

Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

**СИСТЕМА УПРАВЛІННЯ
КОНФІГУРАЦІЯМИ ПРИСТРОЇВ В МЕРЕЖІ
ПІДПРИЄМСТВА**

**Текстова частина до курсової роботи за
спеціальністю „Комп’ютерні науки” 6.050103**

Керівник курсової роботи

к.т.н., доц. Черкасов Д.І.

(підпис) _____

“ ____ ” _____ 2020 р.

Виконав студент Гурін В.І.

“ ____ ” _____ 2020 р.

Зміст

1.	ВСТУП.....	3
1.1.	Управління конфігураціями пристроїв в мережі підприємства	3
1.2.	Мета системи управління конфігураціями пристроїв в мережі підприємства.....	4
1.3.	Основні складові системи управління конфігураціями:.....	5
2.	Огляд існуючих рішень	5
2.1.	Огляд існуючих систем управління конфігураціями	5
2.2.	Огляд існуючих рішень для створення власної системи управління конфігураціями.....	8
2.3.	Оркестрація, CloudFormation, Terraform.....	14
2.4.	Jenkins, Ansible AWX.....	16
2.5.	Висновок	17
3.	Структурна розробка власного рішення	17
3.1.	Структурна схема власного рішення,	17
3.2.	Перелік компонентів, їх характеристики, опис	18
3.3.	Опис функціонування системи.....	29
4.	Детальна розробка компонента.....	33
	Список літератури.....	37
5.	Додаток А.....	38

1. ВСТУП

1.1. Управління конфігураціями пристроїв в мережі підприємства

Мережа сучасного підприємства -- складна система, яка містить в собі велику кількість пристроїв. Для узгодженого функціонування в мережі кожен з пристроїв має свою індивідуальну конфігурацію.

Протягом життєвого циклу мережі відбуваються зміни конфігурацій пристроїв для відслідковування нових умов, вирішення нових задач, підтримки нових сервісів. Під час застосування оновленої конфігурації пристроїв існує вірогідність помилок, в наслідок чого потрібно повернути на пристрій одну з попередніх версій конфігурації. У разі виходу з ладу пристрою і заміні його новим необхідне завантаження конфігурації нового пристрою.

Швидке та ефективне оновлення конфігурацій пристроїв до потрібних версій може бути здійснено автоматизованою системою управління конфігураціями пристроїв.

Підсумовуючи, автоматизована система управління конфігураціями це система яка має допомагати адміністратору у :

- швидкому знаходженні оптимального курсу дій у разі виникнення проблем у системі, спираючись на минулі версії конфігурацій.
- встановленні конфігурацій для нових пристроїв системи ;
- запобіганні виникнення проблем які можуть виникнути у майбутньому, внаслідок не коректної конфігурації пристроїв(в тому числі проблеми безпеки).

1.2. Мета системи управління конфігураціями пристроїв в мережі підприємства

Мета системи управління конфігураціями пристроїв в мережі підприємства це:

- Можливість забезпечення того, щоб усі програмні та апаратні активи, якими володіє компанія, були відомі та відслідковуються у будь-який час.
- Зниження витрат завдяки повному контролю інформації про конфігурації пристроїв і як наслідок усунення можливих проблем.
- Швидке відновлення роботи. У випадку відключення пристрою, можливість швидкого відновлення системи, оскільки конфігурації пристроїв задокументовані та автоматизовані.
- Перевірка коректності і консистентності конфігурацій
- Ефективне управління змінами і контроль версіями. Наявність початкової конфігурації і історії змін дозволяє відслідковувати помилки значно швидше.
- Покращений досвід для клієнтів та внутрішнього персоналу шляхом швидкого виявлення та виправлення неправильних конфігурацій, які можуть негативно вплинути на продуктивність.
- -Налаштування роботи з іншими мережами підприємств

1.3. Основні складові системи управління конфігураціями:

- Репозиторій з системою контролю версій на git
- Компоненти для доступу до пристроїв
- Програмні модулі власної розробки
- Система автоматизації Ansible
- Веб-інтерфейс адміністратора
- Система контролю коректності конфігурацій
- Систему повідомлень
- Система модифікацій конфігурацій згідно змін даних в інших мережах

2. Огляд існуючих рішень

2.1. Огляд існуючих систем управління конфігураціями

На ринку представлений величезний набір інструментів та програм. Деякі платні, а інші - безкоштовні та з відкритим кодом, і вибір серед них не простий. Платне рішення не обов'язково пропонує хорошу якість, а безкоштовне рішення не обов'язково погане.

Порівняльна таблиця систем управління конфігураціями ManageEngine Network Configuration Manager та SolarWinds Network Configuration Manager(Таблиця 1):

	ManageEngine Network Configuration Manager	SolarWinds Network Configuration Manager
Мережеве відкриття	+	+
Резервні копії конфігурацій пристроїв	+	+
Контроль доступу через інформаційну панель, що дозволяє адміністраторам визначати, які користувачі мають доступ до якої інформації	-	+
Сповіщення	Електронна пошта, сторінки, текстові мови, пастки SNMP, SMS, запуск зовнішніх програм, скрипти, повідомлення Syslog	Електронна пошта, SMS, Запуск сценаріїв
Функції журналу, запис змін користувачів у налаштування	+	-
Користувацький користувальницький	Реалізація детальна і розгорнута	Реалізація присутня

інтерфейс / панелі інструментів		
Система контролю коректності конфігурацій	-	-
Багатоадресна передача	-	+
Моніторинг маршруту	-	+
Звітність	Вбудована та настроювана звітність	Вбудована та настрюювана звітність

Таблиця 1.

У SolarWinds Network Configuration Manager акцент зроблений на легкість і швидкість процесу встановлення. Під час встановлення Network Configuration Manager проводить сканування системи, щоб автоматично ідентифікувати всі ваші мережеві пристрої, реєструючи знімки їх поточних конфігурацій. Наявність цього журналу може бути корисним для подальшого використання, якщо, наприклад, ви хочете відновити попередню конфігурацію. Присутні гарні особливості роботи з обладнанням cisco.

У ManageEngine Network Configuration Manager акцент зроблено на різні підходи до візуалізації. Також, як і SolarWinds, ManageEngine пропонує набір інструментів для моніторингу ІТ-служб, які можна інтегрувати для створення єдиної системи управління та моніторингу. ManageEngine Network Configuration Manager підтримує середовища для багатьох постачальників і може керувати цілим набором конфігурацій, включаючи маршрутизатори, брандмауери та комутатори.

2.2. Огляд існуючих рішень для створення власної системи управління конфігураціями

Для розробки власної системи управління конфігурацій є 4 популярні рішення: Chef, Puppet, Ansible, та SaltStack

Порівняльна таблиця(Таблиця 2):

	Chef	Puppet	Ansible	SaltStack
Доступність	+	+	+	+
Масштабованість	Високо масштабний	Високо масштабний	Високо масштабний	Високо масштабний
Мова конфігурації	DSL (Ruby)	DSL (PuppetDSL)	YAML (Python)	YAML (Python)
Складність налаштування	Середня	Середня	Легка	Середня
Складність управління	Середня	Середня	Легка	Легка
Підтримка спільноти	Низька	Низька	Велика	Середня
GUI	Opscode Manage	Puppet Enterprise	AWX(Ansible Tower)	SaltStack Enterprise

Архітектура	Клієнт-сервер	Клієнт-сервер	Сервер	Клієнт-сервер
-------------	---------------	---------------	--------	---------------

Таблиця 2.

Доступність

Усі інструменти мають або декілька серверів, або декілька екземплярів(masters), що робить їх високо доступними.

- Chef - якщо на сервері сталася помилка, існує запасний сервер.
- Puppet – реалізована multi-master архітектура. Якщо головний master не відповідає інші master-и беруть його функції на себе.
- Ansible – працює з одним активним вузлом, який називається “Primary instance” (основний екземпляр). У разі відмови, перемикається на інший екземпляр.
- SaltStack – є можливість налаштування кількох masters. Якщо один не працює, система перемикається на інші masters.

Масштабованість

Усі рішення підтримують є досить гарними у підтримці великої інфраструктури. Достатньо вказати IP-адреси та ім'я хостів вузлів, які ви хочете налаштувати, а решта завдання буде оброблятися за допомогою цих інструментів.

