Network Backups & Restore with Ansible

Rohit Thakur
Principal Engineer, Ansible Networking
GitHub: rohitthakur2590
Matrix: @rothakur:ansible.im

ANSIBLE

Agenda

- Introduction
- Why Network Backups Are Hard?
- Solution Overview
- Architecture Overview
- Idempotent Backups
- Vendor Neutral Designs
- Diff Severity Scoring
- Backup Verification
- SCM Integration
- Q&A
- Key Takeaways

Introduction

Here's what we'll cover today:

Problem Statement

- Discuss challenges with traditional network backup approaches
- Then I'll show you our solution architecture

Key Features

- Dive into idempotent backups & vendor-neutral design
- Demonstrate diff severity scoring with rules & ML enhancement
- Cover SHA-256 hash verification for backup integrity

Live Demo

- See everything working together in a live demonstration

The Problem: Why Network Backups Are Hard

Challenge 1: Vendor Lock-in

- Every vendor has different backup methods
- Cisco IOS uses different commands than Juniper Junos
- NX-OS, EOS, IOS-XR all have their own quirks
- This means maintaining separate scripts for each platform

Challenge 2: False Positives

- Timestamps change on every backup, even if config hasn't changed
- Metadata differences create noise in version control
- This leads to unnecessary commits and PRs

Challenge 3: No Change Prioritization

- All changes look the same – a BGP change is treated the same as a description change
- No way to know if a change is critical or low-risk
- Everything requires manual review

Challenge 4: Backup Integrity

- How do you know if a backup file is corrupted?
- What if someone tampered with the backup?
- No verification before restore operations

Solution Overview

Core Principles:

Idempotent:

- Only backup when actual configuration changes occur

Vendor-Neutral:

- Single playbook works across all platforms

Vendor-Neutral:

- Single playbook works across all platfforms

Intelligent:

- Automatic severity scoring to prioritize changes

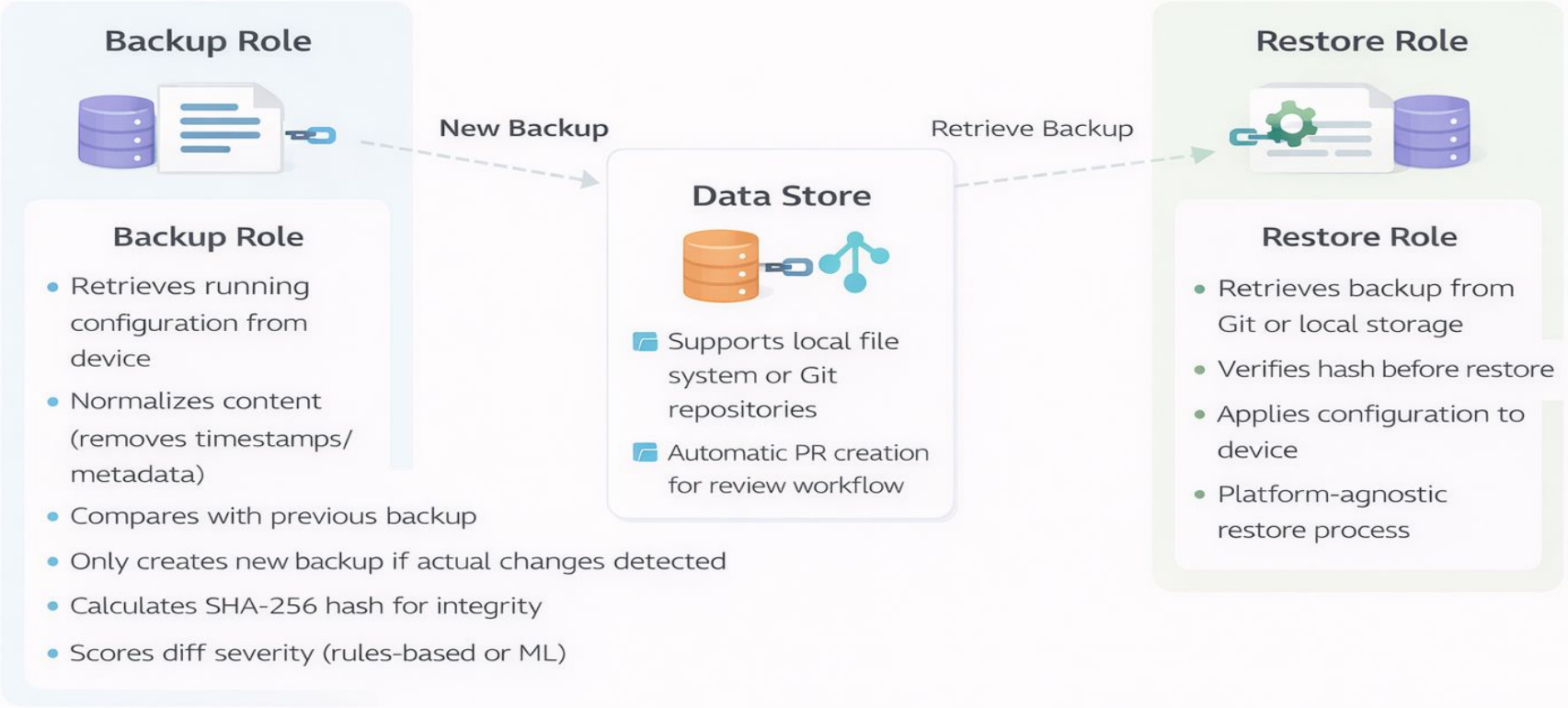
Secure:

