Deployment Runbook

Overview

This runbook provides step-by-step procedures for deploying the HX-Infrastructure-Ansible automation platform across different environments.

Prerequisites

System Requirements

- Ansible 6.0+ installed
- Python 3.9+ with required dependencies
- SSH access to target hosts
- Appropriate permissions for deployment user

Access Requirements

- SSH keys configured for target environments
- Vault passwords for encrypted variables
- Network access to all target hosts
- Required service accounts and API tokens

Pre-Deployment Checklist

1. Environment Validation

```
# Verify Ansible installation
ansible --version

# Test connectivity to target hosts
ansible all -i environments/production/inventory.yml -m ping

# Validate inventory configuration
ansible-inventory -i environments/production/inventory.yml --list

# Check vault access
ansible-vault view environments/production/group_vars/all/vault.yml
```

2. Code Quality Verification

```
# Run syntax check
ansible-playbook playbooks/deployment/deploy.yml --syntax-check

# Run linting
ansible-lint playbooks/deployment/deploy.yml

# Run dry run
ansible-playbook playbooks/deployment/deploy.yml -i environments/production/invent-
ory.yml --check --diff
```

3. Backup Verification

```
# Verify backup systems are operational
ansible-playbook playbooks/maintenance/backup.yml -i environments/production/invent-
ory.yml --check

# Confirm backup retention policies
./scripts/backup/verify_backups.sh
```

Deployment Procedures

Development Environment Deployment

1. Preparation

```
# Switch to development environment
export ANSIBLE_INVENTORY=environments/development/inventory.yml
export ANSIBLE_VAULT_PASSWORD_FILE=.vault_pass_dev

# Verify environment
ansible-inventory --list
```

2. Deployment Execution

```
# Deploy base infrastructure
ansible-playbook playbooks/deployment/deploy.yml \
  -i environments/development/inventory.yml \
  --tags "base,security" \
  --limit development

# Deploy application components
ansible-playbook playbooks/deployment/deploy.yml \
  -i environments/development/inventory.yml \
  --tags "application" \
  --limit development

# Verify deployment
ansible-playbook tests/smoke/development_smoke_tests.yml \
  -i environments/development/inventory.yml
```

3. Post-Deployment Validation

```
# Check service status
ansible all -i environments/development/inventory.yml \
  -m service -a "name=nginx state=started"

# Verify application endpoints
curl -f http://dev-app.example.com/health

# Run integration tests
pytest tests/integration/test_development.py
```

Staging Environment Deployment

1. Preparation

```
# Switch to staging environment
export ANSIBLE_INVENTORY=environments/staging/inventory.yml
export ANSIBLE_VAULT_PASSWORD_FILE=.vault_pass_staging

# Verify staging readiness
ansible-playbook playbooks/deployment/pre_deploy_checks.yml \
  -i environments/staging/inventory.yml
```

2. Blue-Green Deployment

```
# Deploy to green environment
ansible-playbook playbooks/deployment/deploy.yml \
  -i environments/staging/inventory.yml \
  --extra-vars "deployment_slot=green" \
  --tags "all"

# Run smoke tests on green
ansible-playbook tests/smoke/staging_smoke_tests.yml \
  -i environments/staging/inventory.yml \
  --extra-vars "target_slot=green"

# Switch traffic to green
ansible-playbook playbooks/deployment/switch_traffic.yml \
  -i environments/staging/inventory.yml \
  --extra-vars "active_slot=green"

# Verify traffic switch
curl -f http://staging-app.example.com/health
```

3. Rollback Procedure (if needed)

```
# Switch traffic back to blue
ansible-playbook playbooks/deployment/switch_traffic.yml \
  -i environments/staging/inventory.yml \
  --extra-vars "active_slot=blue"

# Verify rollback
curl -f http://staging-app.example.com/health
```

Production Environment Deployment

1. Pre-Production Checklist

- [ ] Staging deployment successful
- [ ] All tests passing
- [ ] Security scans completed
- [ ] Change management approval obtained
- [ ] Maintenance window scheduled
- [ ] Rollback plan reviewed
- [ ] Team notifications sent

2. Production Deployment

```
# Switch to production environment
export ANSIBLE_INVENTORY=environments/production/inventory.yml
export ANSIBLE_VAULT_PASSWORD_FILE=.vault_pass_prod

# Create deployment snapshot
ansible-playbook playbooks/maintenance/create_snapshot.yml \
  -i environments/production/inventory.yml

# Deploy with rolling update strategy
ansible-playbook playbooks/deployment/deploy.yml \
  -i environments/production/inventory.yml \
  --extra-vars "deployment_strategy=rolling" \
  --serial 2

# Monitor deployment progress
watch -n 5 'ansible all -i environments/production/inventory.yml -m service -a
"name=app status=status"'
```

