Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики

Магістерська Робота
Освітній ступінь – магістр

на тему: «**СИСТЕМА КЕРУВАННЯ КОНФІГУРАЦІЯМИ ПРИСТРОЇВ В МЕРЕЖІ ПІДПРИЄМСТВА**»

Виконав : студент 2-го року навчання,
Спеціальності
121 Інженерія програмного забезпечення
Мирошник Дмитро Олександрович
Керівник Черкасов Д. І.
Старший викладач, Кандидат технічних наук

Рецензент_____

Магістерська робота захищена з оцінкою_____

Секретар ЕК_____
«___»_____20___р.

Київ - 2023

ЗМІСТ

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	01.02.2023	
2.	Огляд технічної літератури за темою роботи	15.02.2023	
3.	Огляд сучасних рішень	19.02.2023	
3.	Аналіз рішень, та постановка вимог до розроблюваного застосунку	01.03.2023	
4.	Розробка застосунку	25.04.2023	
5.	Написання пояснювальної записки	31.05.2023	
6.	Написання пояснювальної записки	24.04.2023	

Студент _____

Керівник _____

“____”_____

Керівник курсової роботи

Анотація

Ця робота присвячена розробці системи керування конфігураціями пристроїв в мережі підприємства, а саме резервне копіювання конфігурацій пристроїв, та застосування змін в конфігураціях пристроїв. У роботі було проведено дослідження наявних рішень у сфері роботи пристроїв у мережі, проведено аналіз можливих компонент в розроблюваній системі та впроваджено власне рішення на основі розглянутих застосунків. Результатом роботи є застосунок, який може вивантажувати і зберігати конфігурації пристроїв у сховищі, та в рамках практичної імплементації надані файли конфігурацій роботи в Jenkins та плейбуки Ansible.

ВСТУП

Запит на розробку системи керування конфігураціями пристроїв в мережі підприємства є досить актуальною задачею в сучасних умовах розвитку технологій та інформаційних систем. Одним із ключових елементів IT-інфраструктури підприємства є мережа, що забезпечує швидке та надійне з'єднання між різними пристроями та ресурсами на підприємстві. Для забезпечення стабільної роботи мережі необхідно мати правильно сконфігуруванні пристрої, такі як маршрутизатори, комутатори, та інше, тому проблема керування конфігураціями пристроїв набуває все більшої актуальності.

Система керування конфігураціями пристроїв в мережі - це комплексна система, яка дозволяє керувати конфігураціями мережевих пристроїв на підприємстві. Її основна мета полягає у забезпеченні стабільної та безпечної роботи мережі за допомогою автоматизованого збору, контролю та збереження конфігураційних даних пристроїв, що забезпечують роботу мережі. Така система дозволяє автоматизовано виявляти та відслідковувати зміни конфігурацій пристроїв, забезпечує можливість швидко відновити працездатність мережі в разі виникнення проблем, а також знижує ризик виникнення помилок під час внесення змін у конфігурацію пристроїв.

Система може включати в себе такі елементи, як системи моніторингу, системи управління версіями, системи автоматичного резервного копіювання конфігурацій, системи розгортання конфігурацій та інші. Крім того вона може бути інтегрована з IPAM (IP Address Management), DNS (Domain Name System) та DHCP (Dynamic Host Configuration Protocol). Інтеграція мережевих сервісів із системами керування конфігурацією може допомогти організаціям заощадити час і зменшити помилки під час керування мережевою інфраструктурою.

Основними задачами цієї роботи є огляд існуючих рішень та інструментів, вибір складових компонентів розроблюваної системи та реалізація основного функціоналу збору конфігурацій.

РОЗДІЛ 1.

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

Для системи керування конфігураціями пристроїв в мережі підприємства необхідно визначити необхідні інструменти для забезпечення вимог до системи.

Огляд існуючих рішень необхідний для аналізу та визначення вимог та необхідних компонентів для розроблюваної системи

1.1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Разом із розвитком технологій та підходів функціонування систем конфігураціями пристроїв, збільшуються й запити до таких систем. Компанії можуть збільшуватись, а разом з ними збільшуватись і кількість пристроїв які необхідно адмініструвати, впроваджувати вдосконалення та налаштовувати під запити та необхідності компанії.

Наразі існують рішення які відрізняються наявністю корисного функціоналу, системами сповіщення, підтримкою контролю версій та зручним та інтуїтивно зрозумілим веб-інтерфейсом. Якісь з них можливо масштабувати для постійно зростаючої команди, якісь з них можуть підходити невеликим компаніям.

Unimus

Unimus - це інструмент керування конфігурацією мережевих пристроїв від багатьох виробників таких як Cisco, Juniper, MikroTik та багатьох інших. Він надає мережевим адміністраторам централізовану платформу для керування конфігураціями своїх мережевих пристроїв, включаючи маршрутизатори та комутатори.

Він використовує веб-інтерфейс користувача рис.1 , щоб забезпечити простий та інтуїтивно зрозумілий спосіб керування конфігураціями мережі. Інструмент дозволяє адміністраторам створювати резервні копії конфігурацій пристрою, порівнювати версії конфігурації та повертати конфігурації до попередніх версій. Він також надає контрольні журнали та звіти, щоб допомогти адміністраторам відстежувати зміни конфігурації та відповідати нормативним вимогам.

Unimus надає комплексне рішення для керування конфігурацією мережевих пристроїв, яке спрощує адміністрування мережі та підвищує надійність мережі.

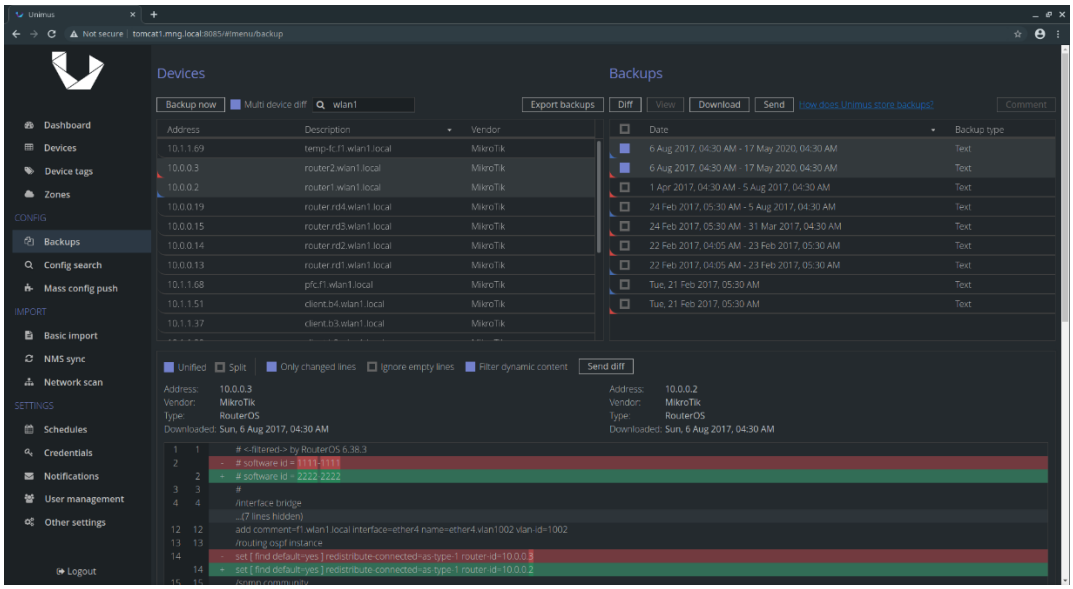

Рис.1 Інтерфейс Unimus

Перевагами цієї системи є підтримка багатьох постачальників, веб-інтерфейс користувача, мультиплатформеність (macOS , Windows, Linux), інтеграція API, контроль доступу на основі ролей, система моніторингу та контролю версій.

Проте мінусами даного інструмент, є його ціна, обмежені можливості автоматизації та обмежена інтеграція з інструментами та платформами поза мережею.

Oxidized

Oxidized - це безкоштовний інструмент керування конфігурацією мережевих пристроїв із відкритим кодом, який дозволяє мережевим адміністраторам автоматизувати резервне копіювання та відстеження конфігурацій пристроїв. Він розроблений для роботи з широким колом постачальників мереж і підтримує такі популярні протоколи, як SSH, Telnet і SNMP.

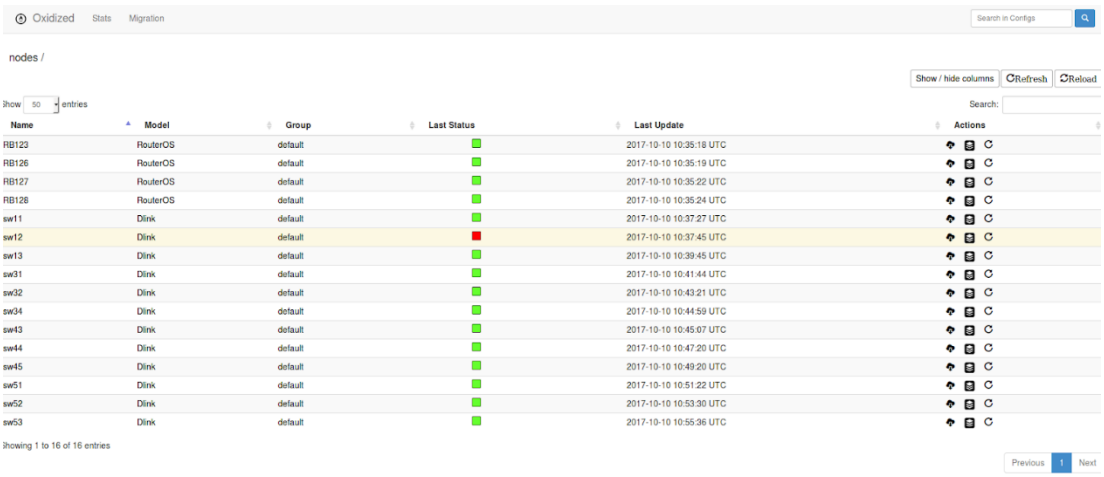

Рис.2 Інтерфейс Oxidized

Oxidized використовує веб-інтерфейс користувача, який забезпечує простий та інтуїтивно зрозумілий спосіб керування конфігураціями мережі. Він також надає можливості автоматизації, такі як автоматичне виявлення змін та інтеграція з інструментами автоматизації, такими як Ansible. Oxidized може інтегруватися з іншими інструментами та службами через свій RESTful API,

що дозволяє бездоганно працювати з іншим програмним забезпеченням у вашому стеку керування мережею.

