# Automatización de Clúster HPC con Ansible

Documentación Técnica del Proyecto

*Pau Santana*

May 14, 2025

# Resumen


## English

This technical project documents the implementation of a High-Performance Computing (HPC) cluster using Ansible for automation. The project focuses on creating a scalable, maintainable, and secure HPC environment that can be easily deployed and managed. The documentation covers the complete lifecycle of the project, from initial planning to implementation and maintenance, including detailed technical specifications, cost analysis, and operational procedures.

## Español

Este proyecto técnico documenta la implementación de un clúster de Computación de Alto Rendimiento (HPC) utilizando Ansible para la automatización. El proyecto se centra en crear un entorno HPC escalable, mantenible y seguro que pueda ser desplegado y gestionado fácilmente. La documentación cubre el ciclo de vida completo del proyecto, desde la planificación inicial hasta la implementación y mantenimiento, incluyendo especificaciones técnicas detalladas, análisis de costos y procedimientos operativos.

# Contents

# List of Figures

# List of Tables

## 0.1 Visión General del Proyecto

Este proyecto implementa un sistema automatizado de despliegue y gestión de clústeres de Computación de Alto Rendimiento (HPC) utilizando Ansible. El sistema está diseñado para proporcionar un entorno HPC completo y listo para producción con todos los servicios esenciales para la computación científica.

### 0.1.1 Objetivos del Proyecto

Los objetivos principales de este proyecto son:

- Crear un sistema de despliegue completamente automatizado para clústeres HPC
- Implementar roles de Ansible modulares y reutilizables para cada componente del clúster
- Garantizar la alta disponibilidad y fiabilidad de los servicios principales
- Proporcionar capacidades completas de monitoreo y generación de informes
- Facilitar el mantenimiento y las actualizaciones de la infraestructura del clúster
- Gestionar software científico a través de Spack y contenedores
- Implementar las mejores prácticas de seguridad en todo el clúster

### 0.1.2 Alcance del Proyecto

El proyecto abarca los siguientes componentes clave:

1. **Infraestructura de Computación**
    - Planificador de trabajos y gestor de recursos SLURM
    - Aprovisionamiento y configuración de nodos de cómputo
    - Configuración y gestión de nodos de acceso
    - Monitoreo y gestión de energía
2. **Infraestructura de Almacenamiento**
    - Configuración de servidor y cliente NFS
    - Almacenamiento compartido para directorios de usuario
    - Gestión de espacio para aplicaciones y datos temporales
    - Gestión automatizada de montajes
3. **Seguridad y Autenticación**
    - Servicios de directorio OpenLDAP
    - Gestión centralizada de usuarios
    - Configuración de firewall

- Implementación de políticas de seguridad

4. **Monitoreo y Generación de Informes**
    - Recopilación de métricas con Prometheus
    - Paneles de control con Grafana
    - Despliegue de Node Exporter
    - Soluciones de monitoreo personalizadas
    - Sistema automatizado de informes
5. **Gestión de Software**
    - Integración del gestor de paquetes Spack
    - Despliegue de aplicaciones en contenedores
    - Gestión de entornos de software científico
6. **Gestión de Infraestructura**
    - Configuración de servicio DNS
    - Implementación del sistema de respaldo
    - Marco de pruebas automatizadas
    - Documentación y procedimientos de mantenimiento

### 0.1.3 Beneficios del Proyecto

La implementación de esta solución automatizada de clúster HPC proporciona varios beneficios clave:

- **Tiempo de Despliegue Reducido:** La automatización reduce el tiempo de configuración del clúster de días a horas
- **Configuración Consistente:** Garantiza una configuración uniforme en todos los componentes del clúster
- **Mantenimiento Sencillo:** Actualizaciones y mantenimiento simplificados mediante automatización
- **Escalabilidad:** Fácil adición de nuevos nodos y servicios
- **Fiabilidad:** Pruebas y validación automatizadas de los componentes del clúster
- **Seguridad:** Políticas de seguridad consistentes y actualizaciones de seguridad automatizadas
- **Monitoreo:** Sistema completo de monitoreo y alertas
- **Documentación:** Generación automatizada de documentación de configuración

### 0.1.4 Partes Interesadas

Las partes interesadas clave en este proyecto son:

- **CientiGO:** La organización implementadora
- **Administradores del Clúster:** Personal técnico responsable de la gestión del clúster
- **Usuarios Finales:** Investigadores y científicos que utilizan el clúster
- **Integradores de Sistemas:** Equipos responsables de integrar el clúster con la infraestructura existente
- **Personal de Soporte:** Personal de soporte técnico

### 0.1.5 Criterios de Éxito

El proyecto se considerará exitoso cuando:

1. Todos los servicios principales estén desplegados y operativos
2. El proceso de despliegue automatizado esté validado y documentado
3. Los sistemas de monitoreo y generación de informes estén completamente funcionales
4. Las medidas de seguridad estén implementadas y verificadas
5. El rendimiento cumpla o supere los requisitos
6. La documentación esté completa y precisa
7. Los materiales de capacitación estén disponibles para administradores y usuarios

## 0.2 Planificación del Proyecto y Asignación de Recursos

### 0.2.1 Cronograma del Proyecto

El proyecto está dividido en varias fases, cada una con hitos y entregables específicos:

1. **Fase 1: Planificación y Diseño (Semanas 1-2)**
   - Recopilación y análisis de requisitos
   - Diseño de arquitectura
   - Selección de stack tecnológico
   - Planificación de recursos
   - Evaluación de riesgos
2. **Fase 2: Desarrollo de Infraestructura Base (Semanas 3-6)**
   - Desarrollo de roles Ansible para servicios principales
   - Diseño de estructura de inventario
   - Implementación de gestión de variables
   - Desarrollo de playbooks básicos
   - Marco inicial de pruebas
3. **Fase 3: Implementación de Servicios (Semanas 7-12)**
   - Automatización del despliegue de SLURM
   - Implementación del sistema de almacenamiento
   - Configuración del sistema de autenticación
   - Despliegue de infraestructura de monitoreo
   - Integración del sistema de gestión de software
4. **Fase 4: Pruebas y Validación (Semanas 13-14)**
   - Pruebas de componentes
   - Pruebas de integración
   - Pruebas de rendimiento
   - Pruebas de seguridad
   - Revisión de documentación
5. **Fase 5: Despliegue y Capacitación (Semanas 15-16)**
   - Despliegue en producción
   - Validación del sistema
   - Capacitación de administradores
   - Documentación de usuario
   - Transferencia de conocimiento

### 0.2.2 Asignación de Recursos

#### 0.2.2.1 Recursos Humanos

| Rol | Responsabilidades | Asignación de Tiempo |
|---|---|---|
| Gerente de Proyecto | Coordinación general del proyecto | Tiempo completo |
| Arquitecto de Sistemas | Diseño y revisión de arquitectura | Tiempo completo |
| Desarrollador Ansible | Desarrollo y pruebas de roles | 2 Tiempo completo |
| Ingeniero DevOps | Infraestructura y CI/CD | Tiempo completo |
| Especialista en Seguridad | Implementación de seguridad | Tiempo parcial |
| Redactor de Documentación | Documentación técnica | Tiempo parcial |

Table 1: Asignación de Recursos Humanos

#### 0.2.2.2 Recursos Técnicos

| Recurso | Especificación | Cantidad |
|---|---|---|
| Servidores de Desarrollo | 2x CPU, 16GB RAM, 500GB SSD | 4 |
| Entorno de Pruebas | 4x CPU, 32GB RAM, 1TB SSD | 2 |
| Control de Versiones | GitLab Enterprise | 1 |
| Pipeline CI/CD | GitLab CI | 1 |
| Sistema de Documentación | Confluence | 1 |
| Sistema de Monitoreo | Prometheus/Grafana | 1 |

Table 2: Asignación de Recursos Técnicos

### 0.2.3 Gestión de Riesgos

| Riesgo | Impacto | Probabilidad | Mitigación |
|---|---|---|---|
| Complejidad Técnica | Alto | Media | Implementación por fases |
| Restricciones de Recursos | Medio | Baja | Planificación temprana |
| Vulnerabilidades de Seguridad | Alto | Media | Auditorías regulares |
| Problemas de Integración | Medio | Media | Pruebas exhaustivas |
| Problemas de Rendimiento | Alto | Baja | Pruebas de rendimiento |

Table 3: Evaluación y Mitigación de Riesgos

### 0.2.4 Aseguramiento de Calidad

El proyecto implementa varias medidas de aseguramiento de calidad:

1. **Calidad del Código**

- Proceso de revisión de código
- Linting automatizado
- Pruebas unitarias
- Pruebas de integración

2. **Documentación**
    - Documentación técnica
    - Guías de usuario
    - Documentación de API
    - Guías de despliegue
3. **Pruebas**
    - Pruebas automatizadas
    - Pruebas de rendimiento
    - Pruebas de seguridad
    - Pruebas de aceptación
4. **Monitoreo**
    - Monitoreo de salud del sistema
    - Monitoreo de rendimiento
    - Monitoreo de seguridad
    - Monitoreo de uso

### 0.2.5 Plan de Comunicación

Se establecen canales de comunicación y reuniones regulares:

- Reuniones diarias de seguimiento
- Revisiones semanales de progreso
- Actualizaciones quincenales a partes interesadas
- Informes mensuales de estado del proyecto
- Actualizaciones de documentación
- Seguimiento y resolución de incidencias

### 0.2.6 Gestión del Cambio

El proyecto implementa un proceso estructurado de gestión del cambio:

1. Presentación de solicitud de cambio
2. Análisis de impacto
3. Proceso de aprobación
4. Planificación de implementación
5. Pruebas y validación
6. Despliegue
7. Actualización de documentación

### 0.2.7 Entregables del Proyecto

Entregables clave para cada fase:

1. **Fase 1**
    - Plan del proyecto
    - Documento de diseño de arquitectura
    - Documentación del stack tecnológico
    - Informe de evaluación de riesgos
2. **Fase 2**
    - Roles Ansible principales
    - Estructura de inventario
    - Playbooks básicos
    - Marco de pruebas
3. **Fase 3**
    - Documentación de implementación de servicios
    - Guías de configuración
    - Resultados de pruebas de integración
    - Informes de pruebas de rendimiento
4. **Fase 4**
    - Informes de pruebas
    - Resultados de auditoría de seguridad
    - Puntos de referencia de rendimiento
    - Documentación del sistema

