Inventory Migration Example: INI to YAML

Overview

This document provides a practical example of migrating from INI format inventory to the standardized YAML format used in HX Infrastructure.

Example Migration

Original INI Format (Legacy)

File: `inventory/legacy.ini`

```
[webservers]
web1.example.com ansible_host=192.168.1.10 server_role=primary
web2.example.com ansible_host=192.168.1.11 server_role=secondary

[databases]
db1.example.com ansible_host=192.168.1.20 postgresql_role=master
db2.example.com ansible_host=192.168.1.21 postgresql_role=replica

[loadbalancers]
lb1.example.com ansible_host=192.168.1.30 keepalived_priority=110

[webservers:vars]
http_port=80
https_port=443
ssl_enabled=true

[databases:vars]
postgresql_version=13
backup_enabled=true

[all:vars]
ansible_user=admin
environment=production
```

Converted YAML Format (Standard)

File: `inventory/environments/production/hosts.yml`

```yaml
all:
  children:
    # Web Services Group
    ui:
      children:
        web_interfaces:
          hosts:
            web1.example.com:
              ansible_host: 192.168.1.10
              server_role: primary
              services: ['nginx', 'application_server']
              os_type: linux
              memory_gb: 8
              cpu_cores: 4

            web2.example.com:
              ansible_host: 192.168.1.11
              server_role: secondary
              services: ['nginx', 'application_server', 'failover']
              os_type: linux
              memory_gb: 8
              cpu_cores: 4

          vars:
            http_port: 80
            https_port: 443
            ssl_enabled: true

        load_balancers:
          hosts:
            lb1.example.com:
              ansible_host: 192.168.1.30
              server_role: primary_load_balancer
              services: ['nginx', 'keepalived', 'ssl_termination']
              os_type: linux
              memory_gb: 4
              cpu_cores: 2
              keepalived_priority: 110

    # Operations Services Group
    operations:
      children:
        databases:
          hosts:
            db1.example.com:
              ansible_host: 192.168.1.20
              server_role: postgresql_primary
              services: ['postgresql', 'backup_service']
              os_type: linux
              memory_gb: 16
              cpu_cores: 8
              storage_gb: 500
              postgresql_role: master

            db2.example.com:
              ansible_host: 192.168.1.21
              server_role: postgresql_replica
              services: ['postgresql', 'read_replica']
              os_type: linux
              memory_gb: 16
              cpu_cores: 8
              storage_gb: 500
```

```
            postgresql_role: replica

        vars:
          postgresql_version: 13
          backup_enabled: true

  vars:
    # Global environment variables
    environment: production
    domain_name: example.com
    ansible_user: admin

    # Common configuration
    timezone: UTC
    ntp_servers:
      - 0.pool.ntp.org
      - 1.pool.ntp.org

    # Security configuration
    firewall_enabled: true
    fail2ban_enabled: true
    automatic_updates: true

    # Monitoring configuration
    monitoring_enabled: true
    log_level: info

# Service-specific groupings for targeted deployments
ui_services:
  children:
    web_interfaces:
    load_balancers:

operations_services:
  children:
    databases:

# Platform-specific groupings
linux_hosts:
  children:
    ui:
    operations:

# Role-specific groupings for maintenance
primary_services:
  hosts:
    web1.example.com:
    db1.example.com:
    lb1.example.com:

secondary_services:
  hosts:
    web2.example.com:
    db2.example.com:
```

Migration Benefits Demonstrated

1. Enhanced Structure

Before (INI): Flat group structure

```
[webservers]
[databases]
[loadbalancers]
```

After (YAML): Hierarchical service organization

```yaml
all:
  children:
    ui:
      children:
        web_interfaces:
        load_balancers:
    operations:
      children:
        databases:
```

2. Rich Metadata

Before (INI): Limited variable support

```
web1.example.com ansible_host=192.168.1.10 server_role=primary
```

After (YAML): Comprehensive host metadata

```yaml
web1.example.com:
  ansible_host: 192.168.1.10
  server_role: primary
  services: ['nginx', 'application_server']
  os_type: linux
  memory_gb: 8
  cpu_cores: 4
```

3. Flexible Groupings

Before (INI): Single-level grouping

```
[webservers:vars]
http_port=80
```

After (YAML): Multiple grouping strategies

```yaml
# Service-specific groupings
ui_services:
  children:
    web_interfaces:
    load_balancers:

# Platform-specific groupings
linux_hosts:
  children:
    ui:
    operations:

# Role-specific groupings
primary_services:
  hosts:
    web1.example.com:
    db1.example.com:
```

Migration Process

Step 1: Analyze Current INI Structure

```
# Identify all INI files
find . -name "*.ini" -type f

# Analyze group structure
grep -E "^\[.*\]" inventory/*.ini

# Identify host variables
grep -E "^[^#\[].*=" inventory/*.ini
```

Step 2: Create YAML Structure

```
# Create new directory structure
mkdir -p inventory/environments/production
mkdir -p inventory/environments/staging
mkdir -p inventory/environments/development

# Create group_vars directories
mkdir -p inventory/group_vars
mkdir -p inventory/environments/production/group_vars
```

Step 3: Convert and Validate

```
# Convert INI to YAML (manual process using above example)
# Validate new inventory
ansible-inventory --list

# Test connectivity
ansible all -m ping

# Compare group memberships
ansible-inventory --graph
```

Step 4: Update Configuration

```
# Update ansible.cfg
sed -i 's|inventory = .*|inventory = inventory/environments/production|' ansible.cfg

# Test with new configuration
ansible-playbook --check playbooks/site.yml
```

Validation Checklist

- [ ] All hosts from INI are present in YAML
- [ ] All group memberships are preserved
- [ ] All host variables are migrated
- [ ] All group variables are migrated
- [ ] New hierarchical structure is logical
- [ ] Service groupings are appropriate
- [ ] Platform groupings are accurate
- [ ] Role-based groupings are useful
- [ ] Ansible inventory parsing succeeds
- [ ] Host connectivity tests pass
- [ ] Playbook execution works correctly
- [ ] Variable resolution is correct

Common Migration Issues

Issue 1: Complex Host Variables

Problem: INI format with many host variables becomes unwieldy
Solution: Use YAML's structured format for better organization

Issue 2: Group Inheritance

Problem: INI group inheritance doesn't translate directly
Solution: Use YAML's children structure for clear relationships

Issue 3: Variable Precedence

Problem: Variable precedence may change between formats
Solution: Test variable resolution thoroughly after migration

Issue 4: Special Characters

Problem: Special characters in hostnames or variables
Solution: Use YAML quoting and escaping as needed

Related Documentation

- Inventory Management Standards (../inventory/README.md)
- Ansible Configuration (../ansible/README.md)
- Documentation Standards (../standards/Documentation_Standards.md)