Variable Usage Examples

This document provides practical examples of how to use the HX Infrastructure variable system for different scenarios and environments.

Basic Environment Setup

Development Environment

```yaml
# inventories/dev/group_vars/all.yml
---
# Environment Configuration
environment: development
domain_name: dev-test.hana-x.ai
network_subnet: 192.168.10.0/24

# Development-specific overrides
debug_mode: true
ssl_verification: false
backup_enabled: false
log_level: DEBUG

# Resource allocation (smaller for dev)
postgresql_max_connections: 100
redis_maxmemory: "2GB"
ollama_max_loaded_models: 2

# Security (relaxed for development)
security_ssh_port: 22
firewall_enabled: true
ssl_self_signed_enabled: true
```

Production Environment

```yaml
# inventories/prod/group_vars/all.yml
---
# Environment Configuration
environment: production
domain_name: hana-x.ai
network_subnet: 10.0.0.0/24

# Production-specific settings
debug_mode: false
ssl_verification: true
backup_enabled: true
log_level: WARN
high_availability: true
performance_monitoring: true

# Resource allocation (optimized for production)
postgresql_max_connections: 500
redis_maxmemory: "16GB"
ollama_max_loaded_models: 5

# Security (strict for production)
security_ssh_port: 2222
firewall_enabled: true
ssl_letsencrypt_enabled: true
```

Service-Specific Configurations

AI/ML Service Customization

```yaml
# inventories/group_vars/ai_ml.yml
---
# Custom Python environment
python_version: "3.11"
python_pip_requirements:
  - "torch>=2.1.0"
  - "transformers>=4.35.0"
  - "custom-ml-package>=1.0.0"

# LiteLLM customization
litellm_workers: 8
litellm_timeout: 600
litellm_models:
  - model_name: "gpt-4-turbo"
    litellm_params:
      model: "ollama/llama3.2:70b"
      api_base: "http://{{ groups['llm_services'][0] }}:11434"
  - model_name: "code-assistant"
    litellm_params:
      model: "ollama/codellama:34b"
      api_base: "http://{{ groups['llm_services'][0] }}:11434"

# Ollama customization
ollama_models:
  - name: "llama3.2:70b"
    size: "70B"
    quantization: "Q4_K_M"
    priority: 1
  - name: "codellama:34b"
    size: "34B"
    quantization: "Q4_K_M"
    priority: 2
  - name: "mistral:7b-instruct"
    size: "7B"
    quantization: "Q4_K_M"
    priority: 3

# GPU configuration
gpu_enabled: true
cuda_version: "12.1"
ai_ml_gpu_memory_fraction: 0.95
ollama_gpu_layers: -1  # Use all GPU layers
```

Database Optimization

```yaml
# inventories/group_vars/operations.yml
---
# PostgreSQL performance tuning
postgresql_max_connections: 400
postgresql_shared_buffers: "4GB"
postgresql_effective_cache_size: "12GB"
postgresql_work_mem: "32MB"
postgresql_maintenance_work_mem: "512MB"
postgresql_checkpoint_completion_target: 0.9
postgresql_wal_buffers: "64MB"

# Custom database configuration
postgresql_databases:
  - name: "webui"
    owner: "webui_user"
    encoding: "UTF8"
    locale: "en_US.UTF-8"
  - name: "analytics"
    owner: "analytics_user"
    encoding: "UTF8"
    locale: "en_US.UTF-8"
  - name: "model_registry"
    owner: "model_registry_user"
    encoding: "UTF8"
    locale: "en_US.UTF-8"

# Redis optimization
redis_maxmemory: "8GB"
redis_maxmemory_policy: "allkeys-lru"
redis_save_rules:
  - "900 1"
  - "300 10"
  - "60 10000"

# Monitoring configuration
prometheus_retention_time: "180d"
prometheus_retention_size: "100GB"
grafana_plugins:
  - grafana-piechart-panel
  - grafana-worldmap-panel
  - grafana-clock-panel
  - grafana-kubernetes-app
```

