

Наступним буде розгортатися base, який налаштовує NTP, як бачимо на рисунку 3.20 необхідні зміни пройшли і процес пішов далі.

```
PLAY [Базові налаштування інфраструктури (NTP)] *****

TASK [base : Налаштування NTP Master на маршрутизаторі]
[WARNING]: To ensure idempotency and correct diff the i
changed: [router-1]
skipping: [sw-core-1]
skipping: [sw-core-2]
skipping: [sw-dist-1]
skipping: [sw-dist-2]
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

TASK [base : Налаштування NTP клієнтів на комутаторах]
skipping: [router-1]
changed: [sw-core-1]
changed: [sw-core-2]
changed: [sw-dist-1]
changed: [sw-dist-2]
changed: [sw-acc-1]
changed: [sw-acc-2]
changed: [sw-acc-3]
```

Рисунок 3.20 – Роль base в розгортанні

Відомості щодо таску про VLAN дивитися в додатках Т.

Відомості щодо таску про routing дивитися на рисунку 3.21.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

```

PLAY [Налаштування динамічної маршрутизації (OSPF)] *****

TASK [routing : Перевірка стану маршрутизації (наявність 'no ip routing')] *
skipping: [router-1]
ok: [sw-core-1]
ok: [sw-core-2]
ok: [sw-dist-1]
ok: [sw-dist-2]

TASK [routing : Увімкнення IP-маршрутизації (тільки для L3-комутаторів)] ***
skipping: [router-1]
skipping: [sw-core-1]
skipping: [sw-core-2]
changed: [sw-dist-1]
changed: [sw-dist-2]

TASK [routing : Базове налаштування процесу OSPF] *****
changed: [router-1]
changed: [sw-core-1]
changed: [sw-core-2]
changed: [sw-dist-1]
changed: [sw-dist-2]

```

Рисунок 3.21 – Роль routing в розгортанні

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Відомості щодо таску про HSRP дивитися на рисунку 3.22.

```
PLAY [Налаштування HSRP-резервування] *****

TASK [hsrp : Налаштування HSRP на SW-DIST-1 (Active)] *
changed: [sw-dist-1] => (item=20)
changed: [sw-dist-1] => (item=30)
changed: [sw-dist-1] => (item=40)
skipping: [sw-dist-2] => (item=20)
skipping: [sw-dist-2] => (item=30)
skipping: [sw-dist-2] => (item=40)
skipping: [sw-dist-2]

TASK [hsrp : Налаштування HSRP на SW-DIST-2 (Standby)]
skipping: [sw-dist-1] => (item=20)
skipping: [sw-dist-1] => (item=30)
skipping: [sw-dist-1] => (item=40)
skipping: [sw-dist-1]
changed: [sw-dist-2] => (item=20)
changed: [sw-dist-2] => (item=30)
changed: [sw-dist-2] => (item=40)
```

Рисунок 3.22 – Роль HSRP в розгортанні

Відомості щодо таску про DHCP дивитися в додатках У.

Відомості щодо таску про security дивитися в додатках Ф.

Фінальний вивід команди продемонстровано на рисунку

```
PLAY RECAP *****
router-1      : ok=5    changed=4    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
sw-acc-1     : ok=8    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-acc-2     : ok=8    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-acc-3     : ok=8    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-core-1    : ok=6    changed=3    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
sw-core-2    : ok=6    changed=3    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
sw-dist-1    : ok=10   changed=7    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-dist-2    : ok=10   changed=7    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
```

Рисунок 3.23 – Підсумок за плейбуком

Внесення конфігураційних змін через Merge Request

Другий сценарій демонструє повний цикл внесення зміни до конфігурації мережі через процес Merge Request – від створення робочої гілки до автоматичного деплою на staging та ручного підтвердження на production.

Було штучно змінено деякі файли, щоб гіт міг розпізнати зміни, всі файли було додані та закомічені та запущені на GitLab, процес переведення можна побачити на рисунку 3.24. Після успішного заливання, на головній сторінці проєкту з'являється MR на впровадження коміту, це показано на рисунку 3.25.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Після створення MR, запускається пайплайн, який запускає lint та dry-run, це показано на рисунках 3.26-27.

```
student@gitlab-ansible-server:~/network-automation$ git checkout -b feature/vars-change
Switched to a new branch 'feature/vars-change'
student@gitlab-ansible-server:~/network-automation$ git add .
student@gitlab-ansible-server:~/network-automation$ git commit -m "fix: changed vars.yml"
[feature/vars-change 200bb53] fix: changed vars.yml
4 files changed, 15 insertions(+), 26 deletions(-)
student@gitlab-ansible-server:~/network-automation$ git push -u origin feature/Add-ACL
error: src refspec feature/Add-ACL does not match any
error: failed to push some refs to 'http://10.0.10.10/dyplomna/network-automation.git'
student@gitlab-ansible-server:~/network-automation$ git push -u origin feature/vars-change
Username for 'http://10.0.10.10': root
Password for 'http://root@10.0.10.10':
Enumerating objects: 93, done.
Counting objects: 100% (89/89), done.
Delta compression using up to 2 threads
Compressing objects: 100% (59/59), done.
Writing objects: 100% (63/63), 8.08 KiB | 826.00 KiB/s, done.
Total 63 (delta 23), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: To create a merge request for feature/vars-change, visit:
remote:   http://10.0.10.10/dyplomna/network-automation/-/merge_requests/new?merge_request%5Bsource_branch%5D=feature%2Fvars-change
remote:
To http://10.0.10.10/dyplomna/network-automation.git
 * [new branch]      feature/vars-change -> feature/vars-change
branch 'feature/vars-change' set up to track 'origin/feature/vars-change'.
student@gitlab-ansible-server:~/network-automation$
```

Рисунок 3.24 – Процес заливки даних на гіт

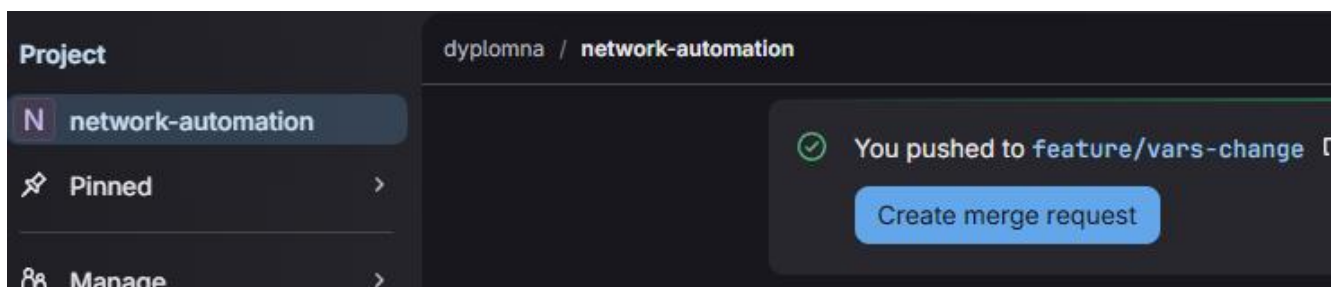


Рисунок 3.25 – Запит на впровадження змін

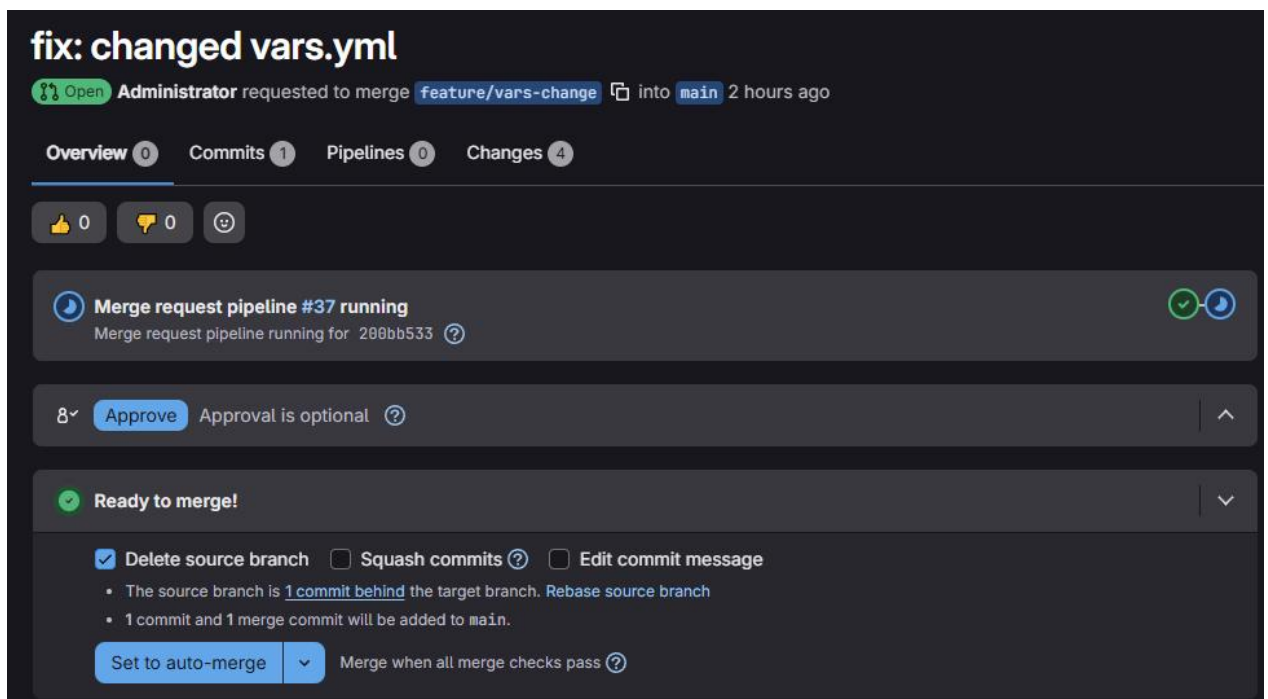


Рисунок 3.26 – Стан MR, пройшов етап lint, проходить dry-run

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 3.27 – Стан MR, пройшов обидва етапи

Після успішного першого пайплайну з'являється можливість переведення коду в основну гілку main, якщо натиснути merge так і станеться, дивитися стан на рисунках 3.28.