Хоча Oxidized не надає розширених можливостей візуалізації топології мережі, він дозволяє адміністраторам переглядати основну інформацію про топологію мережі та візуалізувати зміни конфігурації для кожного пристрою. Він також забезпечує підтримку метаданих пристроїв, таких як IP-адреси та назви пристроїв, щоб допомогти адміністраторам відстежувати свої мережеві пристрої.

Oxidized — це потужний і гнучкий інструмент керування конфігурацією мережевих пристроїв, який може допомогти організаціям оптимізувати процеси керування мережею та покращити стан безпеки мережі. Однак він має деякі обмеження, такі як обмежена підтримка немережевих пристроїв.

SolarWinds

SolarWinds Network Configuration Manager (NCM) — це інструмент керування конфігураціями мережевих пристроїв, завдяки якому адміністратори мережі можуть автоматизувати резервне копіювання конфігурацій, їх відстеження та керування конфігураціями пристроїв.

Однією з унікальних особливостей NCM є його можливості керування відповідністю. NCM дозволяє адміністраторам створювати власні політики та правила для мережевих конфігурацій на основі таких галузевих стандартів, як CIS, NIST і PCI DSS. Інструмент також надає звіти та сповіщення, щоб допомогти адміністраторам переконатися, що їхні мережеві конфігурації відповідають цим стандартам.

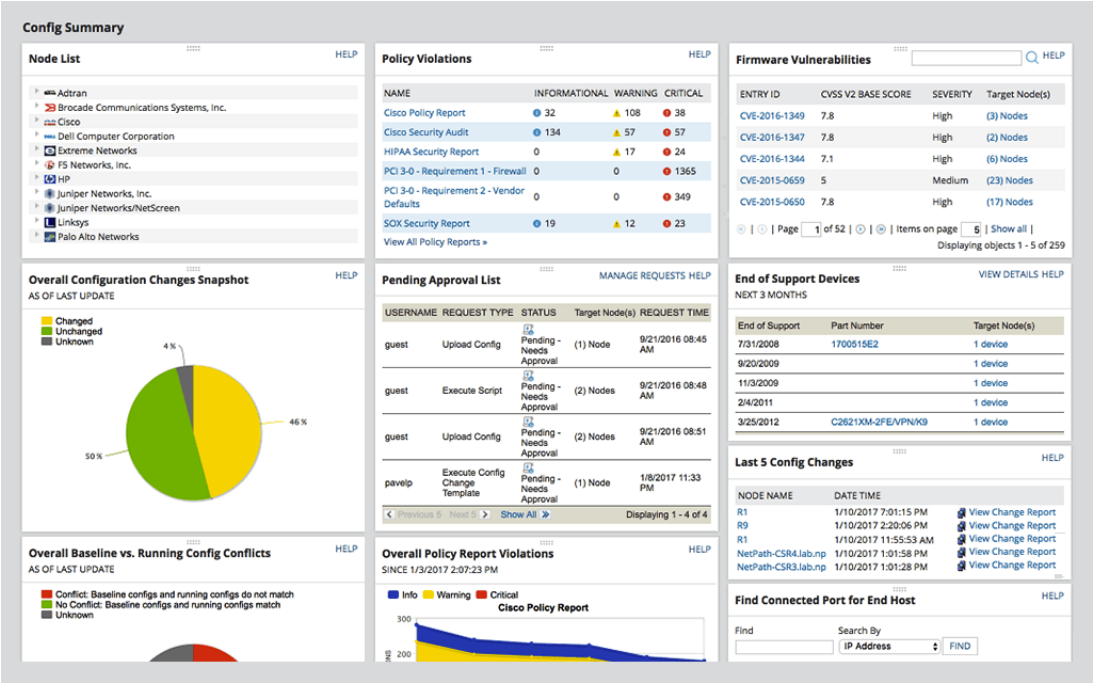

Рис.3 Інтерфейс SolarWinds

NCM також забезпечує підтримку метаданих пристроїв, таких як IP-адреси та назви пристроїв, щоб допомогти адміністраторам відстежувати свої мережеві пристрої. Він також надає базові можливості візуалізації топології мережі, щоб допомогти адміністраторам візуалізувати свої мережеві пристрої та підключення.

Загалом, SolarWinds Network Configuration Manager — це потужний і комплексний інструмент керування конфігурацією мережевих пристроїв, який може допомогти організаціям оптимізувати процеси керування мережею, покращити стан безпеки мережі та забезпечити відповідність галузевим стандартам.

RANCID

RANCID (Really Awesome New Cisco confIg Differ) — це безкоштовний інструмент керування конфігурацією мережевих пристроїв із відкритим вихідним кодом, який спочатку був розроблений для пристроїв Cisco, але тепер підтримує багатьох інших мережевих постачальників. RANCID - це інструмент з інтерфейсом командного рядка, який можна запускати в системах Linux, Unix і macOS. Він використовує набір сценаріїв для підключення до мережевих пристроїв і отримання конфігураційних файлів через Telnet, SSH або інші підтримувані протоколи. Після отримання конфігураційних файлів RANCID зберігає їх у системі контролю версій, наприклад CVS або Git.

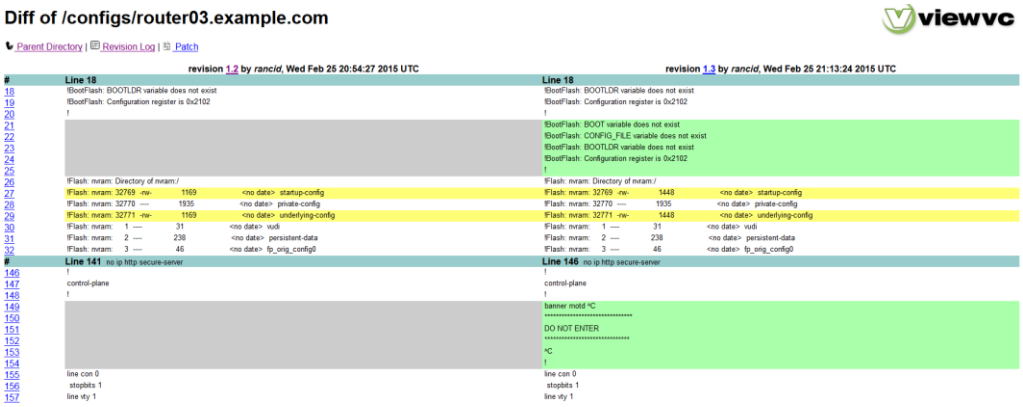

Рис.4 Інтерфейс Rancid

Однією з ключових переваг RANCID є його здатність підтримувати широкий спектр мережевих пристроїв від різних постачальників. RANCID використовує модульну архітектуру, яка дозволяє легко додавати підтримку нових постачальників або протоколів. Інструмент також надає базові можливості візуалізації топології мережі, щоб допомогти адміністраторам візуалізувати свої мережеві пристрої та підключення.

Загалом RANCID є корисним інструментом для мережевих адміністраторів, яким потрібно керувати та відстежувати конфігурації мережевих пристроїв. Його здатність підтримувати широкий спектр мережевих постачальників і протоколів робить його універсальним інструментом для керування складними мережевими середовищами.

Таблиця 1. Порівняльна характеристика

Feature	UNIMUS	OXIDIZED	SOLARWINDS NMC	RANCID
Підтримувані протоколи	SSH, TELNET, SNMP, HTTT(S)	SSH, TELNET, HTTT(S)	SSH, TELNET, SNMP, HTTT(S)	SSH, TELNET
Інтерфейс користувача	Веб-графічний інтерфейс	Веб-графічний інтерфейс	Веб-графічний інтерфейс	Інтерфейс командного рядка
Управління змінами	+	-	+	-
Візуалізація топології мережі	+	-	базова	базова
Ціна	5.90$ за девайс у рік	Open source	Від 1.20$ за девайс у рік	Open source

Дослідивши ці інструменти можна зробити висновок, що платні варіанти мають розширений функціонал з підтримкою та модернізаціями.

Тому варто підходити то вибору виходячи з того який функціонал необхідний та наскільки велика компанія.

1.2 ПОБУДОВА ВИМОГ ТА ФУНКЦІОНАЛУ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

Однією із основних вимог до системи є автоматизований збір поточних конфігурацій пристроїв. Це дозволяє організаціям керувати і контролювати працездатність компонентів у мережі. Він надає цінну інформацію про те, які програмне та апаратне засоби встановлені на пристроях, які настройки застосовані та як вони взаємодіють між собою. Це дозволяє легше впроваджувати, змінювати та відстежувати поточні стани пристроїв в масштабах організації.

Збір поточних конфігурацій пристроїв допомагає виявити проблеми та несумісності між різними пристроями або компонентами. Наприклад, він може показати, чи всі пристрої оновлені до останніх версій програмного забезпечення, чи всі налаштування відповідають потребам організації. Постійний збір поточних конфігурацій пристроїв допомагає також виявити вразливості компонент систем з можливістю їх усунення в подальшому. В разі відмови пристрою або системи знання про його поточну конфігурацію може бути важливим для швидкого відновлення роботи. Збереження резервних копій конфігураційних даних дозволяє відновити пристрій або систему в стан, який був до відмови, з мінімальним часом простою.

Система повинна підтримувати можливість колективної роботи групи адміністраторів з нею, з можливістю рольового розподілу в системі. В кожній системі повинна бути людина яка буде відповідати за стан та працездатність компонентів. Головний адміністратор повинен мати можливість

контролювати усі процеси які відбуваються з компонентами мережі та мати до підтвердження всіх застосованих змін у системі, задля уникнення помилок які можуть допуститись системні адміністратори нижче за рангом.

Система повинна підтримувати можливість перегляду поточних конфігурацій які застосовані в системі, вносити в них зміни, та бачити всю історію змін: ким та коли ці зміни були застосовані. Завдяки системі контролю версій, системні адміністратори повинні мати можливість відстежувати зміни у конфігураціях файлів. Прив'язка до часу може допомогти у вирішенні ряду проблем пов'язаних з нестабільністю системи з прив'язкою до часу. Контроль повинен здійснюватися за кожним окремим пристроєм який наявний у системі.