Web Interface Customization

```yaml
# inventories/group_vars/ui.yml
---
# WebUI customization
webui_name: "HX Enterprise AI Assistant"
webui_app_title: "Enterprise AI Chat"
webui_app_theme: "corporate"
webui_port: 3000

# Authentication configuration
webui_auth_enabled: true
webui_auth_type: "jwt"
webui_auth_token_expiry: 28800  # 8 hours
webui_oauth_enabled: true
webui_oauth_providers:
  - name: "google"
    client_id: "{{ vault_google_oauth_client_id }}"
    client_secret: "{{ vault_google_oauth_client_secret }}"
  - name: "microsoft"
    client_id: "{{ vault_microsoft_oauth_client_id }}"
    client_secret: "{{ vault_microsoft_oauth_client_secret }}"

# Chat configuration
webui_chat_max_message_length: 8000
webui_chat_max_conversation_length: 200
webui_chat_available_models:
  - name: "gpt-4-turbo"
    display_name: "GPT-4 Turbo"
    description: "Most capable model for complex tasks"
    max_tokens: 8192
  - name: "code-assistant"
    display_name: "Code Assistant"
    description: "Specialized for programming tasks"
    max_tokens: 4096
  - name: "quick-chat"
    display_name: "Quick Chat"
    description: "Fast responses for simple queries"
    max_tokens: 2048

# Nginx optimization
nginx_worker_processes: 8
nginx_worker_connections: 2048
nginx_client_max_body_size: "500M"
nginx_gzip_comp_level: 9

# SSL configuration
ssl_letsencrypt_enabled: true
ssl_letsencrypt_domains:
  - "{{ domain_name }}"
  - "www.{{ domain_name }}"
  - "api.{{ domain_name }}"
  - "chat.{{ domain_name }}"
```

Advanced Configuration Patterns

Multi-Environment Variable Management

```yaml
# inventories/group_vars/all.yml (base configuration)
---
# Base settings that apply to all environments
project_name: "HX Infrastructure"
organization: "Hana-X AI"

# Environment-specific overrides using templates
backup_retention_days: >-
  {%- if environment == 'development' -%}
    7
  {%- elif environment == 'test' -%}
    14
  {%- else -%}
    90
  {%- endif -%}

log_level: >-
  {%- if environment == 'production' -%}
    WARN
  {%- elif environment == 'test' -%}
    INFO
  {%- else -%}
    DEBUG
  {%- endif -%}

# Resource allocation based on environment
postgresql_max_connections: >-
  {%- if environment == 'production' -%}
    500
  {%- elif environment == 'test' -%}
    200
  {%- else -%}
    100
  {%- endif -%}
```

Dynamic Resource Allocation

```yaml
# inventories/group_vars/operations.yml
---
# Dynamic PostgreSQL configuration based on system resources
postgresql_memory_shared_buffers: "{{ (ansible_memtotal_mb * 0.25) | int }}MB"
postgresql_memory_effective_cache_size: "{{ (ansible_memtotal_mb * 0.75) | int }}MB"
postgresql_memory_work_mem: "{{ ((ansible_memtotal_mb * 0.25) / postgr-
esql_max_connections) | int }}MB"

# Dynamic Redis configuration
redis_maxmemory: "{{ (ansible_memtotal_mb * 0.6) | int }}MB"

# Dynamic Elasticsearch heap size
elasticsearch_heap_size: "{{ (ansible_memtotal_mb * 0.5) | int }}m"

# CPU-based configuration
nginx_worker_processes: "{{ ansible_processor_vcpus }}"
ollama_threads: "{{ ansible_processor_vcpus }}"
logstash_pipeline_workers: "{{ ansible_processor_vcpus }}"
```