Мова конфігурації

DSL(Ruby) та DSL (PuppetDSL) складніші і орієнтовані більш на розробників, в той час як мови YAML більш прості для людського розуміння.

Складність налаштування

Всі рішення окрім Ansible не є досить легкими у налаштуванні і мають master-agent архітектуру, коли сервер знаходиться на master комп'ютері, а клієнти на всіх agent комп'ютерах.

У Ansible присутній master комп'ютер, на якому знаходиться сервер, але на клієнтських комп'ютерах відсутні агенти. Ansible використовує ssh-з'єднання для входу в клієнтські системи або вузли, які ви хочете налаштувати. Клієнтська машина не вимагає спеціальних налаштувань, тому складність налаштувань значно легша.

Складність управління

Chef та Puppet використовують pull налаштування, а Ansible та SaltStack push. Різниця у тому що при налаштуванні pull клієнтські комп'ютери витягують інформацію із сервера самостійно і без додаткових команд, в той час як за налаштування push сервер має відправляти (пушити) конфігурації до клієнтів. Сервери всіх рішень працюють на операційних системах Linux, або Unix, але клієнтські комп'ютери можуть використовувати і Windows.

- Chef – орієнтований на програмістів, конфігурації на мові Ruby DSL. Pull налаштування.
- Puppet – легший для розуміння ніж Chef, використовує мову Puppet DSL.
- Ansible – гарний для роботи в реальному часі, використовує YAML мову, схожу на англійську, досить легку для розуміння.
- SaltStack – як і Ansible легкий для розуміння, також використовує YAML.

Підтримка спільноти

Досить важливий критерій, що впливає на кількість інформації навколо інструменту. Розглянемо інформацію у google(рис.1) та github(рис.2, рис.3, рис.4, рис.5).

Google:

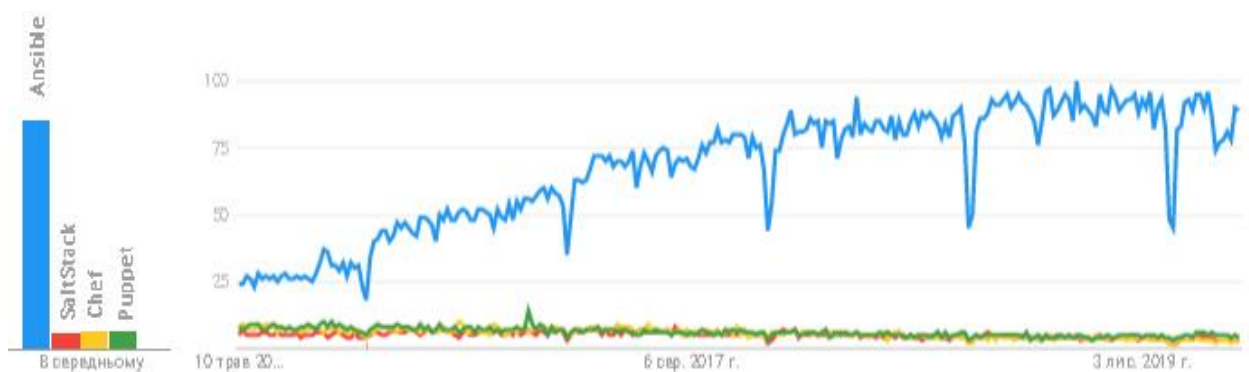



Рис.1 - Графік кількості згадувань Ansible(синій), SaltStack(червоний), Chef(жовтий), Puppet(зелений)

Ansible має набагато більшу поширеність і відповідно підтримку спільноти, порівняно із іншими інструментами.

Github:

Ansible:



Ansible
Ansible is a simple and powerful automation engine.
[See topic](#)

★ Star

120,845 repository results

Sort: Best match ▾


 [ansible/ansible](#)
Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy. Avoid...
[python](#) [ansible](#)
★ 43k ● Python GPL-3.0 license Updated 7 hours ago

Рис.2 - Кількість репозиторіїв із ім'ям “Ansible”, та кількість зірок на основному репозиторії

Chef:

42,883 repository results

Sort: Best match ▾


 [chef/chef](#)
Chef Infra, a powerful automation platform that transforms infrastructure into code automating how infrastructure is ...
[ruby](#) [chef](#) [devops](#) [cfgmgt](#)
★ 6.2k ● Ruby Apache-2.0 license Updated 4 hours ago

Рис.3 - Кількість репозиторіїв із ім'ям “Chef”, та кількість зірок на основному репозиторії

SaltStack:

12,935 repository results

Sort: Best match ▼

 saltstack/salt

Software to automate the management and configuration of any infrastructure or application at scale. Get access to the ...

python event-stream zeromq cloud-providers event-management configuration-management
remote-execution infrastructure-management cloud-management cloud-provisioning

★ 10.8k ● Python Apache-2.0 license Updated 1 hour ago 19 issues need help

Рис.4 - Кількість репозиторіїв із ім'ям "Salt", та кількість зірок на основному репозиторії

Puppet:

43,504 repository results

Sort: Best match ▼

 puppetlabs/puppet

Server automation framework and application

★ 5.7k ● Ruby Apache-2.0 license Updated 3 days ago

Рис.5 - Кількість репозиторіїв із ім'ям "Puppet", та кількість зірок на основному репозиторії

Ansible має набагато більшу аудиторію, порівняно з іншими рішеннями.

2.3. Оркестрація, CloudFormation, Terraform

Як правило, Ansible, Puppet, SaltStack, Chef і SaltStack вважаються інструментами управління конфігураціями та створені для встановлення та управління програмним забезпеченням на існуючих екземплярах сервера (наприклад, встановлення пакетів, запуск послуг, встановлення скриптів або конфігураційних файлів на екземпляр). Вони змушують один чи кілька екземплярів виконувати свої ролі, не потребуючи від користувача вводити команди.

Оркестрація

Такі інструменти, як Terraform та CloudFormation вважаються оркестраторами. Вони призначені для надання серверам самих екземплярів, залишаючи завдання налаштування цих серверів на інструменти управління конфігураціями. Оркестрація забезпечує середовища на більш високому рівні, ніж управління конфігурацією. Основна увага приділяється координації конфігурації у складних середовищах.

Порівняльна таблиця інструментом оркестрації CloudFormation з інструментом оркестрації Terraform та інструментом управління конфігураціями Ansible(Таблиця 3):

	CloudFormation	Terraform	Ansible
Відкритий код (Доступність)	-	+	+
Тип	Інструмент оркестрації конфігурацій	Інструмент оркестрації конфігурацій	Інструмент управління конфігураціями
Мова	Декларативна	Декларативна	Процедурна
Синтаксис	JSON	HCL/JSON	YAML (python)
Підтримка спільноти	Низька	Велика	Велика
Підтримка хмарний технологій	Тільки AWS(Amazon Web Services)	Всі	Всі
Архітектура	Клієнт-сервер	Сервер	Сервер

Таблиця 3.

Загалом, оркеастрація включає у себе управління конфігураціями. Ansible можна поєднувати із CloudFormation та Terraform, тому вони не є конкурентами, а існують для вирішення різних задач.

По мірі зростання складності хмарних технологій, потреба у засобах оркестрації буде збільшуватись. Але для системи управління конфігураціями пристроїв у мережі підприємств доцільніше використовувати інструменти управління конфігураціями.

2.4. Jenkins, Ansible AWX

Jenkins

Jenkins - це інструмент автоматизації з відкритим кодом, написаний на Java, з плагінами, побудованими для цілі постійної інтеграції. Jenkins використовується для безперервного створення та тестування ваших програмних проектів, що полегшує розробникам інтеграцію змін у проект і полегшує користувачам свіжу конструкцію. Це також дозволяє безперервно поставляти програмне забезпечення шляхом інтеграції з великою кількістю технологій тестування та розгортання.

З Jenkins організації можуть прискорити процес розробки програмного забезпечення за допомогою автоматизації. Jenkins інтегрує процеси життєвого циклу розвитку всіх видів, включаючи складання, документування, тест, пакет, етап, розгортання, статичний аналіз та багато іншого.