Системні адміністратори які працюють з пристроями у системі повинні мати можливість вносити зміни у конфігурації пристроїв з якими вони працюють. Можливість зміни налаштувань необхідне для підвищення працездатності системи та задля усунення похибок та несумісностей у системі.

Система повинна бути забезпечена можливістю гарантії відновлення конфігурацій в системі на випадок втрати інформації, задля зменшення часу простою у критичних ситуаціях, таких як вихід із ладу всієї системи корпоративної мережі.

Адміністратори повинні мати можливість використовувати бібліотеку стандартних блоків конфігурацій, які будуть використовуватись як конфігурації для пристроїв у системі. Завдяки цьому, зменшується ймовірність людських помилок при налаштуванні пристроїв, адже автоматичний вибір необхідних конфігурацій будує необхідну конфігурацію автоматично, виходячи з потреб користувача.

Розроблюваний застосунок, повинен надавати можливість системним адміністраторам переглядати перелік наявних в системі маршрутизаторів, дій

які відбуваються в поточний момент в системі та дії які були вже виконані, задля контролю за системою. Додатковою перевагою системи має бути наявність звітів про функціонування системи.

В разі успішного застосування змін, або в разі помилок які виникають у системі, головний системний адміністратор повинен отримувати сповіщення яке надає можливість швидкого реагування на можливі збої в системі або ж для постійного контролю за працездатністю системи.

Також система повинна виконувати збір конфігурацій за розкладом. Це необхідно для того, щоб оперативно виявити неробочі компоненти та усунути проблему.

РОЗДІЛ 2. ВИБІР КОМПОНЕНТ СИСТЕМИ

Перед розробкою власного застосунку, необхідно обрати компоненти розроблюваної системи які задовольнятимуть поставленим вимогам у першому розділі.

2.1 ДВИГУН ДЛЯ ЗАПУСКУ ANSIBLE ПЛЕЙБУКІВ

Jenkins — це сервер автоматизації з відкритим кодом, який широко використовується для безперервної інтеграції (CI) і безперервної доставки (CD) програмних додатків. Jenkins надає платформу для автоматизації створення, тестування та розгортання програмних додатків, і її можна налаштувати та розширити за допомогою плагінів для підтримки широкого спектру мов програмування, інструментів і технологій.

Однією з ключових особливостей Jenkins є його здатність створювати та керувати складними конвеєрами, які можуть включати кілька етапів і середовищ. Ці конвеєри можуть автоматизувати створення, тестування та розгортання програмних додатків, і їх можна налаштувати відповідно до потреб різних груп розробників і проектів.

Однією з ключових переваг використання Jenkins як двигуна для запуску плейбуків Ansible є можливість створювати автоматизовані конвеєри, які можуть організовувати весь процес доставки програмного забезпечення. Це може включати автоматизоване тестування, створення та розгортання додатків у кількох середовищах, керованих з одного конвеєра.

Ще однією перевагою є можливість легко інтегрувати Ansible з іншими інструментами та службами за допомогою плагінів Jenkins. Можливість

поєднання з GitLab дає змогу ініціювати конвеєрні збірки на основі змін у кодовій базі. Крім того, Jenkins надає потужну платформу для керування виконанням підручників Ansible із такими функціями, як планування завдань, розподілені збірки та надійні можливості моніторингу.

Використання Jenkins як двигуна для запуску плейбуків Ansible може допомогти оптимізувати й автоматизувати процес доставки програмного забезпечення, водночас забезпечуючи потужну платформу для керування та оркестрування виконання плейбуків Ansible.

2.2 СИСТЕМА КОНТРОЛЮ ВЕРСІЙ ТА СХОВИЩЕ КОНФІГУРАЦІЙ

Наявність системи контролю версій в системі є досить важливою складовою. Наявність системи контролю версіями надає можливість переглядати історію змін. Також використання репозиторію як сховище для конфігурацій, допомагає зберегти лаконічність системи, без необхідного додавання компонентів які можна замінити одним.

Задля зменшення кількості компонент у системі була розглянута можливість поєднання сховища та системи контролю версій. Можливими рішеннями поставленої задачі є Amazon S3 та GitLab.

Amazon S3 – це хмарна служба зберігання об’єктів, яка надається Amazon Web Services (AWS). Він призначений для зберігання та отримання великих обсягів даних у високо масштабований та довговічний спосіб. Amazon S3 підтримує можливість інтеграції Git що є однією з ключових потреб розроблюваної системи. Використання хмарної служби, підвищує надійність збережених даних, що може бути важливим для швидкого відновлення у разі виходу з ладу системи.

GitLab — це веб-менеджер сховища Git, який надає повну платформу DevOps для керування вихідним кодом, безперервної інтеграції та розгортання, а також інструменти для співпраці для розробників та операційних команд. Він пропонує широкий спектр функцій та інструментів для підтримки процесу розробки програмного забезпечення. GitLab в даному проекті можливо використати як віддалене сховище завдяки якому збережені конфігурації пристроїв будуть завжди доступні, навіть у випадку виходу із ладу компонентів мережі або всього офісу в цілому.

Перевагою GitLab є вбудовані можливості CI/CD, які дозволяють автоматизувати процес застосування нових конфігурацій на пристроях. При зміні конфігурацій, система повинна автоматично застосовувати зміни на пристроях, і за допомогою GitLab це можливо. GitLab дає змогу створювати та підтверджувати або відхиляти merge реквести, писати коментарі до власних змін, мати доступ до системи контролю версій конфігураціями.

Мінусом використання Amazon S3 є обмежена кількість запитів до сховища, що може викликати низку проблем. При виконанні запитів за розкладом, користувач може вичерпати усі можливості запису та стягнення даних зі сховища, що обмежує можливості користувачів.

Саме тому я надаю перевагу GitLab, тому що він задовольняє потреби розроблюваної системи, та є інтуїтивно зрозумілим кожному розробнику.

2.3. ПІДСИСТЕМА АВТОМАТИЗОВАНОГО ЗБОРУ КОНФІГУРАЦІЙ

Ansible, Chef і Puppet — популярні інструменти керування конфігурацією, які використовуються для автоматизації та керування інфраструктурою. Тому на основі їхніх властивостей необхідно обрати яку з цих систем буде використано у розроблюваній системі

Ansible:

- Використовує SSH і не потребує встановлення додаткового агентського програмного забезпечення на цільових системах.
- Використовує просту мову на основі YAML для визначення підручників, що полегшує розуміння та написання.
- Використовує моделі на основі push, де вузол керування надсилає конфігурації та завдання цільовим системам.
- Використовує SSH і не потребує встановлення додаткового агентського програмного забезпечення на цільових системах.
- Має широкий спектр модулів, доступних для різних завдань, включаючи конфігурацію системи, керування хмарою, автоматизацію мережі тощо.
- Можна запускати з будь-якої машини з доступом SSH, що усуває потребу у виділених серверах чи інфраструктурі.

Chef:

- Вимагає інсталяції клієнтського агента, який називається Chef Client, на цільових системах для зв’язку з центральним сервером Chef.
- Використовує доменно-залежну мову (DSL) на основі Ruby для визначення конфігурацій інфраструктури
- Працює на основі моделі на основі витягування, де цільові системи періодично витягують конфігурації з сервера Chef.
- Забезпечує високий рівень розширюваності та гнучкості за допомогою спеціальних ресурсів, що дозволяє користувачам визначати та повторно використовувати власні ресурси.

Puppet:

- Вимагає інсталяції агента під назвою Puppet Agent у цільових системах для зв’язку з центральним сервером Puppet Master.
- Використовує декларативну мову для опису бажаного стану систем.
- Працює на основі моделі на основі вилучення, де цільові системи періодично витягують конфігурації та маніфести з сервера Puppet Master.

Підводячи підсумок, Ansible виділяється своїм підходом без агентів і підходом, що базується на натисканні, тоді як Chef і Puppet покладаються на моделі на основі агентів і на основі витягування. Тому Ansible є в даному випадку найкращим вибором серед трьох представлених варіантів

Ansible буде інтегрований в Jenkins для запуску його плейбуків за допомогою задач.

2.4 ЕМУЛЯЦІЯ ПРИСТРОЇВ CISCO ДЛЯ ТЕСТУВАННЯ СТВОРЕНОГО ЗАСТОСУНКУ

Для того щоб перевірити правильність написаних конфігурацій та роботи програми, необхідно перевірити працездатність системи, зробивши емуляцію мережі з пристроями в ній. Так як розроблювана система на поточному етапі підтримує лише пристрої Cisco, та ця компанія впроваджує додатки та застосунки які можна використати у тестуванні, було обрано платформу Cisco DevNet.

Cisco DevNet –це платформа, надана Cisco Systems, яка пропонує ресурси, інструменти та підтримку для розробників, які створюють програми та інтегрують технології Cisco. DevNet надає доступ до наборів для розробки програмного забезпечення (SDK), API, пісочниці, навчальних лабораторій та

низки інших ресурсів, щоб допомогти розробникам створювати та тестувати свої програми.

Однією з пісочних програм DevNet є Cisco Network Services Orchestrator (NSO) яку я використовував для тестування свого застосунку. Пісочниця Cisco NSO надає розробникам віртуальне лабораторне середовище для дослідження й експериментів із платформою мережевої автоматизації та оркестровки Cisco.

NSO — це програмна платформа, яка дозволяє мережевим адміністраторам автоматизувати конфігурацію та керування мережевими пристроями та послугами. Платформа надає повний набір API для автоматизації та оркестровки мережі, що дозволяє користувачам автоматизувати типові мережеві завдання та робочі процеси.

Пісочниця Cisco NSO надає розробникам практичне середовище для вивчення та експериментів з платформою NSO. Він містить попередньо налаштоване середовище NSO та різноманітні зразки сценаріїв і конфігурацій, які допоможуть розробникам розпочати роботу

Платформа Cisco DevNet надає розробникам велику кількість ресурсів та інструментів, які допомагають їм створювати програми та інтегрувати технології Cisco, включаючи платформу NSO.

Завдяки розвитку своїх платформ, вибір DevNet є найкращим вибором для тестування пристроїв Cisco.

