Título (por hacer)

Pau Santana

Grupo 38

Project motivation & introduction

ENGLISH

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

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 que sea escalable, mantenible y seguro que pueda ser desplegado fácilmente por los administradores del clúster. 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.

Índice (por hacer)

0.1 Visión general del proyecto

Este proyecto implementa un sistema automatizado de despliegue y gestión de clústeres de HPC, usando Ansible cómo la herramienta principal. El sistema está diseñado para proporcionar un entorno HPC completo y listo para producción en todos los servicios esenciales para la computación científica.

0.0.1 Objetivos del proyecto

Los objetivos principales del 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 (HA) y fiabilidad de los servicios principales del clúster.
- Proporcionar capacidades completas de monitoreo, generación de informes y visibilidad del estado del clúster y servicios clave.
- Facilitar el mantenimiento y las actualizaciones de infraestructura del clúster.
- Punto centralizado de gestión del stack de software científico usando Spack y contenedores.
- Implementar mejores prácticas de seguridad a través de todos los componentes del clúster.

0.0.2 Alcance del proyecto

El proyecto abarca los siguientes componentes clave:

1. Infraestructura de computación
    - Planificador de trabajos y gestor de recursos/colas: SLURM
    - Aprovisionamiento y configuración de los nodos de cómputo.
    - Configuración y gestión de nodos de acceso. (login nodes)
    - Monitoreo y gestión de energía.

2. Infraestructura de almacenamiento.
    - Configuración de un servidor y cliente NFS.
    - Almacenamiento compartido para directorios de usuarios y volúmenes de datos.
    - Gestión de espacio para aplicaciones y datos temporales.
    - Gestión automatizada de montajes vía autofs.

3. Seguridad y autenticación
    - Servicio de directorio OpenLDAP con SSL.
    - 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 monitorización/observación en Grafana.
    - Despliegue de Node Exporter.
    - Soluciones de monitoreo personalizadas.
    - Sistema automatizado de informes.

5. Gestión del software
    - Integración del gestor de paquetes científicos Spack.
    - Despliegue de aplicaciones usando contenedores.
    - Gestión de entornos de software cíentifico.

6. Gestión de infraestructura.
    - Configuración de un servicio DNS y DHCP.
    - Implementación de un sistema de backups.
    - 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 considerablemente:** 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 validaciones automatizadas de los distintos 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 dinámica.

0.1.4 Partes interesadas

Las partes interesadas en este proyecto son:

- **CientiGO:** Como la organización implementadora de la solución.
- **Administradores del clúster:** Personal técnico responsable de la gestión del clúster.
- **Usuarios finales:** Investigadores que quieren usar el clúster.
- **Personal de soporte técnico:** Personal de soporte técnico del clúster.
- **Integradores de sistemas:** Equipo responsable de la integración del clúster con la infraestructura ya existente. (este en caso de entornos con complejidad añadida)

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 automatización del despliegue 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/entrenamiento para administradores y usuarios estén disponibles.

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 sus hitos y objetivos en específico:

Fase 1: Planificación y Diseño (Semana 1)

- Recopilación y análisis de requisitos
- Diseño de arquitectura
- Selección de stack tecnológico
- Planificación de recursos
- Evaluación de riesgos

Fase 2: Desarrollo de Infraestructura Base (Semanas 2-3)

- 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

Fase 3: Implementación de Servicios (Semanas 4-6)

- 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

Fase 4: Pruebas y Validación (Semana 6)

- Pruebas de componentes
- Pruebas de integración

- Pruebas de rendimiento
- Pruebas de seguridad
- Revisión de documentación

Fase 5: Despliegue y Capacitación (Semana 7)

- 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 (caso empresa real)

Rol	**Responsabilidades**	**Asignación del tiempo**
Project Manager	Coordinación del proyecto	Tiempo completo
Systems Architect	Diseño y revisión de arquitectura	Tiempo completo
Ansible Developer	Desarrollo y pruebas de roles	Tiempo completo
DevOps Engineer	Infraestructura y CI/CD	Tiempo parcial
Security Specialist	Implementación de seguridad	Tiempo parcial

Tabla 1: Asignación de Recursos Humanos.

0.2.2.2 Recursos técnicos (en un caso real)

Recurso	**Especificación**	**Cantidad**
Servidores para el desarrollo	2x CPU, 16GB RAM, 500GB SSD	2
Entorno de pruebas	4x CPU, 32GB RAM, 1024 GB SSD	2
Control de versiones	GitHub Enterprise o GitLab Enterprise	1
Pipeline CI/CD	GitHub Actions (implementación actual) o GitLab CI.	1
Sistema de documentación	Confluence	1
Sistema de monitoreo	Prometheus/Grafana	1

Tabla 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

Tabla 3: Evaluación y mitigación de riesgos

0.2.4 Aseguramiento de calidad

El proyecto implementa varias medidas para asegurar la calidad del código.

