HX Infrastructure Development Guide

Development Standards and Best Practices

This guide outlines the development standards, workflows, and best practices for contributing to the HX Infrastructure Ansible project.

Table of Contents

Development Environment Setup

Prerequisites

```
# Required software versions
ansible >= 2.15.0
python >= 3.8
git >= 2.25
docker >= 20.10 (for molecule testing)
```

Local Setup

```
# Clone repository
git clone https://github.com/hanax-ai/HX-Infrastructure-Ansible.git
cd HX-Infrastructure-Ansible

# Setup development environment
make dev-setup

# Install pre-commit hooks
pre-commit install

# Verify installation
make lint test
```

IDE Configuration

VS Code Settings

```
{
  "ansible.python.interpreterPath": "/usr/bin/python3",
  "ansible.validation.enabled": true,
  "ansible.validation.lint.enabled": true,
  "yaml.schemas": {
    "https://raw.githubusercontent.com/ansible/ansible-lint/main/src/ansiblelint/
schemas/ansible.json": [
      "playbooks/*.yml",
      "roles/*/tasks/*.yml",
      "roles/*/handlers/*.yml"
    ]
  },
  "files.associations": {
    "*.yml": "ansible",
    "*.yaml": "ansible"
  }
}
```

Recommended Extensions

- `redhat.ansible`
- `redhat.vscode-yaml`
- `ms-python.python`
- `ms-vscode.vscode-json`

Code Standards

YAML Formatting

```
---
# Document start required
# Use 2-space indentation
# Maximum line length: 120 characters
# Use explicit boolean values: true/false

- name: Example task with proper formatting
  ansible.builtin.package:
    name: "{{ package_name }}"
    state: present
  when: package_name is defined
  become: true
  tags:
    - packages
    - installation
```

Variable Naming Conventions

```yaml
# Role variables (snake_case with role prefix)
hx_ca_trust_certificate_path: "/etc/ssl/certs"
hx_ca_trust_update_command: "update-ca-certificates"

# Group variables (descriptive names)
database_connection_string: "postgresql://user:pass@host:5432/db"
application_log_level: "INFO"

# Host variables (specific to host)
server_hostname: "web01.example.com"
server_ip_address: "192.168.1.100"
```

Task Structure

```yaml
---
# tasks/main.yml - Standard task structure
- name: Include validation tasks
  ansible.builtin.include_tasks: validate.yml
  tags: [validation, always]

- name: Include preparation tasks
  ansible.builtin.include_tasks: prepare.yml
  tags: [preparation]

- name: Include installation tasks
  ansible.builtin.include_tasks: install.yml
  tags: [installation]

- name: Include configuration tasks
  ansible.builtin.include_tasks: configure.yml
  tags: [configuration]

- name: Include security tasks
  ansible.builtin.include_tasks: security.yml
  tags: [security]
```

Error Handling

```yaml
- name: Example with comprehensive error handling
  block:
    - name: Attempt primary action
      ansible.builtin.service:
        name: "{{ service_name }}"
        state: started
      register: service_result

  rescue:
    - name: Log error details
      ansible.builtin.debug:
        msg: "Failed to start {{ service_name }}: {{ ansible_failed_result.msg }}"

    - name: Attempt fallback action
      ansible.builtin.command:
        cmd: "systemctl reset-failed {{ service_name }}"
      changed_when: false

    - name: Retry primary action
      ansible.builtin.service:
        name: "{{ service_name }}"
        state: started

  always:
    - name: Verify service status
      ansible.builtin.service_facts:
      register: service_facts

    - name: Assert service is running
      ansible.builtin.assert:
        that:
          - "service_facts.ansible_facts.services[service_name + '.service'].state ==
'running'"
        fail_msg: "Service {{ service_name }} is not running"
```

Role Development

Standardized Role Structure

```
roles/hx_example_standardized/
├── README.md                    # Role documentation
├── meta/
│   └── main.yml                # Role metadata and dependencies
├── defaults/
│   └── main.yml                # Default variables
├── vars/
│   └── main.yml                # Role variables
├── tasks/
│   ├── main.yml                # Main task entry point
│   ├── validate.yml            # Input validation
│   ├── prepare.yml             # Preparation tasks
│   ├── install.yml             # Installation tasks
│   ├── configure.yml           # Configuration tasks
│   └── security.yml            # Security hardening
├── handlers/
│   └── main.yml                # Event handlers
├── templates/
│   └── config.j2               # Jinja2 templates
├── files/
│   └── static_file.conf        # Static files
└── tests/
    ├── inventory               # Test inventory
    └── test.yml                # Test playbook
```