5. **Fase 5**

    - Guía de despliegue en producción
    - Documentación de usuario
    - Materiales de capacitación
    - Procedimientos de mantenimiento

## 0.3 Antecedentes Técnicos

### 0.3.1 Computación de Alto Rendimiento (HPC)

La Computación de Alto Rendimiento se refiere al uso de procesamiento paralelo para ejecutar programas avanzados de manera eficiente, confiable y rápida. Los sistemas HPC típicamente consisten en:

- **Nodos de Cómputo:** Servidores optimizados para tareas computacionales
- **Nodos de Acceso:** Puntos de entrada para que los usuarios accedan al clúster
- **Nodos de Gestión:** Controlan y monitorean el clúster
- **Sistemas de Almacenamiento:** Almacenamiento de alto rendimiento para datos
- **Red de Alta Velocidad:** Interconexión para comunicación entre nodos

### 0.3.2 Ansible

Ansible es una plataforma de automatización de código abierto utilizada para gestión de configuración, despliegue de aplicaciones y automatización de tareas. Características principales:

- **Arquitectura sin Agente:** Utiliza SSH para ejecución remota
- **Lenguaje Declarativo:** Playbooks basados en YAML
- **Idempotencia:** Seguro de ejecutar múltiples veces
- **Extensibilidad:** Módulos y plugins personalizados
- **Gestión de Inventario:** Gestión dinámica de hosts

### 0.3.3 Gestor de Cargas de Trabajo SLURM

SLURM (Simple Linux Utility for Resource Management) es un planificador de trabajos y gestor de recursos de código abierto. Componentes principales:

- **slurmctld:** Demonio de gestión central
- **slurmd:** Demonio de nodo de cómputo
- **slurmdbd:** Demonio de base de datos de contabilidad
- **sacctmgr:** Herramienta de gestión de cuentas
- **squeue:** Gestión de cola de trabajos

### 0.3.4 Tecnologías de Almacenamiento

#### 0.3.4.1 Sistema de Archivos en Red (NFS)

NFS es un protocolo de sistema de archivos distribuido que permite acceso remoto a archivos:

- **NFSv4:** Última versión con seguridad mejorada
- **Ajuste de Rendimiento:** Optimizado para cargas de trabajo HPC
- **Seguridad:** Autenticación Kerberos
- **Alta Disponibilidad:** Configuraciones de respaldo

#### 0.3.4.2 AutoFS

Sistema de montaje automático para compartidos NFS:

- **Montaje Bajo Demanda:** Monta cuando se accede
- **Desmontaje Automático:** Ahorra recursos
- **Mapas Directos:** Puntos de montaje estáticos
- **Mapas Indirectos:** Puntos de montaje dinámicos

### 0.3.5 Autenticación y Seguridad

#### 0.3.5.1 OpenLDAP

Implementación del Protocolo Ligero de Acceso a Directorios:

- **Estructura de Directorio:** Organización jerárquica de datos
- **Replicación:** Replicación multi-maestro
- **Seguridad:** Cifrado TLS/SSL
- **Esquema:** Definiciones de atributos personalizados

#### 0.3.5.2 Seguridad del Sistema

Medidas de seguridad implementadas:

- **Firewall:** Reglas iptables/nftables
- **SELinux:** Control de acceso obligatorio
- **SSH:** Autenticación basada en claves
- **Auditoría:** Registro de actividad del sistema

### 0.3.6 Monitoreo y Métricas

#### 0.3.6.1 Prometheus

Base de datos de series temporales para métricas:

- **Modelo de Datos:** Series temporales con etiquetas
- **Lenguaje de Consulta:** PromQL
- **Alertas:** Alertas basadas en reglas
- **Descubrimiento de Servicios:** Detección dinámica de objetivos

#### 0.3.6.2 Grafana

Plataforma de visualización y análisis:

- **Paneles:** Visualizaciones personalizadas
- **Alertas:** Alertas basadas en umbrales
- **Plugins:** Funcionalidad extendida
- **Autenticación:** Acceso basado en roles

### 0.3.7 Gestión de Software

#### 0.3.7.1 Spack

Gestor de paquetes para software científico:

- **Multi-versión:** Múltiples versiones de software
- **Dependencias:** Resolución automática
- **Entornos:** Pilas de software aisladas
- **Compiladores:** Soporte para múltiples compiladores

#### 0.3.7.2 Contenedores

Containerización de aplicaciones:

- **Docker:** Runtime de contenedores
- **Singularity:** Contenedores orientados a HPC
- **Registro:** Almacenamiento de imágenes
- **Orquestación:** Gestión de contenedores

### 0.3.8 Infraestructura de Red

#### 0.3.8.1 DNS

Sistema de Nombres de Dominio:

- **BIND:** Software de servidor DNS
- **Zonas:** Gestión de dominios
- **Registros:** Registros de recursos
- **Seguridad:** DNSSEC

#### 0.3.8.2 Configuración de Red

Configuración y gestión de red:

- **Subredes:** Segmentación de red
- **Enrutamiento:** Rutas estáticas y dinámicas
- **VLANs:** Redes LAN virtuales
- **QoS:** Calidad de Servicio

### 0.3.9 Respaldo y Recuperación

#### 0.3.9.1 Sistemas de Respaldo

Estrategias de protección de datos:

- **Respaldos Completos:** Respaldo completo del sistema
- **Incremental:** Solo datos modificados
- **Instantáneas:** Copias en un punto en el tiempo
- **Replicación:** Copia en tiempo real

#### 0.3.9.2 Recuperación ante Desastres

Procedimientos de recuperación:

- **RPO:** Objetivo de Punto de Recuperación
- **RTO:** Objetivo de Tiempo de Recuperación
- **Pruebas:** Validación de recuperación
- **Documentación:** Procedimientos de recuperación

### 0.3.10 Optimización de Rendimiento

#### 0.3.10.1 Ajuste del Sistema

Técnicas de optimización de rendimiento:

- **Parámetros del Kernel:** Ajuste del sistema
- **Red:** Optimización TCP/IP
- **Almacenamiento:** Optimización de E/S
- **Memoria:** Gestión de caché

#### 0.3.10.2 Monitoreo y Perfilado

Herramientas de análisis de rendimiento:

- **Node Exporter:** Métricas del sistema
- **SLURM Exporter:** Métricas de trabajos
- **Monitoreo de Energía:** Uso de energía
- **Perfilado de Aplicaciones:** Optimización de código

## 0.4 Análisis de Costos

### 0.4.1 Costos de Licencias de Software

#### 0.4.1.1 Componentes de Código Abierto

El proyecto utiliza principalmente software de código abierto, que no tiene costos directos de licencia:

| Componente | Licencia | Costo |
|---|---|---|
| Ansible | GPL-3.0 | $0 |
| SLURM | GPL-2.0 | $0 |
| OpenLDAP | OpenLDAP | $0 |
| Prometheus | Apache 2.0 | $0 |
| Grafana | AGPL-3.0 | $0 |
| Spack | Apache 2.0 | $0 |
| Docker | Apache 2.0 | $0 |

Table 4: Costos de Licencias de Software

#### 0.4.1.2 Componentes Comerciales

Componentes comerciales opcionales y sus costos:

| Componente | Proveedor | Tipo de Licencia | Costo Anual |
|---|---|---|---|
| Red Hat Enterprise Linux | Red Hat | Por Socket | $799/socket |
| Ansible Tower | Red Hat | Por Nodo | $10,000/100 nodos |
| Grafana Enterprise | Grafana Labs | Por Usuario | $20/usuario/mes |
| Prometheus Enterprise | Grafana Labs | Por Instancia | $1,000/instancia |

Table 5: Costos de Software Comercial

### 0.4.2 Costos de Hardware

#### 0.4.2.1 Infraestructura de Cómputo

Costos estimados de hardware para un clúster de tamaño medio:

| Componente | Especificación | Cantidad | Costo Total |
|---|---|---|---|
| Nodos de Cómputo | 2x CPU, 256GB RAM, 2TB NVMe | 10 | $250,000 |
| Nodos de Acceso | 2x CPU, 128GB RAM, 1TB SSD | 2 | $20,000 |
| Nodos de Gestión | 2x CPU, 64GB RAM, 500GB SSD | 2 | $12,000 |
| Servidor de Almacenamiento | 4x CPU, 256GB RAM, 100TB | 2 | $80,000 |
| Switch de Red | 100GbE, 48 puertos | 2 | $40,000 |

Table 6: Costos de Hardware

#### 0.4.2.2 Infraestructura de Almacenamiento

Costos del sistema de almacenamiento:

| Componente | Especificación | Cantidad | Costo Total |
|---|---|---|---|
| Almacenamiento Primario | 100TB NVMe | 1 | $50,000 |
| Almacenamiento de Respaldo | 200TB HDD | 1 | $30,000 |
| Biblioteca de Cintas | LTO-9, 100TB | 1 | $25,000 |
| Red de Almacenamiento | 100GbE | 1 | $20,000 |

Table 7: Costos de Infraestructura de Almacenamiento

### 0.4.3 Costos de Implementación

#### 0.4.3.1 Servicios Profesionales

Costos de implementación y soporte:

| Servicio | Duración | Tarifa | Costo Total |
|---|---|---|---|
| Arquitectura del Sistema | 2 semanas | $150/hora | $12,000 |
| Implementación | 8 semanas | $120/hora | $38,400 |
| Pruebas | 2 semanas | $100/hora | $8,000 |
| Capacitación | 1 semana | $1,500/día | $7,500 |

Table 8: Costos de Implementación

#### 0.4.3.2 Costos de Mantenimiento

Costos anuales de mantenimiento y soporte:

| Servicio | Tipo | Frecuencia | Costo Anual |
|---|---|---|---|
| Soporte de Hardware | Premium | 24/7 | $50,000 |
| Soporte de Software | Empresarial | 24/7 | $30,000 |
| Actualizaciones del Sistema | Regular | Mensual | $15,000 |
| Capacitación | Refresco | Trimestral | $12,000 |

Table 9: Costos de Mantenimiento

### 0.4.4 Costos Operativos

#### 0.4.4.1 Energía y Refrigeración

Costos operativos anuales:

