



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Hacia un Centro de Datos Distribuido:
Redefinición de la infraestructura
organizacional mediante tecnologías open
source.**

TESIS

Que para obtener el título de
Ingeniero Eléctrico Electrónico

P R E S E N T A

Juan Carlos Frausto Martínez

DIRECTORA DE TESIS

M.I. Abigail Serralde Ruiz



Ciudad Universitaria, Cd. Mx., 2026



**PROTESTA UNIVERSITARIA DE INTEGRIDAD Y
HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)**



De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado HACIA UN CENTRO DE DATOS DISTRIBUIDOS: REDEFINICION DE LA INFRAESTRUCTURA ORGANIZACIONAL NEDIANTE TECNOLOGIAS OPEN SOURCE que presenté para obtener el título de INGENIERO ELÉCTRICO ELECTRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.



JUAN CARLOS FRAUSTO MARTINEZ
Número de cuenta: 315173165

Dedicatoria

A Dios, por el vasto universo; a mi madre, por su fuerza; a mi padre, por su sabiduría; y a quienes contribuyeron a forjar mi ser.

Esta dedicatoria se dirige a quienes han acompañado una vida marcada por el afecto, la confianza y el aprendizaje compartido. Su presencia ha dado sentido a cada etapa de este trayecto. Que cada persona encuentre plenitud en la felicidad y en la tristeza, reconociendo el valor de ambas experiencias; que conserve gratitud por lo recibido y determinación para alcanzar sus anhelos. El aprecio hacia quienes acompañan este camino excede lo que puede expresarse en estas líneas; permanece como un vínculo leal y perdurable. La siguiente reflexión sintetiza ese afecto:

$$P_n := \prod_{k=1}^n (1 + e_k) \geq 1 + \sum_{k=1}^n e_k, \quad e_k \geq 0.$$

En esta ecuación¹ cada e_k simboliza un pequeño esfuerzo, un instante de amor, una chispa de esperanza. El producto refleja que la grandeza no nace de un solo gesto heroico, sino del eco de muchos actos sencillos que, día tras día, se multiplican silenciosamente hasta formar algo inmenso. Así como $1^n = 1$ permanece inmutable, incluso la más leve mejora, $(1 + \varepsilon)^n$, crece sin límites cuando se cultiva con constancia y fe. Que esta ecuación te recuerde que todo lo perdurable se construye poco a poco, que las pequeñas acciones son las que mueven el mundo, y que el amor, como el conocimiento, se expande cuando se comparte.

Con amor y gratitud,

Juan Carlos Frausto Martínez

¹Se trata de la desigualdad de Bernoulli

Agradecimientos

Esta obra se desarrolló en un entorno que inspira, forma y transforma. La Universidad Nacional Autónoma de México constituyó el punto de partida: un espacio que fomenta el cuestionamiento, la creación y una mirada crítica del mundo. Se reconoce profundamente a esta institución y al tiempo de formación brindado. La Facultad de Ingeniería se convirtió en más que una escuela; fue un hogar intelectual que permitió descubrir una vocación orientada a resolver y construir, así como el valor del esfuerzo compartido. Dentro de sus muros, la Coordinación de Planeación y Desarrollo brindó guía, confianza y las herramientas necesarias para continuar creciendo. Este camino fue posible gracias a las personas que ofrecieron su apoyo cuando el horizonte parecía distante. Cada palabra de aliento, cada gesto de apoyo y cada acto de amistad contribuyeron a formar a la persona que culmina este trabajo. A familiares, amistades, profesores y colegas que acompañaron este trayecto se les reconoce no solo la culminación de una carrera, sino también la certeza de que el conocimiento adquiere sentido cuando se comparte con amor y gratitud.

Índice General

Dedicatoria	1
Agradecimientos	2
Introducción	6
Hipótesis	10
Objetivo General	11
Objetivos Específicos	11
Capítulo I – Fundamentos teóricos	14
1.1 Recursos fundamentales del cómputo	15
1.2 Concepto y evolución de los centros de datos	15
1.3 Modelos de servicio y despliegue en la nube	21
1.4 Infraestructura programable	23
1.5 Contenedorización y orquestación	26
1.6 Redes	29
1.7 Seguridad	30
Capítulo II – Principios de diseño arquitectónico y selección del stack	34
2.1 Análisis de caso: Facultad de Ingeniería	35
2.2 Principios de diseño	38
2.3 Protoarquitectura funcional	43
2.4 Selección del stack	48
2.5 Evaluación de interoperabilidad y coherencia del stack	51
2.6 Justificación técnica de la selección final	53
Capítulo III – Arquitectura del datacenter	58
3.1 Marco de descripción arquitectónica	59
3.2 Elementos transversales	60
3.3 Nivel de Contexto	65
3.4 Nivel de Contenedores	73
3.5 Nivel de Componentes	79
3.6 Síntesis de la arquitectura propuesta	85
Capítulo IV – Estrategia de transición hacia el ecosistema distribuido	92
4.1 Roadmap de implementación por fases	93
4.2 Fase 0 – Constitución del ente gestor y diagnóstico de madurez	94
4.3 Fase 1 – Prerrequisitos de conectividad e identidad	96

