HX Infrastructure User Guide

Operational Procedures and User Manual

This guide provides comprehensive instructions for using the HX Infrastructure Ansible automation platform for day-to-day operations, deployments, and maintenance tasks.

Table of Contents

1. Getting Started
2. Environment Management
3. Deployment Procedures
4. Configuration Management
5. Monitoring and Maintenance
6. Troubleshooting
7. Security Operations
8. Backup and Recovery
9. Performance Tuning
10. Best Practices

Getting Started

Prerequisites Checklist

Before using the HX Infrastructure Ansible platform, ensure you have:

- [ ] Ansible 2.15+ installed
- [ ] Python 3.8+ with required modules
- [ ] SSH access to target systems
- [ ] Vault password file configured
- [ ] Required permissions on target systems
- [ ] Network connectivity to all target hosts

Initial Setup

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

# 2. Install dependencies
make install

# 3. Verify setup
make lint test

# 4. Configure vault password
echo "your-vault-password" > .vault_pass
chmod 600 .vault_pass

# 5. Test connectivity
ansible all -i inventories/development/hosts.yml -m ping
```

Quick Start Commands

```
# Deploy to development environment
ansible-playbook -i inventories/development/hosts.yml playbooks/site.yml

# Deploy specific role
ansible-playbook -i inventories/development/hosts.yml playbooks/infrastructure/ca-
trust.yml

# Run with check mode (dry run)
ansible-playbook -i inventories/development/hosts.yml playbooks/site.yml --check --
diff

# Deploy to specific hosts
ansible-playbook -i inventories/development/hosts.yml playbooks/site.yml --limit web_s
ervers
```

Environment Management

Environment Structure

The HX Infrastructure supports multiple environments with isolated configurations:

```
inventories/
├── production/          # Production environment
│   ├── hosts.yml       # Production inventory
│   ├── group_vars/     # Production group variables
│   └── host_vars/      # Production host variables
├── staging/            # Staging environment
│   ├── hosts.yml       # Staging inventory
│   ├── group_vars/     # Staging group variables
│   └── host_vars/      # Staging host variables
├── development/        # Development environment
│   ├── hosts.yml       # Development inventory
│   ├── group_vars/     # Development group variables
│   └── host_vars/      # Development host variables
└── testing/            # Testing environment
    ├── hosts.yml       # Testing inventory
    ├── group_vars/     # Testing group variables
    └── host_vars/      # Testing host variables
```

Inventory Management

Adding New Hosts

```yaml
# inventories/production/hosts.yml
all:
  children:
    web_servers:
      hosts:
        web01.example.com:
          ansible_host: 192.168.1.10
          server_role: frontend
        web02.example.com:
          ansible_host: 192.168.1.11
          server_role: frontend

    database_servers:
      hosts:
        db01.example.com:
          ansible_host: 192.168.1.20
          server_role: database
          postgresql_version: "14"

    application_servers:
      hosts:
        app01.example.com:
          ansible_host: 192.168.1.30
          server_role: application
          java_version: "11"
```

Group Variables Configuration

```yaml
# inventories/production/group_vars/web_servers.yml
---
# Web server specific configuration
nginx_version: "1.20"
ssl_enabled: true
ssl_certificate_path: "/etc/ssl/certs/example.com.crt"
ssl_private_key_path: "/etc/ssl/private/example.com.key"

# Security settings
firewall_enabled: true
allowed_ports:
  - 80
  - 443
  - 22

# Performance tuning
worker_processes: "auto"
worker_connections: 1024
keepalive_timeout: 65
```

Environment Promotion

Development to Staging

```
# 1. Test in development
ansible-playbook -i inventories/development/hosts.yml playbooks/site.yml --check

# 2. Deploy to development
ansible-playbook -i inventories/development/hosts.yml playbooks/site.yml

# 3. Run validation tests
ansible-playbook -i inventories/development/hosts.yml tests/validation.yml

# 4. Promote to staging
ansible-playbook -i inventories/staging/hosts.yml playbooks/site.yml --check --diff

# 5. Deploy to staging
ansible-playbook -i inventories/staging/hosts.yml playbooks/site.yml
```

Staging to Production

```
# 1. Final staging validation
ansible-playbook -i inventories/staging/hosts.yml tests/comprehensive.yml