- Hash verification ensures backup integrity

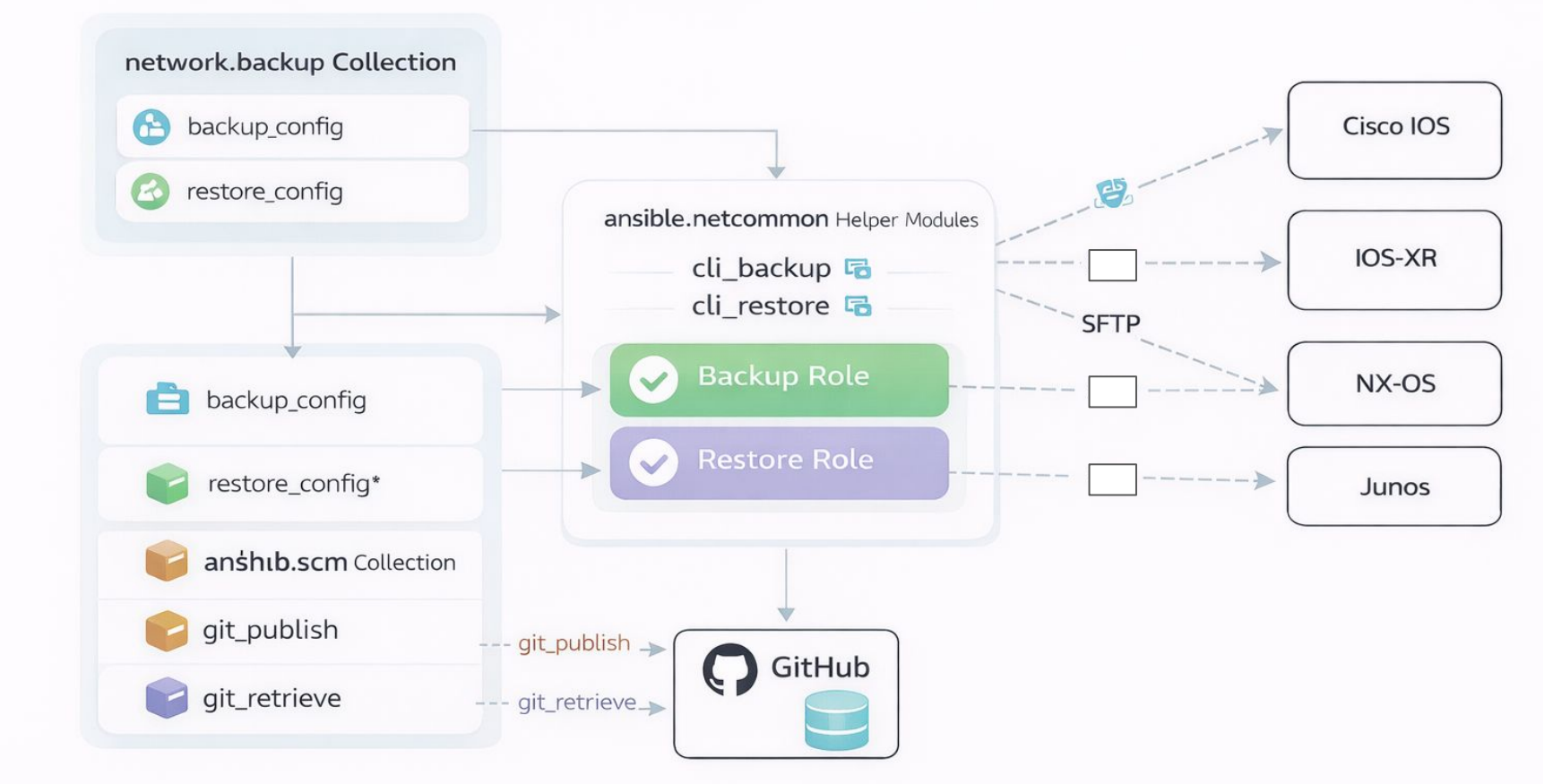
- Built on Ansible's **network.backup** collection
- Works with IOS, IOS-XR, NX-OS, EOS, Junos out of the box
- Integrates with **Git** for version control and collaboration