2.5 ІНСТРУМЕНТ ДЛЯ ВІДОБРАЖЕННЯ ТОПОЛОГІЇ МЕРЕЖІ ТА ТИПІВ ПІДКЛЮЧЕННЯ

Batfish — це потужний інструмент із відкритим кодом, який можна використовувати для емуляції мережі та тестування топології. За допомогою

Batfish ви можете емулювати мережеві топології, налаштовувати мережеві пристрої та тестувати мережеві політики та конфігурації.

Після налаштування топології мережі можна використовувати Batfish для перевірки мережевих політик і конфігурацій. Batfish може допомогти виявити неправильні конфігурації, перевірити політику безпеки та виявити можливі збої в мережі до їх виникнення. Batfish можна використовувати для тестування нових проектів і конфігурацій мережі перед розгортанням їх у робочому середовищі.

Однією з додаткових можливостей Batfish є перевірка змін в мережі та валідація мережевих полісів. Перед розгортанням змін конфігурації у робочій мережі Batfish може змоделювати зміни та оцінити їхній вплив на поведінку мережі. Це допомагає запобігти потенційним проблемам і гарантує, що зміни узгоджуються з бажаною поведінкою мережі. Batfish може перевірити, чи належним чином реалізовані в мережевих конфігураціях заплановані мережеві політики, такі як правила брандмауера, списки доступу або політики маршрутизації. Це допомагає виявити неправильні конфігурації, порушення політики або потенційні загрози безпеці.

Batfish використаний в системі задля розуміння системними адміністраторами принципу роботи мережі з якою вони співпрацюють.

2.6 БІБЛІОТЕКА СТАНДАРТНИХ БЛОКІВ КОНФІГУРАЦІЙ

Ймовірно, при написанні конфігурацій для пристроїв власноруч, системний адміністратор може припуститись різного роду помилок. Наприклад синтаксичних, та не помітити це, або ж просто не включити у набір конфігурацій якусь змінну. Тому задля покращення написаних конфігурацій та зменшення кількості помилок, в систему було додано YangSuite.

YangSuite - це інтегроване середовище розробки (IDE), розроблене спеціально для розробки та тестування моделей даних YANG. YANG — це мова моделювання даних, яка використовується для опису даних конфігурації та стану мережевих елементів, і це ключова технологія, що використовується в мережевій автоматизації та програмуванні.

Моделі даних YANG забезпечують стандартизований спосіб опису елементів даних і зв'язків у конфігурації або стані мережевого пристрою. Вони використовуються системами керування мережею для конфігурації та моніторингу мережевих пристроїв, а також у платформах мережевої автоматизації та оркестровки.

YangSuite надає ряд функцій, які допомагають розробникам створювати та тестувати моделі даних YANG, зокрема:

- забезпечує підсвічування синтаксису та перевірку моделей даних YANG, гарантуючи, що моделі правильно відформатовані та відповідають правилам синтаксису YANG.
- містить графічне представлення моделі даних, що дозволяє легко візуалізувати структуру та зв'язки моделі даних.
- може генерувати код кількома мовами програмування з моделей даних YANG, включаючи Java, Python і C++. Це дозволяє легко інтегрувати моделі даних YANG у програми.
- містить низку інструментів тестування та перевірки, щоб переконатися, що моделі даних YANG працюють правильно.
- можна інтегрувати з іншими інструментами мережевої автоматизації та оркестровки, такими як Cisco Network Services Orchestrator (NSO), щоб спростити процес розробки.

Метою YangSuite є забезпечення комплексного середовища розробки для моделей даних YANG, що полегшує розробникам створення та тестування

моделей даних для автоматизації та оркестровки мережі. Це допомагає спростити процес розробки та підвищити якість моделей даних YANG, що, у свою чергу, може підвищити ефективність і надійність систем управління мережею та автоматизації.

2.7. КОНТЕЙНЕРИЗАЦІЯ

Для простоти розгортання застосунку та для ізоляції застосунку в окремих контейнерах, усі застосунки були контейнеризовані.

Контейнеризація — це технологія віртуалізації, яка дозволяє вам упакувати програму разом із її залежностями, бібліотеками та конфігураційними файлами в єдиний контейнер.

Контейнери забезпечують ізоляцію процесу, що означає, що кожен контейнер працює у своєму власному ізольованому середовищі. Ця ізоляція покращує безпеку та допомагає запобігти конфліктам між програмами та їхніми залежностями.

Наразі існують такі застосунки як Podman, OpenShift та Docker. Проте найпопулярнішим та завдяки великій підтримці та розгорнутій документації, застосунок буде контейнеризована за допомогою Docker.

2.8. АРХІТЕКТУРА РОЗРОБЛЮВАНОГО ЗАСТОСУНКУ

Архітектуру розроблюваного застосунку можна побачити на рис.5. Основна взаємодія користувача відбувається з GitLab та Jenkins. На сервер Jenkins було встановлено плагін Ansible, і завдяки Jenkins, плейбуки Ansible

запускаються та виконують написані конфігурації. Виконання завдань Jenkins включає в себе як збір так і збереження конфігурацій у репозиторій GitLab. Завдяки Jenkins можна переглянути логи запущених задач, їхній статус, та відстежити всі помилки які могли трапитись протягом виконання задач. Jenkins детально описує кожен свій крок, та дає розгорнуту інформацію про стани задач.

GitLab в свою чергу виконує роль сховища та застосунку для перегляду та зміни конфігурацій. Всі зміни які вносяться у систему мають розподіл по користувачам, та всі рішення про зміни конфігурацій приймає головний користувач якому буде призначена ця роль.

Завдяки Batfish аналізуються конфігурації на наявність помилок в написаних конфігураціях та завдяки ньому можливо уникнути типових помилок. Також додатково цей застосунок відображає топологію мережі, в залежності від поточних конфігурацій пристроїв. Тобто фактично необхідно завантажити конфігурації пристроїв, для того щоб відобразити топологію мережі.

YangSuite був доданий як зручний інтерфейс, в якому можна завантажити, опрацювати та проаналізувати конфігурації пристроїв. Завдяки ньому Системний адміністратор може створити свою власну конфігурацію завдяки зручному веб інтерфейсу. Перевагою YangSuite в даному випадку є підтримка багатьох протоколів, найпопулярнішим з яких є netConf та restConf.

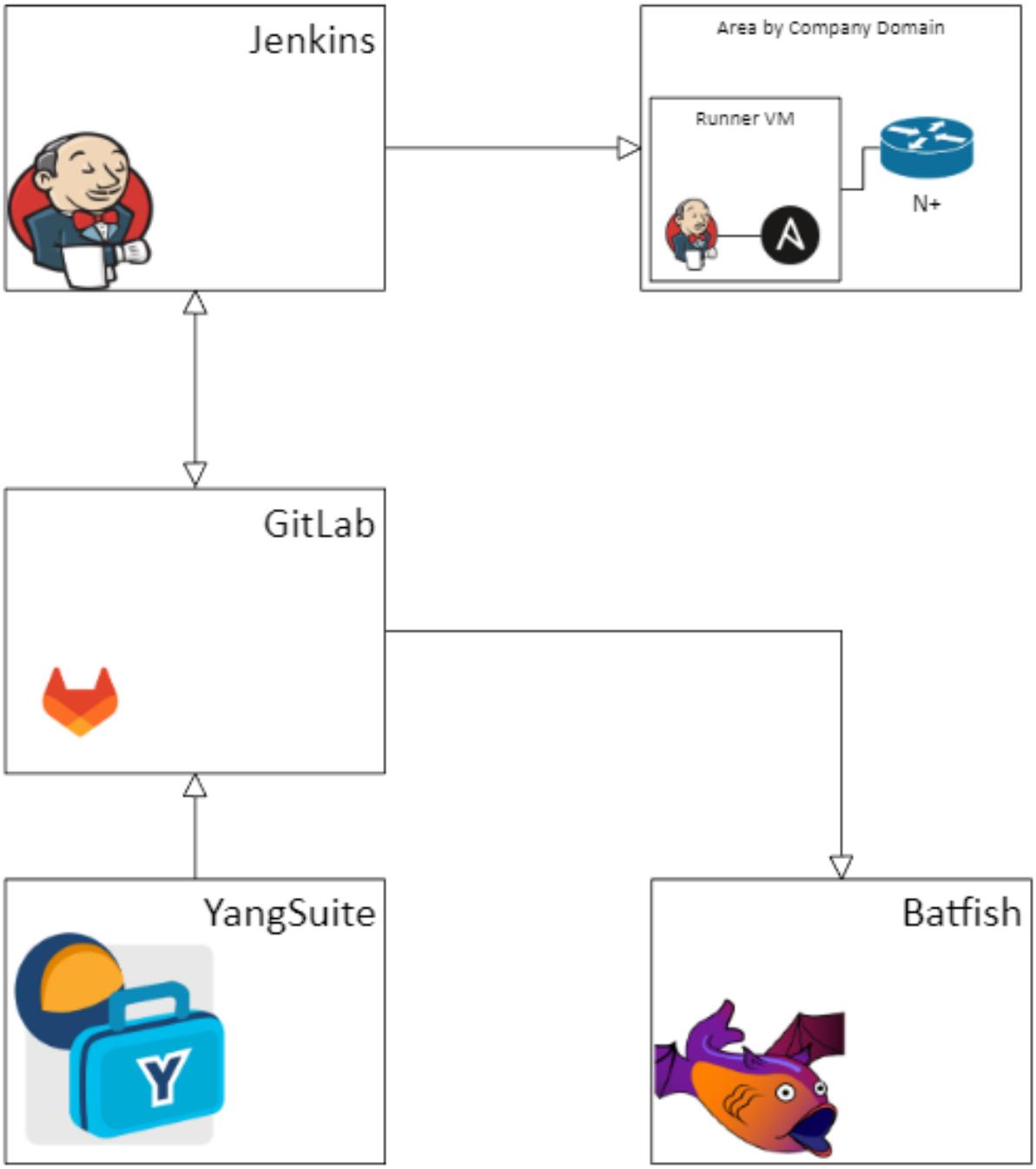

Рис.5 Архітектура розроблюваного застосунку

РОЗДІЛ 3 ОПИС РОБОТИ РЕАЛІЗОВАНОЇ СИСТЕМИ

У цьому розділі буде описана робота системи, взаємодія компонентів та як працювати з системою.