1. Calidad del código
    - Proceso de revisión del código vía CI
    - Linting automatizado
    - Pruebas unitarias usando Molecule
    - Pruebas de integración

2. Documentación
    - Documentación técnica
    - Guías de usuario
    - Guías de despliegue

3. Pruebas
    - Pruebas automatizadas
    - Pruebas de rendimiento
    - Pruebas de seguridad

4. Monitoreo
    - Monitoreo de la salud del sistema (Health checks)
    - Monitoreo del rendimiento
    - Monitoreo de seguridad
    - Monitoreo de uso

0.2.5 Plan de comunicación (caso real con empresas)

Se establecen canales de comunicación y reuniones regulares en caso de que dichas sean solicitadas:

- Reuniones de seguimiento
- Reuniones de progreso
- Informes mensuales del estado del proyecto
- Actualizaciones de documentación
- Seguimiento y resolución de incidencias.

0.2.6 Gestión del cambio

El proyecto adopta un proceso estructurado de gestión del cambio, aprovechando las ventajas del modelo Open Source:

1. **Solicitud de cambio (Pull Request):**
   Los cambios propuestos se presentan mediante una *pull request* en GitHub, partiendo de la suposición de que el repositorio es un *fork* del proyecto principal (*upstream*). Esta solicitud detalla qué se modifica y por qué.

2. **Análisis de impacto:**
   Se evalúan los efectos potenciales del cambio en el funcionamiento del sistema, identificando posibles riesgos, dependencias y beneficios.

3. **Proceso de decisión colaborativo:**
   La comunidad o los responsables del proyecto revisan la *pull request*, decidiendo si aceptan o rechazan el cambio. Este proceso puede incluir comentarios, sugerencias y revisiones adicionales.

4. **Planificación de la implementación:**
   Una vez aprobado, se elabora un plan para aplicar el cambio de forma ordenada, minimizando interrupciones y asegurando compatibilidad.

5. **Pruebas y validación:**
   Se ejecutan pruebas para verificar que el cambio funciona como se espera, sin introducir errores o efectos no deseados.

6. **Despliegue e integración con el repositorio principal:**
   Si las pruebas son satisfactorias, el cambio se despliega y se fusiona (*merge*) con la rama principal del proyecto o el repositorio *upstream*.

7. **Actualización de la documentación:**
   Finalmente, se actualiza la documentación técnica para reflejar las modificaciones realizadas, garantizando su trazabilidad y comprensión futura.

0.2.7 Entregables del proyecto

Entregables clave para cada fase:

Fase 1

- Plan de proyecto
- Documento de diseño de arquitectura
- Documento del stack tecnológico
- Informe de evaluación de riesgos

Fase 2

- Roles Ansible principales
- Estructura de inventario
- Playbooks básicos
- Marco de pruebas

Fase 3

- Documentación de la implementación de los servicios
- Guías de configuración
- Resultados de pruebas de integración
- Informes de pruebas de rendimiento.

Fase 4

- Informes de pruebas
- Resultados de auditoría de seguridad
- Puntos de referencia de rendimiento
- Documentación del sistema

Fase 5

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

0.3 Antecedentes técnicos

0.3.1 Computación de alto rendimiento (HPC)

La computación de alto rendimiento (HPC) se refiere al uso de procesamiento paralelo entre distintos sistemas de altas prestaciones de manera eficiente, confiable y rápida para el proceso de información masiva o cálculos computacionalmente muy exigentes. Sus campos de aplicación más comunes dentro del mundo de la ciencia son la simulación climática, secuenciación genómica, etc.

Los clústeres de HPC consisten en:

- **Nodos de cómputo:** Servidores de altas prestaciones optimizados para tareas computacionales.
- **Nodos de acceso:** Puntos de acceso para que los usuarios comunes puedan conectarse al clúster.
- **Nodos de gestión:** Controlan y monitorean el clúster. Entre ellos, podría ser una base de datos de SLURM, un host master de SGE, etc.
- **Sistemas de almacenamiento:** Almacenamiento de altas prestaciones para el análisis de datos de gran volúmen.
- **Red de alta velocidad:** Interconexión de alta velocidad para comunicación entre nodos.

0.3.2 Ansible

Ansible es una plataforma de automatización de código abierto desarrollada por Red Hat Inc para gestión de configuraciones, despliegue de aplicaciones y automatización de tareas, siendo muy útil y versátil en entornos a gran escala. Sus características principales son:

- **Agentless architecture (arquitectura sin agente)**: Usa SSH para ejecución remota de comandos y módulos de Python, no hace falta instalación previa en los hosts objetivo.
- **Lenguaje declarativo:** Ansible también tiene soporte para lenguajes imperativos. Playbooks basados en YAML.
- **Idempotencia:** Doble ejecución de playbooks con seguridad de no comprometer nada.
- **Extensibilidad:** Módulos personalizados y capacidad de adición de plugins.