Role Metadata Template

```yaml
---
# meta/main.yml
galaxy_info:
  role_name: hx_example_standardized
  author: HX Infrastructure Team
  description: Standardized role for example service deployment
  company: Hanax AI
  license: MIT
  min_ansible_version: "2.15"

  platforms:
    - name: Ubuntu
      versions:
        - "20.04"
        - "22.04"
    - name: CentOS
      versions:
        - "8"
        - "9"
    - name: RedHat
      versions:
        - "8"
        - "9"

  galaxy_tags:
    - system
    - security
    - infrastructure
    - hx

dependencies:
  - role: common
    when: hx_example_use_common | default(true)

collections:
  - ansible.posix
  - community.general
```

Variable Documentation

```yaml
---
# defaults/main.yml with comprehensive documentation

# Service Configuration
hx_example_service_name: "example-service"  # Name of the service
hx_example_service_port: 8080               # Service listening port
hx_example_service_user: "example"          # Service user account
hx_example_service_group: "example"         # Service group

# Installation Options
hx_example_package_name: "example-package"  # Package name to install
hx_example_package_version: "latest"        # Package version (latest/specific)
hx_example_install_method: "package"        # Installation method (package/source/
docker)

# Security Settings
hx_example_enable_ssl: true                 # Enable SSL/TLS
hx_example_ssl_cert_path: "/etc/ssl/certs/example.crt"  # SSL certificate path
hx_example_ssl_key_path: "/etc/ssl/private/example.key" # SSL private key path
hx_example_firewall_enabled: true          # Enable firewall rules

# Performance Tuning
hx_example_max_connections: 1000           # Maximum concurrent connections
hx_example_worker_processes: "auto"        # Number of worker processes
hx_example_memory_limit: "512M"            # Memory limit

# Monitoring
hx_example_enable_monitoring: true         # Enable monitoring
hx_example_metrics_port: 9090             # Metrics endpoint port
hx_example_health_check_path: "/health"   # Health check endpoint

# Backup Configuration
hx_example_backup_enabled: false          # Enable automated backups
hx_example_backup_schedule: "0 2 * * *"   # Backup cron schedule
hx_example_backup_retention: 30           # Backup retention days
```

Testing Framework

Unit Testing with Molecule

```yaml
---
# molecule/default/molecule.yml
dependency:
  name: galaxy
  options:
    requirements-file: requirements.yml

driver:
  name: docker

platforms:
  - name: ubuntu-20.04
    image: ubuntu:20.04
    pre_build_image: true
    privileged: true
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro
    command: /lib/systemd/systemd

  - name: ubuntu-22.04
    image: ubuntu:22.04
    pre_build_image: true
    privileged: true
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro
    command: /lib/systemd/systemd

provisioner:
  name: ansible
  config_options:
    defaults:
      callbacks_enabled: profile_tasks
      stdout_callback: yaml
  inventory:
    host_vars:
      ubuntu-20.04:
        hx_example_test_variable: "test_value_20"
      ubuntu-22.04:
        hx_example_test_variable: "test_value_22"

verifier:
  name: ansible

scenario:
  test_sequence:
    - dependency
    - lint
    - cleanup
    - destroy
    - syntax
    - create
    - prepare
    - converge
    - idempotence
    - side_effect
    - verify
    - cleanup
    - destroy
```

Test Playbooks

```yaml
---
# molecule/default/converge.yml
- name: Converge
  hosts: all
  become: true
  gather_facts: true

  pre_tasks:
    - name: Update package cache (Ubuntu)
      ansible.builtin.apt:
        update_cache: true
        cache_valid_time: 3600
      when: ansible_os_family == "Debian"

    - name: Update package cache (RedHat)
      ansible.builtin.yum:
        update_cache: true
      when: ansible_os_family == "RedHat"

  roles:
    - role: hx_example_standardized
      vars:
        hx_example_service_port: 8080
        hx_example_enable_ssl: false
        hx_example_install_method: "package"
```