#### 0.4.4.2 Costos de Personal

Costos anuales de personal:

| Componente | Uso de Energía | Costo por kWh | Costo Anual |
|---|---|---|---|
| Nodos de Cómputo | 50kW | $0.12 | $52,560 |
| Almacenamiento | 10kW | $0.12 | $10,512 |
| Refrigeración | 60kW | $0.12 | $63,072 |
| Red | 5kW | $0.12 | $5,256 |

Table 10: Costos Operativos

| Posición | Nivel | Salario Anual | Beneficios |
|---|---|---|---|
| Administrador de Sistemas | Senior | $120,000 | $30,000 |
| Ingeniero HPC | Intermedio | $100,000 | $25,000 |
| Especialista en Almacenamiento | Senior | $110,000 | $27,500 |
| Ingeniero de Redes | Intermedio | $95,000 | $23,750 |

Table 11: Costos de Personal

## 0.4.5 Costo Total de Propiedad (TCO)

### 0.4.5.1 Inversión Inicial

Costos del primer año:

- Hardware: $457,000
- Licencias de Software: $50,000
- Implementación: $65,900
- Capacitación: $7,500
- Inversión Inicial Total: $580,400

### 0.4.5.2 Costos Operativos Anuales

Costos anuales recurrentes:

- Mantenimiento: $107,000
- Energía y Refrigeración: $131,400
- Personal: $431,250
- Soporte de Software: $30,000
- Costo Operativo Anual Total: $699,650

## 0.4.6 Estrategias de Optimización de Costos

### 0.4.6.1 Optimización de Hardware

Estrategias de reducción de costos:

- **Despliegue por Fases:** Expansión gradual según la demanda
- **Hardware Estándar:** Uso de componentes de servidor estándar
- **Eficiencia Energética:** Selección de hardware optimizado para energía
- **Compartir Recursos:** Infraestructura multi-tenant

#### 0.4.6.2 Optimización de Software

Reducción de costos de software:

- **Código Abierto:** Maximizar el uso de soluciones de código abierto
- **Soporte de la Comunidad:** Aprovechar recursos de la comunidad
- **Licencias por Volumen:** Negociar acuerdos empresariales
- **Integración con la Nube:** Opciones de despliegue híbrido

### 0.4.7 Retorno de Inversión (ROI)

#### 0.4.7.1 Beneficios Directos

Retornos cuantificables:

- **Tiempo de Despliegue Reducido:** 75% más rápido en configuración del clúster
- **Mantenimiento Inferior:** 50% de reducción en tareas manuales
- **Ahorro de Energía:** 30% de reducción en uso de energía
- **Mayor Utilización:** 40% mejor uso de recursos

#### 0.4.7.2 Beneficios Indirectos

Retornos no cuantificables:

- **Mejor Fiabilidad:** Reducción del tiempo de inactividad
- **Mejor Seguridad:** Protección mejorada
- **Satisfacción del Usuario:** Mejor calidad de servicio
- **Transferencia de Conocimiento:** Mejor documentación

### 0.4.8 Opciones de Financiamiento

Métodos de financiamiento disponibles:

- **Gasto de Capital:** Compra directa
- **Arrendamiento Operativo:** Pagos mensuales
- **Servicios en la Nube:** Modelo de pago según uso
- **Programas de Financiamiento:** Financiamiento de proveedor

## 0.5 Implementación Técnica

### 0.5.1 Estructura de Roles de Ansible

El proyecto implementa una arquitectura modular basada en roles, donde cada rol es responsable de un componente específico del clúster. Los roles están organizados de la siguiente manera:

```
roles/
          common/                % Configuración común para todos
   los nodos
          login/                 % Configuración de nodos de login
          compute/               % Configuración de nodos de
   cómputo
          storage/               % Configuración de almacenamiento
          slurm/                 % Configuración de SLURM
          monitoring/            % Configuración de monitoreo
          auth/                  % Configuración de autenticación
          spack/                 % Gestión de software científico
```

### 0.5.2 Detalles de Implementación de Roles

#### 0.5.2.1 Gestión de SLURM

**Rol slurmctld** El rol del controlador SLURM implementa el demonio de gestión central:

```
% tasks/main.yml
- name: Instalar paquetes del controlador SLURM
  package:
    name: "{{ item }}"
    state: present
  with_items:
    - slurm
    - slurm-slurmctld
    - slurm-slurmdbd

- name: Configurar controlador SLURM
  template:
    src: slurm.conf.j2
    dest: /etc/slurm/slurm.conf
    mode: '0644'
  notify: reiniciar slurmctld

- name: Habilitar e iniciar controlador SLURM
  systemd:
    name: slurmctld
    state: started
    enabled: yes
```

**Rol slurmdbd** El rol de base de datos SLURM gestiona la contabilidad:

```
% tasks/main.yml
- name: Instalar MariaDB
  package:
    name: mariadb-server
    state: present

- name: Configurar MariaDB
  template:
    src: mariadb.cnf.j2
    dest: /etc/my.cnf.d/slurm.cnf
  notify: reiniciar mariadb

- name: Inicializar base de datos SLURM
  mysql_db:
    name: slurm_acct_db
    state: present

- name: Configurar slurmdbd
  template:
    src: slurmdbd.conf.j2
    dest: /etc/slurm/slurmdbd.conf
    mode: '0600'
  notify: reiniciar slurmdbd
```

#### 0.5.2.2 Gestión de Almacenamiento

**Rol nfs_server** El rol de servidor NFS implementa almacenamiento compartido:

```
% tasks/main.yml
- name: Instalar servidor NFS
  package:
    name: nfs-utils
    state: present

- name: Crear directorios NFS
  file:
    path: "{{ item.path }}"
    state: directory
    mode: "{{ item.mode }}"
  with_items: "{{ nfs_shares }}"

- name: Configurar exportaciones NFS
  template:
    src: exports.j2
    dest: /etc/exports
  notify: reiniciar nfs-server

- name: Habilitar e iniciar NFS
  systemd:
    name: nfs-server
    state: started
```

```
    enabled: yes
```

**Rol nfs_client** El rol de cliente NFS configura los montajes del cliente:

```
% tasks/main.yml
- name: Instalar cliente NFS
  package:
    name: nfs-utils
    state: present

- name: Configurar montajes NFS
  mount:
    path: "{{ item.mount }}"
    src: "{{ item.server }}:{{ item.export }}"
    fstype: nfs
    opts: "{{ item.options }}"
    state: mounted
  with_items: "{{ nfs_mounts }}"
```

#### 0.5.2.3 Autenticación

**Rol openldap** El rol OpenLDAP implementa autenticación centralizada:

```
% tasks/main.yml
- name: Instalar OpenLDAP
  package:
    name: "{{ item }}"
    state: present
  with_items:
    - openldap-servers
    - openldap-clients

- name: Configurar OpenLDAP
  template:
    src: slapd.conf.j2
    dest: /etc/openldap/slapd.conf
  notify: reiniciar slapd

- name: Inicializar base de datos LDAP
  ldap_entry:
    dn: "{{ item.dn }}"
    objectClass: "{{ item.objectClass }}"
    attributes: "{{ item.attributes }}"
  with_items: "{{ ldap_entries }}"

- name: Configurar SSSD
  template:
    src: sssd.conf.j2
    dest: /etc/sssd/sssd.conf
    mode: '0600'
  notify: reiniciar sssd
```

#### 0.5.2.4 Monitoreo

**Rol prometheus** El rol Prometheus implementa la recopilación de métricas:

```
% tasks/main.yml
- name: Instalar Prometheus
  package:
    name: prometheus
    state: present

- name: Configurar Prometheus
  template:
    src: prometheus.yml.j2
    dest: /etc/prometheus/prometheus.yml
  notify: reiniciar prometheus

- name: Configurar objetivos
  template:
    src: targets.yml.j2
    dest: /etc/prometheus/targets.yml
  notify: recargar prometheus

- name: Habilitar e iniciar Prometheus
  systemd:
    name: prometheus
    state: started
    enabled: yes
```

**Rol grafana** El rol Grafana implementa la visualización:

```
% tasks/main.yml
- name: Instalar Grafana
  package:
    name: grafana
    state: present

- name: Configurar Grafana
  template:
    src: grafana.ini.j2
    dest: /etc/grafana/grafana.ini
  notify: reiniciar grafana

- name: Importar paneles
  grafana_dashboard:
    url: "{{ item.url }}"
    grafana_url: "{{ grafana_url }}"
    grafana_user: "{{ grafana_user }}"
    grafana_password: "{{ grafana_password }}"
  with_items: "{{ grafana_dashboards }}"

- name: Habilitar e iniciar Grafana
  systemd:
```

```
    name: grafana-server
    state: started
    enabled: yes
```

#### 0.5.2.5 Gestión de Software

**Rol spack** El rol Spack implementa la gestión de software científico:

```
% tasks/main.yml
- name: Clonar repositorio Spack
  git:
    repo: https://github.com/spack/spack.git
    dest: "{{ spack_root }}"
    version: "{{ spack_version }}"

- name: Configurar Spack
  template:
    src: config.yaml.j2
    dest: "{{ spack_root }}/etc/spack/config.yaml"

- name: Instalar compiladores
  command: "{{ spack_root }}/bin/spack install {{ item }}"
  with_items: "{{ spack_compilers }}"

- name: Instalar paquetes
  command: "{{ spack_root }}/bin/spack install {{ item }}"
  with_items: "{{ spack_packages }}"
```

**container_apps Role** The container apps role manages containerized applications:

```
% tasks/main.yml
- name: Install Docker
  package:
    name: "{{ item }}"
    state: present
  with_items:
    - docker-ce
    - docker-ce-cli
    - containerd.io

- name: Configure Docker
  template:
    src: daemon.json.j2
    dest: /etc/docker/daemon.json
  notify: restart docker

- name: Pull container images
  docker_image:
    name: "{{ item.name }}"
    tag: "{{ item.tag }}"
    source: pull
```

```
    with_items: "{{ container_images }}"

- name: Create container volumes
  docker_volume:
    name: "{{ item }}"
    state: present
  with_items: "{{ container_volumes }}"
```