Architecture Overview (High Level)

Here's how it all fits together:



Collection and Module Architecture



Key Features At Glance

Idempotent Backups
Only backup when config actually changes

Vendor-Neutral
One playbook, all platforms

Diff Severity Scoring
Automatic risk assessment

SHA-256 Hash Verification
Backup integrity guarantee

Git Integration
Version control and collaboration

ML-Enhanced Scoring
Optional ML model for complex scenarios

Idempotent Backups

Let's start with idempotent backups – this is foundational.

The Problem:

- Traditional backups create a file every time, even if nothing changed
- Timestamps, metadata create false positives
- Git history becomes noisy with empty commits

Our Solution:

- Normalization **process** removes timestamps and metadata
- Only actual configuration content is compared
- If normalized content is identical → skip backup

Normalization Examples:

- Removes: `!Time: 2026-02-01 14:30:25`
- Removes: `!Command: show running-config`
- Removes: `!No configuration change since last restart`
- Keeps: Actual configuration

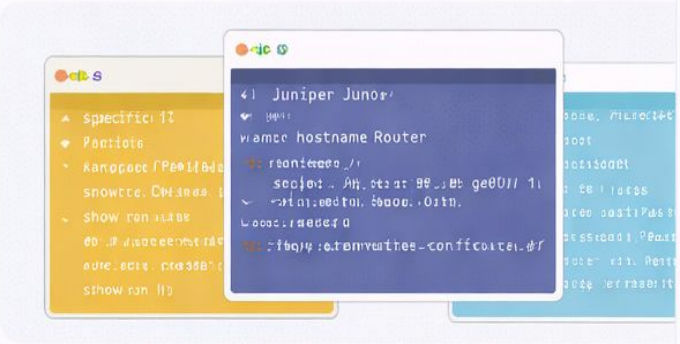
Result:

- Clean Git history
- Only meaningful changes tracked
- Reduced storage and SCM noise

Vendor Neutral Design

The Challenge:

- Each vendor has different CLI commands
- Different output formats
- Different restore methods



Our Approach:

- Uses Ansible's network resource modules
- Abstracts vendor differences
- Single playbook works across platforms

Vendor Neutral Design

Supported Platforms:

- Cisco IOS/IOS-XE
- Cisco IOS-XR
- Cisco NX-OS
- Arista EOS
- Juniper Junos

Code Example:

```
- name: Create backup
  - ansible.builin.include_role:
    - name: network.backup.backup
  , vars:
    - type: "cdiff"
    - data_store:
      - scm:
        - orign: "gi@githubcom.user/repo.git"
          - filename: "{{inventory_hostname}}.ttx"
    }
```

Same playbook works for all platforms!