Останній етап можна запустити лише вручну як і показано на рисунку 3.29. Завершення роботи пайплайну показано на рисунку 3.30.

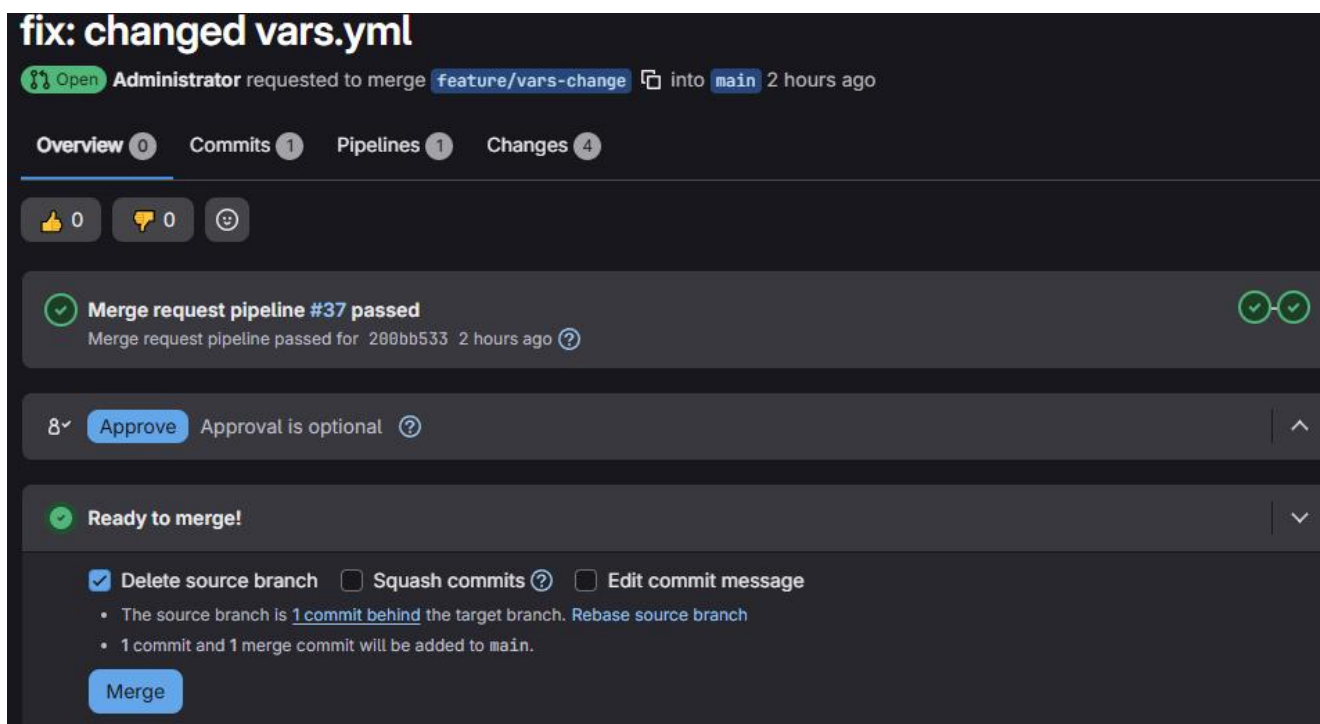


Рисунок 3.28 – Стан MR, пройшов обидва етапи

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

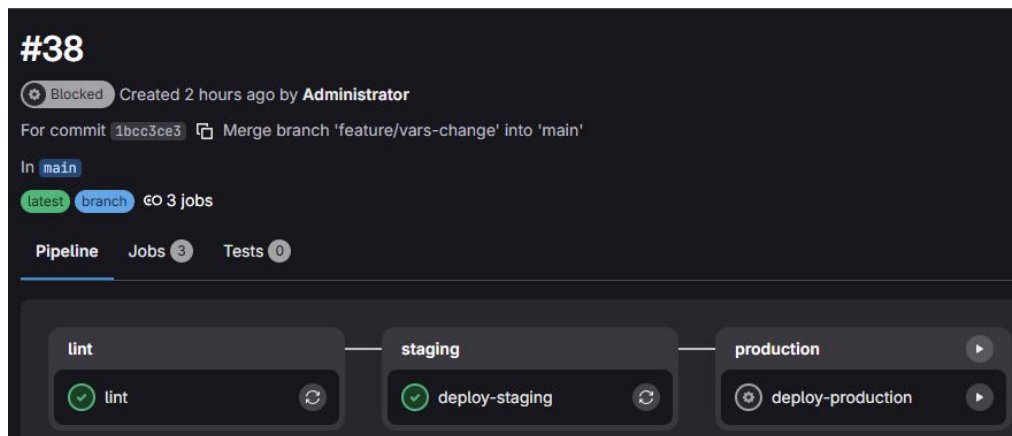


Рисунок 3.29 – Pipeline MR, останній етап лише з ручного нажиму

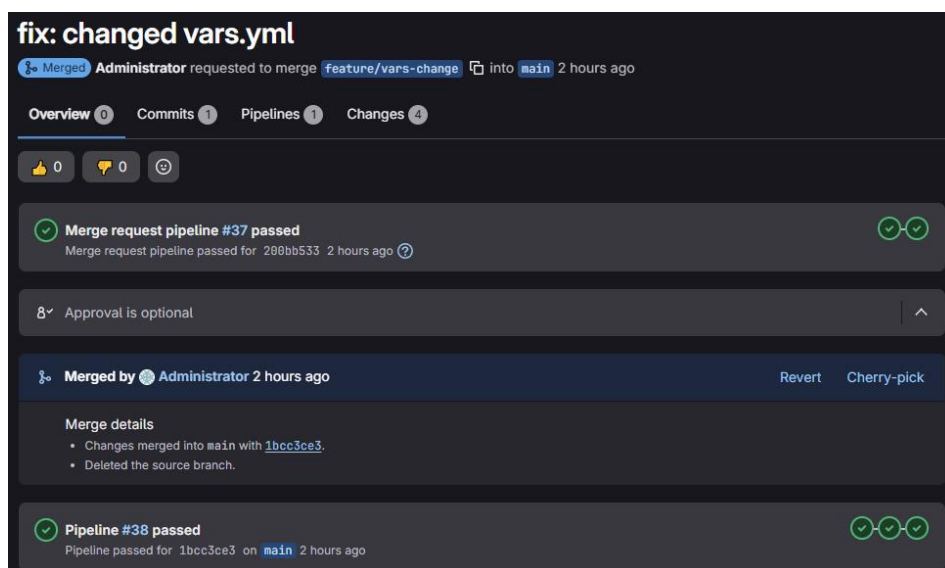


Рисунок 3.30 – Завершений стан MR

Опис файлу .gitlab-ci.yml

Файл.gitlab-ci.yml є серцем і головним інструментом автоматизації роботи. Він описує логіку, етапи та правила роботи CI/CD pipeline для автоматизації мережі. Вміст файлу в додатку X.

Етап 1: LINT. Виконує базову перевірку YAML-коду на наявність синтаксичних помилок (--syntax-check) у файлах site.yml та verify.yml. Це швидкий тест: він не підключається до обладнання, а просто перевіряє, чи не забули ви десь відступ або лапки. Запускається автоматично при створенні Merge Request та при будь-якому коміті в головну гілку main.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Етап 2: DRY-RUN (Холостий запуск). Підключається до обладнання зі списку staging, але не вносить жодних змін. Завдяки прапорцям --check --diff, Ansible лише моделює ситуацію і показує, що саме змінилося б, якби скрипт спрацював по-справжньому. Запускається тільки під час створення Merge Request `$CI_PIPELINE_SOURCE == "merge_request_event"`. Це дозволяє адміністратору переглянути наслідки змін у коді до того, як вони будуть затверджені.

Етап 3: DEPLOY STAGING (Автоматичне розгортання). Це перше реальне застосування конфігурації. Запускає розгортання `site.yml` на тестовій групі пристроїв `inventories/staging/`. Робить паузу на 30 секунд. Запускає плейбук верифікації `verify.yml`, який автоматично перевіряє, чи працює тестова мережа так, як очікувалося. Запускається автоматично, як тільки код потрапляє у головну гілку `main`.

Етап 4: DEPLOY PRODUCTION (Бойовий деплой). Виконує ті самі кроки, що й на етапі Staging, але цього разу код застосовується до справжнього обладнання `inventories/production/`. Запускається доступний лише для гілки `main`, містить критичний запобіжник – `when: manual`. Це означає, що GitLab не запустить цей етап автоматично. Пайплайн зупиниться і чекатиме, поки адміністратор зайде в інтерфейс GitLab і натисне кнопку "Play", підтверджуючи готовність оновити реальну мережу після успішних тестів на Staging.

3.4 Тестування, верифікація та аналіз результатів

Автоматична верифікація стану мережевих протоколів (плейбук `verify.yml`)

Після кожного деплою система автоматично запускає верифікаційний плейбук `verify.yml`, який виконує активне тестування мережі – підключається до пристроїв, збирає фактичний стан та перевіряє його відповідність очікуваному. На відміну від пасивного аналізу логів, такий підхід дає однозначну відповідь щодо коректності функціонування кожного сервісу.

Отриманий результат після запуску продемонстровано на рисунку 3.31. Стани тестів можна переглянути в додатку Ц.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

ПАС: NTP-ієрархію побудовано, сервер доступний.

PLAY RECAP *****
router-1      : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-acc-1      : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-acc-2      : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-acc-3      : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-core-1     : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-core-2     : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-dist-1     : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0
sw-dist-2     : ok=6    changed=0    unreachable=0    failed=0    skipped=4    rescued=0    ignored=0

```

Рисунок 3.31 – Підсумок по verify.yml

Оцінка та перевірка властивості ідемпотентності системи керування

Ідемпотентність є однією з ключових властивостей, що відрізняє декларативну систему автоматизації від простого набору скриптів. Система вважається ідемпотентною, якщо повторний запуск того самого плейбука без змін у коді не вносить жодних модифікацій до конфігурації пристроїв – незалежно від того, скільки разів він був виконаний раніше.

Як видно на рисунку 3.32 вся система є ідемпотентною.