### 0.5.3 Variable Management

#### 0.5.3.1 Global Variables

Global variables are defined in ‘group$_v ars/all/main.yml$‘ :

```
% group_vars/all/main.yml
% Cluster configuration
cluster_name: "hpc_cluster"
domain_name: "cluster.local"
timezone: "UTC"

% Network configuration
internal_network: "10.0.0.0/24"
external_network: "192.168.0.0/24"

% Security settings
firewall_enabled: true
selinux_mode: "enforcing"
ldap_base_dn: "dc=cluster,dc=local"

% Monitoring configuration
prometheus_port: 9090
grafana_port: 3000
alertmanager_port: 9093

% Storage configuration
nfs_server: "storage.cluster.local"
nfs_exports:
  - path: "/home"
    options: "rw,sync,no_root_squash"
  - path: "/shared"
    options: "rw,sync,no_root_squash"
```

#### 0.5.3.2 Role-Specific Variables

Each role has its own variables in ‘defaults/main.yml‘:

```
% roles/slurmctld/defaults/main.yml
slurm_version: "21.08.8"
slurm_config:
  ClusterName: "{{ cluster_name }}"
  ControlMachine: "{{ inventory_hostname }}"
```

```
  SlurmUser: "slurm"
  SlurmctldPort: 6817
  SlurmdPort: 6818
  StateSaveLocation: "/var/spool/slurmctld"
  SlurmdSpoolDir: "/var/spool/slurmd"
  MaxTasksPerNode: 128
  ReturnToService: 2
  SlurmctldPidFile: "/var/run/slurmctld.pid"
  SlurmdPidFile: "/var/run/slurmd.pid"
  ProctrackType: "proctrack/linuxproc"
  TaskPlugin: "task/affinity"
  SchedulerType: "sched/backfill"
  SelectType: "select/cons_tres"
  SelectTypeParameters: "CR_Core_Memory"
  AccountingStorageType: "accounting_storage/slurmdbd"
  AccountingStorageHost: "{{ slurmdbd_host }}"
  AccountingStoragePort: 6819
  AccountingStorageUser: "slurm"
  AccountingStorageLoc: "slurm_acct_db"
```

### 0.5.4 Inventory Structure

The inventory is organized by node function:

```
% inventory/hosts
[management]
slurmctld ansible_host=10.0.0.1
slurmdbd ansible_host=10.0.0.2
nfs_server ansible_host=10.0.0.3

[login]
login[01:02] ansible_host=10.0.0.[10:11]

[compute]
compute[01:10] ansible_host=10.0.0.[20:29]

[monitoring]
prometheus ansible_host=10.0.0.100
grafana ansible_host=10.0.0.101
node_exporter ansible_host=10.0.0.102

[storage]
nfs_server ansible_host=10.0.0.3

[all:vars]
ansible_user=admin
ansible_become=yes
ansible_become_method=sudo
```

### 0.5.5 Playbook Organization

The playbooks are organized by function:

```
playbooks/
          site.yml              % Main playbook
          core/                 % Core services
                slurm.yml
                storage.yml
                auth.yml
          monitoring/           % Monitoring
                prometheus.yml
                grafana.yml
          software/            % Software management
              spack.yml
              containers.yml
```

### 0.5.6 Testing Framework

The testing framework includes:

```
% tests/test_slurm.yml
- name: Test SLURM installation
  hosts: management
  tasks:
    - name: Check slurmctld status
      systemd:
        name: slurmctld
        state: started
      register: slurmctld_status

    - name: Verify slurmctld is running
      assert:
        that: slurmctld_status.status.ActiveState == "active"

    - name: Test SLURM commands
      command: "{{ item }}"
      register: cmd_result
      with_items:
        - sinfo
        - squeue
        - scontrol show config
      changed_when: false

    - name: Verify command output
      assert:
        that: cmd_result.rc == 0
```

### 0.5.7 Documentation

Each role includes documentation:

```
% roles/slurmctld/README.md
% SLURM Controller Role

% Description
% This role installs and configures the SLURM controller daemon
   (slurmctld).

% Requirements
% - Ansible 2.9+
% - RHEL/CentOS 8+
% - MariaDB (for accounting)

% Role Variables
% - `slurm_version`: SLURM version to install
% - `slurm_config`: SLURM configuration parameters
% - `slurmdbd_host`: SLURM database host

% Dependencies
% - epel
% - mariadb

% Example Playbook
```yaml
- hosts: management
  roles:
    - role: slurmctld
      vars:
        slurm_version: "21.08.8"
        slurmdbd_host: "slurmdbd.cluster.local"
```

% License
% MIT
```

## 0.6 Arquitectura y Diseño

### 0.6.1 Visión General de la Arquitectura del Sistema

La arquitectura del clúster HPC está diseñada pensando en la modularidad, escalabilidad y alta disponibilidad. El sistema está dividido en varios componentes clave:

[node distance=2cm] [draw, rectangle] (login) Nodos de Acceso; [draw, rectangle, right=of login] (compute) Nodos de Cómputo; [draw, rectangle, above=of compute] (storage) Almacenamiento; [draw, rectangle, below=of compute] (monitoring) Monitoreo; [draw, rectangle, left=of monitoring] (auth) Autenticación; [draw, rectangle, above=of auth] (management) Gestión;
[->] (login) – (compute); [->] (compute) – (storage); [->] (compute) – (monitoring); [->] (login) – (auth); [->] (management) – (compute); [->] (management) – (storage); [->] (management) – (monitoring);

Figure 1: Arquitectura General del Sistema

### 0.6.2 Arquitectura de Componentes

#### 0.6.2.1 Infraestructura de Cómputo

La infraestructura de cómputo consiste en:

- **Nodos de Acceso**
  - Puntos de acceso de usuarios
  - Envío de trabajos
  - Entorno de desarrollo
  - Procesamiento ligero
- **Nodos de Cómputo**
  - Procesamiento de alto rendimiento
  - Asignación de recursos
  - Ejecución de trabajos
  - Gestión de energía
- **Nodos de Gestión**
  - Controlador SLURM
  - Servidor de base de datos
  - Gestión de configuración
  - Monitoreo del sistema

#### 0.6.2.2 Arquitectura de Almacenamiento

El sistema de almacenamiento implementa un enfoque por niveles:

- **Directorios de Usuario**
  - Almacenamiento de datos de usuario
  - Acceso basado en NFS
  - Respaldos regulares
  - Gestión de cuotas
- **Almacenamiento Compartido**
  - Datos de proyecto
  - Almacenamiento de aplicaciones
  - Acceso de alto rendimiento
  - Gestión del ciclo de vida de datos
- **Espacio Temporal**
  - Almacenamiento temporal
  - Acceso de alta velocidad
  - Limpieza automática
  - Optimizado para rendimiento

#### 0.6.2.3 Arquitectura de Red

El diseño de red implementa:

- **Red de Gestión**
  - Administración del sistema
  - Tráfico de monitoreo
  - Gestión de configuración
  - Acceso seguro
- **Red de Cómputo**
  - Interconexión de alta velocidad
  - Comunicación entre trabajos
  - Tráfico MPI
  - Baja latencia
- **Red de Almacenamiento**
  - Acceso a datos
  - Tráfico de respaldo
  - Alto ancho de banda
  - Rutas redundantes

### 0.6.3 Arquitectura de Servicios

#### 0.6.3.1 Planificación de Trabajos

El planificador de trabajos SLURM implementa:

- **Gestión de Recursos**
  - Asignación de CPU
  - Gestión de memoria
  - Planificación de GPU
  - Selección de nodos
- **Control de Trabajos**
  - Envío de trabajos
  - Gestión de prioridades
  - Uso equitativo
  - Prevención
- **Contabilidad**
  - Seguimiento de uso
  - Asignación de recursos
  - Integración con facturación
  - Informes

#### 0.6.3.2 Autenticación y Seguridad

La arquitectura de seguridad incluye:

- **Servicios de Directorio**
  - OpenLDAP
  - Gestión de usuarios
  - Políticas de grupo
  - Control de acceso
- **Seguridad de Red**
  - Reglas de firewall
  - Aislamiento de red
  - Acceso VPN
  - Detección de intrusiones
- **Seguridad del Sistema**
  - SELinux
  - Permisos de archivos
  - Registro de auditoría
  - Actualizaciones de seguridad

#### 0.6.3.3 Monitoreo y Métricas

El sistema de monitoreo implementa:

- **Monitoreo del Sistema**
  - Estado de nodos
  - Uso de recursos
  - Métricas de rendimiento
  - Alertas
- **Monitoreo de Trabajos**
  - Estado de trabajos
  - Utilización de recursos
  - Análisis de rendimiento
  - Métricas de eficiencia
- **Monitoreo de Energía**
  - Uso de energía
  - Eficiencia energética
  - Seguimiento de costos
  - Optimización

### 0.6.4 Arquitectura de Software

#### 0.6.4.1 Gestión de Software

La pila de software incluye:

- **Software del Sistema**
  - Sistema operativo
  - Bibliotecas del sistema
  - Herramientas de desarrollo
  - Entornos de ejecución
- **Software Científico**
  - Paquetes Spack
  - Imágenes de contenedores
  - Aplicaciones personalizadas
  - Módulos de entorno
- **Software de Gestión**
  - Automatización con Ansible
  - Herramientas de monitoreo
  - Sistemas de informes
  - Soluciones de respaldo

#### 0.6.4.2 Arquitectura de Contenedores

El sistema de contenedores implementa:

- **Runtime de Contenedores**
  - Docker Engine
  - Gestión de imágenes
  - Orquestación
  - Seguridad
- **Imágenes de Contenedores**
  - Aplicaciones científicas
  - Entornos de desarrollo
  - Herramientas de análisis
  - Servicios de soporte
- **Gestión de Contenedores**
  - Registro privado
  - Control de versiones
  - Actualizaciones
  - Políticas de seguridad

### 0.6.5 Alta Disponibilidad Diseño

#### 0.6.5.1 Redundancia de Servicios

El sistema implementa redundancia para servicios críticos:

- **Servicios de Gestión**
  - Failover del controlador SLURM
  - Replicación de base de datos
  - Balanceo de carga
  - Monitoreo de servicios
- **Redundancia de Almacenamiento**
  - Matrices RAID
  - Sistemas de respaldo
  - Replicación de datos
  - Recuperación ante desastres
- **Redundancia de Red**
  - Rutas redundantes
  - Agrupación de enlaces
  - Enrutamiento de failover
  - Monitoreo de red