Verification Tests

```yaml
---
# molecule/default/verify.yml
- name: Verify
  hosts: all
  gather_facts: false
  tasks:
    - name: Verify service is running
      ansible.builtin.service_facts:
      register: service_facts

    - name: Assert service is active
      ansible.builtin.assert:
        that:
          - "service_facts.ansible_facts.services['example-service.service'].state ==
'running'"
        fail_msg: "Service is not running"

    - name: Verify service is listening on correct port
      ansible.builtin.wait_for:
        port: 8080
        host: "{{ ansible_default_ipv4.address }}"
        timeout: 30
      register: port_check

    - name: Verify configuration file exists
      ansible.builtin.stat:
        path: "/etc/example/config.conf"
      register: config_file

    - name: Assert configuration file exists
      ansible.builtin.assert:
        that:
          - config_file.stat.exists
        fail_msg: "Configuration file not found"

    - name: Verify log file is created
      ansible.builtin.stat:
        path: "/var/log/example/service.log"
      register: log_file

    - name: Assert log file exists
      ansible.builtin.assert:
        that:
          - log_file.stat.exists
        fail_msg: "Log file not found"
```

Security Guidelines

Secrets Management

```
# Use Ansible Vault for sensitive data
# group_vars/all/vault.yml (encrypted)
$ANSIBLE_VAULT;1.1;AES256
66386439653...

# Reference in tasks
- name: Configure database connection
  ansible.builtin.template:
    src: database.conf.j2
    dest: /etc/app/database.conf
    mode: '0600'
    owner: app
    group: app
  vars:
    db_password: "{{ vault_database_password }}"
  no_log: true
```

Privilege Escalation

```
# Minimal privilege escalation
- name: Install package (requires root)
  ansible.builtin.package:
    name: "{{ package_name }}"
    state: present
  become: true
  become_user: root

# Service-specific user for application tasks
- name: Configure application
  ansible.builtin.template:
    src: app.conf.j2
    dest: /etc/app/app.conf
  become: true
  become_user: "{{ app_user }}"
```

Input Validation

```yaml
# tasks/validate.yml
- name: Validate required variables are defined
  ansible.builtin.assert:
    that:
      - hx_example_service_name is defined
      - hx_example_service_name | length > 0
      - hx_example_service_port is defined
      - hx_example_service_port | int > 0
      - hx_example_service_port | int < 65536
    fail_msg: "Required variables are not properly defined"

- name: Validate SSL configuration
  ansible.builtin.assert:
    that:
      - hx_example_ssl_cert_path is defined
      - hx_example_ssl_key_path is defined
    fail_msg: "SSL certificate and key paths must be defined when SSL is enabled"
  when: hx_example_enable_ssl | default(false)

- name: Validate installation method
  ansible.builtin.assert:
    that:
      - hx_example_install_method in ['package', 'source', 'docker']
    fail_msg: "Installation method must be one of: package, source, docker"
```

Documentation Standards

Role Documentation Template

~~~
# Role Name: hx_example_standardized

## Description

Brief description of what this role does and its purpose in the HX infrastructure.

## Requirements

- Ansible 2.15+
- Target OS: Ubuntu 20.04+, CentOS 8+
- Required privileges: sudo access

## Role Variables

### Required Variables

| Variable | Type | Description | Example |
|----------|------|-------------|---------|
| `hx_example_service_name` | string | Name of the service | `"example-service"` |
| `hx_example_service_port` | integer | Service port | `8080` |

### Optional Variables

| Variable | Type | Default | Description |
|----------|------|---------|-------------|
| `hx_example_enable_ssl` | boolean | `true` | Enable SSL/TLS |
| `hx_example_package_version` | string | `"latest"` | Package version |

## Dependencies

- `common` role (optional)
- `ansible.posix` collection
- `community.general` collection

## Example Playbook

```yaml
- hosts: servers
  roles:
    - role: hx_example_standardized
      vars:
        hx_example_service_port: 8080
        hx_example_enable_ssl: true
~~~

Testing

```
# Run molecule tests
cd roles/hx_example_standardized
molecule test

# Run specific scenario
molecule test -s ubuntu-20.04
```

License

MIT

Author Information

HX Infrastructure Team - infrastructure@hanax.ai

~~~
## Git Workflow

### Branch Strategy

```mermaid
graph LR
    A[main] --> B[feature/new-role]
    A --> C[hotfix/security-fix]
    A --> D[release/v1.2.0]

    B --> E[Pull Request]
    C --> F[Pull Request]
    D --> G[Pull Request]

    E --> A
    F --> A
    G --> A
~~~

Commit Message Format

```
type(scope): subject

body

footer
```

Types:
- `feat`: New feature
- `fix`: Bug fix
- `docs`: Documentation changes
- `style`: Code style changes
- `refactor`: Code refactoring
- `test`: Test additions/modifications
- `chore`: Maintenance tasks