4.4 Fase 2 — Despliegue de la capa IaaS	97
4.5 Fase 3 — Activación de la capa CaaS	103
4.6 Fase 4 — Consolidación y madurez operativa	104
4.7 Gestión del cambio organizacional	107
Capítulo V — Conclusiones	112
5.1 Evaluación de la propuesta frente al problema planteado	112
5.2 Verificación de la hipótesis	113
5.3 Implicaciones técnicas y organizacionales	115
5.4 Limitaciones del estudio	116
5.5 Líneas de investigación futura	118
Índice de Figuras	121
Índice de Tablas	123
Referencias	126
Bibliografía	126

Introducción

Existen numerosas definiciones de «tecnología»; para este trabajo resulta especialmente pertinente el siguiente enfoque:

La tecnología es la aplicación práctica del conocimiento para que algo totalmente nuevo pueda hacerse, o para que algo pueda hacerse de una manera completamente nueva. [1]

La constitución y adopción de tecnologías siempre ha obedecido al contexto social y a las necesidades concretas de los grupos que las emplean. Pinch y Bijker [2] sostienen que son precisamente esas necesidades y no la superioridad técnica de un artefacto las que determinan qué tecnologías prosperan y cuáles caen en desuso. Lo anterior se observa con claridad en las Revoluciones Industriales registradas a lo largo de la historia.

El término **Revoluciones Industriales** es utilizado por historiadores y analistas para describir al menos cinco grandes procesos de transformación impulsados por la tecnología. Estas revoluciones han remodelado fundamentalmente los sistemas productivos, tecnológicos y sociales, alterando la organización del trabajo, la competitividad empresarial y la estructura socioeconómica global.

Las tres primeras revoluciones industriales se enfocaron estrictamente en escalar la productividad mediante el vapor, la electrificación y la digitalización, los paradigmas recientes han redefinido el objetivo industrial. La Industria 4.0 introdujo la hiperconectividad y la autonomía de las fábricas inteligentes, sentando las bases tecnológicas para la Industria 5.0. Esta última representa un cambio filosófico que prioriza la sostenibilidad, la resiliencia y la colaboración directa entre humanos y robots (cobots), integrando por primera vez dimensiones éticas y ambientales en el núcleo del diseño de los sistemas. En la Tabla 1 se muestra un resumen de las características clave de cada Revolución Industrial. [3], [4]

TABLA 1

EVOLUCIÓN Y CARACTERÍSTICAS DE LAS REVOLUCIONES INDUSTRIALES.

Industria	Época	Tecnologías clave	Características	Retos
1.0	Finales s. XVIII	Energía hidráulica, vapor, mecánica	Transición a sistemas mecánicos	Baja flexibilidad, jornadas extensas
2.0	Finales s. XIX - Inicios s. XX	Electricidad, líneas de ensamblaje	Producción en masa, modelo lineal	Poca adaptabilidad y personalización
3.0	1970s	Electrónica, TI, microprocesadores, PLC	Automatización parcial, software de gestión	Falta de interoperabilidad
4.0	Siglo XXI	IoT, CPS, Nube, IA	Fábricas inteligentes, hiperconectividad	Ciberseguridad, desplazamiento laboral
5.0	Actualidad	Cobots, Digital Twins, IA avanzada	Enfoque humanocéntrico, sostenible y resiliente	Brecha de habilidades, riesgos éticos

