HX-Infrastructure Ansible Directory Structure Guide

Overview

This document provides a comprehensive guide to the standardized directory structure implemented for the HX-Infrastructure Ansible project. The structure follows Ansible best practices and implements clear separation of concerns for enterprise-grade infrastructure automation.

Root Directory Structure

```
hx-infrastructure-ansible/
├── ansible.cfg                  # Ansible configuration
├── requirements.yml             # Galaxy role dependencies
├── site.yml                    # Main site playbook
├── .gitignore                  # Git ignore patterns
├── .ansible-lint               # Ansible linting configuration
├── inventories/                # Environment-specific inventories
├── group_vars/                # Group-specific variables
├── host_vars/                 # Host-specific variables
├── vault/                     # Encrypted secrets per environment
├── roles/                     # Reusable automation roles
├── playbooks/                # Orchestration playbooks
├── vars/                     # Additional variable files
├── templates/                # Jinja2 templates
├── files/                    # Static files for deployment
├── tests/                    # Testing framework
├── scripts/                  # Utility and maintenance scripts
├── docs/                     # Documentation
├── ci/                       # CI/CD workflows and pipelines
├── evidence/                 # Audit and compliance evidence
├── backup/                   # Backup configurations and data
├── monitoring/               # Monitoring configurations
├── security/                 # Security policies and certificates
├── logs/                     # Log storage and rotation
└── tmp/                      # Temporary files and staging
```

Detailed Directory Descriptions

Inventories Structure

Purpose: Environment-specific host definitions and inventory management

```
inventories/
├── dev/                    # Development environment
├── test/                   # Test environment
├── prod/                   # Production environment
└── shared/                 # Shared inventory components
```

Usage: Each environment directory contains:
- `hosts.yml` - Host definitions

- `group_vars/` - Environment-specific group variables
- `host_vars/` - Environment-specific host variables

Group Variables Structure

Purpose: Hierarchical variable management following Variable Hierarchy Design

```
group_vars/
├── all/                    # Variables for all hosts
├── dev/                    # Development environment variables
├── test/                   # Test environment variables
├── prod/                   # Production environment variables
├── web_servers/            # Web server group variables
├── database_servers/       # Database server group variables
├── load_balancers/         # Load balancer group variables
├── monitoring_servers/     # Monitoring server group variables
└── backup_servers/         # Backup server group variables
```

Variable Precedence: Follows Ansible's variable precedence order with environment-specific overrides.

Vault Structure

Purpose: Encrypted secrets management per environment

```
vault/
├── dev/                    # Development secrets
├── test/                   # Test secrets
└── prod/                   # Production secrets
```

Security: Each environment maintains separate vault files with appropriate encryption keys.

Roles Structure

Purpose: Reusable automation components for services and infrastructure

```
roles/
├── common/                 # Common system configuration
├── web_server/             # Web server automation
├── database/               # Database management
├── load_balancer/          # Load balancer configuration
├── monitoring/             # Monitoring setup
├── backup/                 # Backup automation
├── security/               # Security hardening
├── logging/                # Logging configuration
├── networking/             # Network configuration
└── storage/                # Storage management
```

Role Structure: Each role follows standard Ansible role structure with tasks, handlers, templates, files, vars, defaults, and meta directories.

Playbooks Structure

Purpose: Orchestration workflows for deployment and operations

```
playbooks/
├── site/                    # Site-wide playbooks
├── infrastructure/          # Infrastructure deployment
├── applications/            # Application deployment
├── maintenance/             # Maintenance procedures
├── deployment/              # Deployment workflows
├── backup/                  # Backup procedures
└── monitoring/              # Monitoring setup
```

Testing Structure

Purpose: Comprehensive testing framework for validation

```
tests/
├── unit/                    # Unit tests for individual components
├── integration/             # Integration tests for workflows
├── molecule/                # Molecule testing scenarios
└── functional/              # Functional testing suites
```

Scripts Structure

Purpose: Utility and maintenance automation

```
scripts/
├── deployment/              # Deployment utilities
├── maintenance/             # Maintenance scripts
├── backup/                  # Backup utilities
├── monitoring/              # Monitoring scripts
└── utilities/               # General utilities
```

Documentation Structure

Purpose: Comprehensive project documentation

```
docs/
├── architecture/            # System architecture documentation
├── deployment/              # Deployment procedures
├── operations/              # Operational procedures
├── troubleshooting/         # Troubleshooting guides
├── security/                # Security documentation
└── compliance/              # Compliance documentation
```

CI/CD Structure

Purpose: Continuous integration and deployment automation

```
ci/
├── pipelines/               # CI/CD pipeline definitions
├── workflows/               # Workflow configurations
├── templates/               # Pipeline templates
└── scripts/                 # CI/CD utility scripts
```

Evidence Structure

Purpose: Audit and compliance evidence tracking

```
evidence/
├── audits/                  # Audit evidence and reports
├── compliance/              # Compliance documentation
├── security/                # Security assessment evidence
└── backups/                 # Backup verification evidence
```

Separation of Concerns

Environment Separation

- **Development**: Isolated development and testing environment
- **Test**: Pre-production validation environment
- **Production**: Live production systems with strict change control

Service Separation

- **Infrastructure**: Core infrastructure components (networking, storage, security)
- **Applications**: Application-specific automation and deployment
- **Operations**: Day-to-day operational procedures and maintenance

Security Separation

- **Vault**: Encrypted secrets isolated per environment
- **Certificates**: SSL/TLS certificates and key management
- **Policies**: Security policies and compliance frameworks

Variable Hierarchy Implementation

The directory structure implements a comprehensive variable hierarchy:

1. **Global Variables**: `group_vars/all/`
2. **Environment Variables**: `group_vars/{env}/`
3. **Service Group Variables**: `group_vars/{service_group}/`
4. **Host Variables**: `host_vars/{hostname}/`
5. **Vault Variables**: `vault/{env}/`

Best Practices Implementation

File Organization

- All directories include `.keep` files for Git tracking
- Consistent naming conventions across all components
- Logical grouping of related functionality

Security Implementation

- Environment-specific vault separation
- Certificate and key management structure
- Security policy documentation framework

Testing Framework

- Multiple testing levels (unit, integration, functional)

- Molecule integration for role testing
- Evidence collection for compliance

Documentation Standards

- Comprehensive documentation structure
- Architecture and operational guides
- Troubleshooting and compliance documentation

Next Steps for Development

1. **Role Development**: Begin implementing individual roles following the established structure
2. **Inventory Population**: Populate environment-specific inventories with actual host definitions
3. **Variable Configuration**: Implement the variable hierarchy with actual configuration values
4. **Vault Setup**: Configure vault encryption and populate with environment secrets
5. **Testing Implementation**: Set up molecule testing scenarios for role validation
6. **CI/CD Pipeline**: Implement continuous integration and deployment workflows
7. **Documentation**: Populate documentation directories with detailed operational guides

Compliance and Audit Readiness

The structure is designed to support:
- **SOC 2 Compliance**: Evidence collection and audit trail maintenance
- **Security Audits**: Comprehensive security documentation and evidence
- **Change Management**: Structured deployment and rollback procedures
- **Backup Verification**: Automated backup testing and validation

This directory structure provides a solid foundation for enterprise-grade Ansible automation while maintaining flexibility for future expansion and customization.