Ansible AWX

AWX - це веб-рішення, яке робить Ansible простішим у використанні, даючи йому веб інтерфейс. Він розроблений як центр для всіх завдань з автоматизації. Він має API REST, що дозволяє переглядати доступ, графічно керувати або синхронізувати рекламні ресурси з різним хмарними джерелами, реєструвати всі ваші завдання та добре інтегруватися з протоколом легкого доступу до каталогу (LDAP). AWX дозволяє керувати Ansible playbook-ами, inventories та планувати завдання, які потрібно запускати за допомогою веб-інтерфейсу.

Два інструменти можна використовувати разом через AWX API. Але для виконання даної роботи цілком достатньо функціоналу Ansible AWX.

2.5. Висновок

За сукупністю характеристик, розглянутих у порівняльному аналізі, Ansible є оптимальним вибором платформи автоматизації. І у якості веб-рішення для легшого управління Ansible було обрано Ansible AWX.

3. Структурна розробка власного рішення

Задача: створення нової системи шляхом інтеграції платформи для автоматизації, веб-інтерфейсу, повідомлень, планувальника та оригінального скриптингу із додаванням функціональності контролю коректності конфігурацій.

3.1. Структурна схема власного рішення,

Схема реалізації інтерфейсу адміністратора, звітування і планувальника через AWX(Рисунок 6).

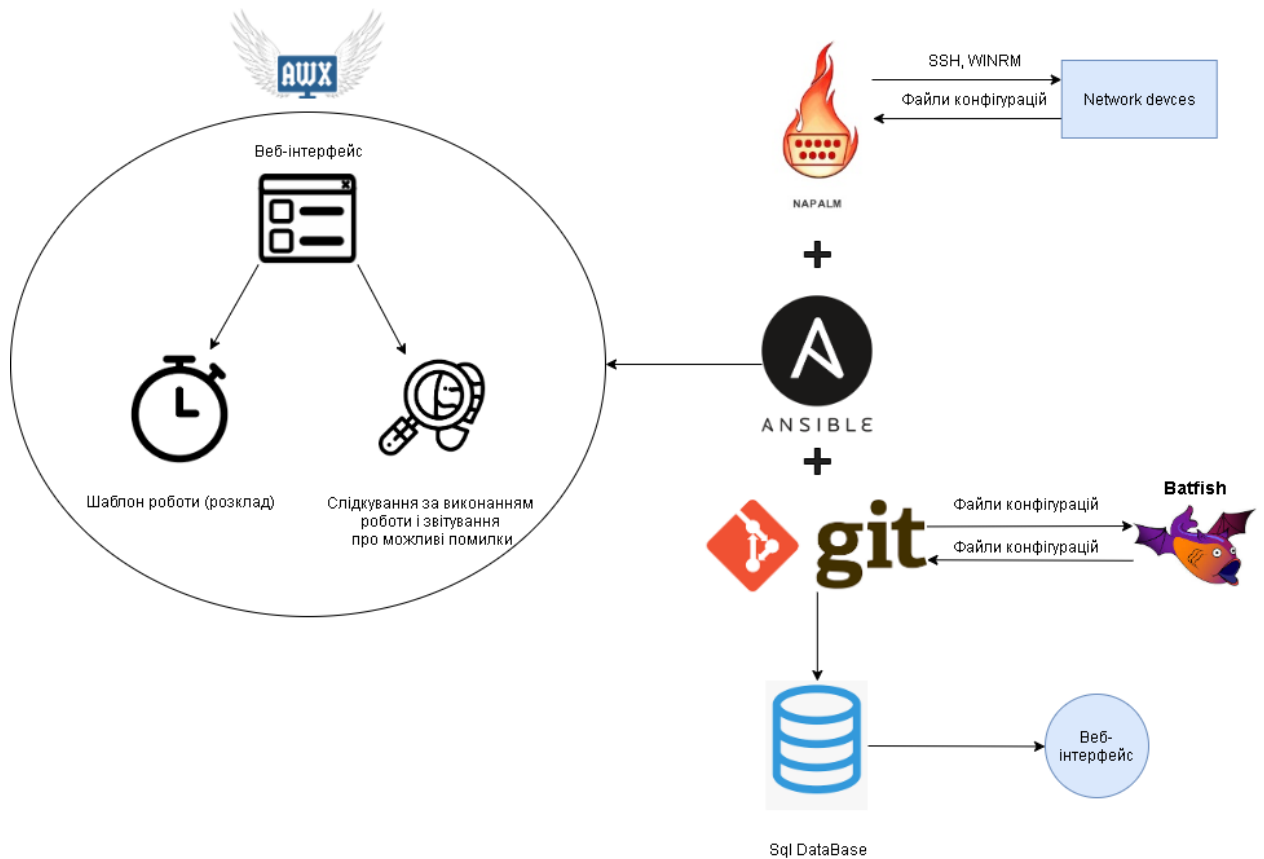


Рисунок 6.

3.2. Перелік компонентів, їх характеристики, опис

Веб інтерфейс AWX

AWX використовується як веб-інтерфейс, що виконує функції делегування, сповіщення (зворотного зв'язку про помилки і т.д.) і розклад робочого процесу.

Вигляд веб-інтерфейсу(Рисунок 7):

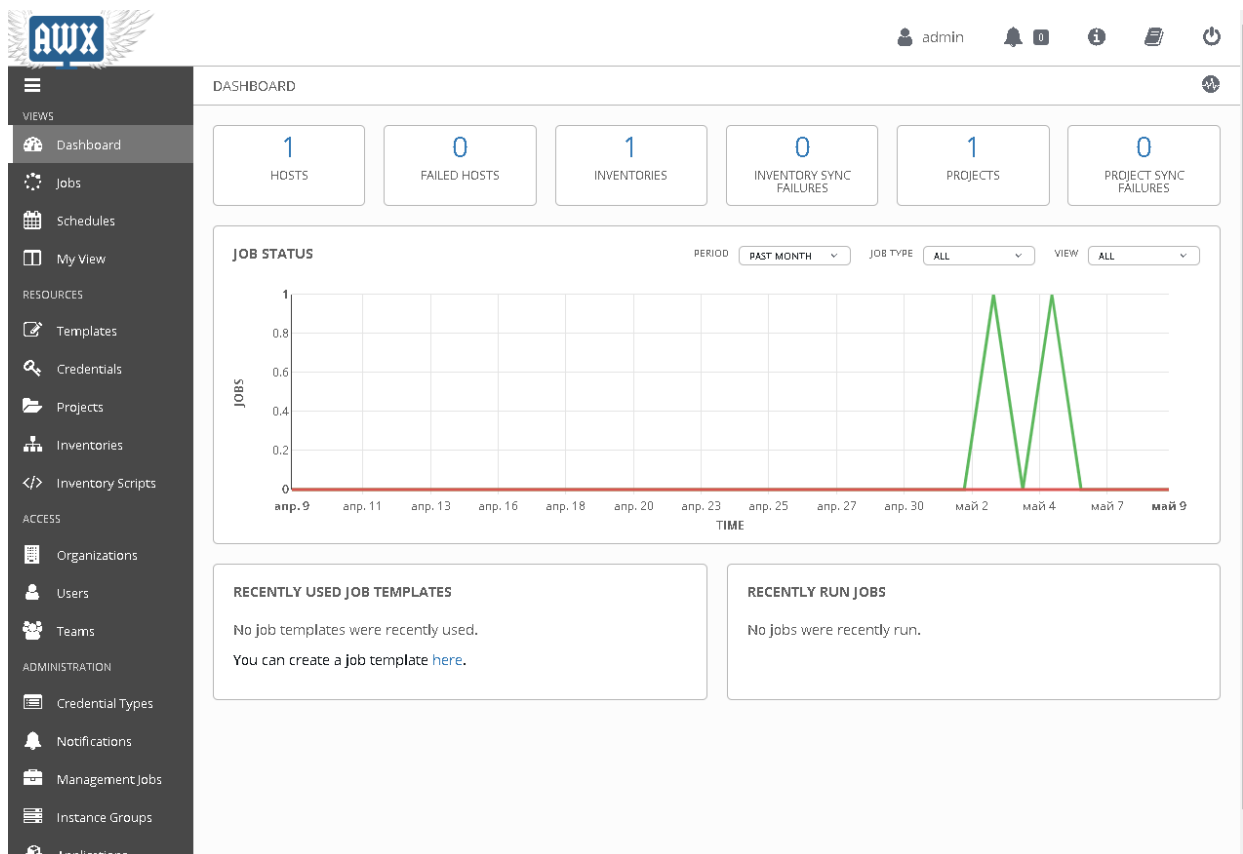


Рисунок 7.

