HX Infrastructure Variable Management System

Overview

The HX Infrastructure Variable Management System provides a comprehensive, hierarchical approach to configuration management across all environments and services. This system implements best practices for variable organization, validation, and environment-specific customization.

Architecture

Variable Hierarchy

The variable system follows a clear hierarchy from most general to most specific:

1. **Global Defaults** (`group_vars/templates/all.yml.j2`)
2. **Service Role Defaults** (`role_defaults/*/defaults/main.yml`)
3. **Group Variables** (`inventories/group_vars/*.yml`)
4. **Environment Variables** (`inventories/{env}/group_vars/*.yml`)
5. **Host Variables** (`inventories/{env}/host_vars/*.yml`)

Variable Precedence

Variables are resolved in order of precedence (highest to lowest):

1. Host variables (most specific)
2. Environment group variables
3. Global group variables
4. Role defaults
5. Template defaults (lowest precedence)

Variable Categories

1. Global Variables (`all.yml`)

Core infrastructure variables that apply to all hosts and services:

```yaml
# Project Information
project_name: "HX Infrastructure"
project_version: "3.0.0"
organization: "Hana-X AI"
environment: "{{ environment | mandatory }}"

# Domain and Network
domain_name: "{{ domain_name | mandatory }}"
network_subnet: "{{ network_subnet | mandatory }}"
vip_address: "{{ vip_address | default(network_subnet | ipaddr('100') | ipad-
dr('address')) }}"

# Authentication
default_user: "{{ default_user | default('agent0') }}"
admin_email: "{{ admin_email | default(default_email) }}"
```

2. Infrastructure Variables

Windows-based services (Domain Controllers, Certificate Authorities):

```yaml
# Active Directory Configuration
active_directory:
  install_dns: "{{ ad_install_dns | default(true) }}"
  install_dhcp: "{{ ad_install_dhcp | default(true) }}"
  organizational_units: "{{ ad_organizational_units | de-
fault(organizational_units_template) }}"

# Certificate Authority
certificate_authority:
  ca_name: "{{ ca_name | default(organization + '-CA') }}"
  key_length: "{{ ca_key_length | default(4096) }}"
  validity_period: "{{ ca_validity_period | default('10 years') }}"
```

3. AI/ML Variables

Machine learning and inference services:

```yaml
# Python Environment
python:
  version: "{{ python_version | default('3.11') }}"
  venv_path: "{{ python_venv_path | default('/opt/ai-ml/venv') }}"

# LiteLLM Configuration
litellm:
  port: "{{ litellm_port | default(4000) }}"
  workers: "{{ litellm_workers | default(4) }}"
  models: "{{ litellm_models | default(models_template) }}"

# Ollama Configuration
ollama:
  port: "{{ ollama_port | default(11434) }}"
  models: "{{ ollama_models | default(models_template) }}"
  gpu:
    enabled: "{{ gpu_enabled | default(false) }}"
    memory_fraction: "{{ ollama_gpu_memory_fraction | default(0.9) }}"
```

4. Operations Variables

Database, caching, monitoring, and logging services:

```
# PostgreSQL Configuration
postgresql:
  version: "{{ postgresql_version | default('15') }}"
  max_connections: "{{ postgresql_max_connections | default(200) }}"
  memory:
    shared_buffers: "{{ postgresql_memory_shared_buffers |
default((ansible_memtotal_mb * 0.25) | int) }}MB"

# Redis Configuration
redis:
  version: "{{ redis_version | default('7') }}"
  memory:
    maxmemory: "{{ redis_maxmemory | default((ansible_memtotal_mb * 0.75) | int) }}MB"

# Monitoring Stack
prometheus:
  retention_time: "{{ prometheus_retention_time | default('90d') }}"
  scrape_configs: "{{ prometheus_scrape_configs | default(scrape_configs_template) }}"
```

5. UI Variables

Web interfaces and load balancing:

```
# WebUI Configuration
webui:
  port: "{{ webui_port | default(3000) }}"
  auth:
    enabled: "{{ webui_auth_enabled | default(true) }}"
    secret_key: "{{ vault_jwt_secret }}"

# Nginx Configuration
nginx:
  worker_processes: "{{ nginx_worker_processes | default('auto') }}"
  upstreams: "{{ nginx_upstreams | default(upstreams_template) }}"

# SSL Certificates
ssl_certificates:
  letsencrypt:
    enabled: "{{ ssl_letsencrypt_enabled | default(true) }}"
    domains: "{{ ssl_letsencrypt_domains | default([domain_name, 'www.' +
domain_name]) }}"
```

Variable Templates

Template System

Variables use Jinja2 templating for dynamic configuration:

```
# Dynamic defaults based on other variables
ca_name: "{{ ca_name | default(organization + '-CA') }}"
grafana_root_url: "{{ grafana_root_url | default('https://' + domain_name + '/
grafana/') }}"

# Conditional defaults
ssl_self_signed_enabled: "{{ ssl_self_signed_enabled | default(environment == 'develop-
ment') }}"

# Dynamic resource allocation
postgresql_memory_shared_buffers: "{{ postgresql_memory_shared_buffers | de-
fault((ansible_memtotal_mb * 0.25) | int) }}MB"
```