- **Gestión de inventario:** Gestión dinámica de hosts, permitiéndonos a los administradores crear separar por grupos, subgrupos y más. También permite formato YAML, al igual que los playbooks.

0.3.3 Gestor de cargas de trabajo (Workload Manager) SLURM

SLURM (Simple Linux Utility for Resource Management) es un planificador de trabajos (de aquí en adelante referenciado cómo: "workload manager") y gestor de recursos de código abierto. Principalmente se compone de:

- **slurmctld**: Demonio de gestión central, nodo controlador.
- **slurmd:** Demonio de nodo de cómputo.
- **slurmdbd:** Demonio gestor de la base de datos de accounting.
- **sacctmgr:** Herramienta de gestión de cuentas y grupos de SLURM.
- **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 distribuidos que permite acceso remoto a archivos:

- **NFSv4:** Última versión con seguridad mejorada
- **Ajuste de rendimiento:** Optimizado para cargas de trabajo HPC.
- **Seguridad:** Autenticación usando Kerberos.
- **Alta disponibilidad:** Configuraciones de backups.

0.3.4.2 AutoFS

AutoFS es un sistema de montaje automático para sistemas de archivos que usan NFS:

- **Montaje bajo demanda:** Monta el directorio cuando se accede a él, únicamente.
- **Desmontaje automático:** Ahorra recursos.
- **Mapeado directo e indirecto:** Tiene soporte para ambos puntos de montaje estáticos o dinámicos.

0.3.5 Autenticación y seguridad

0.3.5.1 OpenLDAP

Implementación del Protocolo Ligero de Acceso a Directorios (LDAP):

- **Estructura de directorio:** Obtenemos una organización jerárquica de datos.
- **Replicación:** Replicación multi-maestro, con HA.
- **Seguridad:** Cuenta con soporte para cifrado TLS/SSL.
- **Esquema:** Definiciones de atributos personalizados. (e.g.: admins-hpc cómo grupo admin de HPC, admins-it cómo admins generales de la empresa que NO son administradores de HPC)

0.3.5.2 Seguridad del sistema

Medidas de seguridad implementadas:

- **Firewall:** Reglas iptables/firewall-cmd.
- **SELinux:** Control de acceso obligatorio y enforced.
- **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 sobre métricas.
- **Descubrimiento de servicios:** Detección dinámica de objetivos.

0.3.6.2 Grafana

Plataforma de observabilidad, análisis y visualización de datos:

- **Paneles:** Visualizaciones personalizadas.
- **Alertas:** Alertas basadas en umbrales. (e.g.: >80% uso de cpu)
- **Plugins:** Funcionalidad extendida.
- **Autenticación:** Acceso con RBAC e integración con LDAP.

0.3.7 Gestión de software científico

0.3.7.1 Spack

Spack es un gestor de paquetes para softwares científicos:

- **Multi-versión:** Múltiples versiones de software
- **Dependencias:** Resolución e instalación automática de dependencias.
- **Entornos:** Stacks de software aislados del entorno.
- **Compiladores:** Soporte para múltiples compiladores.

0.3.7.2 Contenedores

Contenerización de aplicaciones:

- **Docker:** Sirve cómo el runtime de los contenedores así cómo partes de la infraestructura que no recaen sobre la performance necesaria para HPC.
- **Singularity:** Contenedores orientados a workloads para HPC.
- **Registry:** Almacenamiento de imágenes por versiones o por tipo de computación, GPU o CPU. Docker y Singularity tienen retrocompatibilidad.
- **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 y aisladas.
- **QoS:** Calidad de servicio

0.3.9 Backups y disaster recovery

0.3.9.1 Backups

Estrategias de protección de datos:

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

0.3.9.2 Disaster recovery

Procedimientos de recuperación:

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

0.3.10 Optimización de rendimiento

0.3.10.1 Técnicas de optimización Linux.

Técnicas de optimización aplicadas por defecto:

- **Parámetros del kernel:** Ajustes del sistema.
- **Red:** Optimización TCP/IP.
- **Almacenamiento:** Optimización del sistema I/O (E/S, en español)
- **Memoria:** Gestión de caché.

0.3.10 Monitoreo y perfilado

Herramientas de perfilado y análisis de rendimiento:

- **Node Exporter:** Métricas del sistema.
- **SLURM exporter:** Métricas de uso de particiones, jobs, alocación de CPUs, etc.
- **Monitoreo de energía:** Uso de energía integrado con varios plugins de SLURM.
- **Perfilado de aplicaciones:** Optimización de código dónde sea aplicable.