Делегування

Основною функцією AWX є можливість створювати користувачів та групувати їх у команди. Після чого ви можете призначити доступ та правила для інвентарю, облікових даних та ігрових книг на індивідуальному рівні або на рівні команди. Це дає можливість налаштувати доступ до складної автоматизації та контролювати, хто може ним користуватися та де він може запускати.

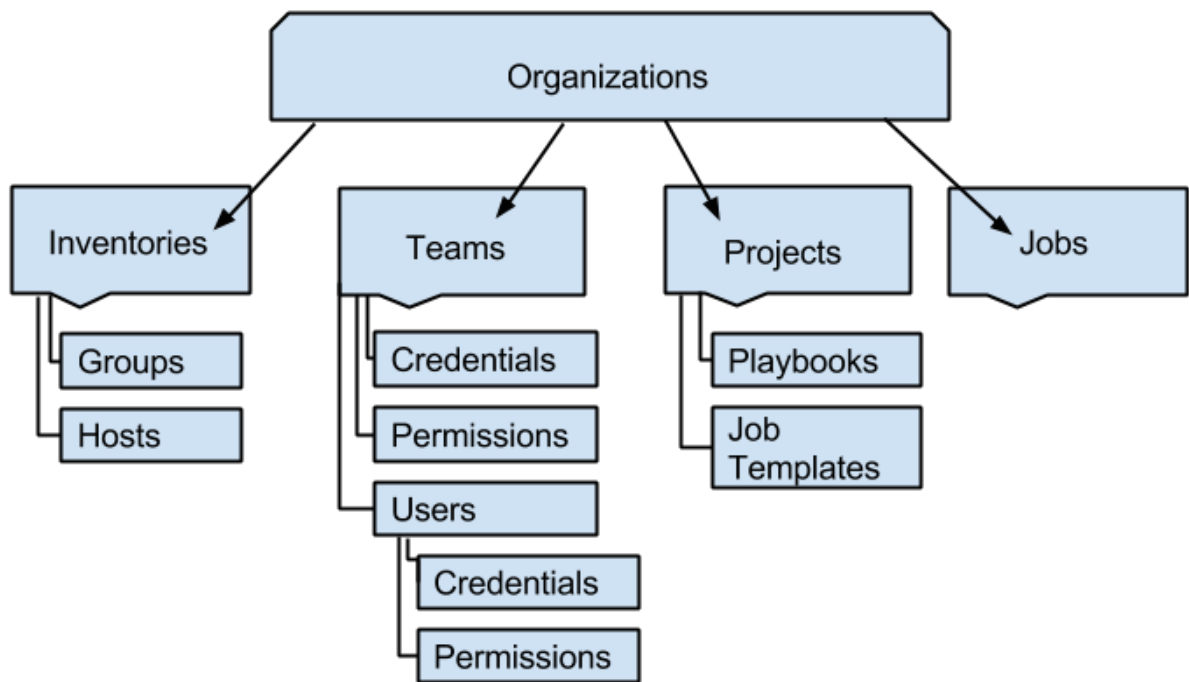


Рисунок 8. Ієрархія об'єктів в AWX

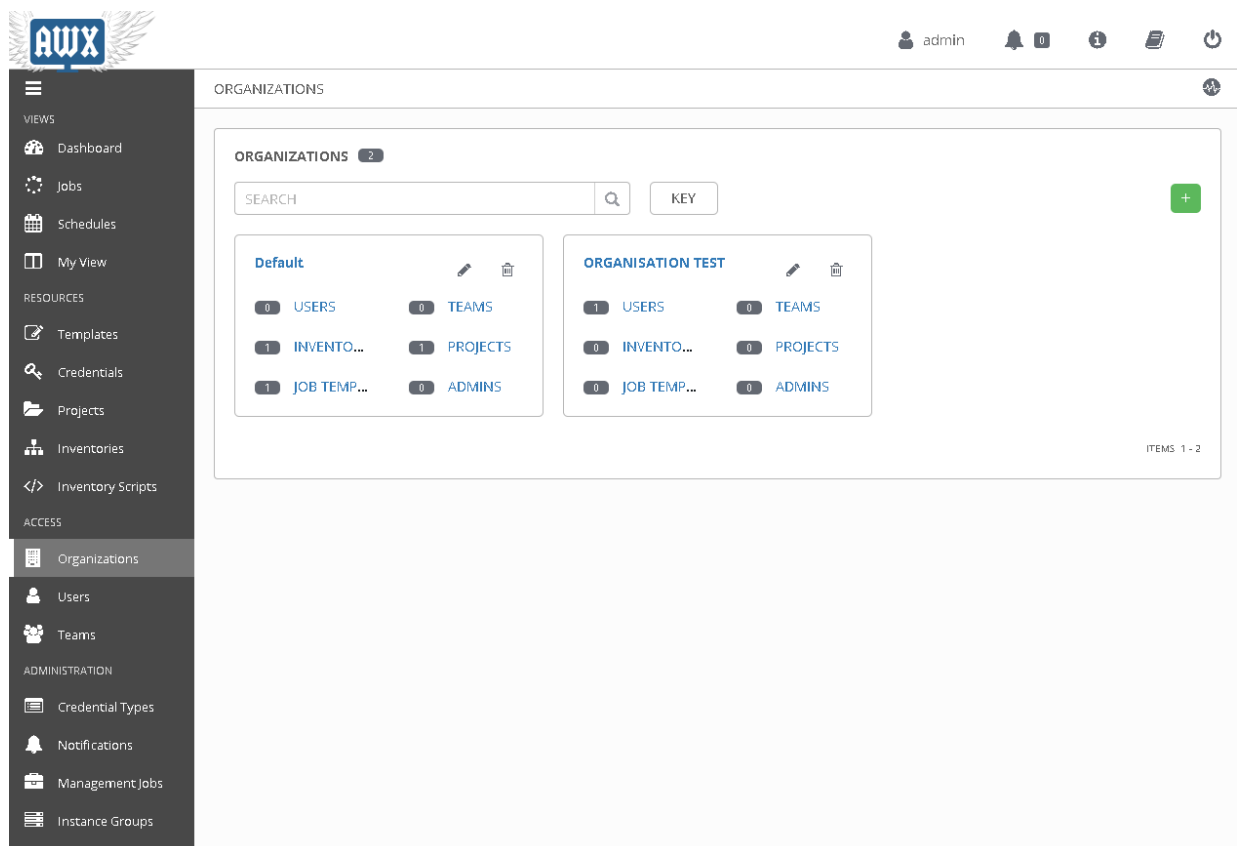


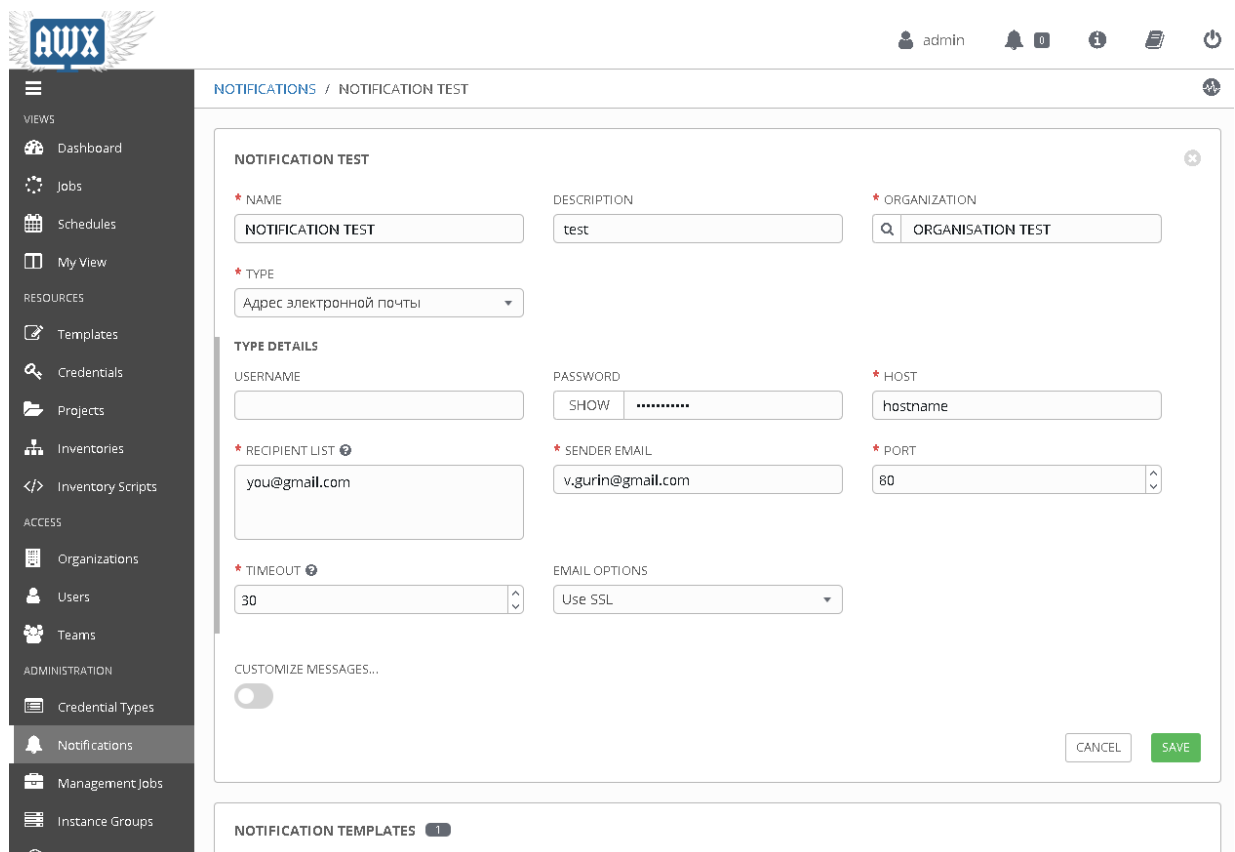
Рисунок 9. Створення нової організації і додавання користувача admin до неї в AWX

Сповіщення(Повідомлення зворотного зв'язку)

AWX додає користувацький плагін зворотного виклику в програму Ansible playbook, яка фіксує події та виводить дані в режимі реального часу. На інформаційній панелі представлений підсумок високого рівня збереженого виводу, який забезпечує огляд виконання завдань, збоїв та успіхів, а також розбиття успіхів та збоїв у інвентарі. Можна одразу дізнатися, які playbook-и запуснені та проблемні місця, які потребують уваги, з деталями.

На рівні playbook є можливість отримувати доступ до результатів і виводити окремі завдання на певному хості. Загалом, швидкість усунування проблем значно пришвидшується.

Коли робота проходить успішно або не вдається, обробник помилок або успіх витягне список відповідних шаблонів сповіщень, використовуючи описану вище процедуру. Потім буде створено об'єкт сповіщень для кожного, який містить відповідні деталі завдання, а потім надсилатиме його до місця призначення (адреси електронної пошти, SMS-номери тощо).



The screenshot displays the AWX web interface. On the left is a dark sidebar with a menu containing sections like VIEWS, RESOURCES, ACCESS, and ADMINISTRATION. The 'Notifications' option is highlighted. The main content area is titled 'NOTIFICATIONS / NOTIFICATION TEST'. It features a form for configuring a notification test. The form includes fields for NAME (set to 'NOTIFICATION TEST'), DESCRIPTION (set to 'test'), and ORGANIZATION (set to 'ORGANISATION TEST'). There is a dropdown for TYPE (set to 'Адрес электронной почты'). Below this is a 'TYPE DETAILS' section with fields for USERNAME, PASSWORD (with a 'SHOW' toggle), HOST (set to 'hostname'), RECIPIENT LIST (set to 'you@gmail.com'), SENDER EMAIL (set to 'v.gurin@gmail.com'), PORT (set to '80'), TIMEOUT (set to '30'), and EMAIL OPTIONS (set to 'Use SSL'). At the bottom of the form is a 'CUSTOMIZE MESSAGES...' toggle switch. 'CANCEL' and 'SAVE' buttons are located at the bottom right of the form. Below the form, there is a section for 'NOTIFICATION TEMPLATES' with a toggle switch.

Рисунок 10. Створення повідомлення через електронну пошту.

Є можливість налаштовувати текстовий вміст кожного з типів сповіщень(початкове повідомлення, повідомлення помилки, повідомлення успіху і т.д.).

CUSTOMIZE MESSAGES...

Use custom messages to change the content of notifications sent when a job starts, succeeds, or fails. Use curly braces to access information about the job: `{{ job_friendly_name }}`, `{{ url }}`, or attributes of the job such as `{{ job.status }}`. You may apply a number of possible variables in the message. Refer to the [Ansible Tower documentation](#) for more details.

START MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

START MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

SUCCESS MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

SUCCESS MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

ERROR MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

ERROR MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

WORKFLOW APPROVED MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

WORKFLOW APPROVED MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

WORKFLOW DENIED MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

WORKFLOW DENIED MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

WORKFLOW RUNNING MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

WORKFLOW RUNNING MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

WORKFLOW TIMED OUT MESSAGE

1 `{{ job_friendly_name }}` `#{{ job.id }}` `'{{ job.name }}'` `{{ job.status }}`: `{{ url }}`

WORKFLOW TIMED OUT MESSAGE BODY

1 `{{ job_friendly_name }}` `#{{ job.id }}` had status `{{ job.status }}`, view details at `{{ url }}`
2
3 `{{ job_metadata }}`

CANCEL

SAVE

Рисунок 11. Налаштування кожного типу помилки у AWX.

Розклад робочого процесу

Створивши шаблон завдання можна зібрати облікові данні, playbook-и, та inventory у купу. Шаблон представляє виконання командного рядка "ansible-playbook", за винятком того, що нема потреби користуватися командним рядком. Натомість можна запускати шаблон завдання за потреби та дивитись вихідну playbook в реальному часі у веб-браузері. Або запланувати запуск пізніше або на періодичній основі та отримати повний доступ до результатів, коли це потрібно.

Також до запуску одного playbook, можна створити шаблон завдання за допомогою редактора робочих процесів, який поєднує в собі кілька запусків playbook-ів.

The screenshot displays the 'Demo Job Template' configuration page in the AWX web interface. The sidebar on the left contains navigation links for Views (Dashboard, Jobs, Schedules, My View) and Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts, Organizations, Users, Teams, Administration, Credential Types, Notifications, Management Jobs, Instance Groups, Applications). The main content area shows the configuration for the 'Demo Job Template'. The 'PLAYBOOK' field is set to 'hello_world.yml'. Other fields include NAME (Demo Job Template), DESCRIPTION, JOB TYPE (Run), INVENTORY (Demo Inventory), PROJECT (Demo Project), CREDENTIALS (Demo Credential), FORKS (0), LIMIT, VERBOSITY (0 (Normal)), JOB TAGS, SKIP TAGS, LABELS, INSTANCE GROUPS, JOB SLICING (1), TIMEOUT (0), SHOW CHANGES (checked), and OPTIONS (ENABLE PRIVILEGE ESCALATION, ENABLE PROVISIONING CALLBACKS, ENABLE WEBHOOK, ENABLE CONCURRENT JOBS, ENABLE FACT CACHE). The 'EXTRA VARIABLES' field is set to '1'.

Рисунок 12. Створення шаблону роботи playbook-y hello_world.yml.

AWX

admin

TEMPLATES / Demo Job Template / SCHEDULES

Demo Job Template

DETAILS PERMISSIONS NOTIFICATIONS COMPLETED JOBS **SCHEDULES**

SEARCH KEY

NAME	FIRST RUN	NEXT RUN	FINAL RUN	ACTIONS
<input checked="" type="checkbox"/> TEST	11.5.2020 12:03:03	11.5.2020 12:03:03	27.5.2020 12:03:03	

ITEMS 1 - 1

TEMPLATES 1

SEARCH KEY

Compact Expanded Name (Ascending)

Demo Job Template	Job Template		
--------------------------	--------------	--	--

ITEMS 1 - 1

Рисунок 13. Додавання розкладу виконання шаблону роботи (кожні 4 дні від 11.05.2020 до 27.05.2020).