Para este trabajo es de particular interés la Industria 4.0, la cual puede comprenderse como un ecosistema tecnológico estructurado en niveles jerárquicos. En el nivel fundamental se encuentran las tecnologías habilitadoras, de las cuales destacan la conectividad (redes de alta disponibilidad, protocolos de comunicación y 5G) y las infraestructuras de cómputo y almacenamiento (centros de datos, computación en la nube y edge computing), que permiten la transmisión y gestión de grandes volúmenes de información. En un segundo nivel se ubican las tecnologías de gestión y explotación de datos, como el Internet de las Cosas Industrial, el Big Data y la analítica avanzada, responsables de capturar, estructurar y procesar la información generada por los sistemas productivos. Finalmente, en el nivel superior se encuentran las tecnologías de aplicación y autonomía operativa, tales como la inteligencia artificial², los sistemas ciberfísicos, la robótica avanzada y la manufactura aditiva, que transforman los datos en decisiones, optimización de procesos y producción inteligente. Esta jerarquía evidencia que las tecnologías de automatización dependen de las capas inferiores de conectividad e infraestructura digital. A partir de lo anterior se vuelve evidente que una correcta implementación de la Industria 4.0 requiere de planes de transformación digital³. [6], [7], [8]

²La inteligencia artificial, por sí misma, puede considerarse una nueva revolución industrial

³De acuerdo con Amazon Web Services, la transformación digital es el proceso mediante el cual una organización integra tecnología digital a todas las áreas empresariales y la hace crecer. [5]

Tres niveles interdependientes · las tecnologías visibles de automatización dependen estructuralmente de las capas inferiores de conectividad e infraestructura digital

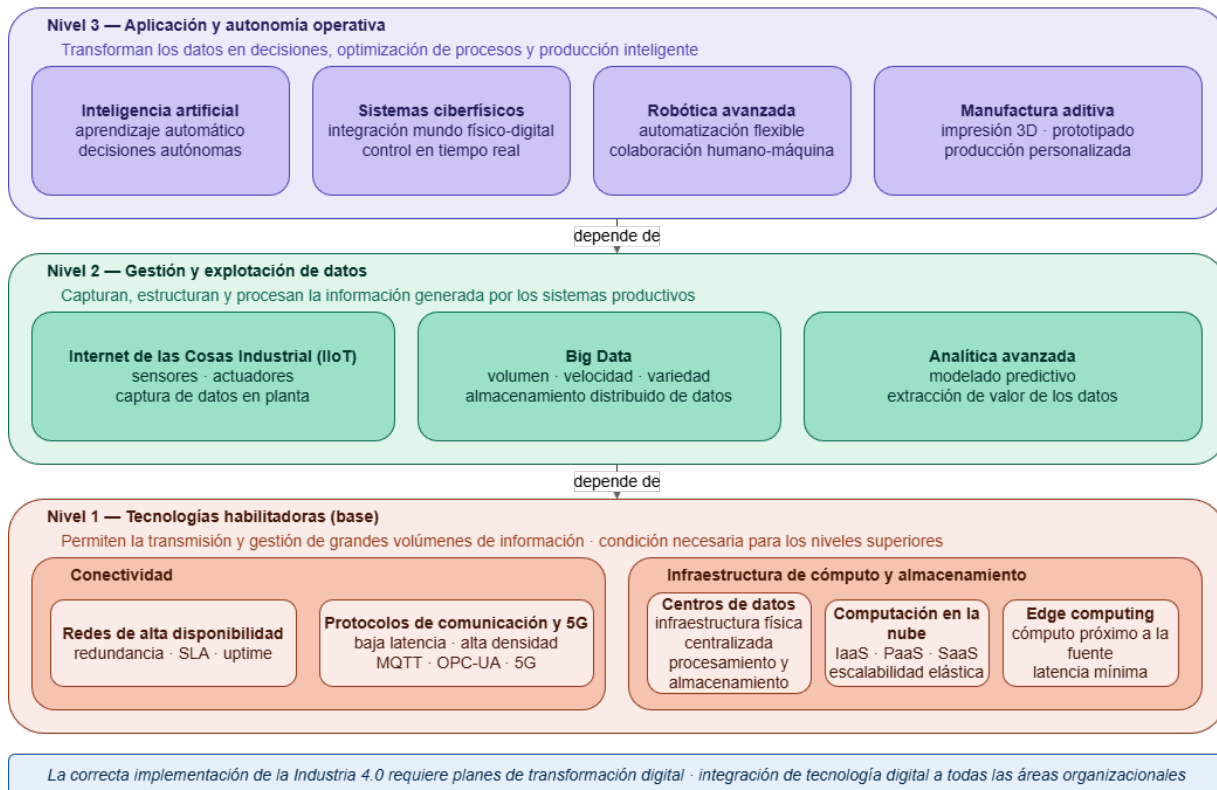


Fig. 1. Jerarquía de las tecnologías en la industria 4.0

Antes de definir el problema central de este texto es indispensable analizar el estado actual de la Industria 4.0 en México. El Informe de Madurez Digital 2025 (IMD), elaborado por American Chamber México, EY, KIO Networks y Needed Education, proporciona una medida cuantitativa de cómo las empresas mexicanas están avanzando en su transformación digital y revela que solo el 41.7 % de las empresas participantes ha alcanzado un nivel de transformación digital, porcentaje que todavía está por debajo del umbral ideal considerado competitivo ($\approx 70\%$) [9].