Conditional Service Configuration

```
# inventories/group_vars/ai_ml.yml
---
# GPU-specific configuration
ollama_server_config: >-
  {%- if gpu_enabled -%}
  {
    "max_loaded_models": {{ ollama_max_loaded_models | default(5) }},
    "gpu_layers": {{ ollama_gpu_layers | default(-1) }},
    "gpu_memory_fraction": {{ ai_ml_gpu_memory_fraction | default(0.9) }},
    "flash_attention": true
  }
  {%- else -%}
  {
    "max_loaded_models": {{ ollama_max_loaded_models | default(2) }},
    "cpu_threads": {{ ansible_processor_vcpus }},
    "use_mmap": true,
    "use_mlock": true
  }
  {%- endif -%}

# Model selection based on GPU availability
ollama_models: >-
  {%- if gpu_enabled -%}
  [
    {"name": "llama3.2:70b", "priority": 1},
    {"name": "codellama:34b", "priority": 2},
    {"name": "mistral:7b-instruct", "priority": 3}
  ]
  {%- else -%}
  [
    {"name": "llama3.2:7b", "priority": 1},
    {"name": "phi3:mini", "priority": 2},
    {"name": "tinyllama:1.1b", "priority": 3}
  ]
  {%- endif -%}
```

Host-Specific Overrides

GPU-Enabled Hosts

```yaml
# inventories/prod/host_vars/hx-ollama-01.prod.hana-x.ai.yml
---
# GPU-specific configuration
gpu_enabled: true
gpu_memory_gb: 48
cuda_version: "12.1"

# Optimized for large models
ollama_max_loaded_models: 3
ai_ml_gpu_memory_fraction: 0.95
ollama_gpu_layers: -1

# Performance tuning
performance_swappiness: 1
performance_vm_dirty_ratio: 5
ai_ml_performance_transparent_hugepages: "always"

# Custom model configuration
ollama_models:
  - name: "llama3.2:70b"
    size: "70B"
    quantization: "Q4_K_M"
    priority: 1
  - name: "codellama:34b"
    size: "34B"
    quantization: "Q4_K_M"
    priority: 2
```

Database Primary/Replica Configuration

```yaml
# inventories/prod/host_vars/hx-db-01.prod.hana-x.ai.yml (Primary)
---
postgresql_role: master
postgresql_replication_enabled: true
postgresql_max_wal_senders: 10
postgresql_wal_level: "replica"
postgresql_archive_mode: true
postgresql_archive_command: "cp %p /backup/postgresql/archive/%f"

# Performance optimization for primary
postgresql_checkpoint_completion_target: 0.9
postgresql_wal_buffers: "64MB"
postgresql_max_connections: 500
```

```yaml
# inventories/prod/host_vars/hx-db-02.prod.hana-x.ai.yml (Replica)
---
postgresql_role: replica
postgresql_primary_host: "hx-db-01.prod.hana-x.ai"
postgresql_hot_standby: true
postgresql_hot_standby_feedback: true
postgresql_max_standby_streaming_delay: "30s"

# Read-only optimization
postgresql_max_connections: 200
postgresql_default_statistics_target: 50
```

Vault Variable Examples

Secure Credential Management

```yaml
# inventories/group_vars/vault.yml (encrypted with ansible-vault)
---
# Database credentials
vault_postgres_password: "SecureP@ssw0rd123!"
vault_postgres_replication_password: "ReplicaP@ssw0rd456!"
vault_redis_password: "RedisSecure789!"

# Application secrets
vault_jwt_secret: "jwt-super-secret-key-that-is-very-long-and-secure-123456789"
vault_session_secret: "session-encryption-key-that-is-also-very-secure-987654321"

# Database user credentials
vault_webui_db_user: "webui_user"
vault_webui_db_password: "WebUISecure123!"
vault_model_registry_user: "model_registry_user"
vault_model_registry_password: "ModelRegSecure456!"
vault_monitoring_db_user: "monitoring_user"
vault_monitoring_db_password: "MonitorSecure789!"
vault_grafana_db_user: "grafana_user"
vault_grafana_db_password: "GrafanaSecure012!"

# Admin credentials
vault_grafana_admin_password: "GrafanaAdmin345!"
vault_grafana_secret_key: "grafana-secret-key-for-encryption-678901234"

# Load balancer
vault_keepalived_password: "KeepAliv"  # Exactly 8 characters

# OAuth credentials
vault_google_oauth_client_id: "123456789-abcdefghijklmnop.apps.googleusercontent.com"
vault_google_oauth_client_secret: "GOCSPX-abcdefghijklmnopqrstuvwxyz"
vault_github_oauth_client_id: "Iv1.abcdefghijklmnop"
vault_github_oauth_client_secret: "abcdefghijklmnopqrstuvwxyz123456789012"

# SMTP credentials
vault_smtp_server: "smtp.gmail.com"
vault_smtp_port: 587
vault_smtp_username: "alerts@hana-x.ai"
vault_smtp_password: "SMTPSecurePassword123!"
vault_smtp_from: "HX Infrastructure <alerts@hana-x.ai>"

# SSL certificates (if using custom CA)
vault_ssl_private_key: |
  -----BEGIN PRIVATE KEY-----
  MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC...
  -----END PRIVATE KEY-----

vault_ssl_certificate: |
  -----BEGIN CERTIFICATE-----
  MIIDXTCCAkWgAwIBAgIJAKoK/OvD8w5EMA0GCSqGSIb3DQEBCwUA...
  -----END CERTIFICATE-----
```