```

PLAY RECAP *****
router-1      : ok=5    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
sw-acc-1      : ok=8    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-acc-2      : ok=8    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-acc-3      : ok=8    changed=0    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
sw-core-1     : ok=6    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
sw-core-2     : ok=6    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
sw-dist-1     : ok=9    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
sw-dist-2     : ok=9    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0

```

Рисунок 3.32 – Завершений стан MR

Висновки до третього розділу

У третьому розділі виконано практичну реалізацію спроектованої системи, проведено її налагодження та всебічне тестування.

Розгорнуто повноцінне DevOps-середовище на базі Ubuntu Server із встановленим GitLab CE, зареєстрованим Shell-runner та Ansible Core з колекцією cisco.ios. Виконано початкове ручне налаштування SSH-доступу на всіх восьми пристроях мережі (ROUTER-1, SW-CORE-1/2, SW-DIST-1/2, SW-ACC-1/2/3) як необхідної передумови для подальшої роботи Ansible. Підтверджено мережеву зв'язність між GitLab-Ansible-Server та усіма пристроями в діапазоні 10.0.10.1–10.0.10.23.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Розроблено та реалізовано шість Ansible-ролей: base (NTP, SSH, hostname), vlans (база VLAN, trunk-інтерфейси), routing (OSPF, увімкнення ip routing на L3-комутаторах), hsrp (резервування першого хопу), dhcp (пули, виключені адреси, ip helper-address на Distribution), security (Port-Security, динамічний VLAN на портах доступу). Підтверджено коректність виконання кожної ролі за результатами виводу PLAY RECAP – нульова кількість збоїв (failed=0) при першому запуску та нульова кількість змін (changed=0) при повторному запуску, що підтверджує ідемпотентність системи.

Налагоджено конвеєр CI/CD та відпрацьовано повний цикл внесення змін через Merge Request: від створення feature-гілки до автоматичного деплою на staging і ручного підтвердження деплою на production. Практично перевірено чотири Merge Request із реальними змінами конфігурації. Відпрацьовано сценарій налагодження – виправлення помилки конфігурації Ansible через новий MR з автоматичним перезапуском пайплайну. Верифікаційний плейбук verify.yml підтвердив коректний стан мережі за критеріями: наявність VLAN 30 (USERS), OSPF-сусідство у стані FULL, активність Port-Security, мережева доступність (Ping) та синхронізація NTP. Практично підтверджено роботу DHCP – кінцеві вузли WS-1 та WS-12 отримали адреси з відповідних пулів (10.0.20.12/24 та 10.0.30.15/24) через DHCP-relay на Distribution-рівні.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання – розроблено та практично реалізовано систему автоматизованого розгортання конфігурації корпоративної мережі з використанням практик DevNet/DevOps.

За результатами виконаної роботи досягнуто таких конкретних результатів:

1. Досліджено теоретичну базу. Проаналізовано трирівневу ієрархічну модель Cisco та методологію NetDevOps, обґрунтовано вибір Ansible як агентлесс-інструменту автоматизації з нативною підтримкою колекції cisco.ios, а також визначено чотиристадійну модель CI/CD конвеєра як основу контрольованого процесу керування змінами.

2. Спроектовано мережеву інфраструктуру. Розроблено повну топологію корпоративної мережі з 8 пристроями Cisco у середовищі GNS3, структурований адресний план із синхронізацією VLAN-ID та третього октету IP-адреси, схему HSRP-резервування та Point-to-Point магістральних з'єднань. Спроектовано модульну структуру Ansible-проєкту з розмежуванням середовищ та моделлю розгалуження Git.

3. Реалізовано систему автоматизації. Розгорнуто DevOps-інфраструктуру на базі GitLab CE з Shell-runner. Розроблено шість Ansible-ролей, що охоплюють повний стек конфігурації: базові налаштування, VLAN, маршрутизацію OSPF, резервування HSRP, DHCP-сервіс та безпеку портів. Загальна кількість керованих параметрів охоплює 8 пристроїв та понад 30 конфігураційних завдань.

4. Підтверджено ключові властивості системи. Верифікаційний плейбук `verify.yml` підтвердив коректний стан мережі за всіма п'ятьма перевірочними критеріями. Повторний запуск плейбуків без змін у коді дав результат `changed=0` на всіх пристроях, що підтверджує ідемпотентність системи. Практичне відпрацювання чотирьох Merge Request із реальними змінами конфігурації продемонструвало повний аудит-trail кожної зміни з прив'язкою до автора, коду та результату пайплайну.

5. Підтверджено практичну цінність підходу. На відміну від ручного CLI-адміністрування, розроблена система забезпечує: відтворюваність – один і той

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

самий Ansible-плейбук дає ідентичний результат незалежно від початкового стану пристроїв; контрольованість змін – будь-яка модифікація конфігурації мережі простежується до конкретного коміту, автора та результату dry-run; захист від помилок – стадія dry-run показує точний diff змін на реальному обладнанні ще до їх застосування, а ручне підтвердження деплою у production унеможливорює випадкове застосування неперевіреної конфігурації.

Таким чином, розроблена система підтверджує доцільність і практичну застосовність підходу Network as Code для управління конфігурацією корпоративної мережі та може бути використана як основа для впровадження аналогічних рішень на реальних підприємствах.

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Systems. Hierarchical Network Design Overview. Cisco Press, 2022. URL: <https://www.cisco.com/c/en/us/support/docs/architecture-design/hierarchical-network-design/13608-hierarchical-network-design.html> (дата звернення: 15.05.2026).
2. Odom W. CCNA 200-301 Official Cert Guide, Volume 1. Cisco Press, 2020. 800 p.
3. Oppitz M., Tomsu P. Enterprise Network Architecture. Springer, 2018. 412 p.
4. Edelman J., Lowe S. S., Oswalt J. Network Programmability and Automation: Skills for the Next-Generation Network Engineer. O'Reilly Media, 2018. 582 p
5. CI/CD YAML syntax reference / GitLab Documentation. GitLab CE Architecture Guide, 2025. URL: <https://docs.gitlab.com/ee/ci/yaml/> (дата звернення: 15.05.2026).
6. Using CI/CD to Automate Network Configuration and Deployment. NVIDIA Developer Blog. URL: <https://developer.nvidia.com/blog/using-ci-cd-to-automate-network-configuration-and-deployment/> (дата звернення: 15.05.2026).
7. Article 3177851. Cisco Press. URL: <https://www.ciscopress.com/articles/article.asp?p=3177851> (дата звернення: 25.05.2026).
8. NetDevOps. Network to Code. URL: <https://networktoencode.com/what-we-do/netdevops/> (дата звернення: 25.05.2026).
9. NetDevOps Concepts: Infrastructure as Code (IaC). Network to Code Blog. URL: <https://networktoencode.com/blog/netdevops-concepts-iac/> (дата звернення: 25.05.2026).
10. Ansible vs Puppet vs Chef. IPCisco. URL: <https://ipcisco.com/lesson/ansible-vs-puppet-vs-chef/> (дата звернення: 27.05.2026).
11. Ansible Documentation. URL: <https://docs.ansible.com/> (дата звернення: 27.05.2026).

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

12. CI/CD Demo NetDevOps. Cisco Developer Blog. URL: <https://blogs.cisco.com/developer/ci-cd-demo-netdevops-9> (дата звернення: 27.05.2026).

13. Three-Tier Architecture. Network Academy. URL: <https://www.networkacademy.io/ccna/network-fundamentals/three-tier-architecture> (дата звернення: 04.06.2026).

14. Cisco Access Control Lists (ACL). Cisco Community. URL: <https://community.cisco.com/t5/networking-knowledge-base/cisco-access-control-lists-acl/ta-p/4182349> (дата звернення: 04.06.2026).

15. Cisco Packet Tracer. Networking Academy. URL: <https://www.netacad.com/cisco-packet-tracer> (дата звернення: 06.06.2026).

16. EVE-NG: Emulated Virtual Environment Next Generation. URL: <https://www.eve-ng.net/> (дата звернення: 06.06.2026).

17. GNS3 Documentation. URL: <https://docs.gns3.com/docs/> (дата звернення: 06.06.2026).

18. RFC 1058. Routing Information Protocol. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc1058> (дата звернення: 11.06.2026).

19. RFC 7868. Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP). IETF. URL: <https://datatracker.ietf.org/doc/html/rfc7868> (дата звернення: 11.06.2026).

20. RFC 2328. OSPF Version 2. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc2328> (дата звернення: 11.06.2026).

21. RFC 3022. Traditional IP Network Address Translator (Traditional NAT). IETF. URL: <https://datatracker.ietf.org/doc/html/rfc3022> (дата звернення: 11.06.2026).

22. RFC 2281. Cisco Hot Standby Router Protocol (HSRP). IETF. URL: <https://www.rfc-editor.org/info/rfc2281/> (дата звернення: 11.06.2026).

23. RFC 5798. Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc5798> (дата звернення: 11.06.2026).