3. Production Validation

```
# Run comprehensive smoke tests
ansible-playbook tests/smoke/production_smoke_tests.yml \
  -i environments/production/inventory.yml

# Verify all services
ansible-playbook playbooks/monitoring/service_check.yml \
  -i environments/production/inventory.yml

# Check application metrics
curl -f http://prod-app.example.com/metrics

# Verify database connectivity
ansible-playbook tests/integration/test_database.yml \
  -i environments/production/inventory.yml
```

Monitoring and Alerting

Deployment Monitoring

```
# Monitor deployment logs
tail -f /var/log/ansible/deployment.log

# Check system metrics
ansible all -i environments/production/inventory.yml \
  -m setup -a "filter=ansible_memory_mb"

# Monitor application logs
ansible all -i environments/production/inventory.yml \
  -m shell -a "tail -n 50 /var/log/app/application.log"
```

Alert Configuration

```
# Enable deployment alerts
ansible-playbook playbooks/monitoring/enable_alerts.yml \
  -i environments/production/inventory.yml \
  --extra-vars "alert_level=deployment"

# Test alert channels
ansible-playbook playbooks/monitoring/test_alerts.yml \
  -i environments/production/inventory.yml
```

Rollback Procedures

Automated Rollback

```
# Execute automated rollback
ansible-playbook playbooks/deployment/rollback.yml \
  -i environments/production/inventory.yml \
  --extra-vars "rollback_version=previous"

# Verify rollback success
ansible-playbook tests/smoke/production_smoke_tests.yml \
  -i environments/production/inventory.yml
```

Manual Rollback Steps

1. **Stop new deployment**
   `bash`
   `ansible-playbook playbooks/deployment/stop_deployment.yml \`
   `-i environments/production/inventory.yml`

2. **Restore from backup**
   `bash`
   `ansible-playbook playbooks/maintenance/restore_backup.yml \`
   `-i environments/production/inventory.yml \`
   `--extra-vars "backup_timestamp=YYYY-MM-DD-HH-MM-SS"`

3. **Restart services**
   `bash`
   `ansible-playbook playbooks/deployment/restart_services.yml \`
   `-i environments/production/inventory.yml`

4. **Verify rollback**
   `bash`
   `ansible-playbook tests/smoke/production_smoke_tests.yml \`
   `-i environments/production/inventory.yml`

Troubleshooting

Common Issues

Deployment Failures

```
# Check deployment logs
ansible-playbook playbooks/troubleshooting/collect_logs.yml \
  -i environments/production/inventory.yml

# Verify connectivity
ansible all -i environments/production/inventory.yml -m ping

# Check disk space
ansible all -i environments/production/inventory.yml \
  -m shell -a "df -h"
```

Service Issues

```
# Check service status
ansible all -i environments/production/inventory.yml \
  -m systemd -a "name=nginx state=status"

# Restart failed services
ansible all -i environments/production/inventory.yml \
  -m systemd -a "name=nginx state=restarted"

# Check service logs
ansible all -i environments/production/inventory.yml \
  -m shell -a "journalctl -u nginx -n 50"
```

Performance Issues

```
# Check system resources
ansible all -i environments/production/inventory.yml \
  -m shell -a "top -bn1 | head -20"

# Monitor network connectivity
ansible all -i environments/production/inventory.yml \
  -m shell -a "netstat -tuln"

# Check application performance
ansible all -i environments/production/inventory.yml \
  -m uri -a "url=http://localhost/health"
```

Post-Deployment Tasks

1. Documentation Updates

- [ ] Update deployment logs
- [ ] Record configuration changes
- [ ] Update runbook if needed
- [ ] Document any issues encountered

2. Team Communication

- [ ] Notify stakeholders of completion
- [ ] Update status dashboards
- [ ] Schedule post-deployment review
- [ ] Update change management system

3. Monitoring Setup

- [ ] Verify monitoring is active
- [ ] Check alert configurations
- [ ] Update dashboards
- [ ] Schedule health checks

Emergency Contacts

On-Call Rotation

- **Primary**: DevOps Team Lead
- **Secondary**: Senior Infrastructure Engineer
- **Escalation**: Engineering Manager

Communication Channels

- **Slack**: #infrastructure-alerts
- **Email**: devops-team@company.com
- **Phone**: Emergency hotline

Appendix

Environment-Specific Variables

- Development: `environments/development/group_vars/`
- Staging: `environments/staging/group_vars/`
- Production: `environments/production/group_vars/`

Useful Commands

```
# Quick health check
ansible all -i inventory.yml -m ping

# Service status check
ansible all -i inventory.yml -m service -a "name=nginx"

# Disk usage check
ansible all -i inventory.yml -m shell -a "df -h"

# Memory usage check
ansible all -i inventory.yml -m shell -a "free -m"

# Process check
ansible all -i inventory.yml -m shell -a "ps aux | grep nginx"
```

Related Documentation

- Troubleshooting Guide (TROUBLESHOOTING_GUIDE.md)
- Security Procedures (SECURITY_PROCEDURES.md)
- Monitoring Guide (MONITORING_GUIDE.md)
- Backup and Recovery (BACKUP_RECOVERY.md)