SCHEDULED JOBS 5

SEARCH KEY

NAME	TYPE	NEXT RUN	ACTIONS
<input checked="" type="checkbox"/> TEST	Playbook Run	11.5.2020 12:03:03	
<input checked="" type="checkbox"/> Cleanup Activity Schedule	Management Job	12.5.2020 15:40:12	
<input checked="" type="checkbox"/> Cleanup Job Schedule	Management Job	10.5.2020 15:40:12	
<input checked="" type="checkbox"/> Cleanup Expired OAuth 2 Tokens	Management Job		
<input checked="" type="checkbox"/> Cleanup Expired Sessions	Management Job		

ITEMS 1 - 5

Рисунок 14. Список всіх розкладів робіт у графі Schedules.

Репозиторій конфігурацій на основі системи контролю версій Git

На сьогодні найбільш широко застосовуваною в світі сучасною системою управління версіями є Git. Git активно підтримуваний проект з відкритим кодом. Величезна кількість програмних проектів покладається на Git для контролю версій, включаючи комерційні проекти, а також відкритий код. Розробники, які співпрацювали з Git, добре представлені в наборі наявних талантів щодо розробки програмного забезпечення, і він добре працює в широкому діапазоні операційних систем та IDE.

Маючи розподілену архітектуру, Git є прикладом DVCS (розподіленої системи управління версіями). Замість того, щоб мати лише одне місце для повної історії версій програмного забезпечення, як це часто зустрічається в колись популярних системах контролю версій, таких як CVS або Subversion (також відомий як SVN), в Git робоча копія коду кожного розробника є також сховищем, які можуть містити повну історію всіх змін.

Засіб аналізу конфігурації мережі Batfish

Batfish виявляє помилки та гарантує правильність запланованих чи поточних конфігурацій мережі. Це дозволяє безпечно та швидко розвивати мережу, не боячись відключень чи порушень безпеки.

Конфігурації мережі упаковуються і подаються Batfish за допомогою snapshot-ів. Знімок - це сукупність інформації (файли конфігурації, дані про маршрутизацію, стан вузлів та посилань вгору / вниз), які представляють стан мережі. Тому Batfish не вимагає прямого доступу до мережевих пристроїв.

Batfish працює як контейнерна послуга, з даних, що надходять через snapshot. Контейнер - це стандартний блок програмного забезпечення, який пакує код та всі його залежності, тому програма швидко та надійно працює з одного обчислювального середовища в інше. Batfish будує серію моделей, які

потім запитуються або використовуючи SDK python - pybatfish або Batfish Ansible role.

Платформа автоматизації Ansible

Ansible це програмне забезпечення з відкритим кодом, яке автоматизує забезпечення програмного забезпечення, управління конфігурацією та розгортанням додатків, засноване на архітектурі без агенту. Хостами керує автомат управління Ansible через SSH.

Протягом останніх років Ansible став надзвичайно популярним у світі мережевої автоматизації. Все зводиться до Ansible. Основні компоненти Ansible:

- Controller Machine (Серверний комп'ютер)
Тут встановлюється Ansible, звідси хости керуються через SSH
- Inventory (Інвентар)
Містить кожен хост, яким потрібно керувати.
- Playbook
YAML файли, що описують дії, які ми хочемо виконати. Ці дії будуються на основі наступної ієрархії(Рисунок 8):
 - Play
Містить набір завдань.
 - Task (Завдання)
Містить набір модулів.
 - Modules (Модулі)
Модулі управляють системними ресурсами, такими як служби, пакети, файли або виконують системні команди. Виконує вказані дії

```

- name: PLAY 1 - FABRIC TEST
  hosts: all
  gather_facts: no

  vars:
    nxos_ssh:
      host: "{{ ansible_host }}"
      username: "{{ user }}"
      password: "{{ pw }}"
      transport: cli

  tasks:
    - name: "Check Connectivity 1 (ping)"
      nxos_ping:
        provider: "{{ nxos_ssh }}"
        source: 192.168.1.1
        vrf: default

- name: PLAY 2 - FABRIC CONFIGURE
  hosts: all
  gather_facts: no

...

```

Playbook

Play

Task

Module

Рисунок 15. Компоненти Ansible: Playbook(сірий), Play(синій), Task(зелений) Module(яскраво зелений).

У розробці власного додатку в Ansible буде використана python бібліотека NAPALM (Network Automation and Programmability Abstraction Layer with Multivendor support) яка реалізує набір функцій для взаємодії з різними мережевими операційними системами за допомогою уніфікованого API.

Веб-інтерфейс

Реалізація веб-інтерфейсу(Рисунок 16).

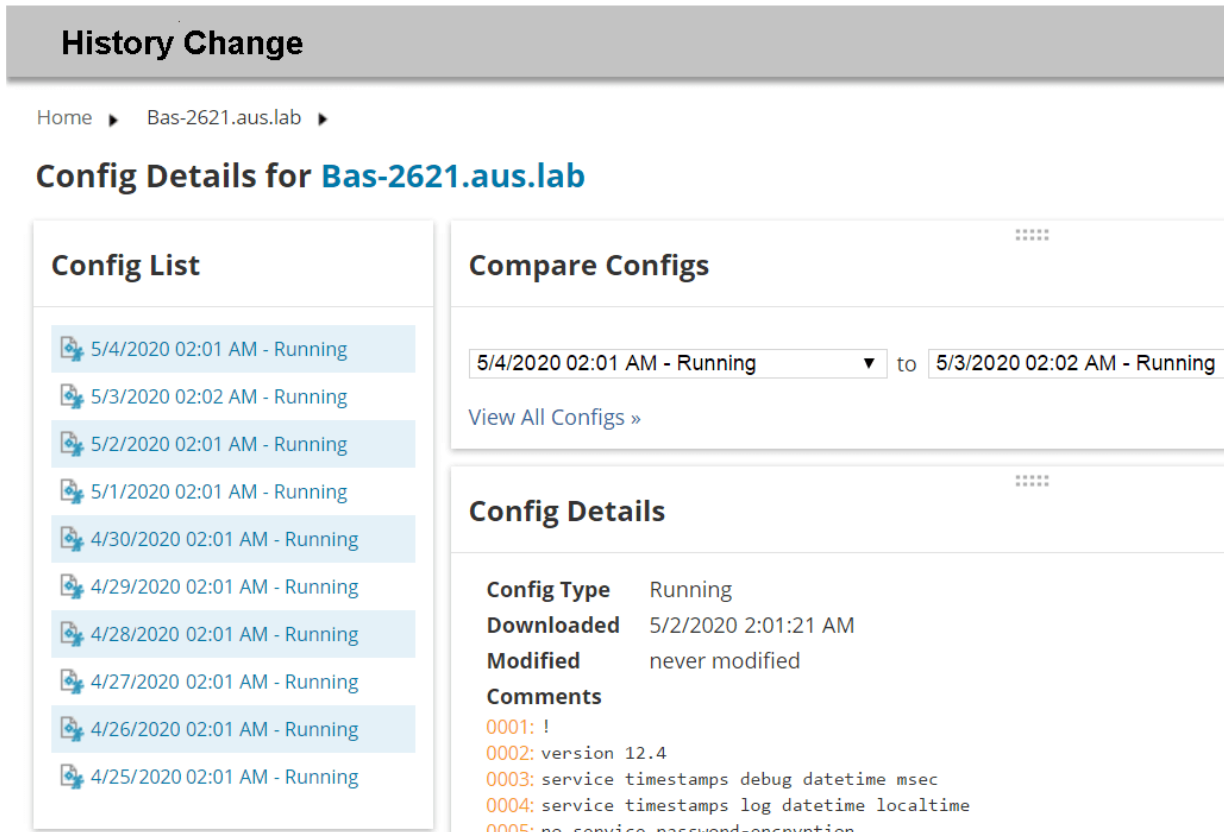


Рисунок 16.

3.3. Опис функціонування системи

- Git використовується в якості контролю змін, щоб відстежувати зміни в конфігурації, які відбуваються
- NAPALM модуль використовується для того щоб запустити зміни на пристрої мережі, які потребують змін.