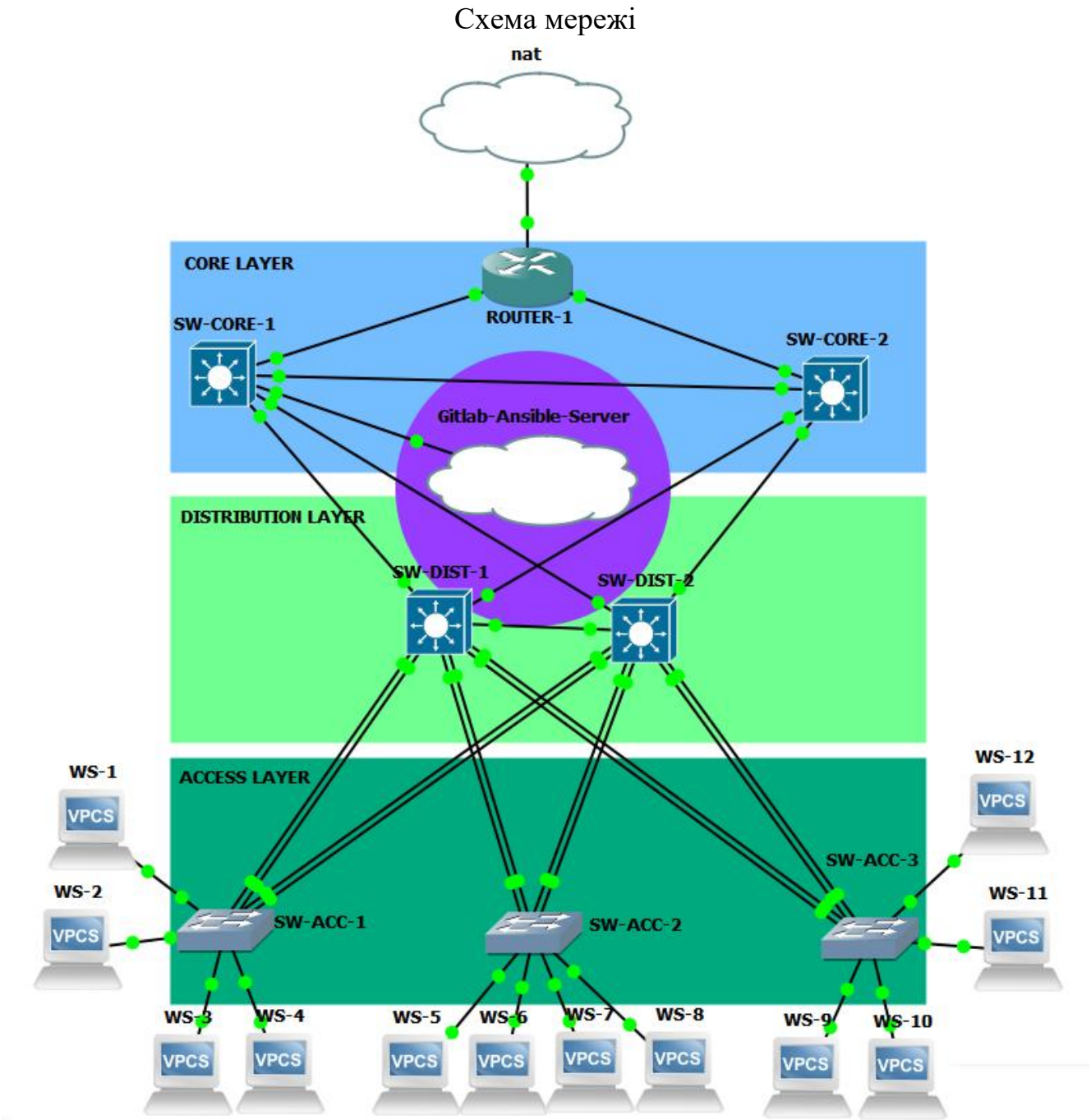
		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

24. IEEE 802.3ad-2000. Aggregation of Multiple Link Segments. URL: <http://www.txgys.com.cn:81/txzj/bzgf/802.3ad-2000-LAG.pdf> (дата звернення: 11.06.2026).
25. RFC 2131. Dynamic Host Configuration Protocol. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc2131> (дата звернення: 11.06.2026).
26. RFC 5905. Network Time Protocol Version 4: Protocol and Algorithms Specification. IETF. URL: <https://www.rfc-editor.org/info/rfc5905/> (дата звернення: 15.06.2026).
27. Puppet Overview. Puppet Documentation. URL: https://www.puppet.com/docs/puppet/6/puppet_overview.html (дата звернення: 15.06.2026).
28. Chef Documentation. URL: <https://docs.chef.io/> (дата звернення: 15.06.2026).
29. Amareswer. Ansible Best Practices and Examples. Medium. URL: <https://medium.com/@amareswer/ansible-best-practices-and-examples-72893d8b19be> (дата звернення: 15.06.2026).
30. GitLab Architecture Overview. GitLab Documentation. URL: <https://docs.gitlab.com/development/architecture/> (дата звернення: 17.06.2026).
31. About GitHub Enterprise Server. GitHub Docs. URL: <https://docs.github.com/en/enterprise-server@3.17/admin/overview/about-github-enterprise-server> (дата звернення: 17.06.2026).
32. CI/CD. GitHub Resources. URL: <https://github.com/resources/articles/ci-cd> (дата звернення: 17.06.2026).
33. Merge Requests. GitLab Documentation. URL: https://docs.gitlab.com/user/project/merge_requests/ (дата звернення: 17.06.2026).
34. Introduction to Ansible Vault. Red Hat Blog. URL: <https://www.redhat.com/en/blog/introduction-ansible-vault> (дата звернення: 17.06.2026).

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		



Порівняльна таблиця утиліт для розгортання мережі

Критерій / Функціонал	Cisco Packet Tracer	GNS3	EVE-NG
Що МОЖЕ програма	Візуалізувати пакети: Покроково показувати рух трафіку (Simulation Mode). Працювати на слабких ПК: Споживає мінімум RAM та процесора. IoT та Розумний дім: Має вбудовані плати (на кшталт Arduino/Raspberry Pi) та датчики. Перевіряти завдання: Автоматично оцінювати лабораторні роботи (.pka).	Емулювати реальні ОС: Запускати справжні прошивки (Cisco IOS, Junos тощо). Мультивендорність: Поєднувати в одну мережу Cisco, Juniper, MikroTik, Arista. Аналізувати трафік: Інтегруватися з Wireshark в один клік для захоплення пакетів. Docker та VM: Підключати реальні контейнери Docker та повноцінні ОС (Windows/Linux).	Працювати через браузер: Керування топологіями через HTML5 без клієнтського софту. Економити пам'ять: Оптимізувати RAM завдяки технології дедуплікації (UKSM). Командна робота: Дозволяти декільком інженерам працювати в одній лабі (у Pro версії). Масштабуватися: Розгортатися на потужних виділених серверах чи в хмарі.
Чого НЕ МОЖЕ програма	Запускати реальні ОС: Використовує лише обмежений набір заздалегідь запрограмованих команд. Працювати з іншими вендорами: Обмежена лише обладнанням Cisco. Інтегрувати реальний софт: Не можна підключити справжню віртуальну машину чи Docker.	Працювати з коробки: Постається абсолютно порожнім, образи пристроїв треба шукати самому. Працювати без софту: Вимагає встановлення локального додатка-клієнта на ПК. Змінювати лінки на льоту: Деякі старі типи пристроїв вимагають зупинки для зміни кабелів.	Дати все безкоштовно: Багато крутих фішок (hot-plug лінків, робота в команді) є лише в платній Pro версії. Працювати з коробки: Так само потребує самостійного пошуку образів пристроїв (.qcow2). Працювати без сервера: Потребує розгортання на віртуальній машині (VMware/ESXi).
Вимоги до заліза	Дуже низькі (чудово працює на будь-якому офісному ПК).	Середні / Високі (залежить від кількості запущених реальних образів).	Високі (найкраще відчувається на виділеному сервері з багатоверсійним процесором та великим об'ємом RAM).
Вартість	Безкоштовно (потрібна лише реєстрація в NetAcad).	Безкоштовно (повністю Open Source).	Є безкоштовна версія (Community) та платна річна підписка (Professional).

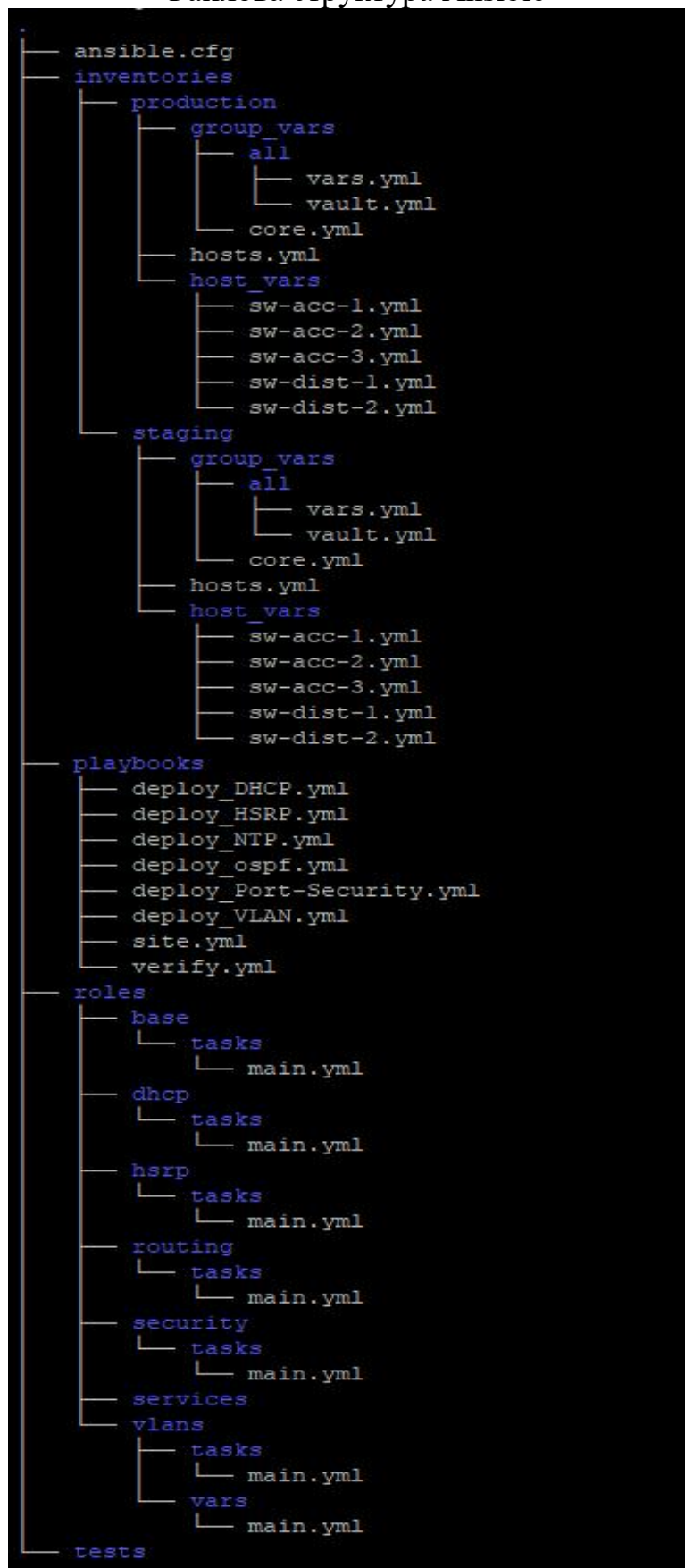
Ansible vs Puppet vs Chef

Критерій порівняння	Ansible	Puppet	Chef
Організація/Власник	Red Hat (IBM)	Perforce	Progress
Архітектура системи	Agentless (без агентів). Керування через SSH / WinRM / API.	Agent-based (з агентами). Клієнт-серверна модель Master-Agent.	Agent-based (з агентами). Модель Workstation - Server - Client.
Модель передачі змін	Push: Керуючий вузол сам ініціює та відправляє конфігурацію.	Pull: Агенти на хостах самі регулярно опитують сервер (кожні 30 хв).	Pull: Клієнти на хостах самі регулярно забирають оновлення з сервера.
Філософія коду	Процедурно-декларативна: Покрокове виконання декларативних модулів.	Суворо декларативна: Опис виключно кінцевого стану ресурсів.	Імперативна: Опис конкретних кроків та алгоритмів для досягнення стану.
Мова написання коду	YAML (простий формат розмітки даних).	Puppet DSL (власна декларативна мова на базі Ruby).	Ruby DSL (повноцінна мова програмування Ruby).
Базовий елемент (одиниця)	Скрипт / Плейбук (Playbook)	Маніфест (Manifest)	Рецепт / Кулінарна книга (Recipe / Cookbook)
Поріг входження	Низький. Легко читається, не потребує навичок програмування.	Середній / Високий. Вимагає вивчення специфічного синтаксису DSL.	Високий. Потребує глибокого розуміння об'єктно-орієнтованого Ruby.
Захист від "дрейфу конфігурації"	Слабкий (потребує регулярного запуску плейбуків вручну або через cron/CI).	Максимальний. Автоматично перезаписує ручні зміни на хостах кожні пів години.	Високий. Автоматично виправляє несанкційні зміни під час запуску клієнта.
Екосистема тестування	Базовий перевірочний запуск (--check), Ansible-lint, Molecule.	Beaker, rspec-puppet, Puppet Linters.	Найпотужніша: Test Kitchen, ChefSpec, InSpec (комплєнс-тести).
Публічний репозиторій	Ansible Galaxy	Puppet Forge	Chef Supermarket

Ansible vs Puppet vs Chef

Критерій NaC	Ansible	Puppet	Chef
Простота інтеграції з Cisco	Найвища. Ідеально підходить для класичних IOS/IOS-XE, GNS3-лабораторій та DevNet Sandbox.	Середня. Потребує налаштування проксі-агентів або підтримки пристроєм Linux Guest Shell (NX-OS).	Середня. Вимагає наявності Ruby-інтерпретатора або проксі-сервера Chef Infra.
Робота з конфігурацією	Текст CLI (ios_config) або сучасні ресурсні модулі (state: merged/overridden).	Абстрактні типи ресурсів (бібліотека netdev_stdlib) + провайдери вендорів.	Нативні кукбуки від Cisco, інтегровані через API пристрою.
Динамічна логіка (IPAM, API)	Можлива через Jinja2-шаблони та інтеграцію з плагінами (напр., NetBox).	Реалізується через ієрархічне сховище даних Hiera.	Максимально гнучка. Код Ruby може "на льоту" робити складні API-запити під час генерації конфігу порту.
Популярність у NetDevOps	Абсолютний лідер завдяки простоті, швидкості ad-hoc команд та відсутності агентів.	Помірна (переважно у великих дата-центрах, де вся інфраструктура вже на Puppet).	Низька (застосовується рідко, переважно великими хмарними провайдерами).

Файлова структура Ansible



		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняльна характеристика GitLab CE та GitHub Enterprise Server для мережевої автоматизації

Критерій порівняння	GitLab Community Edition (CE)	GitHub Enterprise Server (GHES)
Модель ліцензування	Повністю безкоштовна (Open Source) під ліцензією MIT. Ідеально для некомерційного використання, локальних стендів та дипломних робіт.	Суворо комерційна. Вимагає платних ліцензій за кожного користувача (User-based ліцензування). Обмежений тестовий період.
Локальне розгортання (On-Premise / Self-hosted)	Повна автономність. Пакет Omnibus легко розгортається на одній віртуальній машині Ubuntu у закритому контурі (без доступу до Інтернету).	Потребує розгортання у вигляді важкого Virtual Appliance (готові образи OVA для VMware/Hyper-V) з підвищеними вимогами до RAM та CPU.
Архітектура CI/CD конфігурації	Один декларативний файл .gitlab-ci.yml. Логіка стадій (stages) є жорсткою та лінійною.	Модульна система (.github/workflows/*). Гнучка логіка, побудована на подіях (Event-driven) та залежностях між завданнями (needs).
Повторне використання коду	Реалізується через YAML-анкори (як у розробленому стенді для Ansible Vault) або інструкцію include.	Базується на запозиченні готових кроків з публічного або локального Marketplace.
Рівень інтеграції з Ansible	Високий. Shell-runner працює безпосередньо з системним оточенням хоста, де встановлено Ansible.	Високий, але для оптимальної роботи Actions вимагає пакування запусків Ansible у Docker-контейнери.
Автономність в ізольованій мережі (Air-Gapped)	Повна. Платформа не потребує зовнішніх ресурсів під час виконання плейбуків, якщо всі залежності встановлені локально.	Обмежена. Більшість кроків з Marketplace намагаються завантажувати кодові залежності з репозиторіїв GitHub.com під час запуску Pipeline.

Базове налагодження пристрою Router-1

R1