#### 0.6.5.2 Recuperación ante Desastres

El plan de recuperación ante desastres incluye:

- **Estrategia de Respaldos**
  - Respaldos regulares
  - Respaldos incrementales
  - Almacenamiento remoto
  - Verificación de respaldos
- **Procedimientos de Recuperación**
  - Restauración del sistema
  - Recuperación de datos
  - Failover de servicio
  - Procedimientos de prueba
- **Continuidad del Negocio**
  - Objetivos de tiempo de recuperación
  - Objetivos de punto de recuperación
  - Priorización de servicios
  - Documentación

### 0.6.6 Diseño de Escalabilidad

#### 0.6.6.1 Escalabilidad Horizontal

El sistema soporta escalabilidad horizontal:

- **Escalabilidad de Cómputo**
  - Adición de nodos
  - Expansión de recursos
  - Distribución de carga
  - Escalabilidad de rendimiento
- **Escalabilidad de Almacenamiento**
  - Expansión de capacidad
  - Escalabilidad de rendimiento
  - Gestión de niveles
  - Distribución de datos
- **Escalabilidad de Servicios**
  - Replicación de servicio
  - Balanceo de carga
  - Asignación de recursos
  - Optimización de rendimiento

#### 0.6.6.2 Escalabilidad Vertical

El sistema soporta escalabilidad vertical:

- **Mejora de Recursos**
  - Actualizaciones de CPU
  - Expansión de memoria
  - Actualizaciones de almacenamiento
  - Mejora de red
- **Optimización de Rendimiento**
  - Ajuste del sistema
  - Asignación de recursos
  - Optimización de caché
  - Optimización de I/O

## 0.7 Procedimientos Detallados de Implementación

### 0.7.1 Configuración Inicial

#### 0.7.1.1 Preparación del Entorno

1. **Requisitos del Sistema**
   - Instalar sistema operativo base (RHEL/CentOS 8+)
   - Configurar interfaces de red
   - Configurar resolución DNS
   - Configurar hora del sistema (NTP)
2. **Nodo de Control Ansible**
   - Instalar Ansible
   - Configurar claves SSH
   - Configurar inventario
   - Probar conectividad
3. **Configuración del Repositorio**
   - Clonar repositorio
   - Configurar Git
   - Configurar ramas
   - Inicializar submódulos

#### 0.7.1.2 Configuración de Red

1. **Planificación de Red**
   - Definir rangos IP
   - Planificar subredes
   - Configurar enrutamiento
   - Configurar VLANs
2. **Configuración DNS**
   - Instalar BIND
   - Configurar zonas
   - Configurar registros
   - Probar resolución
3. **Configuración de Firewall**
   - Definir reglas
   - Configurar zonas
   - Configurar servicios
   - Probar acceso

### 0.7.2 Despliegue de Servicios Principales

#### 0.7.2.1 Instalación de SLURM

1. **Configuración del Controlador**

```
    % Instalar controlador SLURM
    ansible-playbook playbooks/core/slurm.yml -l management

    % Verificar instalación
    systemctl status slurmctld
    sinfo
```

2. **Configuración de Base de Datos**

```
    % Instalar MariaDB
    ansible-playbook playbooks/core/slurmdbd.yml -l management

    % Inicializar base de datos
    mysql -u root -p
    CREATE DATABASE slurm_acct_db;
    GRANT ALL ON slurm_acct_db.* TO 'slurm'@'localhost';
```

3. **Configuración de Nodos de Cómputo**

```
    % Instalar nodos de cómputo
    ansible-playbook playbooks/core/slurm.yml -l compute

    % Verificar nodos
    sinfo -N
    scontrol show nodes
```

#### 0.7.2.2 Configuración de Almacenamiento

1. **Configuración del Servidor NFS**

```
    % Instalar servidor NFS
    ansible-playbook playbooks/core/storage.yml -l storage

    % Crear compartidos
    mkdir -p /home /shared /scratch
    chmod 755 /home /shared /scratch

    % Configurar exportaciones
    cat > /etc/exports << EOF
    /home     *(rw,sync,no_root_squash)
    /shared   *(rw,sync,no_root_squash)
    /scratch  *(rw,sync,no_root_squash)
    EOF

    % Iniciar NFS
    systemctl enable --now nfs-server
```

2. **Configuración de Cliente**

```
    % Instalar cliente NFS
    ansible-playbook playbooks/core/storage.yml -l
       compute,login

    % Probar montajes
    mount | grep nfs
    df -h
```

#### 0.7.2.3 Configuración de Autenticación

1. **Instalación de OpenLDAP**

```
    % Instalar OpenLDAP
    ansible-playbook playbooks/core/auth.yml -l management

    % Inicializar base de datos
    ldapadd -Y EXTERNAL -H ldapi:/// -f init.ldif
    ldapadd -x -D "cn=admin,dc=cluster,dc=local" -w password
       -f users.ldif
```

2. **Configuración de Cliente**

```
    % Configurar SSSD
    ansible-playbook playbooks/core/auth.yml -l compute,login

    % Probar autenticación
    getent passwd
    id usuario1
```

### 0.7.3 Despliegue de Monitoreo

#### 0.7.3.1 Configuración de Prometheus

1. **Instalación del Servidor**

```
    % Instalar Prometheus
    ansible-playbook playbooks/monitoring/prometheus.yml -l
       monitoring

    % Configurar objetivos
    cat > /etc/prometheus/targets.yml << EOF
    - targets:
      - 'node-exporter:9100'
      - 'slurm-exporter:9101'
      labels:
        job: 'node'
    EOF

    % Iniciar servicio
    systemctl enable --now prometheus
```

2. **Node Exporter**

```
% Instalar node exporter
ansible-playbook playbooks/monitoring/node_exporter.yml
   -l all

% Verificar métricas
curl localhost:9100/metrics
```

#### 0.7.3.2 Configuración de Grafana

1. **Configuración del Servidor**

```
% Instalar Grafana
ansible-playbook playbooks/monitoring/grafana.yml -l
   monitoring

% Configurar fuente de datos
curl -X POST -H "Content-Type: application/json" \
     -d
        '{"name":"Prometheus","type":"prometheus","url":"http://local
        \
     http://admin:admin@localhost:3000/api/datasources

% Importar paneles
for dashboard in dashboards/*.json; do
    curl -X POST -H "Content-Type: application/json" \
         -d @$dashboard \
         http://admin:admin@localhost:3000/api/dashboards/db
done
```

### 0.7.4 Gestión de Software

#### 0.7.4.1 Instalación de Spack

1. **Configuración Base**

```
% Instalar Spack
ansible-playbook playbooks/software/spack.yml -l login

% Inicializar Spack
. /opt/spack/share/spack/setup-env.sh
spack compiler find
spack mirror add local file:///opt/spack/mirror
```

2. **Instalación de Paquetes**

```
% Instalar paquetes comunes
spack install gcc@10.2.0
spack install openmpi@4.1.0
spack install python@3.9.0

```

```
    % Crear entornos
    spack env create scientific
    spack env activate scientific
    spack add gcc openmpi python
    spack install
```

#### 0.7.4.2 Configuración de Contenedores

1. **Instalación de Docker**

```
    % Instalar Docker
    ansible-playbook playbooks/software/containers.yml -l all

    % Configurar Docker
    cat > /etc/docker/daemon.json << EOF
    {
      "registry-mirrors": ["http://registry.local:5000"],
      "insecure-registries": ["registry.local:5000"]
    }
    EOF

    % Reiniciar servicio
    systemctl restart docker
```

2. **Configuración de Registro**

```
    % Instalar registro
    ansible-playbook playbooks/software/registry.yml -l
       management

    % Configurar autenticación
    htpasswd -Bbn admin password > /etc/registry/auth.htpasswd

    % Iniciar servicio
    systemctl enable --now registry
```

### 0.7.5 Pruebas y Validación

#### 0.7.5.1 Pruebas de Rendimiento

1. **Prueba HPL**

```
    % Instalar HPL
    ansible-playbook playbooks/testing/hpl.yml -l compute

    % Ejecutar prueba
    cd /opt/hpl
    mpirun -np 4 ./xhpl
```

2. **Análisis de Resultados**

```
    % Recopilar resultados
    ansible-playbook playbooks/testing/collect_results.yml

    % Generar informe
    python3 scripts/analyze_results.py
```

#### 0.7.5.2 Pruebas de Integración

1. **Pruebas de SLURM**

```
    % Probar envío de trabajos
    sbatch test.job
    squeue
    sacct

    % Verificar contabilidad
    sreport
```

2. **Pruebas de Almacenamiento**

```
    % Probar rendimiento
    fio --name=test --ioengine=libaio --rw=randrw --bs=4k
       --numjobs=4

    % Verificar montajes
    mount | grep nfs
    df -h
```

### 0.7.6 Mantenimiento

#### 0.7.6.1 Actualizaciones del Sistema

1. **Actualizaciones de Software**

```
    % Actualizar sistema
    ansible-playbook playbooks/maintenance/update.yml

    % Verificar servicios
    ansible-playbook playbooks/maintenance/verify.yml
```

2. **Respaldos**

```
    % Ejecutar respaldo
    ansible-playbook playbooks/maintenance/backup.yml

    % Verificar respaldos
    ansible-playbook playbooks/maintenance/verify_backup.yml
```

#### 0.7.6.2 Monitoreo y Mantenimiento

1. **Monitoreo Diario**

```
    % Verificar estado
    ansible-playbook playbooks/maintenance/check_status.yml

    % Revisar alertas
    ansible-playbook playbooks/maintenance/check_alerts.yml
```

2. **Mantenimiento Preventivo**

```
    % Limpiar logs
    ansible-playbook playbooks/maintenance/cleanup.yml

    % Verificar espacio
    ansible-playbook playbooks/maintenance/check_space.yml
```