De acuerdo con este mismo informe, aunque existen avances, la mayoría de las organizaciones y empresas mexicanas no ha completado todavía una adopción madura de tecnologías digitales propias de la Industria 4.0. Apenas el 1 % de las empresas ha alcanzado madurez en inteligencia artificial, tecnología que se puede considerar como indicador de una integración plena de la Industria 4.0 en procesos productivos y de negocio.

Este contraste se acentúa cuando se analiza el tamaño de las empresas: las grandes compañías, especialmente aquellas con inversión extranjera o que forman parte de cadenas globales de valor, tienden a mostrar mayor nivel de digitalización, mientras que las pequeñas y medianas

empresas (PyMES) enfrentan barreras estructurales importantes como: limitaciones de capital, falta de talento especializado y ausencia de estrategias formales de transformación digital que restringen su avance [10].

El estado actual de la conectividad en México refleja un avance significativo en el despliegue de infraestructura digital, particularmente en redes móviles de quinta generación (5G) y fibra óptica. Diversos análisis de mercado señalan que la cobertura 5G ha crecido de manera progresiva en los principales corredores urbanos e industriales del país, aunque todavía existen brechas importantes en zonas rurales y semiurbanas [11]. Paralelamente, la expansión de redes de fibra óptica continúa siendo un factor estratégico para habilitar servicios de alta velocidad, baja latencia y mayor capacidad de transmisión de datos, condición indispensable para soportar aplicaciones industriales, servicios en la nube y entornos de hiperconectividad [12].

En cuanto a infraestructura de centros de datos, México se ha consolidado como uno de los mercados emergentes más relevantes en América Latina, con polos estratégicos como Querétaro y Guadalajara. Reportes especializados proyectan un crecimiento sostenido del mercado, impulsado por la demanda de servicios cloud, edge computing e inteligencia artificial, así como por la llegada de operadores de hiperescala⁴ [13]. Esta expansión está estrechamente vinculada al fortalecimiento de la conectividad nacional, ya que el desarrollo de redes de alta capacidad y baja latencia constituye la base técnica sobre la cual operan los ecosistemas digitales contemporáneos.

Estos datos indican que la Industria 4.0 en México se encuentra en una etapa de transición, con avances significativos en sectores estratégicos, pero con una adopción heterogénea y desigual. El liderazgo tecnológico está concentrado en grandes empresas, mientras que una proporción considerable de PyMES continúa rezagada, lo que amplía la brecha digital industrial y plantea la necesidad de políticas públicas, incentivos y estrategias que permitan una integración más equitativa y rápida de la Industria 4.0.

Si bien México experimenta una expansión en la instalación de centros de datos operados por gigantes tecnológicos como Amazon, Google y Microsoft, esta oferta comercial presenta dos desafíos fundamentales para el sector público y educativo. En primer lugar, la adopción

⁴El termino hace referencia a los proveedores masivos de computo en la nube, en inglés el término es hyperscalers

de estas plataformas frecuentemente conlleva el riesgo de **vendor lock-in**⁵, comprometiendo la independencia tecnológica de las organizaciones. En segundo lugar, los modelos de nube pública no resuelven el problema de la infraestructura local preexistente: instituciones de gran escala, como las universidades públicas, poseen vastos recursos de cómputo adquiridos a lo largo de los años que operan en silos, subutilizados y fragmentados.

En este contexto, la independencia tecnológica no solo implica evitar el vendor lock-in, sino también desarrollar la capacidad de gobernar, unificar y orquestar la infraestructura heterogénea propia. Dicha independencia es un pilar de la soberanía digital y se compone de [15], [16]:

- **Autonomía de gestión:** la capacidad de las organizaciones para elegir, administrar y proteger su infraestructura digital sin depender de restricciones propietarias o de jurisdicciones extranjeras.
- **Control total en entornos críticos:** permite ejecutar cargas de trabajo en nubes privadas, locales o híbridas, manteniendo el cumplimiento de normativas locales y la residencia de datos.
- **Continuidad y Seguridad:** asegura el funcionamiento de servicios esenciales incluso en entornos desconectados o aislados, mitigando riesgos geopolíticos y problemas de vigilancia externa.