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

Tabla 4: Costos de licencias de software, costes en dólares americanos.

0.4.1.2 Componentes comerciales

Componentes comerciales opcionales y sus costos adheridos:

Componente	**Proveedor**	**Tipo de licencia**	**Costo Anual**
RHEL (Red Hat Enterprise Linux)	Red Hat	Por zócalo de CPU.	$799/socket
Ansible Tower	Red Hat	Por Nodo	$100/nodo
Grafana Enterprise	Grafana Labs	Por Usuario	$20/usuario/mes
Prometheus Enterprise	Grafana Labs	Por Instancia	$1.000/instancia

Tabla 5: Costos de software comercial opcional, en dólares americanos

0.4.2 Costos de hardware

0.4.2.1 Infraestructura de cómputo

Costos estimados de hardware para un clúster de un tamaño medio:

Componente	**Especificación**	**Cantidad**	**Costo Total**
Nodos de cómputo	2x CPU (64C/64T), 256GB RAM, 2TB NVMe	10	$250.000
Nodos de acceso	2x CPU (24C/24T), 128GB RAM, 1TB SSD	2	$20.000
Nodos de gestión	2x CPU (12C/12T), 64GB RAM, 500GB SSD	2	$12,000
Servidor de almacenamiento	4x CPU (24C/24T), 256GB RAM, 100 TB	2	$80.000
Switch de red	100 GbE, 48 puertos	2	$40.000

Tabla 6: Costes de hardware, en dólares americanos.

0.4.2.2 Infraestructura de almacenamiento

Costos del sistema de almacenamiento

Componente	**Especificación**	**Cantidad**	**Costo Total**
Almacenamiento primario	100 TB NVMe	1	$50.000
Almacenamiento de backups	200 TB HDD	1	$30.000
Biblioteca de cintas	LTO-9, 100TB	1	$25.000
Red de almacenamiento	100 GbE	1	$20.000

Tabla 7: Costes de infraestructura de almacenamiento, en USD.

0.4.3 Costos de implementación

0.4.3.1 Servicios profesionales

Costos de implementación 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

Tabla 9: Costes de mantenimiento. En dólares americanos.

0.4.4 Costos operativos

0.4.4.1 Energía y refrigeración

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

Tabla 10: Costos operativos. Precio por kWh en España a quince de mayo del 2025.

0.4.4.2 Costos de personal

Costos anuales de persona

Posición	**Nivel**	**Salario Anual**	**Beneficios**
Administrador de Sistemas	Senior	$120.000	$30.000
Ingeniero HPC	Intermedio	$130.000	$25.000
Especialista en administración	Senior	$110.000	$27.500
Ingeniero de redes	Intermedio	$95.000	$23.750

Tabla 11: Costos de personal. En dólares americanos.

0.4.5 Costo total de propiedad (TCO)

0.4.5.1 Inversión inicial

Costos del primer año:

- Hardware: $457.000
- Licencias de software (si aplica): $50.000
- Implementación: $65.900
- Capacitaciones: $7.500
- **Inversión total: $580.400**

0.4.5.2 Costos operativos anuales

Costos anuales recurrentes:

- Mantenimiento: $107.000
- Energía y Refrigeración: $137.000
- 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

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 del 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 de la infraestructura en específico. Los roles están organizados de esta forma:

Categoria	Roles incluidos
Infraestructura base	compute, login_node, nfs_server, nfs_client, dns, epel
	openldap, firewall, autofs, backup
Seguridad y autenticación	prometheus, grafana, node_exporter monitoring, slurm_power_monitoring, reporting, proxmox_monitoring
Monitorización	spack, container_apps, docker, hpl
	foreman, hardware_drivers, ear
Gestión externa y orquestación	foreman, hardware_drivers, ear
	slurmctld, slurmdbd
Gestor de trabajos (SLURM)	

Nota para el lector: Dada la naturaleza de la cantidad de archivos en los roles, no es posible añadir una captura de pantalla con todo un *tree,* así que está detallado con una imagen descriptiva. Esto también se ve reflejado en los playbooks de Ansible, dónde, con su idempotencia, algunos superan las 200 líneas de código. Para los playbooks, dejaré un enlace directo al repositorio de GitHub donde se ubican, para que usted pueda revisar el código con un entorno e interfaz más ideal y adecuado.