# .1 Configuration Files

## .1.1 Ansible Configuration

### .1.1.1 Inventory Structure

```
# inventory/hosts.yml
all:
  children:
    management:
      hosts:
        mgmt01:
          ansible_host: 192.168.1.10
        mgmt02:
          ansible_host: 192.168.1.11
    compute:
      hosts:
        compute[01:10]:
          ansible_host: 192.168.1.{{ 20 + (inventory_hostname |
             regex_replace('compute', '') | int) }}
    login:
      hosts:
        login01:
          ansible_host: 192.168.1.30
        login02:
          ansible_host: 192.168.1.31
    storage:
      hosts:
        storage01:
          ansible_host: 192.168.1.40
        storage02:
          ansible_host: 192.168.1.41
    monitoring:
      hosts:
        monitor01:
          ansible_host: 192.168.1.50
```

### .1.1.2 Group Variables

```
# group_vars/all.yml
---
# Global variables
cluster_name: "hpc-cluster"
domain_name: "cluster.local"
timezone: "UTC"

# Network configuration
internal_network: "192.168.1.0/24"
management_network: "192.168.2.0/24"
storage_network: "192.168.3.0/24"
```

```
12
13 # DNS servers
14 dns_servers:
15   - "8.8.8.8"
16   - "8.8.4.4"
17
18 # NTP servers
19 ntp_servers:
20   - "pool.ntp.org"
21   - "time.google.com"
22
23 # Package repositories
24 package_repos:
25   epel:
26     enabled: true
27     gpgcheck: true
28   centos_base:
29     enabled: true
30     gpgcheck: true
31
32 # Security settings
33 security:
34   selinux: "enforcing"
35   firewall: "enabled"
36   ssh:
37     permit_root_login: "no"
38     password_authentication: "no"
39     port: 22
```

## .1.2 SLURM Configuration

### .1.2.1 SLURM Controller

```
1  # /etc/slurm/slurm.conf
2  ClusterName=hpc-cluster
3  ControlMachine=mgmt01
4  ControlAddr=192.168.1.10
5  SlurmUser=slurm
6  SlurmctldPort=6817
7  SlurmdPort=6818
8  AuthType=auth/munge
9  StateSaveLocation=/var/spool/slurm/ctld
10 SlurmdSpoolDir=/var/spool/slurm/d
11 SwitchType=switch/none
12 MpiDefault=none
13 SlurmctldPidFile=/var/run/slurmctld.pid
14 SlurmdPidFile=/var/run/slurmd.pid
15 ProctrackType=proctrack/cgroup
16 ReturnToService=1
17 SlurmctldTimeout=300
18 SlurmdTimeout=300
```

```
InactiveLimit=0
MinJobAge=300
KillWait=30
Waittime=0
SchedulerType=sched/backfill
SelectType=select/cons_res
SelectTypeParameters=CR_Core_Memory
AccountingStorageType=accounting_storage/slurmdbd
AccountingStorageHost=mgmt01
AccountingStoragePort=6819
AccountingStorageUser=slurm
JobCompType=jobcomp/none
TaskPlugin=task/cgroup

# Node definitions
NodeName=compute[01-10] CPUs=64 RealMemory=256000 State=UNKNOWN
PartitionName=compute Nodes=compute[01-10] Default=YES
   MaxTime=INFINITE State=UP
```

### .1.2.2 SLURM Database

```
# /etc/slurm/slurmdbd.conf
ArchiveEvents=yes
ArchiveJobs=yes
ArchiveResvs=yes
ArchiveSteps=no
ArchiveSuspend=no
ArchiveTXN=no
ArchiveUsage=no
AuthType=auth/munge
DbdHost=mgmt01
DbdPort=6819
DbdUser=slurm
DebugLevel=verbose
PurgeEventAfter=12months
PurgeJobAfter=12months
PurgeResvAfter=1month
PurgeStepAfter=1month
PurgeSuspendAfter=1month
PurgeTXNAfter=12months
PurgeUsageAfter=12months
SlurmUser=slurm
StorageType=accounting_storage/mysql
StorageHost=localhost
StoragePort=3306
StorageUser=slurm
StoragePass=password
StorageLoc=slurm_acct_db
```

## .1.3 Storage Configuration

### .1.3.1 NFS Server

```
# /etc/exports
/home     *(rw,sync,no_root_squash,no_subtree_check)
/shared   *(rw,sync,no_root_squash,no_subtree_check)
/scratch  *(rw,sync,no_root_squash,no_subtree_check)

# /etc/fstab
/dev/mapper/vg_home-lv_home    /home    xfs    defaults    0 0
/dev/mapper/vg_shared-lv_shared /shared  xfs    defaults    0 0
/dev/mapper/vg_scratch-lv_scratch /scratch xfs defaults    0 0
```

### .1.3.2 NFS Client

```
# /etc/fstab
storage01:/home    /home    nfs    rw,vers=4,soft,intr    0 0
storage01:/shared  /shared  nfs    rw,vers=4,soft,intr    0 0
storage01:/scratch /scratch nfs    rw,vers=4,soft,intr    0 0
```

## .1.4 Authentication Configuration

### .1.4.1 OpenLDAP Server

```
# /etc/openldap/slapd.conf
include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/duaconf.schema
include         /etc/openldap/schema/dyngroup.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/java.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/openldap.schema
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema

pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args

database        config
rootdn          "cn=admin,cn=config"
rootpw          {SSHA}password

database        mdb
suffix          "dc=cluster,dc=local"
rootdn          "cn=admin,dc=cluster,dc=local"
rootpw          {SSHA}password
```

```
directory       /var/lib/ldap
index           objectClass eq
index           cn eq
index           uid eq
index           memberUid eq
```

### .1.4.2 SSSD Client

```
# /etc/sssd/sssd.conf
[sssd]
domains = cluster.local
services = nss, pam
config_file_version = 2

[domain/cluster.local]
id_provider = ldap
auth_provider = ldap
ldap_uri = ldap://mgmt01
ldap_search_base = dc=cluster,dc=local
ldap_id_use_start_tls = True
cache_credentials = True
ldap_tls_cacertdir = /etc/openldap/cacerts
```

## .1.5 Monitoring Configuration

### .1.5.1 Prometheus

```
# /etc/prometheus/prometheus.yml
global:
  scrape_interval: 15s
  evaluation_interval: 15s

rule_files:
  - /etc/prometheus/rules/*.yml

scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['node-exporter:9100']
    relabel_configs:
      - source_labels: [__address__]
        target_label: instance

  - job_name: 'slurm'
    static_configs:
      - targets: ['slurm-exporter:9101']
    relabel_configs:
      - source_labels: [__address__]
        target_label: instance

```

```
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

### .1.5.2 Grafana

```
// /etc/grafana/grafana.ini
[server]
http_port = 3000
domain = monitor.cluster.local
root_url = https://monitor.cluster.local/

[security]
admin_user = admin
admin_password = password
secret_key = your-secret-key

[auth.anonymous]
enabled = true
org_name = Main Org.
org_role = Viewer

[databases]
type = sqlite3
path = /var/lib/grafana/grafana.db

[metrics]
enabled = true
interval_seconds = 10
```

## .1.6 Software Management

### .1.6.1 Spack Configuration

```
# /opt/spack/etc/spack/config.yaml
config:
  install_tree: /opt/spack/install
  module_roots:
    tcl: /opt/spack/modules
  build_stage:
    - /tmp/spack-stage
  source_cache: /opt/spack/cache
  misc_cache: /opt/spack/cache
  db_lock_timeout: 120
  connect_timeout: 30
  verify_ssl: true
  dirty: false
  license_dir: /opt/spack/licenses
  checksum: true
  install_path_scheme: '{name}/{version}-{hash}'
```

```
  build_jobs: 4
  concretizer:
    unify: true
  compiler:
    implicit_rpaths: true
  packages:
    all:
      compiler: [gcc@10.2.0:, clang@11.0.0:]
      providers:
        mpi: [openmpi@4.1.0:]
        blas: [openblas]
        lapack: [openblas]
```

### .1.6.2 Docker Configuration

```
// /etc/docker/daemon.json
{
  "storage-driver": "overlay2",
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "3"
  },
  "default-ulimits": {
    "nofile": {
      "name": "nofile",
      "hard": 64000,
      "soft": 64000
    }
  },
  "registry-mirrors": [
    "https://registry.cluster.local"
  ],
  "insecure-registries": [
    "registry.cluster.local:5000"
  ],
  "metrics-addr": "0.0.0.0:9323",
  "experimental": true
}
```

## .1.7 Network Configuration

### .1.7.1 Firewall Rules

```
# /etc/firewalld/zones/internal.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Internal</short>
  <description>Internal network access</description>
  <source address="192.168.1.0/24"/>
```

```
  <service name="ssh"/>
  <service name="nfs"/>
  <service name="rpc-bind"/>
  <service name="mountd"/>
  <service name="slurmctld"/>
  <service name="slurmd"/>
  <service name="slurmdbd"/>
  <service name="prometheus"/>
  <service name="grafana"/>
  <port protocol="tcp" port="2375"/>
  <port protocol="tcp" port="2376"/>
  <port protocol="tcp" port="2377"/>
  <port protocol="tcp" port="7946"/>
  <port protocol="udp" port="7946"/>
  <port protocol="udp" port="4789"/>
</zone>
```

### .1.7.2 DNS Configuration

```
# /etc/named.conf
options {
    listen-on port 53 { 127.0.0.1; 192.168.1.0/24; };
    directory       "/var/named";
    dump-file       "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { localhost; 192.168.1.0/24; };
    recursion yes;
    dnssec-enable yes;
    dnssec-validation yes;
    bindkeys-file "/etc/named.iscdlv.key";
    managed-keys-directory "/var/named/dynamic";
    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";
};

zone "cluster.local" IN {
    type master;
    file "cluster.local.zone";
    allow-update { none; };
};

zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "1.168.192.zone";
    allow-update { none; };
};
```

## .1.8 Backup Configuration

### .1.8.1 Backup Script

```
#!/bin/bash
# /usr/local/bin/backup.sh

# Configuration
BACKUP_DIR="/backup"
RETENTION_DAYS=30
DATE=$(date +%Y%m%d)
LOG_FILE="/var/log/backup.log"

# Backup function
backup() {
    local source=$1
    local dest=$2
    local name=$3

    echo "Starting backup of $name at $(date)" >> $LOG_FILE
    tar -czf $dest/$name-$DATE.tar.gz $source
    if [ $? -eq 0 ]; then
        echo "Backup of $name completed successfully" >> $LOG_FILE
    else
        echo "Backup of $name failed" >> $LOG_FILE
        exit 1
    fi
}