Para transitar hacia una verdadera gobernanza de datos y servicios, es imperativo superar el aislamiento de los sistemas tradicionales. A partir de lo anterior se plantea la siguiente hipótesis de trabajo:

Hipótesis

Es posible diseñar una arquitectura de referencia para la constitución de un centro de datos distribuido, construida exclusivamente con tecnologías de código abierto, cuyos componentes se integren mediante interfaces estandarizadas y no mediante desarrollo a la medida, y que soporte cargas de cómputo de propósito general y especializado sin requerir rediseño estructural ante la incorporación de nuevos recursos.

⁵Vendor lock-in es una práctica comercial en la que un cliente se ve prácticamente obligado a seguir usando el producto o servicio de un proveedor, independientemente de su calidad, porque cambiar a otro no es práctico debido al alto coste o esfuerzo que implicaría. [14]

De lo anterior se establecen los siguientes objetivos:

Objetivo General

Proponer una arquitectura de referencia basada en un stack tecnológico de código abierto que permita a una organización la implementación de un centro de datos distribuido, elástico y soberano.

Los objetivos específicos por lo tanto son:

Objetivos Específicos

Fundamentación de la infraestructura base: Seleccionar y justificar un *stack* tecnológico *Open Source* para la implementación del centro de datos mediante Infraestructura Definida por Software e Infraestructura como Código, que permita la convergencia de recursos de cómputo heterogéneos, almacenamiento y redes en un grupo unificado.

Estandarización del despliegue mediante contenedores: La infraestructura base debe permitir la implementación de microservicios basados en contenedorización para resolver la fragmentación del software institucional, transformando las aplicaciones tradicionales en unidades portables y escalables que consuman ágilmente los recursos del clúster.

Metodología de transición y adopción: Desarrollar un plan de ruta (*roadmap*) para la migración de sistemas aislados hacia el nuevo ecosistema, definiendo estrategias para la integridad de los datos, la alta disponibilidad y la continuidad operativa.

Para considerar que la hipótesis se sustenta se deben cumplir las siguientes condiciones:

- **Suficiencia del stack:** existe al menos un componente de código abierto, con licencia libre y gobernanza documentada, para cada responsabilidad funcional necesaria para el correcto funcionamiento del centro de datos.
- **Coherencia de la integración:** las conexiones entre dichos componentes se establecen mediante estándares abiertos y no mediante adaptadores propietarios, de forma que la sustitución de un componente no obligue a rediseñar los demás.

- **Generalidad y extensibilidad de la arquitectura:** la arquitectura resultante admite cargas de cómputo de propósito general y especializado bajo una misma capa de control, y la incorporación de nuevos recursos de cualquiera de esos tipos no obliga a rediseñar su estructura.

Es importante delimitar el alcance de esta verificación. La comprobación que ofrece este trabajo es de naturaleza documental y de diseño, no experimental: la hipótesis pretende sustentar la **viabilidad arquitectónica** de la propuesta, no su desempeño bajo condiciones reales de operación, para esto último es indispensable la prueba de concepto mediante un prototipo funcional que se identifica como línea de trabajo futura y hasta entonces toda afirmación sobre el comportamiento del sistema en producción permanece fuera de lo que este trabajo puede sostener.

CAPÍTULO

I

Fundamentos teóricos



RESUMEN

Este capítulo establece los fundamentos teóricos necesarios para comprender, evaluar y justificar la arquitectura propuesta. Se explora desde la conceptualización de los recursos clásicos del cómputo hasta la evolución hacia infraestructuras hiperconvergentes (HCI) y modelos de servicio en la nube. Además, se analizan los paradigmas que habilitan los centros de datos modernos: la infraestructura definida por software (SDI), la orquestación de contenedores, las topologías de red y los mecanismos de identidad y seguridad.

CONCEPTOS CLAVE

Sistemas Distribuidos

Arquitecturas que dispersan funciones entre múltiples nodos para garantizar escalabilidad y tolerancia a fallos.

Nube Comunitaria

Modelo de despliegue en el que la infraestructura es aprovisionada para el uso exclusivo de una comunidad con intereses, misiones y normativas comunes.

Infraestructura Hiperconvergente (HCI)

Marco de TI que unifica recursos de cómputo, almacenamiento y redes en una única plataforma definida por software operando sobre hardware de propósito general.

Infraestructura Programable (SDI e IaC)

Paradigma que abstrae los recursos físicos exponiéndolos mediante APIs (SDI), permitiendo su aprovisionamiento automatizado, auditable y reproducible mediante código (IaC).