3.1. ПОКРОКОВИЙ РОЗБІР РОБОТИ СИСТЕМИ

Одним із основних засобів взаємодії користувача є Jenkins – рис. 6. В Jenkins був інтегрований Ansible для того щоб виконувати та керувати задачами. Для збору конфігурацій використовується спеціально написана задача, яка використовує заздалегідь написані плейбуки з конфігураціями які необхідно виконати.

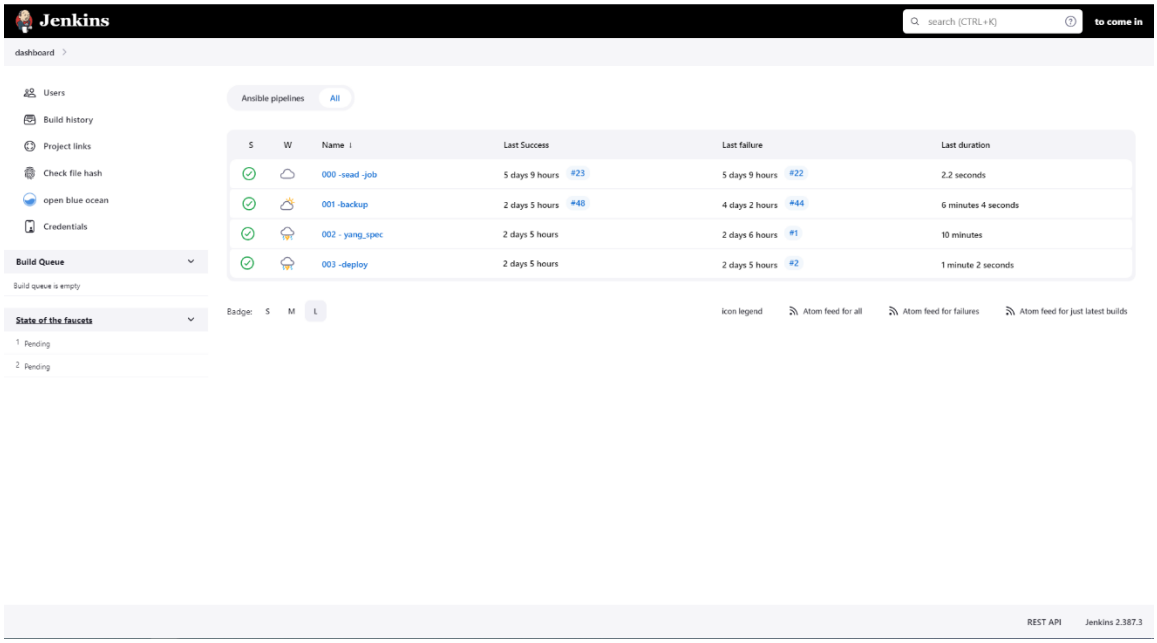

Рис.6 Загальний вигляд Jenkins

Веб-інтерфейс Jenkins надає обширну інформацію для системних адміністраторів які працюватимуть з цим додатком. В Jenkins користувач може переглянути історію всіх запусків збору бекапів, ким це було запущено, коли та скільки часу це зайняло. Jenkins зберігатиме логи усіх операцій що були здійснені в системі, та ким. Кожен системний адміністратор матиме

власний аккаунт завдяки якому він може здійснювати операції в системі, і відповідно кожна операція прив'язана до кожного окремого користувача. Це дає змогу відслідковувати дії кожного користувача з розділенням по ролям та даватиме змогу бачити повний лог подій – рис. 7.

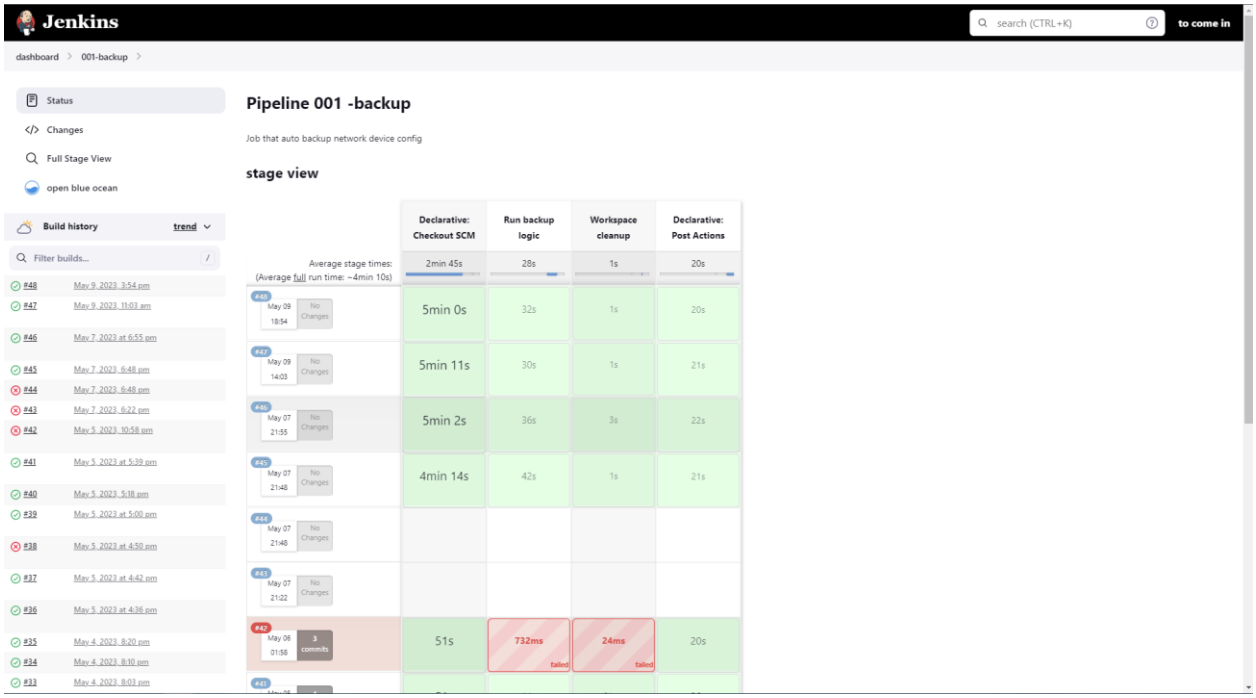

Рис.7 Відображення поточного стану та результатів виконання робіт

Кожний окремий лог операції можна переглянути з детальним описом через Blue Ocean – рис. 8. Системний адміністратор має змогу переглянути що відбувалось під час запуску кожної задачі, які команди виконувались, які тести були успішними. Детальний опис кожної операції включає в себе також логи запуску плейбуку прив'язаного до цієї задачі.

Приклад реалізованої задачі Jenkins можна побачити в додатку Б. Показаний код ділиться на 2 етапи: виконання задачі та відправка імейлу. Загалом в цій задачі визначено які плейбуки мають бути використані, інвентар пристроїв які задіяні, та усі необхідні для роботи параметри такі як ім'я користувача його пароль та ін.

Плейбук який викликається цією роботою продемонстровано у додатку А. Основною задачею цього плейбуку є збір конфігурації з пристрою та додавання зібраної конфігурації з пристрою до віддаленого репозиторію.

У випадку успішного або неуспішного запуску задачі, на вказаний імейл в системі буде відправлене повідомлення з відповідним результатом виконання. Також додатково за результатами виконання кожної задачі створюються артефакти, які можна вивантажити і переглянути.

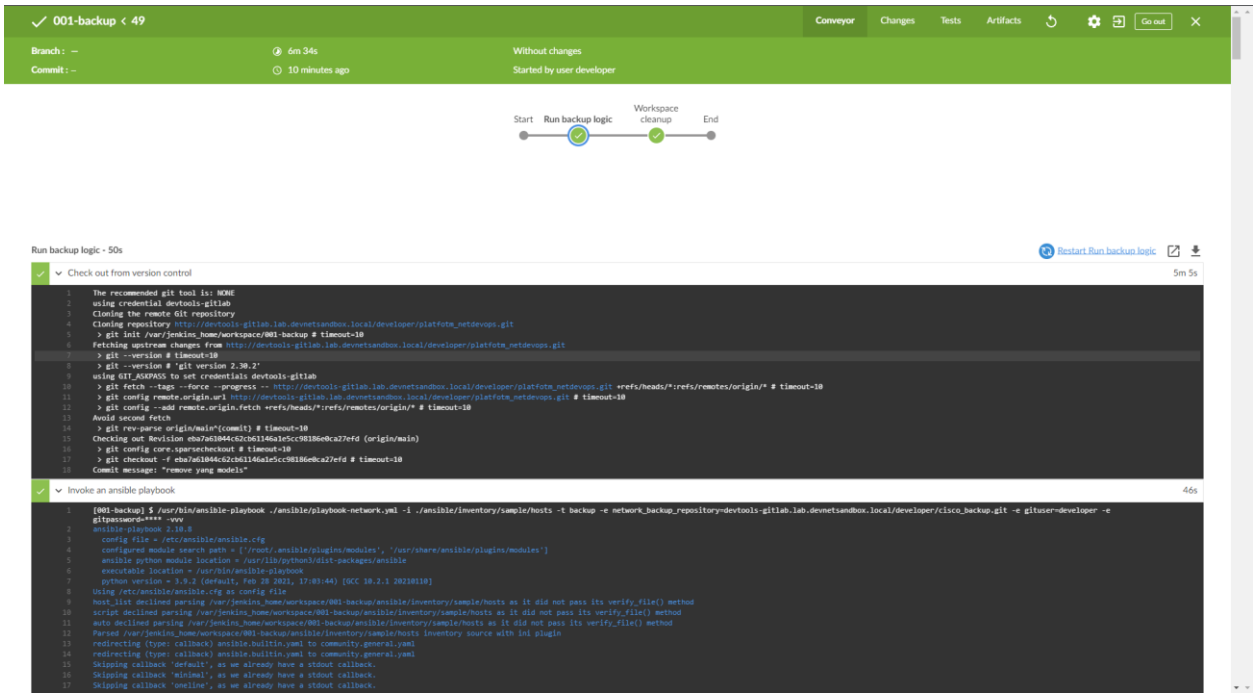

Рис.8 Відображення кожного виконуваного кроку в Jenkins

Виконувана задача, за допомогою плейбуків збирає конфігурації пристроїв які визначені в окремому файлі. Ansible плейбуки легко налаштовуються та можуть бути розширені додатковими функціями та плагінами. Вони забезпечують гнучкий спосіб для автоматизації управління інфраструктурою та розгортання додатків.