# Create backup directory
mkdir -p $BACKUP_DIR

# Backup important directories
backup "/home" "$BACKUP_DIR" "home"
backup "/shared" "$BACKUP_DIR" "shared"
backup "/etc" "$BACKUP_DIR" "config"
backup "/scratch" "$BACKUP_DIR" "scratch"

# Cleanup old backups
find $BACKUP_DIR -type f -mtime +$RETENTION_DAYS -delete

# Sync to remote backup server
rsync -avz --delete $BACKUP_DIR/ backup-server:/backup/
```

### .1.8.2 Backup Crontab

```
# /etc/cron.d/backup
0 2 * * * root /usr/local/bin/backup.sh
```

# .1 Installation Manual

## .1.1 Prerequisites

### .1.1.1 System Requirements

**Hardware Requirements**

- Management Node:
  - CPU: 8+ cores
  - RAM: 32GB minimum
  - Storage: 500GB+ system disk
  - Network: 2x 10GbE ports
- Compute Nodes:
  - CPU: 32+ cores per node
  - RAM: 128GB+ per node
  - Storage: 1TB+ system disk
  - Network: 2x 10GbE ports
- Storage Nodes:
  - CPU: 16+ cores
  - RAM: 64GB+
  - Storage: 10TB+ RAID array
  - Network: 2x 10GbE ports
- Login Nodes:
  - CPU: 16+ cores
  - RAM: 64GB+
  - Storage: 1TB+ system disk
  - Network: 2x 10GbE ports

**Software Requirements**

- Operating System:
  - RHEL/CentOS 8.x or Rocky Linux 8.x
  - Latest security updates
  - Minimal installation with development tools
- Network:
  - Static IP addressing
  - DNS resolution
  - NTP synchronization
  - Firewall configuration
- Storage:
  - RAID controller
  - LVM support
  - XFS or ext4 filesystem
- Security:

- SSH key-based authentication
- SELinux configuration
- Firewall rules
- SSL certificates

## .1.2 Installation Steps

### .1.2.1 Base System Installation

1. Install Operating System:

```
    # Boot from installation media
    # Select "Minimal Install" with "Development Tools"
    # Configure network with static IP
    # Set hostname according to role (e.g., mgmt01, compute01)
    # Create root password and admin user
```

2. Update System:

```
    dnf update -y
    dnf install -y epel-release
    dnf update -y
```

3. Install Base Packages:

```
    dnf install -y \
        vim \
        wget \
        curl \
        git \
        htop \
        iotop \
        iftop \
        net-tools \
        bind-utils \
        chrony \
        nfs-utils \
        openldap-clients \
        sssd \
        sssd-tools \
        sssd-ldap \
        sssd-krb5 \
        sssd-proxy \
        sssd-ipa \
        sssd-ad \
        sssd-common \
        sssd-common-pac \
        sssd-common-pac-devel \
        sssd-common-pac-devel-docs \
        sssd-common-pac-devel-tools \
        sssd-common-pac-devel-tools-docs \
        sssd-common-pac-devel-tools-devel \
```

```
        sssd-common-pac-devel-tools-devel-docs \
        sssd-common-pac-devel-tools-devel-tools \
        sssd-common-pac-devel-tools-devel-tools-docs \
        sssd-common-pac-devel-tools-devel-tools-devel \
        sssd-common-pac-devel-tools-devel-tools-devel-docs
```

4. Configure Network:

```
    # Edit /etc/sysconfig/network-scripts/ifcfg-eth0
    BOOTPROTO=static
    IPADDR=192.168.1.10
    PREFIX=24
    GATEWAY=192.168.1.1
    DNS1=192.168.1.1
    DNS2=8.8.8.8

    # Restart network
    systemctl restart NetworkManager
```

5. Configure Hostname:

```
    # Edit /etc/hostname
    mgmt01.cluster.local

    # Edit /etc/hosts
    192.168.1.10 mgmt01.cluster.local mgmt01
    192.168.1.11 compute01.cluster.local compute01
    192.168.1.12 storage01.cluster.local storage01
    192.168.1.13 login01.cluster.local login01
```

6. Configure NTP:

```
    # Edit /etc/chrony.conf
    server time.google.com iburst
    server time.apple.com iburst

    # Start and enable chronyd
    systemctl enable --now chronyd
```

7. Configure Firewall:

```
    # Install firewalld
    dnf install -y firewalld

    # Start and enable firewalld
    systemctl enable --now firewalld

    # Configure zones
    firewall-cmd --permanent --new-zone=internal
    firewall-cmd --permanent --zone=internal
       --add-source=192.168.1.0/24
    firewall-cmd --permanent --zone=internal --add-service=ssh
```

```
    firewall-cmd --permanent --zone=internal
       --add-service=dhcpv6-client
    firewall-cmd --permanent --zone=internal --add-service=nfs
    firewall-cmd --permanent --zone=internal
       --add-service=rpc-bind
    firewall-cmd --permanent --zone=internal
       --add-service=mountd
    firewall-cmd --permanent --zone=internal
       --add-port=6817-6818/tcp
    firewall-cmd --permanent --zone=internal
       --add-port=3000/tcp
    firewall-cmd --permanent --zone=internal
       --add-port=9090/tcp

    # Apply changes
    firewall-cmd --reload
```

### .1.2.2 Management Node Setup

1. Install Ansible:

```
    dnf install -y ansible

    # Verify installation
    ansible --version
```

2. Configure Ansible:

```
    # Edit /etc/ansible/ansible.cfg
    [defaults]
    inventory = /etc/ansible/hosts
    remote_user = ansible
    host_key_checking = False
    retry_files_enabled = False
    log_path = /var/log/ansible.log

    # Create inventory
    mkdir -p /etc/ansible/inventory
```

3. Install MariaDB:

```
    dnf install -y mariadb-server mariadb

    # Start and enable MariaDB
    systemctl enable --now mariadb

    # Secure installation
    mysql_secure_installation

    # Create SLURM database
    mysql -u root -p << EOF
    CREATE DATABASE slurm_acct_db;
```

```
    CREATE USER 'slurm'@'localhost' IDENTIFIED BY 'password';
    GRANT ALL PRIVILEGES ON slurm_acct_db.* TO
       'slurm'@'localhost';
    FLUSH PRIVILEGES;
    EOF
```

4. Install OpenLDAP:

```
    dnf install -y openldap-servers openldap-clients

    # Start and enable slapd
    systemctl enable --now slapd

    # Configure base DN
    ldapadd -Y EXTERNAL -H ldapi:/// -f
       /etc/openldap/schema/cosine.ldif
    ldapadd -Y EXTERNAL -H ldapi:/// -f
       /etc/openldap/schema/nis.ldif
    ldapadd -Y EXTERNAL -H ldapi:/// -f
       /etc/openldap/schema/inetorgperson.ldif
```

### .1.2.3 Compute Node Setup

1. Install SLURM:

```
    dnf install -y slurm slurm-slurmd slurm-perlapi

    # Create SLURM configuration
    mkdir -p /etc/slurm
```

2. Configure SLURM:

```
    # Copy configuration from management node
    scp mgmt01:/etc/slurm/slurm.conf /etc/slurm/
    scp mgmt01:/etc/slurm/cgroup.conf /etc/slurm/

    # Start and enable slurmd
    systemctl enable --now slurmd
```

3. Install MPI:

```
    dnf install -y openmpi-devel openmpi

    # Configure environment
    echo "module load mpi/openmpi-x86_64" >
       /etc/profile.d/mpi.sh
```

4. Install Development Tools:

```
    dnf groupinstall -y "Development Tools"
    dnf install -y \
        gcc \
```

```
        gcc-c++ \
        gcc-gfortran \
        make \
        cmake \
        autoconf \
        automake \
        libtool \
        m4 \
        flex \
        bison \
        python3-devel
```

#### .1.2.4 Storage Node Setup

1. Install NFS Server:

```
    dnf install -y nfs-utils

    # Start and enable NFS
    systemctl enable --now nfs-server
    systemctl enable --now rpcbind
```

2. Configure NFS Exports:

```
    # Edit /etc/exports
    /shared     192.168.1.0/24(rw,sync,no_root_squash)
    /home       192.168.1.0/24(rw,sync,no_root_squash)
    /opt        192.168.1.0/24(ro,sync,no_root_squash)

    # Apply exports
    exportfs -ra
```

3. Configure RAID:

```
    # Install RAID tools
    dnf install -y mdadm

    # Create RAID array
    mdadm --create /dev/md0 --level=6 --raid-devices=4
       /dev/sd[b-e]

    # Create filesystem
    mkfs.xfs /dev/md0

    # Mount filesystem
    mkdir -p /shared
    mount /dev/md0 /shared
    echo "/dev/md0 /shared xfs defaults 0 0" >> /etc/fstab
```

4. Configure Quotas:

```
    # Install quota tools
    dnf install -y quota

    # Enable quotas
    mount -o remount,usrquota,grpquota /shared
    quotacheck -cugm /shared
    quotaon -v /shared

    # Set default quotas
    edquota -p template -u *
    edquota -p template -g *
```

#### .1.2.5 Login Node Setup

1. Install NFS Client:

```
    dnf install -y nfs-utils

    # Start and enable NFS
    systemctl enable --now rpcbind
```

2. Configure NFS Mounts:

```
    # Edit /etc/fstab
    storage01:/shared /shared nfs defaults 0 0
    storage01:/home   /home   nfs defaults 0 0
    storage01:/opt    /opt    nfs ro,defaults 0 0

    # Mount filesystems
    mount -a
```

3. Install Development Tools:

```
    dnf groupinstall -y "Development Tools"
    dnf install -y \
        gcc \
        gcc-c++ \
        gcc-gfortran \
        make \
        cmake \
        autoconf \
        automake \
        libtool \
        m4 \
        flex \
        bison \
        python3-devel
```

4. Install Environment Modules:

```
    dnf install -y environment-modules

    # Initialize modules
    source /etc/profile.d/modules.sh
```

## .1.3 Post-Installation

### .1.3.1 Verification

1. Verify SLURM:

```
    # Check controller status
    systemctl status slurmctld

    # Check node status
    sinfo -N

    # Submit test job
    srun hostname
```

2. Verify NFS:

```
    # Check exports
    showmount -e storage01

    # Check mounts
    mount | grep nfs

    # Test write access
    touch /shared/test.txt
```

3. Verify LDAP:

```
    # Check server status
    systemctl status slapd

    # Test authentication
    getent passwd
    id <username>
```

4. Verify Monitoring:

```
    # Check Prometheus
    curl localhost:9090/-/healthy

    # Check Grafana
    curl localhost:3000/api/health
```

### .1.3.2 Backup Configuration

1. Configure Backup Script:

```
# Create backup script
cat > /usr/local/bin/backup.sh << 'EOF'
#!/bin/bash

# Backup configuration files
tar -czf /backup/config-$(date +%Y%m%d).tar.gz \
    /etc/slurm \
    /etc/ansible \
    /etc/openldap \
    /etc/prometheus \
    /etc/grafana

# Backup databases
mysqldump -u root -p slurm_acct_db >
   /backup/mysql/slurm_acct_db-$(date +%Y%m%d).sql

# Backup LDAP database
slapcat > /backup/ldap/ldap-$(date +%Y%m%d).ldif

# Clean up old backups
find /backup -type f -mtime +30 -delete
EOF

# Make executable
chmod +x /usr/local/bin/backup.sh
```

2. Configure Backup Schedule:

```
# Add to crontab
echo "0 2 * * * /usr/local/bin/backup.sh" >
   /etc/cron.d/backup
```

### .1.3.3 Security Hardening

1. Configure SELinux:

```
# Set enforcing mode
setenforce 1

# Configure policies
setsebool -P slurm_use_nfs 1
setsebool -P nfs_export_all_ro 1
setsebool -P nfs_export_all_rw 1
```

2. Configure SSH:

```
# Edit /etc/ssh/sshd_config
PermitRootLogin no
PasswordAuthentication no
```

```
    PubkeyAuthentication yes
    AllowUsers ansible

    # Restart SSH
    systemctl restart sshd
```

3. Configure Firewall:

```
    # Block all incoming traffic
    firewall-cmd --permanent --zone=public --set-target=DROP

    # Allow only internal zone
    firewall-cmd --permanent --zone=internal
       --set-target=ACCEPT

    # Apply changes
    firewall-cmd --reload
```

4. Install Security Updates:

```
    # Configure automatic updates
    dnf install -y dnf-automatic

    # Enable automatic updates
    systemctl enable --now dnf-automatic.timer
```

# .1 Troubleshooting Guide

## .1.1 Common Issues and Solutions

### .1.1.1 SLURM Issues

**SLURM Controller Not Starting** 1. Check controller status:

```
systemctl status slurmctld
journalctl -u slurmctld
```

2. Verify configuration:

```
slurmctld -C
slurmctld -v
```

3. Check permissions:

```
ls -l /var/spool/slurm/ctld
ls -l /var/run/slurmctld.pid
```

4. Verify database connection:

```
mysql -u slurm -p -e "SELECT 1"
```

**Compute Nodes Not Responding** 1. Check node status:

```
sinfo -N
scontrol show nodes
```

2. Verify slurmd service:

```
systemctl status slurmd
journalctl -u slurmd
```

3. Check network connectivity:

```
ping mgmt01
telnet mgmt01 6817
```

4. Verify munge authentication:

```
munge -n
munge -n | unmunge
```

**Jobs Not Starting** 1. Check job status:

```
squeue -j <job_id>
scontrol show job <job_id>
```

2. Verify resource availability:

```
sinfo -s
scontrol show nodes
```

3. Check job requirements:

```
scontrol show job <job_id> | grep ReqNodeList
scontrol show job <job_id> | grep ReqMem
```

4. Verify job script:

```
cat <job_script>
sbatch --test-only <job_script>
```

### .1.1.2 Storage Issues

**NFS Mount Failures** 1. Check NFS server:

```
systemctl status nfs-server
showmount -e storage01
```

2. Verify exports:

```
cat /etc/exports
exportfs -v
```

3. Check client mounts:

```
mount | grep nfs
df -h
```

4. Verify network:

```
ping storage01
telnet storage01 2049
```

**Performance Issues** 1. Check I/O statistics:

```
iostat -x 1
nfsstat -c
```

2. Monitor network:

```
iftop
netstat -i
```

3. Check disk usage:

```
df -h
du -sh /*
```

4. Verify quotas:

```
quota -s
repquota -a
```

### .1.1.3 Authentication Issues

**LDAP Connection Problems** 1. Check LDAP server:

```
systemctl status slapd
ldapsearch -x -b "dc=cluster,dc=local"
```

2. Verify SSSD:

```
systemctl status sssd
sssctl domain-list
```

3. Check certificates:

```
openssl s_client -connect mgmt01:636
certutil -L -d /etc/openldap/cacerts
```

4. Test authentication:

```
getent passwd
id <username>
```

**User Access Issues** 1. Check user account:

```
ldapsearch -x -b "dc=cluster,dc=local"
   "uid=<username>"
getent passwd <username>
```

2. Verify group membership:

```
getent group
id <username>
```

3. Check permissions:

```
ls -l /home/<username>
ls -l /shared
```

4. Test login:

```
ssh -v <username>@login01
```

### .1.1.4 Monitoring Issues

**Prometheus Problems** 1. Check service status:

```
systemctl status prometheus
journalctl -u prometheus
```

2. Verify configuration:

```
promtool check config /etc/prometheus/prometheus.yml
curl localhost:9090/-/healthy
```

3. Check targets:

```
curl localhost:9090/api/v1/targets
curl localhost:9090/api/v1/query?query=up
```

4. Verify storage:

```
du -sh /var/lib/prometheus
df -h /var/lib/prometheus
```

**Grafana Issues** 1. Check service status:

```
systemctl status grafana-server
journalctl -u grafana-server
```

2. Verify database:

```
sqlite3 /var/lib/grafana/grafana.db ".tables"
```

3. Check data sources:

```
curl -u admin:admin
   http://localhost:3000/api/datasources
```

4. Verify dashboards:

```
curl -u admin:admin
   http://localhost:3000/api/dashboards
```

### .1.1.5 Network Issues

**Connectivity Problems** 1. Check network interfaces:

```
ip addr show
nmcli device show
```

2. Verify routing:

```
ip route show
traceroute mgmt01
```

3. Check DNS:

```
dig mgmt01.cluster.local
nslookup mgmt01.cluster.local
```

4. Test connectivity:

```
ping mgmt01
telnet mgmt01 22
```

**Firewall Issues** 1. Check firewall status:

```
systemctl status firewalld
firewall-cmd --list-all
```

2. Verify rules:

```
    firewall-cmd --list-services
    firewall-cmd --list-ports
```

3. Check zones:

```
    firewall-cmd --get-active-zones
    firewall-cmd --zone=internal --list-all
```

4. Test access:

```
    telnet mgmt01 6817
    telnet mgmt01 6818
```

## .1.2 Performance Tuning

### .1.2.1 System Tuning

**CPU Tuning** 1. Check CPU settings:

```
    cat /proc/cpuinfo
    cpupower frequency-info
```

2. Verify governor:

```
    cat
       /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

3. Check CPU load:

```
    top
    mpstat -P ALL
```

4. Monitor processes:

```
    ps aux | sort -nrk 3,3
    pidstat -u 1
```

**Memory Tuning** 1. Check memory usage:

```
    free -h
    vmstat 1
```

2. Verify swap:

```
    swapon -s
    cat /proc/sys/vm/swappiness
```

3. Monitor memory pressure:

```
    cat /proc/pressure/memory
```

4. Check huge pages:

```
    cat /proc/meminfo | grep Huge
```

**I/O Tuning** 1. Check I/O statistics:

```
    iostat -x 1
    iotop
```

2. Verify disk usage:

```
    df -h
    du -sh /*
```

3. Monitor I/O wait:

```
    vmstat 1
    pidstat -d 1
```

4. Check file system:

```
    tune2fs -l /dev/sda1
    xfs_info /dev/sda1
```

### .1.2.2 Application Tuning

**MPI Tuning** 1. Check MPI version:

```
    mpirun --version
    ompi_info
```

2. Verify environment:

```
    env | grep MPI
    module list
```

3. Test performance:

```
    mpirun -np 4 ./xhpl
```

4. Monitor processes:

```
    ps aux | grep mpi
    top -p $(pgrep -d',' -f mpi)
```

**Container Tuning** 1. Check container status:

```
    docker ps
    docker stats
```

2. Verify resources:

```
    docker info
    cgroup stats
```

3. Monitor performance:

```
    docker top <container>
    docker logs <container>
```

4. Check networking:

```
    docker network ls
    docker network inspect bridge
```

## .1.3 Recovery Procedures

### .1.3.1 System Recovery

**Node Recovery** 1. Check system status:

```
    systemctl status
    journalctl -b
```

2. Verify services:

```
    systemctl list-units --failed
    systemctl list-dependencies
```

3. Check file systems:

```
    fsck -f /dev/sda1
    xfs_repair /dev/sda1
```

4. Restore from backup:

```
    tar -xzf /backup/config-20230101.tar.gz -C /
```

**Service Recovery** 1. Stop services:

```
    systemctl stop slurmctld slurmd
```

2. Clean up state:

```
    rm -f /var/run/slurmctld.pid
    rm -f /var/run/slurmd.pid
```

3. Restore configuration:

```
    cp /etc/slurm/slurm.conf.bak /etc/slurm/slurm.conf
```

4. Start services:

```
    systemctl start slurmctld slurmd
```

### .1.3.2 Data Recovery

**File Recovery** 1. Check backup status:

```
    ls -l /backup
    du -sh /backup/*
```

2. List available backups:

```
    tar -tvf /backup/home-20230101.tar.gz
```

3. Restore files:

```
    tar -xzf /backup/home-20230101.tar.gz -C /tmp
```

4. Verify restoration:

```
diff -r /home /tmp/home
```

**Database Recovery** 1. Check database status:

```
systemctl status mariadb
mysql -e "SHOW DATABASES;"
```

2. Verify backups:

```
ls -l /backup/mysql
```

3. Restore database:

```
mysql -u root -p < /backup/mysql/slurm_acct_db.sql
```

4. Check integrity:

```
mysqlcheck -u root -p slurm_acct_db
```