# 2. Production pre-deployment checks
ansible-playbook -i inventories/production/hosts.yml playbooks/pre-deployment-
checks.yml

# 3. Production deployment (with extra safety)
make prod-deploy

# 4. Post-deployment validation
ansible-playbook -i inventories/production/hosts.yml tests/post-deployment.yml
```

Deployment Procedures

Standard Deployment Workflow

Full Infrastructure Deployment

```
# 1. Pre-deployment validation
ansible-playbook -i inventories/production/hosts.yml playbooks/validation/pre-deploy-
ment.yml

# 2. Backup current configuration
ansible-playbook -i inventories/production/hosts.yml playbooks/maintenance/backup.yml

# 3. Deploy infrastructure components
ansible-playbook -i inventories/production/hosts.yml playbooks/infrastructure/site.yml

# 4. Deploy applications
ansible-playbook -i inventories/production/hosts.yml playbooks/applications/site.yml

# 5. Apply security hardening
ansible-playbook -i inventories/production/hosts.yml playbooks/security/hardening.yml

# 6. Post-deployment validation
ansible-playbook -i inventories/production/hosts.yml playbooks/validation/post-deploy-
ment.yml
```

Role-Specific Deployments

```
# Deploy CA trust configuration
ansible-playbook -i inventories/production/hosts.yml \
  -e "target_hosts=all" \
  playbooks/infrastructure/ca-trust.yml

# Deploy domain join configuration
ansible-playbook -i inventories/production/hosts.yml \
  -e "target_hosts=domain_members" \
  playbooks/infrastructure/domain-join.yml

# Deploy PostgreSQL authentication
ansible-playbook -i inventories/production/hosts.yml \
  -e "target_hosts=database_servers" \
  playbooks/infrastructure/pg-auth.yml

# Deploy Web UI components
ansible-playbook -i inventories/production/hosts.yml \
  -e "target_hosts=web_servers" \
  playbooks/applications/webui.yml

# Deploy LiteLLM proxy
ansible-playbook -i inventories/production/hosts.yml \
  -e "target_hosts=api_servers" \
  playbooks/applications/litellm-proxy.yml
```

Rolling Deployments

Web Server Rolling Update

```
# Deploy to web servers one at a time
ansible-playbook -i inventories/production/hosts.yml \
  --limit web_servers \
  --serial 1 \
  playbooks/applications/webui.yml

# Deploy to web servers in batches of 2
ansible-playbook -i inventories/production/hosts.yml \
  --limit web_servers \
  --serial 2 \
  playbooks/applications/webui.yml

# Deploy to 25% of web servers at a time
ansible-playbook -i inventories/production/hosts.yml \
  --limit web_servers \
  --serial 25% \
  playbooks/applications/webui.yml
```

Database Rolling Update

```
# Update secondary databases first
ansible-playbook -i inventories/production/hosts.yml \
  --limit database_servers:!primary_db \
  playbooks/infrastructure/pg-auth.yml

# Update primary database
ansible-playbook -i inventories/production/hosts.yml \
  --limit primary_db \
  playbooks/infrastructure/pg-auth.yml
```

Blue-Green Deployment

```
# 1. Deploy to green environment
ansible-playbook -i inventories/production/hosts.yml \
  -e "deployment_color=green" \
  playbooks/deployment/blue-green.yml

# 2. Run health checks on green
ansible-playbook -i inventories/production/hosts.yml \
  -e "deployment_color=green" \
  playbooks/validation/health-check.yml

# 3. Switch traffic to green
ansible-playbook -i inventories/production/hosts.yml \
  -e "active_color=green" \
  playbooks/deployment/traffic-switch.yml

# 4. Validate traffic switch
ansible-playbook -i inventories/production/hosts.yml \
  playbooks/validation/traffic-validation.yml
```

Configuration Management

Variable Management

Encrypted Variables (Vault)