0.5.2 Detalles de la implementación de los roles

0.5.2.1 Gestión de SLURM

Rol slurmctld: El rol del controlador SLURM que implementa el demonio de gestión central del clúster se puede encontrar en este enlace:

https://github.com/psantana5/ansible-hpc/tree/main/roles/slurmctld

El playbook que sirve cómo entry point para el despliegue del software:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/slurm/slurmctld.yml

En este rol, realizamos (según estructura de carpetas para el rol):

- **handlers:** interacciones con systemd para el reinicio del daemon de slurmctld, el firewall de los nodos para permitir el tráfico por el puerto por defecto del demonio controlador de slurm y MUNGE.
- **tasks:** en este rol, en específico, antes de proceder a la instalación, hacemos unas pruebas con llaves SSH, para verificar que el host target tenga la llave del host de origen. Después, instalamos los paquetes slurm-slurmctld y munge desde los repositorios de EPEL, creamos el usuario y grupo "slurm" si no existe con un UID fijo a través de todo el clúster (importante para que munge no detecte inconsistencias), creamos una llave munge y los directorios requeridos para que munge pueda iniciarse.

  Seguidamente, se genera el archivo slurm.conf, que define la configuración de SLURM, que luego es copiado a todos los hosts que caigan debajo del grupo [slurm:children] Más relevantemente, este archivo contiene las particiones y hosts que tendrá el clúster. Desde nuestro Ansible inventory, un jinja template añade los hosts dentro de slurm.conf dinámicamente mientras estén bajo el grupo de [compute]

  También, para una mejor UX en el clúster y un nivel de logging más elevado, crearemos unos simples scripts de prólogo y epílogo.

- **templates:** contiene los archivos de configuración de slurm, munge y ambos los scripts epilogue y prologue en formato de jinja templates (.j2), para que puedan ser editados dinámicamente en función de las variables que haya especificado el usuario.

Rol slurmdbd: El rol de la base de datos SLURM que gestiona la contabilidad y almacena toda la información cuantitativa del clúster. El rol se puede encontrar en este enlace:

https://github.com/psantana5/ansible-hpc/tree/main/roles/slurmdbd

El playbook que sirve cómo entry point para todo el despliegue de slurmdbd se puede encontrar en este enlace:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/slurm/slurmdbd.yml

En este rol, realizamos (según estructura de carpetas para el rol):

- **handlers:** Interacciones con systemd para reiniciar la base de datos, reiniciar el demonio de la base de datos de slurm, recargar el firewall y reiniciar munge.
- **tasks:** En primer lugar, este playbook busca si existen llaves RSA en el host, para seguir o crear una nueva. (en caso de crear una llave nueva, la copia al host remoto). Seguidamente, instala los paquetes requeridos para la instalación de la base de datos (mariadb-server, munge, python3-PyMySQL), crea el usuario y grupo *slurm* con un UID y GID específico para MUNGE, crea el archivo de configuración de la base de datos de SLURM, crea la base de datos y el usuario slurm dentro de MariaDB, genera la munge key, copia una configuración predefinida para el demonio de la base de datos específica para soportar cargas de trabajo mayores, se establece la password para root de la base de datos y por último, inicia todos los demonios.
- **templates:** Contiene los archivos de configuración para mariadb y para la base de datos de SLURM.

0.5.2.2 Gestión de almacenamiento

Rol nfs_server. El rol de servidor NFS que implementa el almacenamiento compartido puede ser encontrado en este enlace:

https://github.com/psantana5/ansible-hpc/tree/main/roles/nfs_server

El playbook que sirve cómo entry point para empezar la instalación del servidor NFS, puede ser encontrado en este enlace:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/core/nfs-server.yml

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **handlers:** en este caso, únicamente necesita recargar los exports usando el comando *exportfs -ra,* no hay interacciones con systemd.
- **tasks:** Este playbook prepara un servidor NFS en sistemas Rocky Linux 8 o 9. Primero instala los paquetes necesarios y activa servicios como nfs-server, rpcbind y nfs-mountd. Si firewalld está activo, abre los puertos requeridos para permitir el tráfico NFS.

  Después, asegura los permisos adecuados en /var/lib/nfs y crea los directorios típicos de un entorno HPC: /apps, /apps/software, /apps/libraries, /apps/data, /apps/spack, /scratch y /home. El directorio /apps/spack recibe permisos más amplios (0775) para permitir colaboración.

  El archivo /etc/exports, que define qué carpetas se comparten y con qué opciones, se genera a partir de una plantilla (exports.j2). También se despliega un archivo /etc/nfs.conf optimizado mediante otra plantilla (nfs.conf.j2), ambos con handlers que reinician o recargan el servicio al cambiar.

  Además, aplica parámetros sysctl para optimizar el rendimiento del servidor NFS (tamaño de buffers, conexiones, etc.), y fuerza el uso de TCP en mountd desactivando UDP en /etc/sysconfig/nfs.

