Troubleshooting Guide

Overview

This guide provides systematic approaches to diagnosing and resolving common issues in the HX-Infrastructure-Ansible automation platform.

General Troubleshooting Methodology

1. Problem Identification

- **Gather Information**: Collect error messages, logs, and symptoms
- **Reproduce Issue**: Attempt to reproduce the problem consistently
- **Check Recent Changes**: Review recent deployments or configuration changes
- **Verify Environment**: Confirm the affected environment and scope

2. Initial Diagnostics

```
# Check Ansible connectivity
ansible all -i inventory.yml -m ping

# Verify inventory configuration
ansible-inventory -i inventory.yml --list

# Check Ansible configuration
ansible-config dump --only-changed

# Review recent playbook runs
tail -f /var/log/ansible/ansible.log
```

3. Systematic Investigation

- **Start with basics**: Network, DNS, authentication
- **Check dependencies**: Services, packages, configurations
- **Review logs**: System logs, application logs, Ansible logs
- **Test incrementally**: Isolate components and test individually

Common Issues and Solutions

Ansible Connectivity Issues

SSH Connection Failures

Symptoms:
- “UNREACHABLE” errors
- SSH timeout messages
- Authentication failures

Diagnosis:

```
# Test SSH connectivity manually
ssh -i ~/.ssh/id_rsa user@target-host

# Check SSH configuration
ansible all -i inventory.yml -m setup -a "filter=ansible_ssh*"

# Verify SSH keys
ssh-add -l

# Test with verbose output
ansible all -i inventory.yml -m ping -vvv
```

Solutions:

```
# Fix SSH key permissions
chmod 600 ~/.ssh/id_rsa
chmod 644 ~/.ssh/id_rsa.pub

# Add SSH key to agent
ssh-add ~/.ssh/id_rsa

# Update inventory with correct SSH settings
# In inventory.yml:
# ansible_ssh_private_key_file: ~/.ssh/id_rsa
# ansible_ssh_user: ubuntu
# ansible_ssh_common_args: '-o StrictHostKeyChecking=no'

# Test connectivity with specific user
ansible all -i inventory.yml -u ubuntu -m ping
```

DNS Resolution Issues

Symptoms:

- “Name or service not known” errors
- Intermittent connectivity issues

Diagnosis:

```
# Test DNS resolution
nslookup target-host
dig target-host

# Check /etc/hosts
cat /etc/hosts

# Test with IP addresses
ansible all -i inventory.yml -m ping --limit "192.168.1.100"
```

Solutions:

```
# Add entries to /etc/hosts
echo "192.168.1.100 target-host" >> /etc/hosts

# Use IP addresses in inventory
# In inventory.yml:
# target-host ansible_host=192.168.1.100

# Configure DNS servers
# In /etc/resolv.conf:
# nameserver 8.8.8.8
# nameserver 8.8.4.4
```

Playbook Execution Issues

Task Failures

Symptoms:

- Tasks failing with specific error messages
- Playbook execution stops unexpectedly

Diagnosis:

```
# Run with increased verbosity
ansible-playbook playbook.yml -i inventory.yml -vvv

# Check specific task
ansible-playbook playbook.yml -i inventory.yml --start-at-task "Task Name"

# Run in check mode
ansible-playbook playbook.yml -i inventory.yml --check --diff

# Test specific module
ansible all -i inventory.yml -m module_name -a "module_args"
```

Solutions:

```
# Fix syntax errors
ansible-playbook playbook.yml --syntax-check

# Update module parameters
# Check module documentation:
ansible-doc module_name

# Handle errors gracefully
# In playbook:
- name: Task that might fail
  module_name:
    param: value
  ignore_errors: yes
  register: result

- name: Handle failure
  debug:
    msg: "Task failed: {{ result.msg }}"
  when: result.failed
```

Variable Issues

Symptoms:
- “undefined variable” errors
- Incorrect variable values
- Template rendering failures

Diagnosis:

```
# Check variable precedence
ansible-playbook playbook.yml -i inventory.yml --list-hosts --list-tasks

# Debug variables
ansible all -i inventory.yml -m debug -a "var=variable_name"

# Check variable files
ansible-vault view group_vars/all/vault.yml

# Test template rendering
ansible all -i inventory.yml -m template -a "src=template.j2 dest=/tmp/test"
```

Solutions:

```
# Define default values
# In playbook:
vars:
  variable_name: "{{ variable_name | default('default_value') }}"

# Check variable scope
# Group variables: group_vars/group_name/vars.yml
# Host variables: host_vars/hostname/vars.yml

# Fix template syntax
# In template.j2:
{{ variable_name | default('default') }}
{% if variable_name is defined %}
  {{ variable_name }}
{% endif %}
```

Performance Issues

Slow Playbook Execution

Symptoms:
- Playbooks taking excessive time to complete
- Tasks hanging or timing out