Diff Severity Scoring

The Problem:

- All configuration changes look the same
- No way to prioritize what needs urgent review
- BGP changes are as important as description changes? No!

Our Solution: Rules-Based Scoring

- Analyzes configuration diff
- Extracts features: BGP, ACL, routing, VLAN, interface, security
- Assigns points based on change type
- Categorizes into severity levels

Scoring Rules:

CRITICAL (≥20 points):

- BGP changes: 10 points each
- ACL changes: 10 points each
- Security changes (AAA, TACACS, RADIUS): 10 points ach

HIGH (10-19 points):

- Multiple routing changes: 5 points each
- VLAN changes: 5 points each

MEDIUM (5-9 points):

- Interface changes: 3 points each

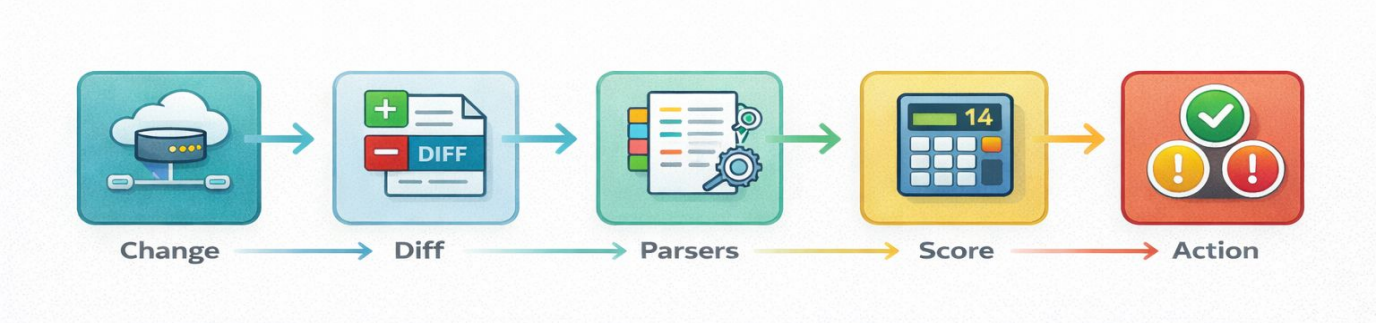
LOW (<5 points):

- Description-only changes: 1 point each

Example Output:

```
- Severity Level: HIGH
- Severity Score: 12
```

Diff Severity Scoring



Change: Device config modified

Diff: Unified diff generated

Parsers: Domain-aware analysis (BGP, ACL, Routing, etc.)

Score: Weighted severity calculation

Action: Auto-approve / Review / Block

Diff Severity Scoring (Rule Based)

When to Use ML:

- Rules-based works great for most cases
- ML adds value for:
  - Complex multi-feature changes
  - Historical pattern recognition
  - Custom organizational risk models

Scoring Rules:

CRITICAL (≥ 20 points):

- BGP changes: 10 points each
- ACL changes: 10 points each
- Security changes (AAA, TACACS, RADLS): 10 points each

HIGH (10-19 points):

- Multiple routing changes: 5 points each
- VLAN changes: 5 points each

MEDIUM (5-9 points):