- **templates:** En último lugar, existen dos templates de archivos de configuración, uno con todos los parámetros requeridos para exportar las carpetas cómo shares NFS con párametros de seguridad y policies al exportar para obtener un mayor rendimiento. También, existe un archivo bajo el nombre nfs.conf.j2, que incluye parámetros para forzar NFSv4, que ofrece ventajas sobre NFSv3, y con un límite de hilos para el demonio de 128.

Rol nfs_client. El rol de cliente NFS que implementa y monta los directorios compartidos en todos los hosts se puede encontrar aquí:

https://github.com/psantana5/ansible-hpc/tree/main/roles/nfs_client/tasks

El playbook que sirve cómo entry point para empezar la configuración del cliente NFS en todos los hosts, puede ser encontrado en este enlace:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/nfs-client.yml

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **tasks:** Este playbook configura un cliente NFS en un nodo que montará directorios remotos desde un servidor. En primer lugar, instala los paquetes necesarios (nfs-utils y rpcbind) para habilitar la funcionalidad NFS en el sistema.

  A continuación, se asegura de que existan los puntos de montaje necesarios para el entorno HPC: /apps, /scratch, /home y /opt/spack, todos con permisos de acceso estándar (0755).

  Después, verifica si ya existe un directorio local en /opt/spack. En caso afirmativo, lo renombra como copia de seguridad (/opt/spack.local.bak) para evitar conflictos con el montaje remoto.

  Luego, monta el directorio /opt/spack desde el servidor NFS, usando la ruta remota {{ nfs_server_host }}:/apps/spack. El montaje se realiza con opciones optimizadas para rendimiento y estabilidad (rw,sync,hard,intr,rsize=1048576,wsize=1048576,noatime,nodiratime).

  Por último, crea un archivo /etc/profile.d/spack.sh que define el entorno de Spack para todos los usuarios. Este script establece SPACK_ROOT y carga el entorno de Spack al iniciar sesión, permitiendo que el sistema utilice la instalación centralizada del software.

0.5.2.3 Autenticación

Rol openldap El rol OpenLDAP que implementa autenticación centralizada se puede encontrar en este enlace:

https://github.com/psantana5/ansible-hpc/tree/main/roles/openldap

El playbook que sirve como entry point para instalar y configurar el servicio OpenLDAP, puede ser encontrado en este enlace:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/auth/openldap.yml

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **handlers:** recarga el servicio de daemons de systemd, debido a que crea su propio service file, reinicia el servicio *slapd* (demonio de ldap). También, a causa de unos problemas con Rocky Linux y OpenLDAP, debe añadir un contexto especial para la ruta de archivos de datos de OpenLDAP y para la ruta de archivos de configuración.
- **tasks:** Este playbook de Ansible tiene como objetivo desplegar y configurar un servidor OpenLDAP desde cero. Lo primero que hace es asegurarse de que la variable ldap_admin_password esté definida, ya que es necesaria para establecer la contraseña del administrador (cn=admin) en el DIT base.

  Luego instala los paquetes esenciales para el funcionamiento de OpenLDAP: tanto el servidor (openldap-servers) como los clientes y bibliotecas necesarias (openldap-clients, python3-ldap, etc.). A continuación, activa e inicia el servicio slapd, que es el demonio principal del servidor LDAP.

  Una vez el servicio está en marcha, crea y ajusta permisos en los directorios clave: el de configuración (/etc/openldap/slapd.d), el de datos (/var/lib/ldap) y los relacionados con certificados TLS. También aplica los contextos SELinux adecuados usando semanage, para evitar conflictos de seguridad al acceder a estos recursos.

  Después, genera el archivo /etc/sysconfig/slapd a partir de una plantilla. Este archivo permite definir cómo se ejecuta el servicio (por ejemplo, si usa IPv6, o qué protocolos acepta). Si se producen cambios en esta plantilla, se reinicia automáticamente el servicio LDAP para aplicarlos.

  Finalmente, el playbook genera los certificados TLS.

- **templates:** Este conjunto de plantillas Jinja2 forma parte del rol Ansible para desplegar y configurar un servidor OpenLDAP de manera automatizada. Cada plantilla genera un archivo de configuración o datos clave para que el servidor LDAP funcione correctamente.

  Primero, las plantillas .ldif.j2 como initial.ldif.j2 y users_groups.ldif.j2 definen los datos iniciales del directorio LDAP: entradas base, usuarios y grupos que se importan al arrancar el servidor. Estas entradas se configuran dinámicamente usando variables para adaptarse al entorno donde se despliega.

  Las plantillas relacionadas con TLS, como ldap-tls.ldif.j2, generan la configuración necesaria para asegurar las conexiones LDAP mediante cifrado, algo imprescindible para proteger las credenciales y datos sensibles durante la comunicación.

  root_dn.ldif.j2 establece el Distinguished Name (DN) del administrador LDAP, que es la cuenta principal con permisos para gestionar todo el directorio.