```
# Create new vault file
ansible-vault create group_vars/production/vault.yml

# Edit existing vault file
ansible-vault edit group_vars/production/vault.yml

# View vault file
ansible-vault view group_vars/production/vault.yml

# Encrypt existing file
ansible-vault encrypt group_vars/production/secrets.yml

# Decrypt file
ansible-vault decrypt group_vars/production/secrets.yml

# Change vault password
ansible-vault rekey group_vars/production/vault.yml
```

Variable Precedence

Understanding Ansible variable precedence (highest to lowest):

1. Extra vars (`-e` command line)
2. Task vars
3. Block vars
4. Role vars
5. Play vars
6. Host vars
7. Group vars
8. Role defaults

Example Variable Structure

```yaml
# group_vars/all/common.yml
---
# Common variables for all hosts
timezone: "UTC"
ntp_servers:
  - "pool.ntp.org"
  - "time.google.com"

log_retention_days: 30
backup_retention_days: 90

# group_vars/all/vault.yml (encrypted)
---
# Encrypted sensitive variables
vault_database_password: "secure_password_123"
vault_api_keys:
  external_service: "api_key_xyz"
  monitoring_service: "monitor_key_abc"

vault_ssl_certificates:
  example_com:
    cert: |
      -----BEGIN CERTIFICATE-----
      ...
      -----END CERTIFICATE-----
    key: |
      -----BEGIN PRIVATE KEY-----
      ...
      -----END PRIVATE KEY-----
```

Template Management

Jinja2 Templates

```
{# templates/nginx.conf.j2 #}
user {{ nginx_user }};
worker_processes {{ nginx_worker_processes | default('auto') }};
pid /run/nginx.pid;

events {
    worker_connections {{ nginx_worker_connections | default(1024) }};
    use epoll;
    multi_accept on;
}

http {
    # Basic Settings
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout {{ nginx_keepalive_timeout | default(65) }};
    types_hash_max_size 2048;

    # SSL Settings
    {% if ssl_enabled | default(false) %}
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    {% endif %}

    # Virtual Hosts
    {% for vhost in nginx_vhosts %}
    server {
        listen {{ vhost.port | default(80) }};
        server_name {{ vhost.server_name }};
        root {{ vhost.document_root }};

        {% if vhost.ssl | default(false) %}
        listen 443 ssl http2;
        ssl_certificate {{ vhost.ssl_cert }};
        ssl_certificate_key {{ vhost.ssl_key }};
        {% endif %}

        location / {
            try_files $uri $uri/ =404;
        }
    }
    {% endfor %}
}
```

File Management

Static File Deployment

```yaml
# Deploy configuration files
- name: Deploy application configuration
  ansible.builtin.copy:
    src: "{{ item }}"
    dest: "/etc/app/{{ item }}"
    owner: app
    group: app
    mode: '0644'
    backup: true
  loop:
    - app.conf
    - logging.conf
    - database.conf
  notify: restart application

# Deploy with template processing
- name: Deploy templated configuration
  ansible.builtin.template:
    src: "{{ item.src }}"
    dest: "{{ item.dest }}"
    owner: "{{ item.owner | default('root') }}"
    group: "{{ item.group | default('root') }}"
    mode: "{{ item.mode | default('0644') }}"
    backup: true
  loop:
    - src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
      owner: nginx
      group: nginx
    - src: app.conf.j2
      dest: /etc/app/app.conf
      owner: app
      group: app
      mode: '0600'
  notify:
    - restart nginx
    - restart application
```

Monitoring and Maintenance

Health Checks

System Health Validation

```
# Run comprehensive health checks
ansible-playbook -i inventories/production/hosts.yml playbooks/monitoring/health-
check.yml

# Check specific services
ansible-playbook -i inventories/production/hosts.yml \
  -e "services=['nginx', 'postgresql', 'application']" \
  playbooks/monitoring/service-check.yml

# Validate SSL certificates
ansible-playbook -i inventories/production/hosts.yml playbooks/monitoring/ssl-
check.yml
```

Custom Health Check Playbook

```yaml
---
# playbooks/monitoring/health-check.yml
- name: Comprehensive Health Check
  hosts: all
  gather_facts: true
  tasks:
    - name: Check system uptime
      ansible.builtin.command: uptime
      register: uptime_result
      changed_when: false