Diagnosis:

```
# Enable timing
export ANSIBLE_CALLBACKS_ENABLED=timer,profile_tasks

# Run with profiling
ansible-playbook playbook.yml -i inventory.yml

# Check system resources
ansible all -i inventory.yml -m shell -a "top -bn1 | head -10"

# Monitor network usage
ansible all -i inventory.yml -m shell -a "netstat -i"
```

Solutions:

```
# Increase parallelism
# In ansible.cfg:
[defaults]
forks = 20
host_key_checking = False

# Use pipelining
[ssh_connection]
pipelining = True

# Optimize gathering facts
# In playbook:
gather_facts: no

# Or gather specific facts:
- name: Gather minimal facts
  setup:
    filter: ansible_os_family
```

Memory Issues

Symptoms:

- Out of memory errors
- System becoming unresponsive

Diagnosis:

```
# Check memory usage
ansible all -i inventory.yml -m shell -a "free -m"

# Monitor memory during execution
watch -n 1 'free -m'

# Check swap usage
ansible all -i inventory.yml -m shell -a "swapon -s"
```

Solutions:

```
# Reduce batch size
# In playbook:
serial: 2

# Limit concurrent tasks
# In ansible.cfg:
[defaults]
forks = 5

# Add swap space
ansible all -i inventory.yml -m shell -a "fallocate -l 2G /swapfile"
ansible all -i inventory.yml -m shell -a "chmod 600 /swapfile"
ansible all -i inventory.yml -m shell -a "mkswap /swapfile"
ansible all -i inventory.yml -m shell -a "swapon /swapfile"
```

Service and Application Issues

Service Start Failures

Symptoms:

- Services failing to start
- Service status showing as failed

Diagnosis:

```
# Check service status
ansible all -i inventory.yml -m systemd -a "name=nginx"

# View service logs
ansible all -i inventory.yml -m shell -a "journalctl -u nginx -n 50"

# Check service configuration
ansible all -i inventory.yml -m shell -a "systemctl cat nginx"

# Test service manually
ansible all -i inventory.yml -m shell -a "nginx -t"
```

Solutions:

```
# Fix configuration errors
ansible all -i inventory.yml -m lineinfile -a "path=/etc/nginx/nginx.conf
line='worker_processes auto;'"

# Restart service
ansible all -i inventory.yml -m systemd -a "name=nginx state=restarted"

# Enable service
ansible all -i inventory.yml -m systemd -a "name=nginx enabled=yes"

# Check dependencies
ansible all -i inventory.yml -m shell -a "systemctl list-dependencies nginx"
```

Port Binding Issues

Symptoms:

- “Address already in use” errors
- Services unable to bind to ports

Diagnosis:

```
# Check port usage
ansible all -i inventory.yml -m shell -a "netstat -tuln | grep :80"

# Find process using port
ansible all -i inventory.yml -m shell -a "lsof -i :80"

# Check firewall rules
ansible all -i inventory.yml -m shell -a "iptables -L"
```

Solutions:

```
# Kill process using port
ansible all -i inventory.yml -m shell -a "pkill -f nginx"

# Change service port
# In service configuration:
listen 8080;

# Open firewall port
ansible all -i inventory.yml -m ufw -a "rule=allow port=80"
```

Security and Permission Issues

Permission Denied Errors

Symptoms:
- “Permission denied” errors
- Unable to write files or execute commands

Diagnosis:

```
# Check file permissions
ansible all -i inventory.yml -m shell -a "ls -la /path/to/file"

# Check user and group
ansible all -i inventory.yml -m shell -a "id"

# Check sudo access
ansible all -i inventory.yml -m shell -a "sudo -l"
```

Solutions:

```
# Fix file permissions
ansible all -i inventory.yml -m file -a "path=/path/to/file mode=0644"

# Change ownership
ansible all -i inventory.yml -m file -a "path=/path/to/file owner=user group=group"

# Use become for privilege escalation
# In playbook:
- name: Task requiring root
  command: some_command
  become: yes
```

SELinux Issues

Symptoms:
- SELinux denials in logs
- Services failing due to SELinux policies

Diagnosis:

```
# Check SELinux status
ansible all -i inventory.yml -m shell -a "getenforce"

# Check SELinux denials
ansible all -i inventory.yml -m shell -a "ausearch -m AVC -ts recent"

# Check file contexts
ansible all -i inventory.yml -m shell -a "ls -Z /path/to/file"
```

Solutions:

```
# Set SELinux to permissive (temporary)
ansible all -i inventory.yml -m shell -a "setenforce 0"

# Fix file contexts
ansible all -i inventory.yml -m shell -a "restorecon -R /path/to/directory"

# Create SELinux policy
ansible all -i inventory.yml -m shell -a "audit2allow -a -M mypolicy"
ansible all -i inventory.yml -m shell -a "semodule -i mypolicy.pp"
```

Advanced Troubleshooting

Log Analysis

Centralized Logging

```
# Set up log aggregation
ansible-playbook playbooks/logging/setup_logging.yml -i inventory.yml

# Search logs
grep -r "ERROR" /var/log/ansible/

# Analyze patterns
awk '/ERROR/ {print $1, $2, $NF}' /var/log/ansible/ansible.log
```