1.1 Recursos fundamentales del cómputo

Originalmente, el término cómputo hacía referencia al acto de realizar cálculos matemáticos. Sin embargo, gracias al desarrollo tecnológico su significado se ha ampliado considerablemente. Hoy en día, la palabra se utiliza como sinónimo tanto de las máquinas que realizan dichos cálculos (las computadoras) como de sus recursos de cómputo [17], [18]. Estos recursos abarcan los componentes de hardware y software esenciales que permiten la ejecución de tareas en un sistema informático. Los componentes fundamentales son los siguientes:

- Unidad Central de Procesamiento (CPU)
- Memoria de acceso aleatorio (RAM)
- Almacenamiento persistente
- Recursos de red
- Unidad de Procesamiento Gráfico (GPU)
- Sistema Operativo (OS)

Resulta conveniente agrupar los recursos de cómputo en categorías funcionales que abstraen los componentes individuales y los organizan de acuerdo con la función principal que desempeñan:

- **Procesamiento:** comprende los componentes responsables de la ejecución de instrucciones y el procesamiento de datos: CPU, GPU y RAM.
- **Almacenamiento:** engloba los dispositivos y sistemas destinados a la persistencia de datos de manera local o distribuida.
- **Red:** incluye los recursos y mecanismos que permiten la interconexión y comunicación entre sistemas.

El sistema operativo se considerará para efectos de este trabajo como un medio base para el aprovechamiento de los recursos de cómputo.

1.2 Concepto y evolución de los centros de datos

1.2.1 Definición de centro de datos

Un centro de datos es una instalación física diseñada específicamente para albergar la infraestructura de cómputo que es necesaria para construir, ejecutar y entregar aplicaciones y servicios, así como para gestionar los datos asociados. Desde una perspectiva técnica, su diseño se fundamenta en una red robusta de recursos de cómputo y almacenamiento. Esta infraestructura incluye:

servidores, switches, routers, firewalls, sistemas de almacenamiento avanzados y controladores de entrega de aplicaciones. [19]

1.2.2 Centro de datos distribuido

Un **Centro de Datos Distribuido** es la evolución de un centro de datos. En lugar de centralizar todo en una única ubicación física, un centro de datos distribuido implementa arquitectura de computación en red que dispersa las funciones de cómputo en nodos física y geográficamente dispersos [20]. Esta dispersión no solo aumenta drásticamente la capacidad de procesamiento total, sino que, garantiza la resiliencia operativa y permite acercar el cómputo a las fuentes de datos, minimizando así la latencia cuando es necesario. En este modelo, la capa de abstracción de recursos permite que el usuario final simplemente acceda a la capacidad de cómputo que necesita, sin preocuparse por la ubicación física de cada componente. En la Fig. 2 se observa como un centro de datos abstrae el acceso a los recursos de computo que ya se han descrito.

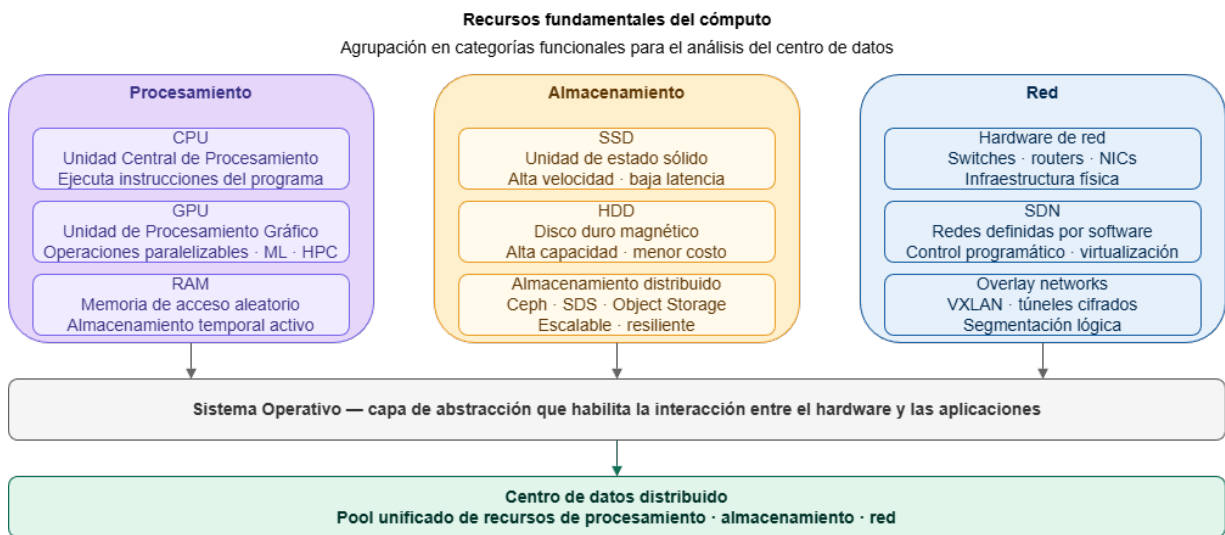


Fig. 2. Abstracción de los recursos de cómputo en un centro de datos

Los sistemas distribuidos son naturalmente más complejos que los sistemas monolíticos. Para que el esfuerzo de su implementación esté justificado, deben satisfacer las siguientes propiedades fundamentales identificadas por Van Steen y Tanenbaum [20]:

1.2.2.1 Compartición de recursos

⁶El termino *pool* se usa en el argot computacional para hacer referencia a un grupo recursos de computo de uso compartido.

Esta característica permite a los usuarios y servicios acceder y compartir un mismo pool⁶ de recursos, abarcando desde capacidad de procesamiento hasta servicios de software completos [20]. La compartición de recursos le da al centro de datos una ventaja fundamental sobre recursos de computo dispersos, esta es, la economía de escala⁷.

1.2.2.2 Transparencia

Un sistema distribuido busca ocultar su complejidad interna al usuario final, haciendo que el acceso a recursos y servicios sea una tarea trivial evitando complejidad [20]. Los autores destacan que este objetivo se logra mediante el uso de middleware, una capa de software que gestiona automáticamente dicha complejidad. En la Tabla 2 se muestran las diferentes capas de transparencia identificadas por Van Steen y Tanenbaum.

TABLA 2

DIFERENTES FORMAS DE TRANSPARENCIA EN UN SISTEMA DISTRIBUIDO [20].

Transparencia	Descripción
Acceso <i>Access</i>	Ocultar diferencias en la representación de datos y cómo se accede a un objeto
Ubicación <i>Location</i>	Ocultar dónde se encuentra ubicado un objeto
Reubicación <i>Relocation</i>	Ocultar que un objeto puede moverse a otra ubicación mientras está en uso
Migración <i>Migration</i>	Ocultar que un objeto puede moverse a otra ubicación
Replicación <i>Replication</i>	Ocultar que un objeto está replicado
Concurrencia <i>Concurrency</i>	Ocultar que un objeto puede ser compartido por varios usuarios independientes
Fallo <i>Failure</i>	Ocultar el fallo y la recuperación de un objeto

1.2.2.3 Apertura

La apertura es la capacidad del sistema de ofrecer componentes que puedan ser fácilmente utilizados o integrados en otros sistemas [20]. Un sistema abierto cumple con tres propiedades:

- **Interoperabilidad:** caracteriza el grado en que dos implementaciones de diferentes fabricantes (o desarrolladores) pueden coexistir y trabajar juntas basándose en estándares comunes

⁷Las economías de escala ocurren cuando el costo de producir un bien o servicio disminuye conforme aumenta el volumen de producción [21].

- **Portabilidad:** describe en qué medida una aplicación puede ejecutarse sin modificación en distintos sistemas que implementen las mismas interfaces.
- **Extensibilidad:** esta característica permite añadir nuevos componentes o reemplazar los existentes sin afectar a los que permanecen en el sistema.

1.2.2.4 Confiabilidad

La confiabilidad se define como el grado en el que se puede confiar en que un sistema computacional operará según lo esperado [20]. Los componentes de la confiabilidad se listan en la Tabla 3.

TABLA 3

CARACTERÍSTICAS CLAVE DE LA CONFIABILIDAD EN SISTEMAS DISTRIBUIDOS [20].

Característica	Descripción
Disponibilidad <i>Availability</i>	Probabilidad de que el sistema opere correctamente en un instante de tiempo dado.
Fiabilidad <i>Reliability</i>	Capacidad del sistema de ejecutarse continuamente sin fallar durante un intervalo sostenido.
Seguridad <i>Safety</i>	Garantía de que una falla temporal no derive en consecuencias catastróficas.
Mantenibilidad <i>Maintainability</i>	Facilidad con la que un sistema fallido puede ser detectado y reparado.

1.2.2.5 Consenso distribuido y quórum

Una propiedad crítica de los sistemas distribuidos es la necesidad de mantener consistencia de estado entre múltiples nodos que comparten información o toma de decisiones. El **consenso distribuido** es el mecanismo mediante el cual un conjunto de nodos acuerda un valor o decisión única a pesar de que algunos nodos puedan fallar o quedar desconectados [20].