Пристрої з яких збираються конфігурації визначаються в Ansible Inventory файлі і мають наступний вигляд який продемонстровано в додатку С. В такому файлі визначається тип пристрою який знаходиться в системі,

його ім’я та IP адреса цього пристрою, задля того щоб до нього можливо було під ’єднатись.

Конфігурації цих пристроїв, якщо вони були оновлені зберігаються у репозиторії GitLab. GitLab має всі можливості системи контролю версій. Завдяки GitLab системний адміністратор має змогу переглянути всі зміни які були внесені у конфігурацію файлу, ким та коли були змінені конфігурації, переглянути яку саме конфігурацію має поточний пристрій. Всі файли зберігаються у GitLab з його IP адресою та назвою пристрою, що допомагає в пошуку необхідної конфігурації у репозиторії.

Завдяки GitLab – рис. 9 ми можемо задовольнити потребу у безпечному збереженні конфігурацій, на випадок ситуацій коли необхідно відтворити втрачені файли, на випадок критичних ситуацій. GitLab матиме останні конфігурації які були використані у системі, і за необхідності їх можна буде вивантажити.

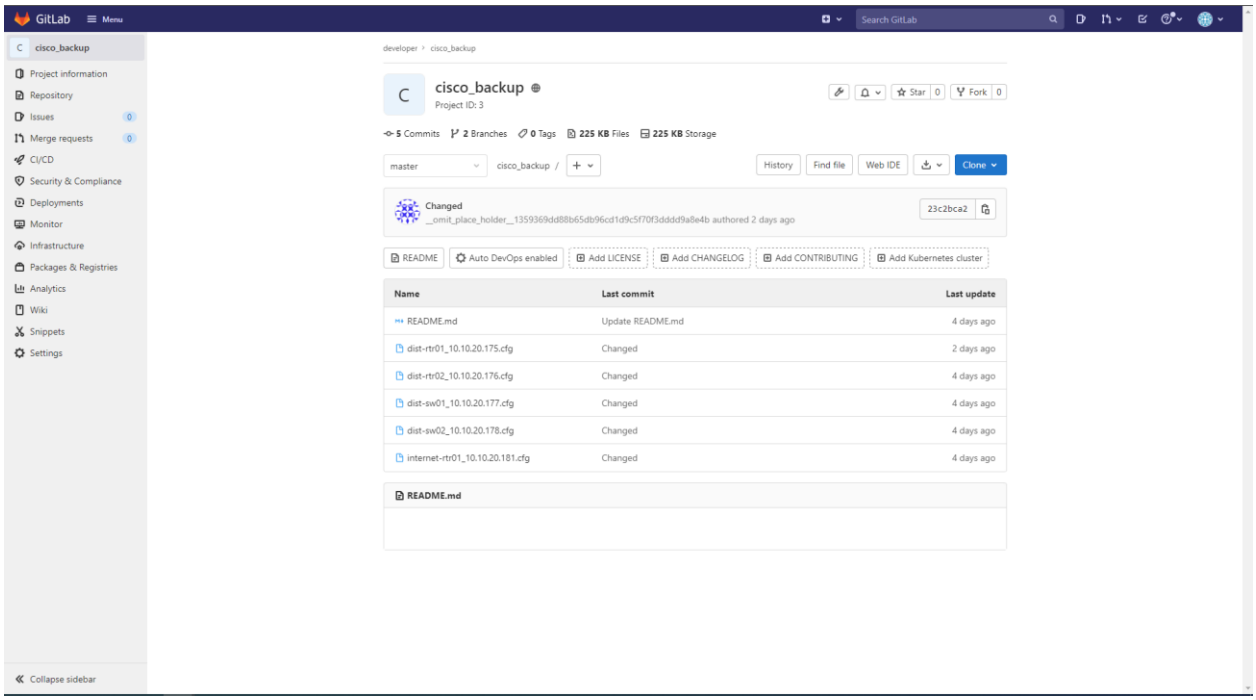

Рис.9 Інтерфейс GitLab з вивантаженими конфігураціями

Web-інтерфейс GitLab дозволяє коригувати конфігурації всередині GitLab, і це дозволяє вносити зміни в файли конфігурації пристроїв. Завдяки інтегрованому Web IDE в GitLab користувач має змогу вносити зміни.

При зміні файлу конфігурації пристроїв створюється merge request який необхідно підтвердити для зміни конфігурації файлу. Після підтвердження зміни конфігурації запускається окремий процес, який застосовує зміни конфігурації на пристроях – рис. 10.

Користувачі матимуть власну змогу налаштувати GitLab для користувачів які мають відстежувати та підтверджувати всі зміни які вносяться у файли конфігурацій.

Також завдяки GitLab можна зручно переглянути всю історію змін файлів, що в них змінювалось, коли та ким ці зміни були застосовані та підтверджені.

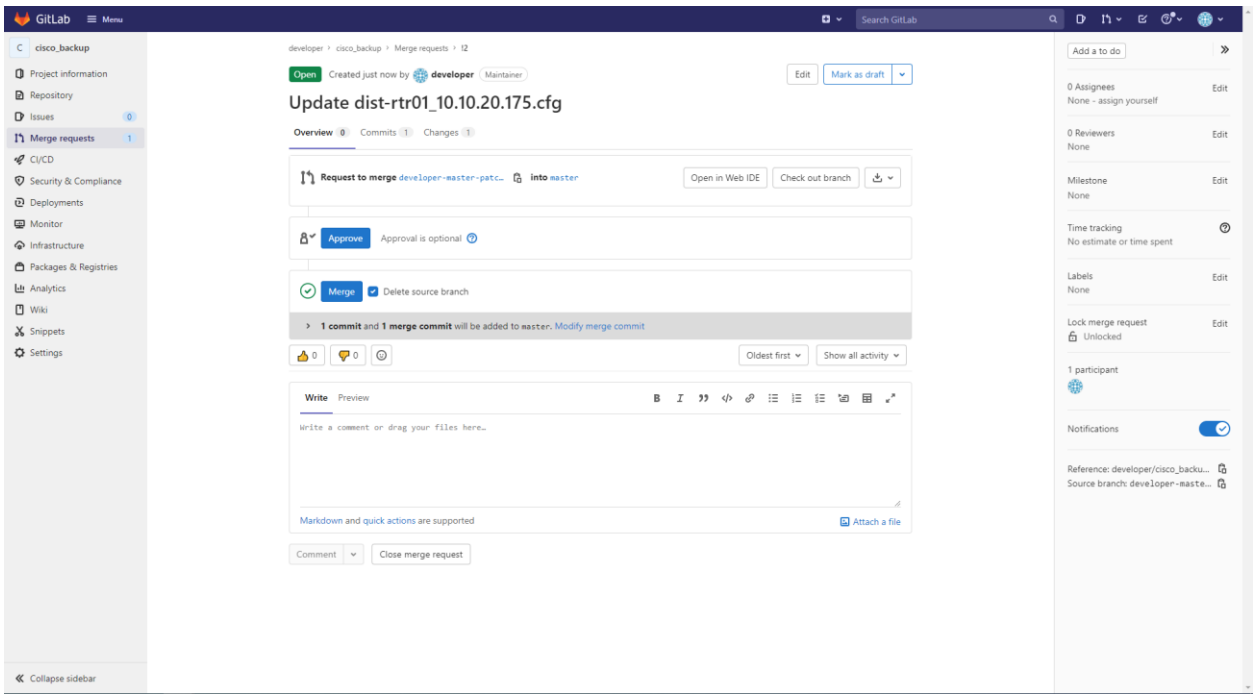

Рис.10 Зміна конфігурацій в GitLab

Ця система була протестована завдяки Cisco DevNet - рис. 11, в якій було використано Cisco Network Services Orchestrator. Cisco Network Services Orchestrator (NSO).

Наразі, було використано лише чотири пристрої з мережі, з конфігураціями котрих було проведено аналіз. На поточний момент система може конфігурувати та працювати лише з пристроями Cisco, завдяки розгорнутим можливостям тестування, та написання конфігурацій. Система може працювати з пристроями з операційних систем NX-OS, IOS XR, IOS та ASA.

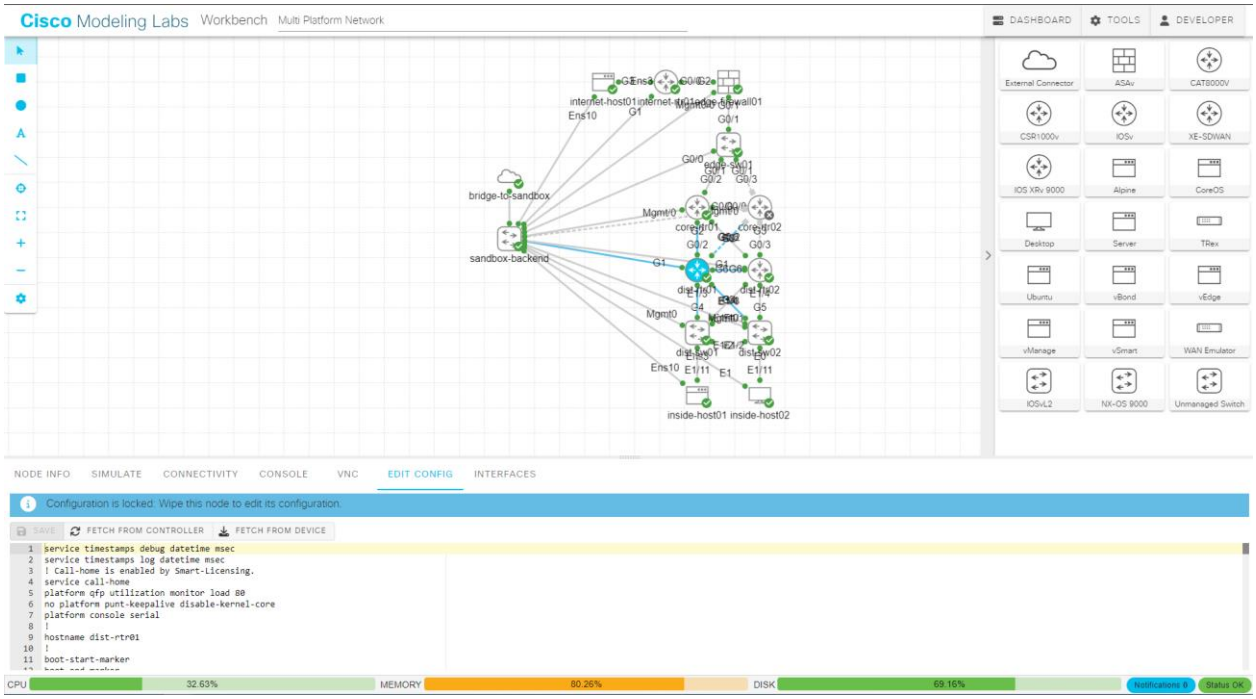

Рис.11 Тестове Середовище DevNet

NX-OS розроблений для середовищ центрів обробки даних і надає розширені функції для високопродуктивної мережі, віртуалізації та автоматизації. NX-OS підтримує модульну архітектуру та пропонує такі функції, як віртуальні контексти пристроїв (VDC).