Validation Examples

Running Variable Validation

```
# Validate all variables for development environment
python vars_validation/validate_vars.py \
  inventories/group_vars/*.yml \
  inventories/dev/group_vars/*.yml

# Validate specific service variables
python vars_validation/validate_vars.py \
  inventories/group_vars/all.yml \
  inventories/group_vars/ai_ml.yml

# Validate with vault variables (decrypt first)
ansible-vault decrypt inventories/group_vars/vault.yml
python vars_validation/validate_vars.py inventories/group_vars/*.yml
ansible-vault encrypt inventories/group_vars/vault.yml
```

Custom Validation Rules

```
# vars_validation/custom_rules.yml
---
# Custom validation for specific deployment
custom_validation_rules:
  - name: "enterprise_security_requirements"
    description: "Enterprise security compliance"
    rules:
      - "ssl_verification == true"
      - "backup_encryption == true"
      - "security_ssh_port != 22"
      - "firewall_enabled == true"

  - name: "gpu_cluster_requirements"
    description: "GPU cluster configuration validation"
    condition: "gpu_enabled == true"
    rules:
      - "cuda_version is defined"
      - "ai_ml_gpu_memory_fraction >= 0.8"
      - "ollama_gpu_layers == -1"
```

Migration Examples

Upgrading from Previous Version

```yaml
# Old configuration (v2.0)
webui_database_host: "192.168.10.40"
webui_database_port: 5432
webui_database_name: "webui"
webui_database_user: "webui_user"

# New configuration (v3.0)
webui:
  database:
    host: "{{ groups['databases'][0] }}"
    port: "{{ webui_database_port | default(5432) }}"
    name: "{{ webui_database_name | default('webui') }}"
    username: "{{ vault_webui_db_user }}"
    password: "{{ vault_webui_db_password }}"
```

Adding New Service

```yaml
# inventories/group_vars/new_service.yml
---
# New service configuration following established patterns
new_service:
  version: "{{ new_service_version | default('latest') }}"
  port: "{{ new_service_port | default(8080) }}"
  host: "{{ new_service_host | default('0.0.0.0') }}"

  # Database connection
  database:
    host: "{{ groups['databases'][0] }}"
    port: 5432
    name: "{{ new_service_database_name | default('new_service') }}"
    username: "{{ vault_new_service_db_user }}"
    password: "{{ vault_new_service_db_password }}"

  # Monitoring
  monitoring:
    enabled: "{{ new_service_monitoring_enabled | default(true) }}"
    port: "{{ new_service_metrics_port | default(9090) }}"

  # Environment-specific overrides
  environment_config:
    development:
      debug_mode: true
      log_level: "DEBUG"
    production:
      debug_mode: false
      log_level: "WARN"
      high_availability: true
```

This comprehensive set of examples demonstrates the flexibility and power of the HX Infrastructure variable system, showing how to configure everything from simple development environments to complex production deployments with advanced features and security requirements.