    - name: Check disk space
      ansible.builtin.shell: df -h | grep -E '^/dev/'
      register: disk_space
      changed_when: false

    - name: Check memory usage
      ansible.builtin.shell: free -h
      register: memory_usage
      changed_when: false

    - name: Check running services
      ansible.builtin.service_facts:
      register: service_facts

    - name: Validate critical services
      ansible.builtin.assert:
        that:
          - "service_facts.ansible_facts.services[item + '.service'].state == 'run-
ning'"
        fail_msg: "Service {{ item }} is not running"
      loop:
        - nginx
        - postgresql
        - ssh

    - name: Check network connectivity
      ansible.builtin.uri:
        url: "http://{{ ansible_default_ipv4.address }}"
        method: GET
        timeout: 10
      register: connectivity_check
      failed_when: connectivity_check.status != 200

    - name: Generate health report
      ansible.builtin.template:
        src: health-report.j2
        dest: "/tmp/health-report-{{ ansible_hostname }}-{{ ansible_date_time.epoch }}
.txt"
      delegate_to: localhost
```

Log Management

Log Collection

```
# Collect logs from all servers
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.fetch \
  -a "src=/var/log/application.log dest=./logs/ flat=yes"

# Collect system logs
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.fetch \
  -a "src=/var/log/syslog dest=./logs/{{ inventory_hostname }}-syslog flat=no"

# Rotate logs
ansible-playbook -i inventories/production/hosts.yml playbooks/maintenance/log-rota-
tion.yml
```

Log Analysis Playbook

```
---
# playbooks/monitoring/log-analysis.yml
- name: Log Analysis and Alerting
  hosts: all
  tasks:
    - name: Check for error patterns in application logs
      ansible.builtin.shell: |
        grep -i "error\|exception\|failed" /var/log/application.log | tail -20
      register: error_logs
      changed_when: false
      failed_when: false

    - name: Check disk space usage
      ansible.builtin.shell: df / | awk 'NR==2 {print $5}' | sed 's/%//'
      register: disk_usage
      changed_when: false

    - name: Alert on high disk usage
      ansible.builtin.debug:
        msg: "WARNING: Disk usage is {{ disk_usage.stdout }}% on {{
inventory_hostname }}"
      when: disk_usage.stdout | int > 80

    - name: Check for authentication failures
      ansible.builtin.shell: |
        grep "authentication failure" /var/log/auth.log | wc -l
      register: auth_failures
      changed_when: false

    - name: Alert on authentication failures
      ansible.builtin.debug:
        msg: "WARNING: {{ auth_failures.stdout }} authentication failures detected"
      when: auth_failures.stdout | int > 10
```

Performance Monitoring

System Performance Check

```
# Run performance monitoring
ansible-playbook -i inventories/production/hosts.yml playbooks/monitoring/perform-
ance.yml

# Check resource utilization
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.shell \
  -a "top -bn1 | head -20"

# Network performance test
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.shell \
  -a "iperf3 -c performance-server -t 10"
```

Troubleshooting

Common Issues and Solutions

Connection Issues

```
# Test SSH connectivity
ansible all -i inventories/production/hosts.yml -m ping

# Debug SSH connection
ansible all -i inventories/production/hosts.yml -m ping -vvv

# Test with different user
ansible all -i inventories/production/hosts.yml -u different_user -m ping

# Check SSH key authentication
ssh-add -l
ssh -T git@github.com
```

Permission Issues

```
# Check sudo permissions
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.shell \
  -a "sudo -l" \
  --become

# Test privilege escalation
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.shell \
  -a "whoami" \
  --become

# Fix file permissions
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.file \
  -a "path=/etc/app/config.conf owner=app group=app mode=0644" \
  --become
```

Service Issues

```
# Check service status
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.systemd \
  -a "name=nginx state=started enabled=yes"

# Restart services
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.systemd \
  -a "name=nginx state=restarted"

# Check service logs
ansible all -i inventories/production/hosts.yml \
  -m ansible.builtin.shell \
  -a "journalctl -u nginx --no-pager -n 50"
```

Debugging Techniques

Verbose Output