Log Rotation

```
# Check log rotation
ansible all -i inventory.yml -m shell -a "logrotate -d /etc/logrotate.conf"

# Force log rotation
ansible all -i inventory.yml -m shell -a "logrotate -f /etc/logrotate.conf"
```

Network Troubleshooting

Connectivity Testing

```
# Test network connectivity
ansible all -i inventory.yml -m shell -a "ping -c 3 google.com"

# Check routing
ansible all -i inventory.yml -m shell -a "ip route show"

# Test specific ports
ansible all -i inventory.yml -m shell -a "telnet target-host 22"
```

Firewall Debugging

```
# Check iptables rules
ansible all -i inventory.yml -m shell -a "iptables -L -n -v"

# Check ufw status
ansible all -i inventory.yml -m shell -a "ufw status verbose"

# Test with firewall disabled (temporarily)
ansible all -i inventory.yml -m shell -a "ufw disable"
```

Database Issues

Connection Problems

```
# Test database connectivity
ansible all -i inventory.yml -m shell -a "mysql -h localhost -u user -p -e 'SELECT 1'"

# Check database status
ansible all -i inventory.yml -m systemd -a "name=mysql"

# Review database logs
ansible all -i inventory.yml -m shell -a "tail -f /var/log/mysql/error.log"
```

Performance Issues

```
# Check database processes
ansible all -i inventory.yml -m shell -a "mysql -e 'SHOW PROCESSLIST'"

# Check database size
ansible all -i inventory.yml -m shell -a "du -sh /var/lib/mysql"

# Optimize tables
ansible all -i inventory.yml -m shell -a "mysqlcheck -o --all-databases"
```

Monitoring and Alerting

Health Checks

```
# Automated health check
ansible-playbook playbooks/monitoring/health_check.yml -i inventory.yml

# Service monitoring
ansible all -i inventory.yml -m shell -a "systemctl is-active nginx"

# Application monitoring
ansible all -i inventory.yml -m uri -a "url=http://localhost/health"
```

Performance Monitoring

```
# System metrics
ansible all -i inventory.yml -m shell -a "vmstat 1 5"

# Disk I/O
ansible all -i inventory.yml -m shell -a "iostat -x 1 5"

# Network statistics
ansible all -i inventory.yml -m shell -a "ss -tuln"
```

Recovery Procedures

Service Recovery

```
# Restart failed services
ansible-playbook playbooks/recovery/restart_services.yml -i inventory.yml

# Restore from backup
ansible-playbook playbooks/recovery/restore_backup.yml -i inventory.yml

# Rollback deployment
ansible-playbook playbooks/deployment/rollback.yml -i inventory.yml
```

Data Recovery

```
# Restore database
ansible-playbook playbooks/recovery/restore_database.yml -i inventory.yml

# Restore configuration files
ansible-playbook playbooks/recovery/restore_configs.yml -i inventory.yml
```

Prevention Strategies

Proactive Monitoring

- Implement comprehensive monitoring
- Set up alerting for critical metrics
- Regular health checks
- Automated testing

Documentation

- Keep runbooks updated
- Document known issues
- Maintain troubleshooting logs
- Share knowledge across team

Testing

- Regular disaster recovery testing
- Automated testing in CI/CD
- Load testing
- Security testing

Emergency Procedures

Incident Response

1. **Assess Impact**: Determine scope and severity
2. **Communicate**: Notify stakeholders
3. **Contain**: Prevent further damage
4. **Investigate**: Find root cause
5. **Resolve**: Implement fix
6. **Document**: Record lessons learned

Emergency Contacts

- **On-Call Engineer**: [Phone/Slack]
- **Team Lead**: [Phone/Email]
- **Manager**: [Phone/Email]
- **Emergency Hotline**: [Phone]

Communication Channels

- **Slack**: #infrastructure-alerts
- **Email**: devops-team@company.com
- **Status Page**: status.company.com

Tools and Resources

Debugging Tools

```
# Ansible debugging
ansible-playbook playbook.yml --step
ansible-playbook playbook.yml --start-at-task "task name"

# System debugging
strace -p PID
tcpdump -i eth0 port 22
wireshark

# Log analysis
grep, awk, sed
journalctl
logrotate
```

Useful Commands

```
# Quick system check
ansible all -i inventory.yml -m shell -a "uptime && free -m && df -h"

# Service status check
ansible all -i inventory.yml -m shell -a "systemctl status nginx mysql"

# Network connectivity
ansible all -i inventory.yml -m shell -a "ping -c 1 8.8.8.8"

# Disk usage
ansible all -i inventory.yml -m shell -a "du -sh /var/log/*"
```

Related Documentation

- Deployment Runbook (DEPLOYMENT_RUNBOOK.md)
- Security Procedures (SECURITY_PROCEDURES.md)
- Monitoring Guide (MONITORING_GUIDE.md)
- Backup and Recovery (BACKUP_RECOVERY.md)