```

ROUTER-1(config)#no ip domain lookup
ROUTER-1(config)#
ROUTER-1(config)#
ROUTER-1(config)#interface FastEthernet0/0
ROUTER-1(config-if)# ip address dhcp
ROUTER-1(config-if)# ip nat outside
ROUTER-1(config-if)# ip virtual-reassembly
ROUTER-1(config-if)# duplex half
ROUTER-1(config-if)#
ROUTER-1(config-if)#interface FastEthernet1/0
ROUTER-1(config-if)# ip address 10.0.0.1 255.255.255.252
ROUTER-1(config-if)# ip nat inside
ROUTER-1(config-if)# ip virtual-reassembly
ROUTER-1(config-if)# duplex half
ROUTER-1(config-if)#
ROUTER-1(config-if)#interface FastEthernet2/0
ROUTER-1(config-if)# ip address 10.0.0.5 255.255.255.252
ROUTER-1(config-if)# ip nat inside
ROUTER-1(config-if)# ip virtual-reassembly
ROUTER-1(config-if)# duplex half
ROUTER-1(config-if)#
ROUTER-1(config-if)#$list NAT-USERS interface FastEthernet0/0 overload
ROUTER-1(config)#ip route 10.0.10.0 255.255.255.0 10.0.0.2
ROUTER-1(config)#
ROUTER-1(config)#ip access-list standard NAT-USERS
ROUTER-1(config-std-nacl)# permit 10.0.0.0 0.255.255.255
ROUTER-1(config-std-nacl)#
ROUTER-1(config-std-nacl)#ip domain-name kulchytskyi.net
ROUTER-1(config)#username ansible privilege 15 secret cisc0123
ROUTER-1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: ROUTER-1.kulchytskyi.net

% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be non-exportable...
*Jun 23 17:04:23.819: %LINEPROTO-5-UPDOWN: Line protocol on Interface NV10, chan
ged state to up[OK]

ROUTER-1(config)#ip ssh version 2
ROUTER-1(config)#
ROUTER-1(config)#line vty 0 4
ROUTER-1(config-line)# login local
ROUTER-1(config-line)# transport input ssh

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

Базове налагодження пристрою SW-CORE-1

```

SW-CORE-1(config-if)#
SW-CORE-1(config-if)#interface Ethernet1/0
SW-CORE-1(config-if)# switchport trunk encapsulation dot1q
SW-CORE-1(config-if)# switchport mode trunk
SW-CORE-1(config-if)# switchport trunk allowed vlan 1,10
SW-CORE-1(config-if)# no shutdown
SW-CORE-1(config-if)# ex
SW-CORE-1(config)#
SW-CORE-1(config)#interface Ethernet0/1
SW-CORE-1(config-if)# switchport trunk encapsulation dot1q
SW-CORE-1(config-if)# switchport mode trunk
SW-CORE-1(config-if)# switchport trunk allowed vlan 1,10
SW-CORE-1(config-if)# no shutdown
SW-CORE-1(config-if)# ex
SW-CORE-1(config)#
SW-CORE-1(config)#interface Ethernet0/2
SW-CORE-1(config-if)# switchport trunk encapsulation dot1q
SW-CORE-1(config-if)# switchport mode trunk
SW-CORE-1(config-if)# switchport trunk allowed vlan 1,10
SW-CORE-1(config-if)# no shutdown
SW-CORE-1(config-if)# ex
SW-CORE-1(config)#
SW-CORE-1(config)#int range e2/0-3, e3/0-3
SW-CORE-1(config-if-range)# sh
SW-CORE-1(config-if-range)#
SW-CORE-1(config-if-range)#spanning-tree vlan 1,10 root primary
SW-CORE-1(config)#spanning-tree vlan 1,10 priority 24576
SW-CORE-1(config)#
SW-CORE-1(config)#ip route 0.0.0.0 0.0.0.0 10.0.0.1
SW-CORE-1(config)#
SW-CORE-1(config)#ip domain-name kulchytskyi.net
SW-CORE-1(config)#username ansible privilege 15 secret cisco123
SW-CORE-1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: SW-CORE-1.kulchytskyi.net

% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 2 seconds)

SW-CORE-1(config)#ip ssh version 2
SW-CORE-1(config)#
SW-CORE-1(config)#line vty 0 4
SW-CORE-1(config-line)# login local
SW-CORE-1(config-line)# transport input ssh

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

Базове налагодження пристрою SW-DIST-1