- Interface changes: 3 points each

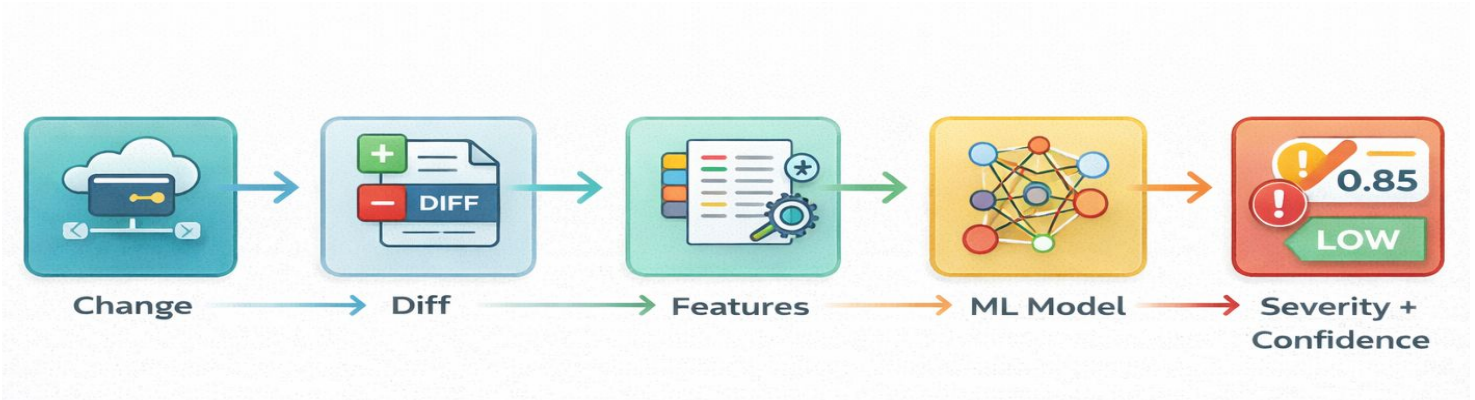
LOW (<5 points):

- Description-only changes: 1 point each

Example:

```
# Train model
python train_ml_model.py —data historical_diffs.json
# Use in playbook
enable_ml: true  '/path/to/model.pkl
```

ML-Enhanced Diff Severity Scoring



Learns from historical changes

Adapts to your network

Produces confidence scores

Backup Verification

The Problem:

- How do you know if a backup file is corrupted?
- What if someone tampered with the backup?
- No way to verify before restore

Our Solution: SHA-256 Hash Verification

- Every backup file gets a SHA-256 hash
- Hash stored in separate .sha256 file

Example:

Backup File:
`ios_device_backup.txt`

Hash File:
`ios_device_backup.txt.sha256`

Our Solution: SHA-256 Hash Verification

- Every backup file gets a SHA-256 hash
- Hash stored in separate .sha256 file
- Hash verified before restore operations

During Backup:

1. Calculate SHA-256 hash of backup file
2. Store hash in 'backup.txt.sha256' file
3. Both files committed to Git together

During Restore:

1. Read expected hash from .sha256' file
2. Calculate actual hash of backup file

✅ MATCH → Restore proceeds
❌ MISMATCH → Restore aborted (safety)

Benefits:

- Detects file corruption
- Detects tampering
- Prevents restoring bad configs
- Cryptographic proof of integrity

Example:

Backup File: `ios_device_backup.txt`
Hash File: `ios_device_backup.txt.sha256`

SCM Integration (git)

Workflow:

- Backup creates PR automatically
- Team reviews changes
- Merge PR to approve backup
- Restore uses merged backup

Benefits:

- Version control for all backups
- Collaborative review process
- Audit trail
- Rollback capability

SSH Key Support:

- Uses SSH keys for authentication
- No tokens needed
- Secure and simple

Our Solution: SHA-256 Hash Verification

- Backup creates PR automatically
- Team reviews changes
- Merge PR to approve backup

Workflow:

Example:

Backup File: `ios_device_backup.txt`

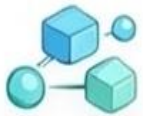
Hash File: `ios_device_backup.txt.sha256`

Demo In Action

Key Takeaways

- **Idempotent backups** eliminate **false positives**
- **Vendor-neutral design** reduces maintenance
- **Diff severity scoring** prioritizes reviews
- **Hash verification** ensures backup integrity
- **Git integration** enables collaboration
- **ML enhancement** available for complex scenarios

Network Backup Resources



Feature

https://github.com/redhat-cop/network.backup/tree/ai_dev_backup

SCM Integration

https://github.com/ansible-collections/ansible.scm

Playbooks

https://github.com/rohitthakur2590/automation-playbooks

Ansible community

Join the forum to participate in discussions and get help!

Want to contribute? Find out how to get involved.

Thanks!

GitHub: **rohitthakur2590**
Matrix: **#community:ansible.com #social:ansible.com**

Ansible community forum:
https://forum.ansible.com/

ANSIBLE