IOS XR розроблений для високоякісних маршрутизаторів, таких як Cisco CRS (Carrier Routing System) і ASR (Aggregation Services Router). Він оптимізований для розгортання великомасштабних постачальників послуг і базової мережі

IOS (Internetwork Operating System використовується маршрутизаторами та комутаторами Cisco для корпоративних мереж. Це універсальна операційна система, яка надає широкий спектр функцій і послуг для маршрутизації, комутації, безпеки та керування

ASA (Adaptive Security Appliance забезпечує вдосконалений брандмауер, VPN (віртуальну приватну мережу) і можливості запобігання вторгненням. Пристрої ASA зазвичай використовуються в архітектурах мережевої безпеки для захисту мереж від несанкціонованого доступу, загроз і атак

Для покращення розуміння системних адміністраторів система була обладнана утилітою Batfish Dashboard. Завдяки ній системний адміністратор може завантажити поточні конфігурації мережевих пристроїв які на поточний стан працюють в системі, і застосунок побудує топологію мережі.

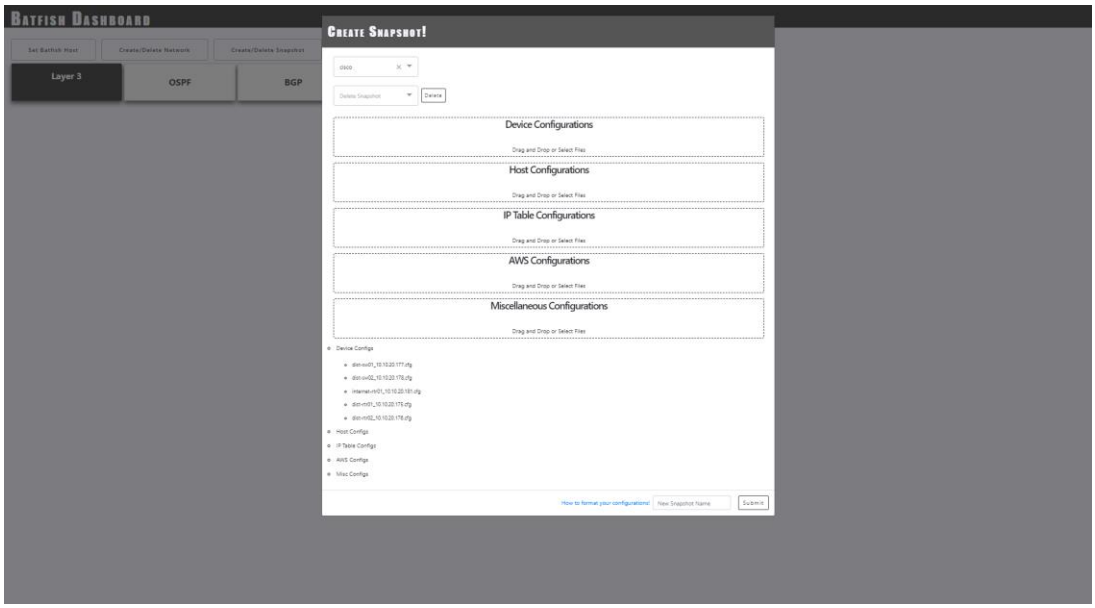

Рис.12 Завантаження конфігурацій в систему

На етапі тестування було використано 2 роутери та 2 свічі Cisco. На кожній окремій вкладниці протоколів маршрутизації можна побачити різну візуалізацію топології мережі – рис. 13.

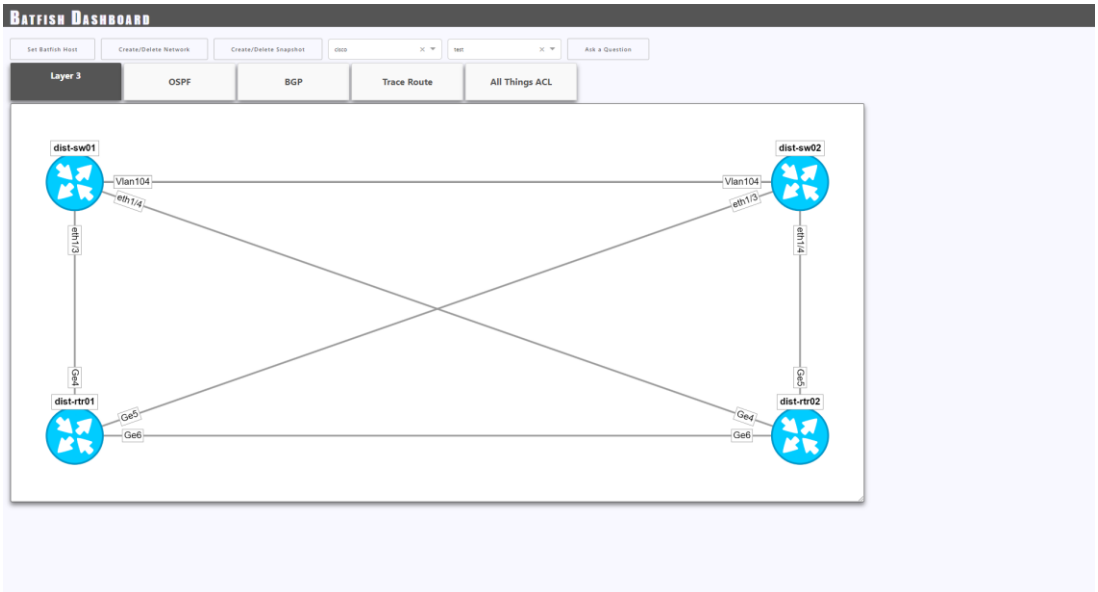

Рис.13 Відображення топології мережі в залежності від завантажених конфігурацій

У комп'ютерних мережах Layer 3 відноситься до третього рівня моделі OSI (взаємозв'язку відкритих систем), яка також відома як мережевий рівень. Цей рівень відповідає за маршрутизацію та пересилання пакетів даних між різними мережами, а також забезпечує логічну адресацію та механізми обробки помилок.

Пристрої Layer 3, такі як маршрутизатори, використовують цю інформацію про логічну адресацію для маршрутизації пакетів через мережу шляхом перевірки IP-адреси призначення та визначення наступного переходу в топології мережі. Вони також можуть виконувати такі завдання, як фільтрація пакетів і трансляція мережевих адрес (NAT), щоб забезпечити функції безпеки та підключення.

Ця інформація дає змогу мережевим адміністраторам побачити по яким портам зроблені підключення, як між собою вони пов'язані та які інтерфейси мережі задіяні.

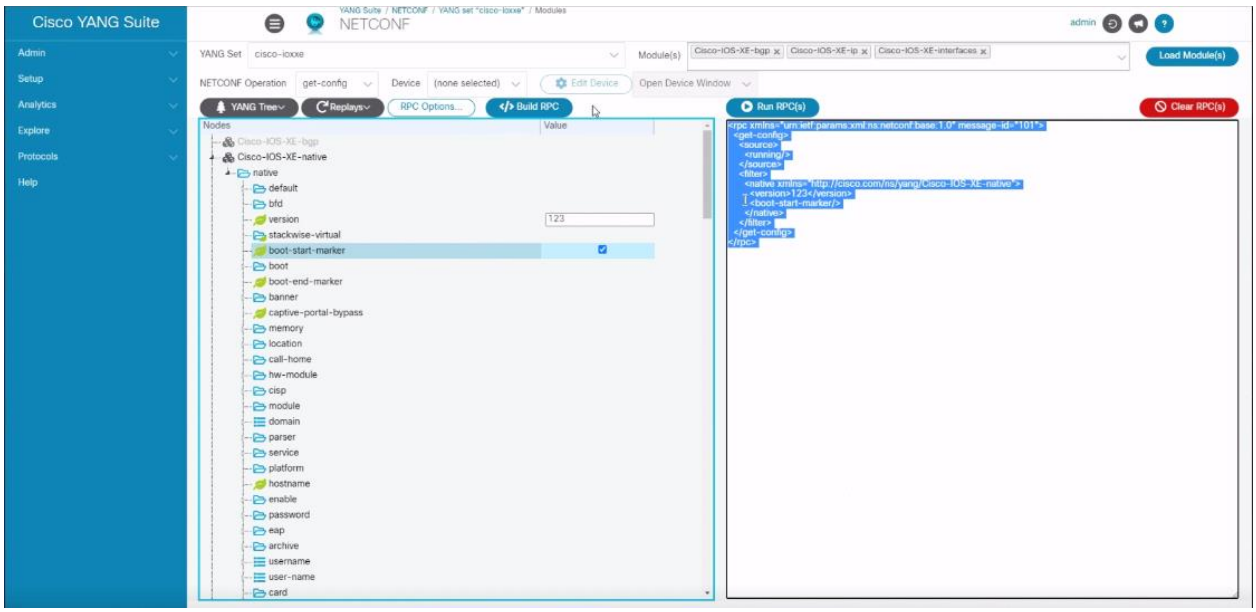

Рис.14 Генерація конфігурацій завдяки YangSuite

У застосунку YangSuite системний адміністратор має можливість згенерувати конфігурації для пристроїв завдяки веб інтерфейсу. У системі

представлено різноманітні види конфігурацій які можуть бути застосованими на пристроях, і системному адміністратору залишається лише обрати необхідні параметри, та застосувати ці зміни на пристроях.