```
# Increase verbosity levels
ansible-playbook -v playbook.yml    # Basic verbose
ansible-playbook -vv playbook.yml   # More verbose
ansible-playbook -vvv playbook.yml  # Very verbose
ansible-playbook -vvvv playbook.yml # Debug level
```

Debug Module Usage

```
# Debug variable values
- name: Debug variable
  ansible.builtin.debug:
    var: variable_name

# Debug with custom message
- name: Debug with message
  ansible.builtin.debug:
    msg: "The value of variable_name is {{ variable_name }}"

# Conditional debug
- name: Debug when condition is met
  ansible.builtin.debug:
    msg: "Condition is true"
  when: some_condition | default(false)

# Debug facts
- name: Debug all facts
  ansible.builtin.debug:
    var: ansible_facts
    verbosity: 2
```

Task Debugging

```yaml
# Register task output for debugging
- name: Run command and register output
  ansible.builtin.command: some_command
  register: command_result

- name: Debug command result
  ansible.builtin.debug:
    var: command_result

# Use failed_when for custom failure conditions
- name: Custom failure condition
  ansible.builtin.command: some_command
  register: result
  failed_when: "'ERROR' in result.stdout"

# Use changed_when for custom change detection
- name: Custom change detection
  ansible.builtin.command: some_command
  register: result
  changed_when: "'CHANGED' in result.stdout"
```

Recovery Procedures

Service Recovery

```bash
# Automated service recovery
ansible-playbook -i inventories/production/hosts.yml playbooks/recovery/service-
recovery.yml

# Database recovery
ansible-playbook -i inventories/production/hosts.yml \
  -e "recovery_point='2023-12-01 10:00:00'" \
  playbooks/recovery/database-recovery.yml

# Configuration rollback
ansible-playbook -i inventories/production/hosts.yml \
  -e "rollback_version=previous" \
  playbooks/recovery/config-rollback.yml
```

Disaster Recovery

```yaml
---
# playbooks/recovery/disaster-recovery.yml
- name: Disaster Recovery Procedure
  hosts: all
  serial: 1
  tasks:
    - name: Stop all services
      ansible.builtin.systemd:
        name: "{{ item }}"
        state: stopped
      loop:
        - nginx
        - application
        - postgresql

    - name: Restore from backup
      ansible.builtin.unarchive:
        src: "{{ backup_location }}/{{ inventory_hostname }}-{{ recovery_date }}.tar.g
z"
        dest: /
        remote_src: true

    - name: Start services in order
      ansible.builtin.systemd:
        name: "{{ item }}"
        state: started
        enabled: true
      loop:
        - postgresql
        - application
        - nginx

    - name: Verify service health
      ansible.builtin.uri:
        url: "http://{{ ansible_default_ipv4.address }}/health"
        method: GET
      retries: 5
      delay: 10
```

Security Operations

Security Hardening

System Hardening Playbook

```bash
# Run security hardening
ansible-playbook -i inventories/production/hosts.yml playbooks/security/hardening.yml

# Apply specific security policies
ansible-playbook -i inventories/production/hosts.yml \
  -e "security_policy=pci_dss" \
  playbooks/security/compliance.yml

# Update security patches
ansible-playbook -i inventories/production/hosts.yml playbooks/security/patch-manage-
ment.yml
```

SSL/TLS Management

```
# Deploy SSL certificates
ansible-playbook -i inventories/production/hosts.yml \
  -e "certificate_domain=example.com" \
  playbooks/security/ssl-deployment.yml

# Renew certificates
ansible-playbook -i inventories/production/hosts.yml playbooks/security/ssl-
renewal.yml

# Validate SSL configuration
ansible-playbook -i inventories/production/hosts.yml playbooks/security/ssl-valida-
tion.yml
```

Access Control

User Management

```
# Add new user
ansible-playbook -i inventories/production/hosts.yml \
  -e "username=newuser" \
  -e "user_groups=['sudo', 'developers']" \
  playbooks/security/user-management.yml