```

SW-DIST-1(config)#vlan 10
SW-DIST-1(config-vlan)# name MGMT
SW-DIST-1(config-vlan)# ex
SW-DIST-1(config)#
SW-DIST-1(config)#interface vlan 10
SW-DIST-1(config-if)# ip address 10.0.10.11 255.255.255.0
SW-DIST-1(config-if)# no shutdown
SW-DIST-1(config-if)# ex
SW-DIST-1(config)#
SW-DIST-1(config)#interface Ethernet0/0
SW-DIST-1(config-if)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if)# switchport mode trunk
SW-DIST-1(config-if)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if)# no shutdown
SW-DIST-1(config-if)# ex
SW-DIST-1(config)#
SW-DIST-1(config)#interface Ethernet0/1
SW-DIST-1(config-if)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if)# switchport mode trunk
SW-DIST-1(config-if)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if)# no shutdown
SW-DIST-1(config-if)# ex
SW-DIST-1(config)#
SW-DIST-1(config)#interface Ethernet0/2
SW-DIST-1(config-if)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if)# switchport mode trunk
SW-DIST-1(config-if)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if)# no shutdown
SW-DIST-1(config-if)# ex

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

SW-DIST-1(config)#int range e1/0-1
SW-DIST-1(config-if-range)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if-range)# switchport mode trunk
SW-DIST-1(config-if-range)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if-range)# channel-group 1 mode active
Creating a port-channel interface Port-channel 1

SW-DIST-1(config-if-range)# exit
SW-DIST-1(config)#
SW-DIST-1(config)#int range e2/0-1
SW-DIST-1(config-if-range)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if-range)# switchport mode trunk
SW-DIST-1(config-if-range)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if-range)# channel-group 2 mode active
Creating a port-channel interface Port-channel 2

SW-DIST-1(config-if-range)# exit
SW-DIST-1(config)#
SW-DIST-1(config)#int range e3/0-1
SW-DIST-1(config-if-range)# switchport trunk encapsulation dot1q
SW-DIST-1(config-if-range)# switchport mode trunk
SW-DIST-1(config-if-range)# switchport trunk allowed vlan 1,10
SW-DIST-1(config-if-range)# channel-group 3 mode active
Creating a port-channel interface Port-channel 3

SW-DIST-1(config-if-range)# exit
SW-DIST-1(config)#
SW-DIST-1(config)#interface Port-channel1
SW-DIST-1(config-if)# spanning-tree port-priority 64
SW-DIST-1(config-if)#interface Port-channel2
SW-DIST-1(config-if)# spanning-tree port-priority 128
SW-DIST-1(config-if)#interface Port-channel3
SW-DIST-1(config-if)# spanning-tree port-priority 128
SW-DIST-1(config-if)# exit
SW-DIST-1(config)#
SW-DIST-1(config)#
SW-DIST-1(config)#int range e2/2-3, e3/2-3
SW-DIST-1(config-if-range)# sh
SW-DIST-1(config-if-range)# ex
SW-DIST-1(config)#
SW-DIST-1(config)#spanning-tree vlan 1,10 priority 28672
SW-DIST-1(config)#
SW-DIST-1(config)#ip default-gateway 10.0.10.1
SW-DIST-1(config)#
SW-DIST-1(config)#ip domain-name kulchytskyi.net
SW-DIST-1(config)#username ansible privilege 15 secret cisco123
SW-DIST-1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: SW-DIST-1.kulchytskyi.net

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

Базове налагодження пристрою SW-ACC-1

```

SW-ACC-1(config)#vlan 10
SW-ACC-1(config-vlan)# name MGMT
SW-ACC-1(config-vlan)# ex
SW-ACC-1(config)#
SW-ACC-1(config)#interface vlan 10
SW-ACC-1(config-if)# ip address 10.0.10.21 255.255.255.0
SW-ACC-1(config-if)# no shutdown
SW-ACC-1(config-if)# ex
SW-ACC-1(config)#
SW-ACC-1(config)#int range e0/0-1
SW-ACC-1(config-if-range)# switchport trunk encapsulation dot1q
SW-ACC-1(config-if-range)# switchport mode trunk
SW-ACC-1(config-if-range)# switchport trunk allowed vlan 1,10
SW-ACC-1(config-if-range)# channel-group 1 mode active
Creating a port-channel interface Port-channel 1

SW-ACC-1(config-if-range)# no shutdown
SW-ACC-1(config-if-range)# exit
SW-ACC-1(config)#
SW-ACC-1(config)#int range e1/0-1
SW-ACC-1(config-if-range)# switchport trunk encapsulation dot1q
SW-ACC-1(config-if-range)# switchport mode trunk
SW-ACC-1(config-if-range)# switchport trunk allowed vlan 1,10
SW-ACC-1(config-if-range)# channel-group 11 mode active
Creating a port-channel interface Port-channel 11

SW-ACC-1(config-if-range)# no shutdown
SW-ACC-1(config-if-range)# exit
SW-ACC-1(config)#
SW-ACC-1(config)#int range e1/2-3
SW-ACC-1(config-if-range)# sh
SW-ACC-1(config-if-range)# ex
SW-ACC-1(config)#
SW-ACC-1(config)#int range e0/2-3
SW-ACC-1(config-if-range)# sh
SW-ACC-1(config-if-range)# ex
SW-ACC-1(config)#
SW-ACC-1(config)#int range e2/0-3
SW-ACC-1(config-if-range)# sh
SW-ACC-1(config-if-range)# ex
SW-ACC-1(config)#int range e3/0-3
SW-ACC-1(config-if-range)# sh
SW-ACC-1(config-if-range)# ex
SW-ACC-1(config)#
SW-ACC-1(config)#ip default-gateway 10.0.10.1
SW-ACC-1(config)#
SW-ACC-1(config)#ip domain-name kulchytskyi.net
SW-ACC-1(config)#username ansible privilege 15 secret cisco123
SW-ACC-1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: SW-ACC-1.kulchytskyi.net

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Клонування репозиторію на локальну машину

```

student@gitlab-ansible-server:~/network-automation$ git clone http://10.0.10.10/dyplomna/network-automation.git
Cloning into 'network-automation'...
Username for 'http://10.0.10.10': root
Password for 'http://root@10.0.10.10':
warning: You appear to have cloned an empty repository.
student@gitlab-ansible-server:~/network-automation$ tree
.
├── ansible.cfg
├── inventories
│   ├── production
│   │   ├── group_vars
│   │   │   └── all
│   │   │       ├── vars.yml
│   │   │       └── vault.yml
│   │   └── hosts.yml
│   ├── staging
│   │   └── group_vars
│   └── network-automation
├── playbooks
│   ├── site.yml
│   └── verify.yml
├── roles
│   ├── base
│   │   └── tasks
│   │       └── main.yml
│   ├── dhcp
│   │   └── tasks
│   │       └── main.yml
│   ├── hsrp
│   │   └── tasks
│   │       └── main.yml
│   ├── routing
│   │   └── tasks
│   │       └── main.yml
│   ├── security
│   │   └── tasks
│   │       └── main.yml
│   ├── services
│   └── vlans
│       ├── tasks
│       │   ├── main.yml
│       │   └── vars
│       │       └── main.yml
└── tests

25 directories, 13 files
student@gitlab-ansible-server:~/network-automation$ █

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст ролі DHCP

```

--
- name: Виключення IP-адрес шлюзів та свічів із роздачі DHCP
  cisco.ios.ios_config:
    lines:
      - ip dhcp excluded-address 10.0.20.1 10.0.20.10
      - ip dhcp excluded-address 10.0.30.1 10.0.30.10
      - ip dhcp excluded-address 10.0.40.1 10.0.40.10
    when: inventory_hostname == 'router-1'

- name: Створення DHCP-пулів для клієнтських мереж
  cisco.ios.ios_config:
    parents: ip dhcp pool {{ item.name }}
    lines:
      - network 10.0.{{ item.subnet }}.0 255.255.255.0
      - default-router 10.0.{{ item.subnet }}.1
      - dns-server 8.8.8.8
    loop:
      - { name: 'SERVERS', subnet: '20' }
      - { name: 'USERS', subnet: '30' }
      - { name: 'VOIP', subnet: '40' }
    when: inventory_hostname == 'router-1'

- name: Налаштування DHCP Relay (ip helper-address) на Distribution
  cisco.ios.ios_config:
    parents: interface Vlan{{ item }}
    lines:
      - ip helper-address 10.0.0.1
    loop:
      - 20
      - 30
      - 40
    when: "'distribution' in group_names"

- name: Гарантуємо, що клієнтські VLAN інтерфейси увімкнені
  cisco.ios.ios_interfaces:
    config:
      - name: "Vlan{{ item }}"
        enabled: true
    state: merged
    loop:
      - 20
      - 30
      - 40
    when: "'distribution' in group_names"

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст полі security

```

GNU nano 8.7.1
- name: Адміністративне увімкнення клієнтських портів
  cisco.ios.ios_interfaces:
    config:
      - name: "{{ item }}"
        enabled: true
    state: merged
  loop:
    - Ethernet2/0
    - Ethernet2/1
    - Ethernet2/2
    - Ethernet2/3
  when: "'access' in group_names"

- name: Налаштування Access-портів (Динамічний VLAN)
  cisco.ios.ios_config:
    parents: interface {{ item.interface }}
    lines:
      - switchport mode access
      - switchport access vlan {{ item.vlan }}
      - description Dynamic Access Port VLAN {{ item.vlan }}
  loop: "{{ access_ports }}"
  when: "'access' in group_names"