  Por otro lado, slapd.conf.j2 y slapd.service.j2 se encargan de configurar el demonio LDAP (slapd). El primero define parámetros de funcionamiento del servidor LDAP, mientras que el segundo genera el archivo de servicio para systemd, garantizando que el servicio se inicie y gestione correctamente.

  Estas plantillas se procesan durante la ejecución del playbook, aplicando valores concretos para cada despliegue, lo que asegura que el servidor LDAP se configure de forma segura, flexible y reproducible en cualquier entorno.

0.5.2.4 Monitoreo

Rol prometheus

El rol Prometheus que implementa la recopilación de métricas se puede encontrar aquí:

https://github.com/psantana5/ansible-hpc/tree/main/roles/prometheus

El playbook que sirve como entry point para instalar y configurar Prometheus puede ser encontrado en este enlace:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/monitoring/monitoring.yml

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **handlers:** interacciones con los servicios de systemd para reiniciar el demonio de Prometheus, únicamente.
- **tasks:** en primer lugar, el playbook descarga la versión 2.45.0 desde el repositorio de GitHub de Prometheus, extrae el contenido del archivo tar, copia los binarios de Prometheus dentro de los binarios locales para aplicaciones de usuario, copia la configuración de Prometheus (prometheus.yml), crea el servicio de systemd y arranca el demonio.
- **templates:** Incluye los Jinja templates para el servicio de Prometheus y el archivo de configuración de Prometheus, este último con sintaxis específica para añadir todos los hosts con las indentaciones correctas dentro del archivo.

Rol Grafana

El rol Grafana que implementa la visualización de métricas se puede encontrar aquí:

https://github.com/psantana5/ansible-hpc/tree/main/roles/grafana

El playbook que sirve como entry point para instalar y configurar Grafana puede ser encontrado aquí:

https://github.com/psantana5/ansible-hpc/blob/main/playbooks/monitoring/monitoring.yml

Para este rol, hágase notar que tenemos dos métodos de implementación, uno vía el rol de monitoreo usando Docker cómo método de instalación (recomendado) o vía el rol de "Grafana", usando RPMs. Para un approach más clásico, documentamos el rol en específico para su despliegue con RPMs.

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **handlers:** reinicio del demonio del servidor de Grafana.
- **tasks:** añade el repositorio de Grafana, instala Grafana, configura la configuración por defecto de Grafana desde un template en Jinja para poder cargar todos los dashboards necesarios sin necesidad de adicionarlos manualmente, copia los dashboards de Grafana a una ubicación accesible para el demonio y por último reinicia el servicio para aplicar los cambios.
- **templates:** Contiene el archivo .ini para la configuración inicial de Grafana. Nótese que este archivo guarda las credenciales para la interfaz de Grafana, con lo que debe cambiarlas una vez se inicie sesión.

0.5.2.5 Gestión de Software

Rol Spack.

El rol Spack que implementa la gestión de software científico se puede encontrar en este enlace:

https://github.com/psantana5/ansible-hpc/tree/main/roles/spack

El playbook que sirve de entry point para instalar y configurar Spack puede ser encontrado aquí:

https://github.com/psantana5/ansible-hpc/blob/main/spack.yml

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

meta: Para este rol en específico, crearemos un nuevo plugin de Ansible que instale Spack. Con esta acción, contribuimos con la comunidad al facilitar la instalación de Spack vía un plugin específico en lugar de un playbook aislado.

tasks: Este playbook de Ansible tiene como objetivo automatizar la instalación de Spack, una herramienta de gestión de entornos y compilación de software científico, en un entorno Linux. La instalación contempla tanto dependencias de sistema como configuraciones personalizadas en función de la arquitectura del procesador.

En primer lugar, se asegura la presencia del paquete git, necesario para clonar el repositorio de Spack. A continuación, se instalan todas las dependencias requeridas para compilar software con Spack, así como utilidades para detectar las capacidades del procesador, como hwloc, util-linux y herramientas estándar de compilación (gcc, make, etc.).

Seguidamente, se comprueba si el directorio de instalación de Spack (/opt/spack) está montado desde un recurso NFS. Si es así, se omiten las tareas de creación de directorios y clonación del repositorio, asumiendo que el contenido ya está disponible. En caso contrario, se crean los directorios necesarios para la instalación, la caché y el stage de compilación, y se clona el repositorio oficial de Spack desde Git.

Después, se configura un entorno personalizado de Spack para cada usuario del sistema que lo requiera (en este caso, psantana). Esto incluye la creación del directorio .spack en el home del usuario y la copia de una configuración básica (config.yaml) mediante plantillas Jinja2.