Examples:

```
feat(roles): add hx_litellm_proxy_standardized role

- Implement LiteLLM proxy configuration
- Add SSL/TLS support
- Include health check endpoints
- Add comprehensive testing

Closes #123
```

Pre-commit Hooks

```yaml
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.4.0
    hooks:
      - id: trailing-whitespace
      - id: end-of-file-fixer
      - id: check-yaml
      - id: check-added-large-files

  - repo: https://github.com/ansible/ansible-lint
    rev: v6.17.2
    hooks:
      - id: ansible-lint

  - repo: https://github.com/adrienverge/yamllint
    rev: v1.32.0
    hooks:
      - id: yamllint
```

Code Review Process

Review Checklist

Functionality

- [ ] Code accomplishes intended purpose
- [ ] All edge cases are handled
- [ ] Error handling is comprehensive
- [ ] Idempotency is maintained

Security

- [ ] No hardcoded secrets
- [ ] Proper privilege escalation
- [ ] Input validation implemented
- [ ] Secure defaults used

Quality

- [ ] Code follows style guidelines
- [ ] Documentation is complete
- [ ] Tests are comprehensive
- [ ] Performance considerations addressed

Ansible Best Practices

- [ ] Tasks are idempotent
- [ ] Proper use of modules
- [ ] Appropriate use of handlers
- [ ] Correct variable scoping

Review Process

1. **Automated Checks**: CI/CD pipeline runs automatically
2. **Peer Review**: At least one team member reviews

3. **Security Review**: Security team reviews for sensitive changes
4. **Testing**: All tests must pass
5. **Documentation**: Documentation must be updated
6. **Approval**: Required approvals before merge

CI/CD Integration

GitHub Actions Workflow

```yaml
# .github/workflows/ci.yml
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.9'

      - name: Install dependencies
        run: |
          pip install ansible-lint yamllint
          ansible-galaxy install -r requirements.yml

      - name: Run ansible-lint
        run: ansible-lint .

      - name: Run yamllint
        run: yamllint .

  security:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Run security scan
        run: |
          make secrets-lint

      - name: Check for vulnerabilities
        run: |
          # Add vulnerability scanning tools
          echo "Security scan completed"

  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: [3.8, 3.9, '3.10']

    steps:
      - uses: actions/checkout@v3

      - name: Set up Python ${{ matrix.python-version }}
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python-version }}

      - name: Install dependencies
        run: |
          pip install molecule molecule-plugins[docker] pytest-testinfra
      - name: Run molecule tests
        run: |
          cd roles/hx_ca_trust_standardized && molecule test
          cd ../hx_domain_join_standardized && molecule test
```

Troubleshooting

Common Issues

Ansible Lint Errors

```
# Fix common lint issues
ansible-lint --fix .

# Ignore specific rules
# .ansible-lint
skip_list:
  - yaml[line-length]
  - name[casing]
```

Molecule Test Failures

```
# Debug molecule tests
molecule --debug test

# Keep containers for debugging
molecule test --destroy never

# Connect to test container
molecule login -h instance-name
```

Variable Precedence Issues

```
# Variable precedence (highest to lowest):
# 1. Extra vars (-e)
# 2. Task vars
# 3. Block vars
# 4. Role vars
# 5. Play vars
# 6. Host vars
# 7. Group vars
# 8. Role defaults

# Use debug to check variable values
- name: Debug variable value
  ansible.builtin.debug:
    var: variable_name
```

Performance Optimization

```
# Optimize Ansible performance
[defaults]
forks = 50
pipelining = True
gathering = smart
fact_caching = jsonfile
fact_caching_connection = /tmp/ansible_facts_cache
fact_caching_timeout = 86400

# Use async for long-running tasks
- name: Long running task
  ansible.builtin.command: /path/to/long/command
  async: 300
  poll: 10
```

Debugging Techniques

```
# Enable verbose output
ansible-playbook -vvv playbook.yml

# Use debug module
- name: Debug variable
  ansible.builtin.debug:
    var: ansible_facts
    verbosity: 2

# Use assert for validation
- name: Validate condition
  ansible.builtin.assert:
    that:
      - condition_to_check
    fail_msg: "Condition failed"
    success_msg: "Condition passed"
```

This development guide provides comprehensive standards and practices for contributing to the HX Infrastructure Ansible project, ensuring code quality, security, and maintainability.