# Remove user access
ansible-playbook -i inventories/production/hosts.yml \
  -e "username=olduser" \
  -e "action=remove" \
  playbooks/security/user-management.yml

# Update SSH keys
ansible-playbook -i inventories/production/hosts.yml \
  -e "username=admin" \
  -e "ssh_key_file=~/.ssh/id_rsa.pub" \
  playbooks/security/ssh-key-management.yml
```

Security Auditing

Security Audit Playbook

```
# Run comprehensive security audit
ansible-playbook -i inventories/production/hosts.yml playbooks/security/audit.yml

# Check for vulnerabilities
ansible-playbook -i inventories/production/hosts.yml playbooks/security/vulnerability-
scan.yml

# Generate compliance report
ansible-playbook -i inventories/production/hosts.yml \
  -e "compliance_standard=soc2" \
  playbooks/security/compliance-report.yml
```

Backup and Recovery

Backup Procedures

Automated Backup

```
# Run full backup
ansible-playbook -i inventories/production/hosts.yml playbooks/backup/full-backup.yml

# Run incremental backup
ansible-playbook -i inventories/production/hosts.yml playbooks/backup/incremental-
backup.yml

# Backup specific services
ansible-playbook -i inventories/production/hosts.yml \
  -e "backup_services=['database', 'application_data']" \
  playbooks/backup/service-backup.yml
```

Backup Validation

```
# Validate backup integrity
ansible-playbook -i inventories/production/hosts.yml playbooks/backup/backup-valida-
tion.yml

# Test restore procedure
ansible-playbook -i inventories/staging/hosts.yml \
  -e "backup_file=/backups/production-20231201.tar.gz" \
  playbooks/backup/restore-test.yml
```

Recovery Procedures

Point-in-Time Recovery

```
# Database point-in-time recovery
ansible-playbook -i inventories/production/hosts.yml \
  -e "recovery_time='2023-12-01 14:30:00'" \
  playbooks/recovery/database-pitr.yml

# File system recovery
ansible-playbook -i inventories/production/hosts.yml \
  -e "recovery_path='/var/www/html'" \
  -e "recovery_date='2023-12-01'" \
  playbooks/recovery/filesystem-recovery.yml
```

Performance Tuning

System Optimization

Performance Tuning Playbook

```
# Apply performance optimizations
ansible-playbook -i inventories/production/hosts.yml playbooks/performance/optimiza-
tion.yml

# Database performance tuning
ansible-playbook -i inventories/production/hosts.yml \
  -e "db_type=postgresql" \
  playbooks/performance/database-tuning.yml

# Web server optimization
ansible-playbook -i inventories/production/hosts.yml playbooks/performance/web-optim-
ization.yml
```

Monitoring Performance

Performance Metrics Collection

```
# Collect performance metrics
ansible-playbook -i inventories/production/hosts.yml playbooks/monitoring/performance-
metrics.yml

# Generate performance report
ansible-playbook -i inventories/production/hosts.yml \
  -e "report_period=7days" \
  playbooks/monitoring/performance-report.yml
```

Best Practices

Operational Excellence

1. **Always use check mode first**
   `bash`
   `ansible-playbook --check --diff playbook.yml`

2. **Limit scope when possible**
   `bash`
   `ansible-playbook --limit web_servers playbook.yml`

3. **Use tags for selective execution**
   `bash`
   `ansible-playbook --tags "configuration,security" playbook.yml`

4. **Backup before changes**
   `bash`
   `ansible-playbook backup.yml && ansible-playbook deployment.yml`

5. **Validate after deployment**
   `bash`
   `ansible-playbook deployment.yml && ansible-playbook validation.yml`

Security Best Practices

1. **Use Ansible Vault for secrets**
2. **Implement least privilege access**
3. **Regular security audits**
4. **Keep systems updated**
5. **Monitor access logs**

Performance Best Practices

1. **Use SSH pipelining**
2. **Optimize fact gathering**
3. **Use async for long tasks**
4. **Implement proper caching**
5. **Monitor resource usage**

This user guide provides comprehensive operational procedures for effectively using the HX Infrastructure Ansible platform in production environments.