El playbook también genera un script de inicialización (/etc/profile.d/spack.sh) que exporta las variables de entorno necesarias para usar Spack de forma global en el sistema.

A continuación, se ejecuta un script en Bash para detectar detalles de la arquitectura del procesador (modelo, flags y microarquitectura). Con esta información, se determinan las opciones óptimas de compilación (-march=native, -O3, -mavx2, etc.) y se almacenan en archivos temporales. Estos datos se utilizan para establecer variables de hecho (set_fact) en Ansible, permitiendo así su reutilización posterior en configuraciones específicas de Spack.

Posteriormente, se crean los ficheros de configuración de Spack (config.yaml, packages.yaml y compilers.yaml) adaptados a la arquitectura detectada, usando plantillas que aplican los flags óptimos de compilación y seleccionan objetivos de arquitectura adecuados (como zen2 o cascadelake, por ejemplo).

Finalmente, se genera un documento de verificación (ARCHITECTURE_CHECKS.md) en el que se explica cómo validar que Spack ha sido correctamente configurado para aplicar optimizaciones específicas de arquitectura. Este documento incluye comandos para consultar la arquitectura detectada, los compiladores configurados, los flags utilizados y cómo comprobar la especificación de compilación de un paquete determinado.

templates: Los archivos de la carpeta templates definen configuraciones personalizadas de Spack usando Jinja2, permitiendo que Ansible genere ficheros adaptados al entorno de cada nodo.

config.yaml.j2: Establece rutas de instalación, compilación y módulos de entorno para Spack, usando variables como spack_install_dir y spack_stage_dir.

packages.yaml.j2: Configura el uso de bibliotecas del sistema y define flags de optimización específicos de CPU, gracias a variables como cpu_target y opt_flags.

compilers.yaml.j2: Registra compiladores disponibles en el sistema con sus rutas y arquitectura objetivo, adaptándose a la máquina mediante variables como gcc_version.

spack.sh.j2: Script que carga Spack automáticamente al iniciar sesión. Se instala en /etc/profile.d/ para que esté disponible globalmente.

ARCHITECTURE_CHECKS.md.j2: Documento informativo que resume comandos de verificación de arquitectura y configuración de Spack, generado dinámicamente.

Rol container_apps

El rol de container_apps que se encarga de gestionar las aplicaciones contenerizadas se puede encontrar aquí:

https://github.com/psantana5/ansible-hpc/blob/main/roles/container_apps/

Según podemos observar en este rol en función de las carpetas que existen, podemos ver que se desempeñan estas funciones:

- **tasks**: Este playbook de Ansible tiene como objetivo automatizar la instalación y configuración de Singularity/Apptainer, un sistema de contenedores específico para entornos HPC. La instalación contempla tanto la configuración del sistema como la preparación de contenedores científicos comunes.

  En primer lugar, se instala el paquete singularity-ce, que es la implementación de Singularity/Apptainer para entornos HPC. A continuación, se crea un directorio centralizado en /opt/containers con los permisos adecuados (0755) para almacenar las imágenes de contenedores.

Seguidamente, se procede a la descarga de contenedores científicos predefinidos desde Docker Hub, incluyendo:

TensorFlow con soporte GPU (tensorflow/tensorflow:latest-gpu)

PyTorch (pytorch/pytorch:latest)

Jupyter Notebook con entorno de ciencia de datos (jupyter/datascience-notebook:latest)

Estos contenedores se almacenan en formato .sif (Singularity Image Format) en el directorio /opt/containers.

Finalmente, se generan scripts de sumisión SLURM para cada contenedor, que se instalan en /usr/local/bin con permisos de ejecución. Estos scripts permiten a los usuarios ejecutar trabajos en los contenedores a través del sistema de colas SLURM sin necesidad de conocer los detalles de configuración de Singularity/Apptainer.

- **templates**: Los archivos de la carpeta templates definen los scripts de sumisión SLURM para cada contenedor usando Jinja2, permitiendo que Ansible genere scripts adaptados al entorno del cluster:

  tensorflow_job.sh.j2: Script para ejecutar trabajos de TensorFlow en el cluster

  pytorch_job.sh.j2: Script para ejecutar trabajos de PyTorch en el cluster

  jupyter_job.sh.j2: Script para ejecutar instancias de Jupyter Notebook en el cluster

  Estos scripts se instalan en /usr/local/bin con permisos de ejecución (0755) y proporcionan una interfaz sencilla para que los usuarios puedan ejecutar sus trabajos en los contenedores a través de SLURM.

  handlers: No se implementa ningún handler en este rol ya que Singularity/Apptainer no utiliza un demonio que requiera reinicio.