- name: Налаштування Port-Security для USERS
  cisco.ios.ios_config:
    parents: interface {{ item }}
    lines:
      - switchport mode access
      - switchport port-security
      - switchport port-security maximum 2
      - switchport port-security mac-address sticky
      - switchport port-security violation restrict
  loop:
    - Ethernet2/0
    - Ethernet2/1
    - Ethernet2/2
    - Ethernet2/3
  when: "'access' in group_names"

# Створення розширених та стандартних ACL на Core-комутаторах
- name: Налаштування списків доступу (ACL) безпеки
  cisco.ios.ios_acls:
    config:
      - afi: ipv4
    acls:
      - name: PROTECT-SERVERS
        acl_type: extended
        aces:
          - grant: permit
            protocol: tcp
            source:
              address: 10.0.30.0
              wildcard: 0.0.0.255
            destination:
              address: 10.0.20.0
              wildcard: 0.0.0.255
            layer4_operator:
              operator: eq
              port: 80

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

- grant: permit
  protocol: tcp
  source:
    address: 10.0.30.0
    wildcard: 0.0.0.255
  destination:
    address: 10.0.20.0
    wildcard: 0.0.0.255
  layer4_operator:
    operator: eq
    port: 443
- grant: deny
  protocol: ip
  source:
    any: true
  destination:
    address: 10.0.20.0
    wildcard: 0.0.0.255
- name: RESTRICT-SSH
  acl_type: standard
  aces:
    - grant: permit
      source:
        address: 10.0.10.10
        wildcard: 0.0.0.0
    - grant: deny
      source:
        any: true
  state: merged
when: "'core' in group_names"

# Навешування ACL PROTECT-SERVERS на інтерфейс VLAN 20 (вхідний трафік до серверів)
- name: Застосування ACL PROTECT-SERVERS до інтерфейсу серверей
  cisco.ios.ios_config:
    lines:
      - ip access-group PROTECT-SERVERS out
    parents: interface Vlan20
  when: "'core' in group_names"

# Обмеження доступу до SSH (VTY ліній) на всіх пристроях мережі
- name: Застосування ACL RESTRICT-SSH на лінії VTY для захисту SSH
  cisco.ios.ios_config:
    lines:
      - access-class RESTRICT-SSH in
    parents: line vty 0 4

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст елемента інвентарю group_vars/all/vars.yml

```

GNU nano 8.7.1
--
ansible_user: ansible
ansible_network_os: cisco.ios.ios
ansible_connection: network_cli

ansible_become: true
ansible_become_method: enable
ansible_password: "{{ vault_ansible_password }}"
ansible_become_password: "{{ vault_ansible_become_password }}"

snmp_auth_pass: "{{ vault_snmp_auth_pass }}"
snmp_priv_pass: "{{ vault_snmp_priv_pass }}"

access_ports:
- { interface: "Ethernet2/0", vlan: 20 }
- { interface: "Ethernet2/1", vlan: 30 }
- { interface: "Ethernet2/2", vlan: 40 }
- { interface: "Ethernet2/3", vlan: 30 }

# ACL rules
corporate_acls:
- name: PROTECT-SERVERS
  type: extended
  rules:
    - action: permit
      protocol: tcp
      source: 10.0.30.0 0.0.0.255      # VLAN 30 USERS
      destination: 10.0.20.0 0.0.0.255 # VLAN 20 SERVERS
      port: eq 80
    - action: permit
      protocol: tcp
      source: 10.0.30.0 0.0.0.255
      destination: 10.0.20.0 0.0.0.255
      port: eq 443
    - action: deny
      protocol: ip
      source: any
      destination: 10.0.20.0 0.0.0.255

- name: RESTRICT-SSH
  type: standard
  rules:
    - action: permit
      source: 10.0.10.10 0.0.0.0      # Тільки наш GitLab-Ansible-Server
    - action: deny
      source: any

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст плейбуку site.yml

```

GNU nano 8.7.1
--
- name: Перевірка зв'язку з пристроями Cisco
  hosts: all
  gather_facts: false

  tasks:
    - name: Збір базової інформації (IOS Facts)
      cisco.ios.ios_facts:
        gather_subset: min
        register: device_facts

- name: Базові налаштування інфраструктури (NTP)
  hosts: all
  gather_facts: false
  roles:
    - base

- name: Налаштування бази VLAN на комутаторах
  hosts: core, distribution, access
  gather_facts: false
  roles:
    - vlans

- name: Налаштування динамічної маршрутизації (OSPF)
  hosts: routers, core, distribution
  gather_facts: false
  roles:
    - routing

- name: Налаштування HSRP-резервування
  hosts: distribution
  gather_facts: false
  roles:
    - hsrp

- name: Налаштування DHCP-сервера та Relay
  hosts: routers, distribution
  gather_facts: false
  roles:
    - dhcp

- name: Налаштування безпеки на рівні доступу (Port-Security)
  hosts: access
  gather_facts: false
  roles:
    - security

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

Виконання задачі по VLAN

```

PLAY [Налаштування бази VLAN на комутаторах] *****

TASK [vlans : Створення бази VLAN] *****
ok: [sw-core-1] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-core-1] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-core-1] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-core-1] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-core-1] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-core-2] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-core-2] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-core-2] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-core-2] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-core-2] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-dist-1] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-dist-1] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-dist-1] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-dist-1] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-dist-1] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-dist-2] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-dist-2] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-dist-2] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-dist-2] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-dist-2] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-acc-1] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-acc-1] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-acc-1] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-acc-1] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-acc-1] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-acc-2] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-acc-2] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-acc-2] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-acc-2] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-acc-2] => (item={'vlan_id': 99, 'name': 'NATIVE'})
ok: [sw-acc-3] => (item={'vlan_id': 10, 'name': 'MGMT'})
ok: [sw-acc-3] => (item={'vlan_id': 20, 'name': 'SERVERS'})
ok: [sw-acc-3] => (item={'vlan_id': 30, 'name': 'USERS'})
ok: [sw-acc-3] => (item={'vlan_id': 40, 'name': 'VOIP'})
ok: [sw-acc-3] => (item={'vlan_id': 99, 'name': 'NATIVE'})

TASK [vlans : Дозвіл клієнтських VLAN на Trunk-інтерфейсах] *
changed: [sw-core-1] => (item=Ethernet0/1)
changed: [sw-core-1] => (item=Ethernet0/2)
changed: [sw-core-1] => (item=Ethernet1/0)
changed: [sw-core-2] => (item=Ethernet0/1)
changed: [sw-core-2] => (item=Ethernet0/2)
changed: [sw-core-2] => (item=Ethernet1/0)
changed: [sw-dist-1] => (item=Ethernet0/0)
changed: [sw-dist-1] => (item=Ethernet0/1)
changed: [sw-dist-1] => (item=Ethernet0/2)
changed: [sw-dist-1] => (item=Port-channel1)
changed: [sw-dist-1] => (item=Port-channel2)
changed: [sw-dist-1] => (item=Port-channel3)
changed: [sw-dist-2] => (item=Ethernet0/0)
changed: [sw-dist-2] => (item=Ethernet0/1)
changed: [sw-dist-2] => (item=Ethernet0/2)
changed: [sw-dist-2] => (item=Port-channel11)
changed: [sw-dist-2] => (item=Port-channel22)
changed: [sw-dist-2] => (item=Port-channel33)
changed: [sw-acc-1] => (item=Port-channel1)
changed: [sw-acc-1] => (item=Port-channel11)
changed: [sw-acc-2] => (item=Port-channel2)
changed: [sw-acc-2] => (item=Port-channel22)

```

		Кульчицький Т.В.		
		Миколайчук В.В.		
Змн.	Арк.	№ докум.	Підпис	Дата

Виконання задачі по DHCP

```

PLAY [Налаштування DHCP-сервера та Relay] *****

TASK [dhcp : Виключення IP-адрес шлюзів та свічів із роздачі DHCP] *****
changed: [router-1]
skipping: [sw-dist-1]
skipping: [sw-dist-2]

TASK [dhcp : Створення DHCP-пулів для клієнтських мереж] *****
changed: [router-1] => (item={'name': 'SERVERS', 'subnet': '20'})
changed: [router-1] => (item={'name': 'USERS', 'subnet': '30'})
changed: [router-1] => (item={'name': 'VOIP', 'subnet': '40'})
skipping: [sw-dist-1] => (item={'name': 'SERVERS', 'subnet': '20'})
skipping: [sw-dist-1] => (item={'name': 'USERS', 'subnet': '30'})
skipping: [sw-dist-1] => (item={'name': 'VOIP', 'subnet': '40'})
skipping: [sw-dist-1]
skipping: [sw-dist-2] => (item={'name': 'SERVERS', 'subnet': '20'})
skipping: [sw-dist-2] => (item={'name': 'USERS', 'subnet': '30'})
skipping: [sw-dist-2] => (item={'name': 'VOIP', 'subnet': '40'})
skipping: [sw-dist-2]

TASK [dhcp : Налаштування DHCP Relay (ip helper-address) на Distribution] *
skipping: [router-1] => (item=20)
skipping: [router-1] => (item=30)
skipping: [router-1] => (item=40)
skipping: [router-1]
changed: [sw-dist-1] => (item=20)
changed: [sw-dist-1] => (item=30)
changed: [sw-dist-1] => (item=40)
changed: [sw-dist-2] => (item=20)
changed: [sw-dist-2] => (item=30)
changed: [sw-dist-2] => (item=40)

TASK [dhcp : Гарантуємо, що клієнтські VLAN інтерфейси увімкнені] *****
skipping: [router-1] => (item=20)
skipping: [router-1] => (item=30)
skipping: [router-1] => (item=40)
skipping: [router-1]
changed: [sw-dist-1] => (item=20)
changed: [sw-dist-1] => (item=30)
changed: [sw-dist-1] => (item=40)
changed: [sw-dist-2] => (item=20)
changed: [sw-dist-2] => (item=30)
changed: [sw-dist-2] => (item=40)

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

Виконання tasky по security