- Використання NAPALM дозволить зафіксувати конфігурації цільових пристроїв перед зміною і запустити призначену конфігурацію для цих пристроїв
- Зміни не будуть вноситися до пристроїв одразу ж, натомість інформація про зміни та нова конфігурація для пристроїв зберігається у нову гілку Git, щоб розпочати етап перевірки цих змін через Batfish, таким чином цей крок може бути виконаний у нормальний робочий час, оскільки це не впливає на мережу.
- Після того як зміни підтверджені, зміни зберігаються в master гілку
- Зміни вносяться до бази даних
- Зміни копіюються із master гілки до пристроїв у мережі і конфігурації застосовуються.

Схема функціонування отримання конфігурацій від пристроїв, історії змін конфігурацій, перевірка коректності конфігурацій і завантаження конфігурацій на пристрої мережі(Рисунок 17) .

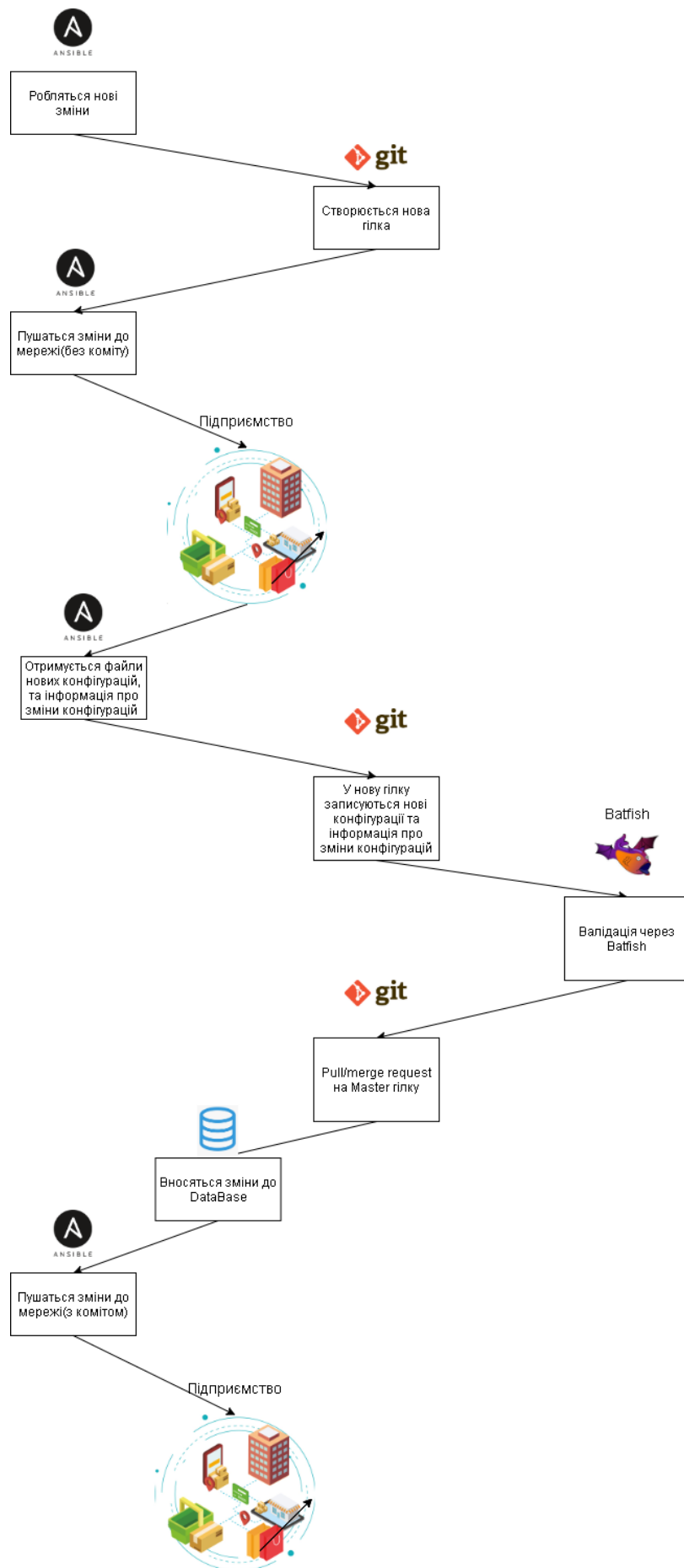


Рисунок 17.

Схема функціонування інтерфейсу адміністратора, звітування і планувальника через AWX(Рисунок 18).

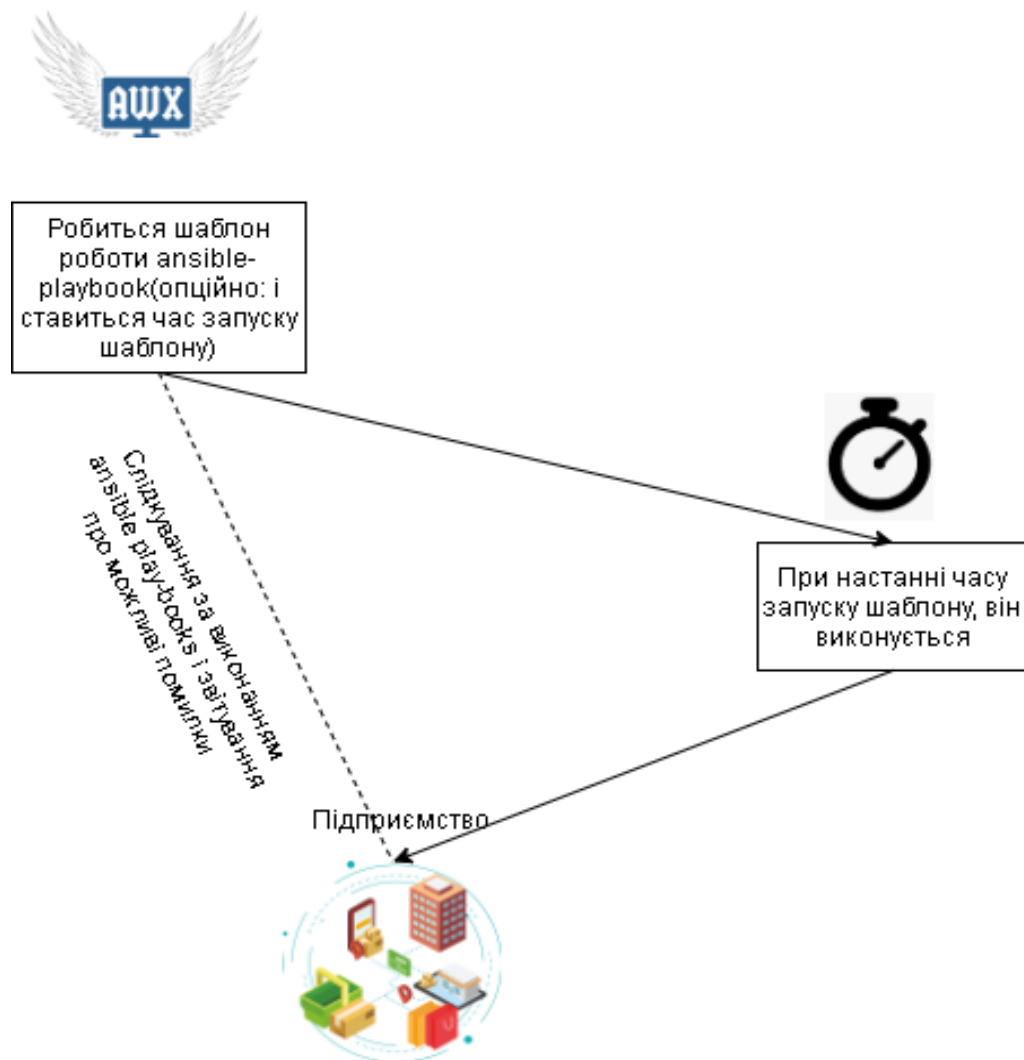


Рисунок 18.

4. Детальна розробка компонента

Виконана розробка вказаного елемента системи(Рисунок 19)



Рисунок 19.

Код

Сотворений код (Додаток А).

Пояснення до коду

Створення папок для роботи із конфігураціями data/configs – папка для зберігання всіх конфігурацій, diff та cand – папки для роботи із інформацією про зміни конфігурацій та резервної копії існуючих конфігурацій відповідно(Рисунок 20).

```
name: Create data directory
file: path=../data state=directory
-
name: Create configs directory
file: path=../data/configs state=directory
-
name: Create diff directory
file: path=../compiled/diff state=directory
-
name: Create candidate directory
file: path=../compiled/cand state=directory
delegate_to: localhost
run_once: yes
```

Рисунок 20.