3.2 ПОЯСНЕННЯ ДО ВИМОГ ТА ЯК ВОНИ БУЛИ ВИРІШЕНІ

Для опису типових дій які повинен виконати адміністратор та для перевірки виконання вимог поставлених до розробленої системи у першому розділі, необхідно описати функціонал системи.

Автоматизований збір поточних конфігурацій пристроїв здійснюється за допомогою окремої задачі в Jenkins, та зберігає зібрані конфігурації в репозиторії GitLab.

Jenkins надає можливість переглянути всі історії запуску кожної окремої задачі з можливістю контролю кожного етапу. У випадку збою в системі системний адміністратор отримає повідомлення на пошту.

Jenkins має розгорнуту історію подій з можливістю покрокового відстеження поточного стану системи. Також він генерує логи системи відповідно до кожної задачі.

Jenkins та GitLab підтримують можливість колективної роботи та розділення на ролі всередині системи. Один окремий системний адміністратор буде отримувати всі нотифікації про стан системи, та в разі збоїв у них, та підтверджувати зміни внесені у конфігурації пристроїв.

Завдяки GitLab системні адміністратори мають можливість переглянути всі наявні в системі пристрої завдяки файлу Ansible-Inventory, та переглянути конфігурації файлів які були застосовані востаннє в системі. Всі файли мають назву яка відповідає його IP адресі та назві пристрою.

GitLab надає змогу системним адміністраторам переглядати історії змін кожного окремого файлу конфігурації, вносити в них зміни та мають прив’язку до часу та окремого користувача який ці зміни вносив.

GitLab також був використаний як сховище для останніх конфігурацій на випадок потреби відновлення конфігурацій.

В систему було додано YangSuite, мета якого допомогти системним адміністраторам в створенні файлів конфігурацій для пристроїв наявних в системі.

Також було додано Batfish Dashboard для відображення топології мережі, що підвищить розуміння системних адміністраторів в роботі з мережею.

ВИСНОВКИ

У першому розділі було проведено порівняльний аналіз існуючих систем. Були наведені приклади систем Unimus, OXIDIZE, RANCID та SolarWinds Network Configuration Manager. Було проведено порівняльний аналіз цих систем за такими категоріями як вартість, наявність інтерфейсу користувача, підтримувані протоколи та наявність топології мережі. До кожної з наведених систем було приведено та відображено переваги та недоліки кожної з систем

У другому розділі було визначено компоненти розроблюваної системи, та була розроблена порівняльна характеристика можливих компонентів системи. Вибір кожної компоненти системи було обґрунтовано її необхідність, яке призначення воно має в розроблюваній системі та які функції та конфігурації вона має. Також у розділі було відображено архітектуру системи та її взаємозв'язки між компонентами.

У третьому розділі було пояснено принципи роботи кожної компоненти, було детально описано як працюють та як взаємодіяти з компонентами системи. Також було продемонстровано код конфігурацій розробленої системи. Було описано яким чином вимоги, поставлені у першому розділі, були задоволені в розробленій системі.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Unimus Електронний ресурс. Доступ 05.02.2023 https://unimus.net/
2. Порівняння з відгуками комерційних проектів Електронний ресурс. Доступ 05.02.2023 https://www.trustradius.com/
3. Oxidized Електронний ресурс. Доступ 05.02.2023 https://github.com/ytti/oxidized
4. Rancid Електронний ресурс. Доступ 05.02.2023 https://shrubbery.net/rancid/
5. Integration an IPAM server Електронний ресурс. Доступ 02.04.2023 https://www.cisco.com/c/dam/en_us/training-events/product-training/dnac-12/IPAM/DNAC12_IntegratingAnIPAMServer.pdf
6. YangSuite by docker Електронний ресурс. Доступ 20.04.2023 http://www.zhaocs.info/install-use-yangsuite-by-docker.html
7. Batfish Електронний ресурс. Доступ 20.04.2023 https://github.com/batfish/ansible/blob/master/docs/bf_assert.rst
8. Batfish Documentation Електронний ресурс. Доступ 23.04.2023 https://batfish.readthedocs.io/en/latest/notebooks/linked/getting-started-with-batfish.html
9. Integrating Ansible with Jenkins in a CI/CD process Електронний ресурс. Доступ 20.04.2023 https://www.redhat.com/en/blog/integrating-ansible-jenkins-cicd-process
10. Документація GitLab Електронний ресурс. Доступ 20.04.2023 https://docs.gitlab.com/
11. GitHub Batfish_DashboardЕлектронний ресурс. Доступ 20.04.2023 https://github.com/drosarius/batfish_dashboard
12. Developer Cisco Sanbox Електронний ресурс. Доступ 20.04.2023 https://developer.cisco.com/site/sandbox/

13. Automation 16. Електронний ресурс. Доступ 20.04.2023 https://karneliuk.com/2022/07/automation-16-how-to-prepare-cisco-nexus-9000-cisco-nx-os-to-be-automated-with-netconf-and-gnmi-complete-guide/

ДОДАТКИ

ДОДАТОК А

```
---
- name: Backup the config
  ios_config:
    backup: yes
  register: config_output

- set_fact:
    temp_backup_file: "{{ config_output.backup_path }}"


- name: Clone the backup repo
  git:
    repo: "{{ schema }}{{ gituser | urlencode }}:{{ gitpassword | urlencode }}@{{
network_backup_repository }}"
    dest: "{{ network_backup_dir }}"
    accept_hostkey: yes
    force: yes
  register: clone_result
  delegate_to: localhost
  run_once: true
  when: network_backup_repository is defined

- include_tasks: "{{ role_path }}/tasks/{{ ansible_network_os }}-backup.yml"

- name: Copy the temp to the destination
  copy:
    src: "{{ temp_backup_file }}"
    dest: "{{ network_backup_file }}"
  register: copy_result
  delegate_to: localhost

- name: Delete the temp file
  file:
    path: "{{ temp_backup_file }}"
    state: absent
  changed_when: False
  delegate_to: localhost

- block:
  - name: Add any new backups to the repository
    shell: "git add *.cfg"
    args:
      chdir: "{{ network_backup_dir }}"
    delegate_to: localhost
    changed_when: False
    run_once: true
```

```
- name: Get the status of the repository
  shell: "git status"
  args:
    chdir: "{{ network_backup_dir }}"
  register: status_results
  changed_when: False
  ignore_errors: true
  delegate_to: localhost
  run_once: true

- name: Commit the changes
  shell: "git commit -am 'Changed'"
  args:
    chdir: "{{ network_backup_dir }}"
  when: status_results.stdout is search('Changes')
  delegate_to: localhost
  run_once: true
  environment:
    GIT_COMMITTER_NAME: "{{ git_name | default(comit) }}"
    GIT_COMMITTER_EMAIL: "{{ git_email | default(comit) }}"
    GIT_AUTHOR_NAME: "{{ git_name | default(comit) }}"
    GIT_AUTHOR_EMAIL: "{{ git_email | default(comit) }}"

- name: Push the changes
  shell: "git push origin master"
  args:
    chdir: "{{ network_backup_dir }}"
  run_once: true
  when: status_results.stdout is search('Changes')
  delegate_to: localhost
  environment:
    GIT_COMMITTER_NAME: "{{ git_name | default(comit) }}"
    GIT_COMMITTER_EMAIL: "{{ git_email | default(comit) }}"
    GIT_AUTHOR_NAME: "{{ git_name | default(comit) }}"
    GIT_AUTHOR_EMAIL: "{{ git_email | default(comit) }}"
when: network_backup_repository is defined
```

Додаток Б

```
pipeline {
    agent { label 'master' }
    stages {
        stage("Run backup logic") {
            steps {
                wrap([$class: 'AnsiColorBuildWrapper', colorMapName: "xterm"]) {
                    withCredentials([usernamePassword(credentialsId: 'devtools-
gitlab', usernameVariable: 'gituser', passwordVariable: 'gitpassword')]) {
                        ansiblePlaybook(
                                playbook: './ansible/playbook-network.yml',
                                installation: 'Ansible_master',
                                inventory: './ansible/inventory/hosts',
                                tags: '${ansible_tags}',
                                extras: "-vvv",
                                extraVars: [
                                    network_backup_repository:
"${network_backup_repository}",
                                    gituser: '$gituser',
                                    gitpassword: '$gitpassword',
                                ],
                                colorized: true)
                    }
                }
            }
        }
        stage("Workspace cleanup") {
            steps {
                cleanWs()
            }
        }
    }
    post {
        success {
            emailext(
                    subject: "SUCCESS: '${env.JOB_NAME}' is SUCCESSFUL",
                    body: """<p>Check console output at <a
her="${env.BUILD_URL}">${env.JOB_NAME}</a></p>
                    \n\n Job successful. \n\n""",
                    to: defaultMailList
            )
        }
        unsuccessful {
            emailext(
                    subject: "UNSUCCESS: '${env.JOB_NAME}' is UN-SUCCESSFUL",
                    body: """<p>Check console output at <a
href="${env.BUILD_URL}">${env.JOB_NAME}</a></p>
                    \n\n Export unsuccessful. \n\n""",
                    to: defaultMailList
```

```
            )
        }
    }
}
```

ДОДАТОК С

```
[cisco_iosxe]
dist-rtr01 ansible_host=10.10.20.175
dist-rtr02 ansible_host=10.10.20.176
internet-rtr01 ansible_host=10.10.20.181

[cisco_nxos]
dist-sw01 ansible_host=10.10.20.177
dist-sw02 ansible_host=10.10.20.178

[cisco_iosxe:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_become=yes
ansible_become_method=enable

[cisco_nxos:vars]
ansible_connection=network_cli
ansible_network_os=nxos
ansible_become=yes
ansible_become_method=enable

[all:vars]
ansible_user=cisco
ansible_password=cisco
ansible_become_pass=cisco
```