```

PLAY [Налаштування безпеки на рівні доступу (Port-Security)] *****

TASK [security : Адміністративне увімкнення клієнтських портів] *****
changed: [sw-acc-1] => (item=Ethernet2/0)
changed: [sw-acc-1] => (item=Ethernet2/1)
changed: [sw-acc-1] => (item=Ethernet2/2)
changed: [sw-acc-1] => (item=Ethernet2/3)
changed: [sw-acc-2] => (item=Ethernet2/0)
changed: [sw-acc-2] => (item=Ethernet2/1)
changed: [sw-acc-2] => (item=Ethernet2/2)
changed: [sw-acc-2] => (item=Ethernet2/3)
changed: [sw-acc-3] => (item=Ethernet2/0)
changed: [sw-acc-3] => (item=Ethernet2/1)
changed: [sw-acc-3] => (item=Ethernet2/2)
changed: [sw-acc-3] => (item=Ethernet2/3)

TASK [security : Налаштування Access-портів (Динамічний VLAN)] *****
changed: [sw-acc-1] => (item={'interface': 'Ethernet2/0', 'vlan': 20})
changed: [sw-acc-1] => (item={'interface': 'Ethernet2/1', 'vlan': 30})
changed: [sw-acc-1] => (item={'interface': 'Ethernet2/2', 'vlan': 40})
changed: [sw-acc-1] => (item={'interface': 'Ethernet2/3', 'vlan': 30})
changed: [sw-acc-2] => (item={'interface': 'Ethernet2/0', 'vlan': 20})
changed: [sw-acc-2] => (item={'interface': 'Ethernet2/1', 'vlan': 30})
changed: [sw-acc-2] => (item={'interface': 'Ethernet2/2', 'vlan': 40})
changed: [sw-acc-2] => (item={'interface': 'Ethernet2/3', 'vlan': 30})
changed: [sw-acc-3] => (item={'interface': 'Ethernet2/0', 'vlan': 20})
changed: [sw-acc-3] => (item={'interface': 'Ethernet2/1', 'vlan': 30})
changed: [sw-acc-3] => (item={'interface': 'Ethernet2/2', 'vlan': 40})
changed: [sw-acc-3] => (item={'interface': 'Ethernet2/3', 'vlan': 30})

TASK [security : Налаштування Port-Security для USERS] *****
changed: [sw-acc-1] => (item=Ethernet2/0)
changed: [sw-acc-1] => (item=Ethernet2/1)
changed: [sw-acc-1] => (item=Ethernet2/2)
changed: [sw-acc-1] => (item=Ethernet2/3)
changed: [sw-acc-2] => (item=Ethernet2/0)
changed: [sw-acc-2] => (item=Ethernet2/1)
changed: [sw-acc-2] => (item=Ethernet2/2)
changed: [sw-acc-2] => (item=Ethernet2/3)
changed: [sw-acc-3] => (item=Ethernet2/0)
changed: [sw-acc-3] => (item=Ethernet2/1)
changed: [sw-acc-3] => (item=Ethernet2/2)
changed: [sw-acc-3] => (item=Ethernet2/3)

TASK [security : Налаштування списків доступу (ACL) безпеки] *****
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

TASK [security : Застосування ACL PROTECT-SERVERS до інтерфейсу серверей] ***
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

TASK [security : Застосування ACL RESTRICT-SSH на лінії VTY для захисту SSH]
changed: [sw-acc-1]
changed: [sw-acc-2]
changed: [sw-acc-3]

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст файлу .gitlab-ci.yml

```

GNU nano 8.7.1 .git
stages:
  - lint
  - dry-run
  - staging
  - production

variables:
  ANSIBLE_HOST_KEY_CHECKING: "false"

# Шаблон для безпечного керування паролями через Ansible Vault
.setup_vault:
  before_script:
    - echo "$ANSIBLE_VAULT_PASSWORD" > .vault_pass.txt
  after_script:
    - rm -f .vault_pass.txt

# --- ЕТАП 1: LINT (Перевірка синтаксису YAML коду) ---
lint:
  stage: lint
  tags:
    - network-automation
  script:
    - ansible-playbook playbooks/site.yml --syntax-check
    - ansible-playbook playbooks/verify.yml --syntax-check
  rules:
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
    - if: '$CI_COMMIT_BRANCH == "main"'

# --- ЕТАП 2: DRY-RUN (Холостий запуск змін без реального заліза) ---
dry-run:
  stage: dry-run
  tags:
    - network-automation
  extends: .setup_vault
  script:
    - ansible-playbook -i inventories/staging/ playbooks/site.yml --check --diff --vault-password-file .vault_pass.txt
  rules:
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'

# --- ЕТАП 3: DEPLOY STAGING (Автоматичне розгортання + Тестування мережі) ---
deploy-staging:
  stage: staging
  tags:
    - network-automation
  extends: .setup_vault
  script:
    - ansible-playbook -i inventories/staging/ playbooks/site.yml --vault-password-file .vault_pass.txt
    - sleep 30
    - ansible-playbook -i inventories/staging/ playbooks/verify.yml --vault-password-file .vault_pass.txt
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'

# --- ЕТАП 4: DEPLOY PRODUCTION (Бойовий деплой з ручним підтвердженням) ---
deploy-production:
  stage: production
  tags:
    - network-automation
  extends: .setup_vault
  script:
    - ansible-playbook -i inventories/production/ playbooks/site.yml --vault-password-file .vault_pass.txt
    - sleep 30
    - ansible-playbook -i inventories/production/ playbooks/verify.yml --vault-password-file .vault_pass.txt
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
      when: manual

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Перелік перевірок в verify.yml

```
student@gitlab-ansible-server:~/network-automation$ ansible-playbook -i inventories/production/ playbooks/verify.yml --ask-vault-pass
Vault password:

PLAY [Автоматична верифікація стану мережі (Testing)] *****

TASK [Отримання інформації про VLAN] *****
skipping: [router-1]
ok: [sw-core-1]
ok: [sw-core-2]
ok: [sw-dist-1]
ok: [sw-dist-2]
ok: [sw-acc-1]
ok: [sw-acc-2]
ok: [sw-acc-3]

TASK [Отримання інформації про OSPF сусідів] *****
ok: [router-1]
ok: [sw-core-1]
ok: [sw-core-2]
ok: [sw-dist-1]
ok: [sw-dist-2]
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

TASK [Отримання інформації про Port-Security] *****
skipping: [router-1]
skipping: [sw-core-1]
skipping: [sw-core-2]
skipping: [sw-dist-1]
skipping: [sw-dist-2]
ok: [sw-acc-1]
ok: [sw-acc-2]
ok: [sw-acc-3]

TASK [Отримання статусу NTP (Associations)] *****
ok: [router-1]
ok: [sw-core-1]
ok: [sw-core-2]
ok: [sw-dist-1]
ok: [sw-dist-2]
ok: [sw-acc-1]
ok: [sw-acc-2]
ok: [sw-acc-3]

TASK [Перевірка зв'язності (Ping від Router до SW-ACC-1)] *****
ok: [router-1]
skipping: [sw-core-1]
skipping: [sw-core-2]
skipping: [sw-dist-1]
skipping: [sw-dist-2]
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]
```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

TASK [ТЕСТ 1 - Чи існує VLAN 30 (USERS)?] ***
skipping: [router-1]
ok: [sw-core-1] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-core-2] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-dist-1] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-dist-2] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-acc-1] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-acc-2] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.
ok: [sw-acc-3] => {
    "changed": false
}

MSG:

ПАС: VLAN 30 успішно знайдено.

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

TASK [ТЕСТ 2 - Чи є OSPF сусіди в стані FULL?] ***
ok: [router-1] => {
    "changed": false
}

MSG:

ПАС: OSPF сусіди знаходяться в стані FULL.
ok: [sw-core-1] => {
    "changed": false
}

MSG:

ПАС: OSPF сусіди знаходяться в стані FULL.
ok: [sw-core-2] => {
    "changed": false
}

MSG:

ПАС: OSPF сусіди знаходяться в стані FULL.
ok: [sw-dist-1] => {
    "changed": false
}

MSG:

ПАС: OSPF сусіди знаходяться в стані FULL.
ok: [sw-dist-2] => {
    "changed": false
}

MSG:

ПАС: OSPF сусіди знаходяться в стані FULL.
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

TASK [ТЕСТ 3 - Чи увімкнено Port-Security на портах доступу?] *****
skipping: [router-1]
skipping: [sw-core-1]
skipping: [sw-core-2]
skipping: [sw-dist-1]
skipping: [sw-dist-2]
ok: [sw-acc-1] => {
    "changed": false
}

MSG:

ПАС: Port-Security активовано на інтерфейсі.
ok: [sw-acc-2] => {
    "changed": false
}

MSG:

ПАС: Port-Security активовано на інтерфейсі.
ok: [sw-acc-3] => {
    "changed": false
}

MSG:

ПАС: Port-Security активовано на інтерфейсі.

TASK [ТЕСТ 4 - Чи успішно проходять Ping-запити (ICMP)?] *****
ok: [router-1] => {
    "changed": false
}

MSG:

ПАС: Мережева доступність (Ping) підтверджена.
skipping: [sw-core-1]
skipping: [sw-core-2]
skipping: [sw-dist-1]
skipping: [sw-dist-2]
skipping: [sw-acc-1]
skipping: [sw-acc-2]
skipping: [sw-acc-3]

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

TASK [ТЕСТ 5 - Чи бачать пристрої NTP-сервер (Associations)?] *****
ok: [router-1] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-core-1] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-core-2] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-dist-1] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-dist-2] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-acc-1] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-acc-2] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.
ok: [sw-acc-3] => {
    "changed": false
}

MSG:

ПАС: NTP-ієрархію побудовано, сервер доступний.

```

		Кульчицький Т.В.			ККІКБ.КР.Б – 123 – 09 – ПЗ	Арк.
		Миколайчук В.В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		