Встановлення конфігурацій на пристрій, де config_file – шлях до файлу звідки будуть братися нові конфігурації для встановлення, diff_file - шлях до файлу, де зберігаються "відмінності" між запущеною конфігурацією та новою конфігурацією і candidate_file – шлях до файлу, що зберігає резервну копію конфігурації з пристрою(Рисунок 21).

```
name: Push new config and get candidate(backup) config with NAPALM
napalm_install_config:
  hostname: '{{ inventory_hostname }}'
  username: '{{ user }}'
  dev_os: '{{ os }}'
  password: '{{ password }}'
  config_file: '../compiled/{{ inventory_hostname }}/running.conf'
  commit_changes: '{{commit| default('no')}}'
  candidate_file: '../compiled/cand/{{ inventory_hostname }}.running.conf'
  diff_file: '../compiled/diff/{{inventory_hostname}}.running.conf'
```

Рисунок 21.

Копіювання файлу, що зберігає резервну копію конфігурації з пристрою у папку data/configs, якій зберігаються всі конфігурації.

```
name: Copy candidate to data folder
copy:
  content: '{{lookup('file','../compiled/cand/{{ inventory_hostname }}.running.conf')}}'
  dest: '../data/configs/{{ inventory_hostname }}.running.conf'
```

Рисунок 22.

Видалення папки cand, оскільки вона більше не потрібна (Рисунок 23).

```
name: Remove first candidate Folder
file: path=../compiled/cand state=absent
```

Рисунок 23.

Створений шаблон роботи системи за розкладом (Рисунок 24).

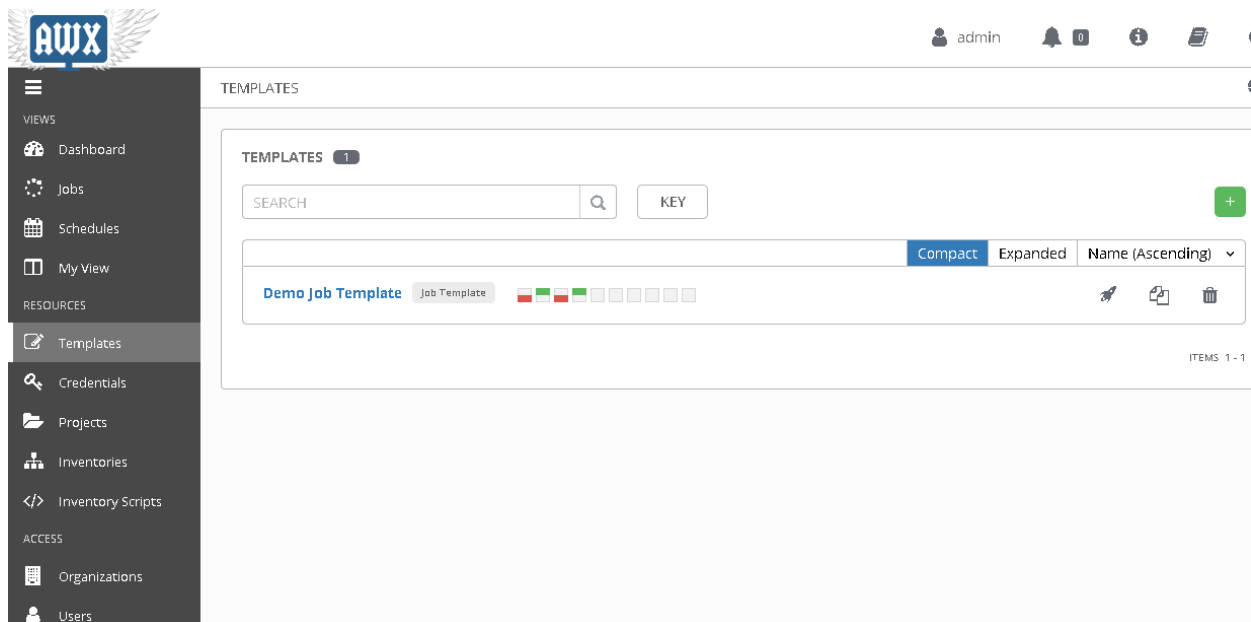


Рисунок 24.

Висновок

Розроблено систему управління конфігураціями зі складовими компонентами:

- репозиторій з системою контролю версій (git)
- компоненти для доступу до пристроїв
 - системи автоматизації Ansible + NAPALM
- інтерфейс адміністратора (AWX)
- планувальник (AWX)
- система контролю коректності конфігурацій (Batfish)
- система повідомлень (AWX)
- SQL База даних та інтерфейс для перегляду змін конфігурацій (власний веб-інтерфейс)

Виконана мета системи управління

- швидке відновлення конфігурації у випадку заміни пристроїв, що відмовили - викнано
- відновлення у разі пошкодження конфігурації внаслідок помилки – викнано
- історичний огляд змін в конфігураціях - викнано
- перевірка коректності і консистентності конфігурацій – викнано
- повідомлення про помилки під час виконання - викнано

Список літератури

1. 10 Best Network Configuration Management Tools. [Електронний ресурс]: dnsstuff.com – 2019. – Режим доступу: <https://www.dnsstuff.com/network-configuration-management-tools>
2. Chef vs Puppet vs Ansible vs Saltstack: Which Works Best For You? [Електронний ресурс]: edureka.co – 2019. – Режим доступу: <https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/>
3. Puppet vs. Chef vs. Ansible vs. SaltStack [Електронний ресурс]: jetpatch.com – 2016. – Режим доступу: <https://jetpatch.com/blog/agent-management/puppet-vs-chef-vs-ansible-vs-saltstack/>
4. Навчальні матеріали для роботи з Git. [Електронний ресурс]. – Режим доступу: <https://www.atlassian.com/git/tutorials/what-is-git>
5. Документація Batfish. [Електронний ресурс]. – Режим доступу: <https://www.batfish.org/>
6. A Hands-on Guide to Multi-Tiered Firewall Changes with Ansible and Batfish (Part 1). [Електронний ресурс]: packetflow.co.uk – 2019. – Режим доступу: <https://www.packetflow.co.uk/a-hands-on-guide-to-multi-tiered-firewall-changes-with-ansible-and-batfish-part-1/>
7. Infrastructure as Code: Chef, Ansible, Puppet, or Terraform? [Електронний ресурс]: ibm.com – 2018. – Режим доступу: <https://www.ibm.com/cloud/blog/chef-ansible-puppet-terraform>
8. Tracking and Validating Configuration Deployment with NAPALM, Ansible and Git [Електронний ресурс]: agileintegratedsolutions.com – Режим доступу: <https://www.agileintegratedsolutions.com/>
9. NAPALM library Documentation [Електронний ресурс]. – Режим доступу: <https://napalm.readthedocs.io/en/latest>
10. Ansible Documentation [Електронний ресурс]. – Режим доступу: <https://docs.ansible.com/>

5. Додаток А

(ДОВІДНИКОВИЙ)

#Playbook for awx

- name: NAPAM Anbile work

hosts: all

tasks:

-

name: Create data directory

file: path=../data state=directory

-

name: Create configs directory

file: path=../data/configs state=directory

-

name: Create diff directory

file: path=../compiled/diff state=directory

-

name: Create candidate directory

file: path=../compiled/cand state=directory

delegate_to: localhost

run_once: yes

-

name: Push new config and get candidate(backup) config with NAPALM

napalm_install_config:

hostname: '{{ inventory_hostname }}'

username: '{{ user }}'

dev_os: '{{ os }}'

password: '{{ password }}'

config_file: '../compiled/{{ inventory_hostname
}}/running.conf'

commit_changes: '{{ commit| default('no') }}'

candidate_file: '../compiled/cand/{{ inventory_hostname
}}.running.conf'

diff_file:

'../compiled/diff/{{ inventory_hostname }}.running.conf'

-

name: Copy candidate to data folder

copy:

content: '{{ lookup('file','../compiled//cand/{{ inventory_hostname
}}.running.conf') }}'

dest: '../data/configs/{{ inventory_hostname }}.running.conf'

-

name: Remove first candidate Folder

file: path=../compiled/cand state=absent