Template Functions

Common template patterns:

```
# Mandatory variables (must be defined)
domain_name: "{{ domain_name | mandatory }}"

# Default with fallback
admin_email: "{{ admin_email | default(default_email) }}"

# Conditional defaults
debug_mode: "{{ debug_mode | default(environment != 'production') }}"

# List manipulation
dns_servers: "{{ dns_servers | default(['8.8.8.8', '8.8.4.4']) }}"

# Network calculations
vip_address: "{{ vip_address | default(network_subnet | ipaddr('100') | ipad-
dr('address')) }}"
```

Environment-Specific Configuration

Development Environment

```
environment_config:
  development:
    debug_mode: true
    ssl_verification: false
    backup_enabled: false
    log_level: "DEBUG"
    auto_download_models: true
```

Test Environment

```
environment_config:
  test:
    debug_mode: true
    ssl_verification: false
    backup_enabled: true
    log_level: "INFO"
    retention_days: 14
```

Production Environment

```yaml
environment_config:
  production:
    debug_mode: false
    ssl_verification: true
    backup_enabled: true
    log_level: "WARN"
    high_availability: true
    performance_monitoring: true
```

Vault Variables

Sensitive variables are stored in Ansible Vault files:

```yaml
# Database Credentials
vault_postgres_password: "{{ vault_postgres_password | default('') }}"
vault_redis_password: "{{ vault_redis_password | default('') }}"

# Application Secrets
vault_jwt_secret: "{{ vault_jwt_secret | default('') }}"
vault_session_secret: "{{ vault_session_secret | default('') }}"

# SSL/TLS Certificates
vault_ssl_private_key: "{{ vault_ssl_private_key | default('') }}"

# OAuth Credentials
vault_google_oauth_client_id: "{{ vault_google_oauth_client_id | default('') }}"
vault_github_oauth_client_secret: "{{ vault_github_oauth_client_secret | default('') }}
"
```

Variable Validation

Validation Rules

All variables are validated against comprehensive rules:

```yaml
# Type validation
- name: "postgresql_max_connections"
  type: "integer"
  min_value: 50
  max_value: 1000

# Pattern validation
- name: "domain_name"
  type: "string"
  pattern: "^[a-zA-Z0-9][a-zA-Z0-9-]{1,61}[a-zA-Z0-9]\\.[a-zA-Z]{2,}$"

# Dependency validation
- condition: "gpu_enabled == true"
  requires: ["cuda_version", "ai_ml_gpu_memory_fraction"]
```

Running Validation

```
# Validate all variables
python vars_validation/validate_vars.py inventories/group_vars/*.yml

# Validate specific environment
python vars_validation/validate_vars.py \
  inventories/group_vars/*.yml \
  inventories/prod/group_vars/*.yml
```

Best Practices

1. Variable Naming

- Use descriptive, hierarchical names: `postgresql_max_connections`
- Group related variables: `webui_auth_enabled`, `webui_auth_secret_key`
- Use consistent prefixes for services: `nginx_*`, `redis_*`, `ollama_*`

2. Default Values

- Provide sensible defaults for all optional variables
- Use dynamic defaults based on system resources
- Make environment-specific adjustments

3. Documentation

- Document all variables with descriptions
- Explain complex relationships and dependencies
- Provide examples for custom configurations

4. Security

- Never store secrets in plain text
- Use Ansible Vault for sensitive data
- Validate password strength and security settings

5. Environment Management

- Use consistent variable names across environments
- Override only necessary values per environment
- Validate environment-specific requirements

Usage Examples

Basic Variable Override

```
# inventories/prod/group_vars/all.yml
postgresql_max_connections: 500
redis_maxmemory: "8GB"
log_level: "WARN"
```

Service-Specific Configuration

```yaml
# inventories/group_vars/ai_ml.yml
ollama_models:
  - name: "llama3.2:70b"
    priority: 1
  - name: "codellama:34b"
    priority: 2

litellm_workers: 8
ai_ml_gpu_memory_fraction: 0.95
```

Environment-Specific Overrides

```yaml
# inventories/dev/group_vars/all.yml
debug_mode: true
ssl_verification: false
backup_enabled: false

# inventories/prod/group_vars/all.yml
debug_mode: false
ssl_verification: true
backup_enabled: true
high_availability: true
```

Troubleshooting

Common Issues

1. **Missing Required Variables**
   `Error: Missing required global variable: domain_name`
   `Solution: Define domain_name in group_vars/all.yml`

2. **Type Validation Errors**
   `Error: postgresql_max_connections: Expected type integer, got string`
   `Solution: Remove quotes around numeric values`

3. **Dependency Validation Failures**
   `Error: GPU-enabled hosts require CUDA configuration`
   `Solution: Define cuda_version when gpu_enabled is true`

Debugging Variables

```bash
# Check variable resolution
ansible-inventory -i inventories/prod --list --yaml

# Test variable templates
ansible all -i inventories/prod -m debug -a "var=postgresql_max_connections"

# Validate specific variables
python vars_validation/validate_vars.py inventories/group_vars/operations.yml
```

Migration Guide

From Previous Versions

1. **Update variable names** to new hierarchical structure
2. **Move sensitive data** to vault files
3. **Add validation rules** for custom variables
4. **Test configurations** in development environment
5. **Update playbooks** to use new variable names

Adding New Variables

1. **Define in appropriate role defaults**
2. **Add validation rules** in `validation_rules.yml`
3. **Document usage** and examples
4. **Test across all environments**
5. **Update templates** if needed

This variable management system provides a robust, scalable foundation for managing complex infrastructure configurations while maintaining security, consistency, and ease of use across all environments.