El concepto de **quórum** es central para implementar consenso: se define como el número mínimo de nodos que deben estar de acuerdo para que una decisión sea válida. En un sistema de n nodos, el quórum típico es $\lfloor \frac{n}{2} \rfloor + 1$, es decir, la mayoría simple. Los algoritmos de consenso de uso extendido en la industria, como Raft [22]⁸, se construyen sobre esta regla de mayoría: mientras la mayoría de los nodos permanezca disponible y comunicada, el sistema puede elegir un líder y confirmar escrituras de forma consistente.

1.2.2.6 Seguridad

⁸Raft es el algoritmo empleado por etcd, el almacén de estado de Kubernetes.

La seguridad es una propiedad indispensable para que un sistema distribuido sea confiable [20], un sistema es seguro si cumple con:

- **Confidencialidad e integridad:** garantizan la divulgación y el acceso autorizado a la información.
- **Autenticación:** permite al sistema verificar que efectivamente está interactuando con la entidad que dice ser
- **Autorización:** determina si una entidad autenticada cuenta con los derechos de acceso adecuados.

1.2.2.7 Escalabilidad

La escalabilidad se ha convertido en uno de los objetivos de diseño más relevantes en los sistemas distribuidos modernos. Esta característica se define como la capacidad de un sistema para crecer y adaptarse a la demanda sin degradar su funcionamiento [20]. En la Tabla 4 se muestran los tipos de escalabilidad esperados en sistemas distribuidos.

TABLA 4

TIPOS DE ESCALABILIDAD EN SISTEMAS DISTRIBUIDOS [20].

Dimensión	Descripción	Desafío principal
Tamaño <i>Size</i>	Capacidad de incorporar más usuarios y recursos sin pérdida notable de rendimiento.	Degradación del rendimiento
Geográfica <i>Geographical</i>	Capacidad de operar correctamente con usuarios y recursos físicamente distantes.	Retardos de comunicación
Administrativa <i>Administrative</i>	Capacidad de mantenerse gestionable abarcando múltiples organizaciones independientes.	Coordinación entre entidades autónomas

A continuación se muestra una figura que resume las características fundamentales que cumple todo sistema distribuido.

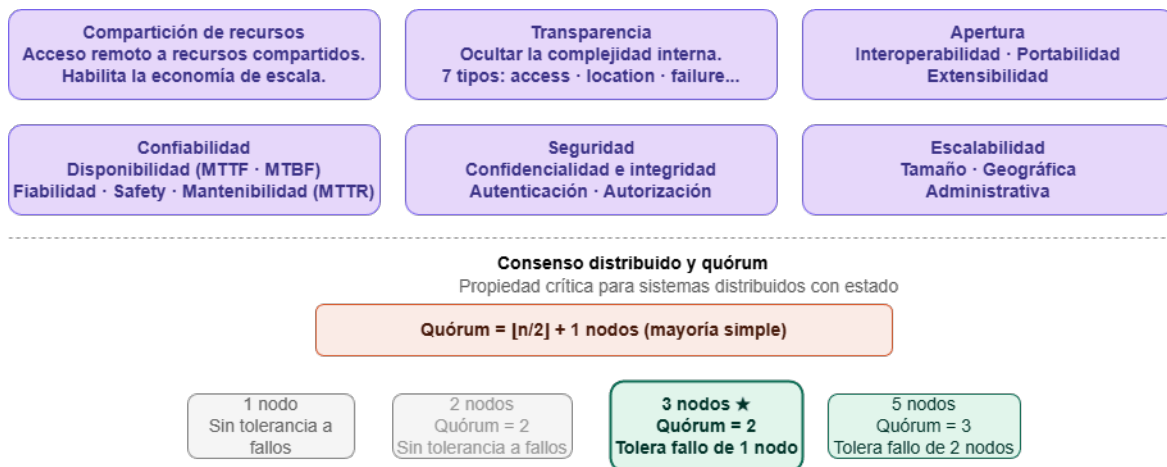


Fig. 3. Características fundamentales de los sistemas distribuidos.

1.2.3 Convergencia e hiperconvergencia

1.2.3.1 Infraestructura convergente (CI)

La Infraestructura Convergente es un paradigma de implementación que simplifica la gestión de centros de datos al integrar el cómputo, el almacenamiento y la red en un paquete único preconfigurado por el fabricante [23]. Esto elimina la complejidad de gestionar componentes fragmentados por separado, ofreciendo un proceso de despliegue y un soporte unificados [24].

1.2.3.2 Infraestructura hiperconvergente (HCI)

La hiperconvergencia se considera como la evolución lógica y más avanzada del CI. La HCI combina almacenamiento, cómputo y redes en un sistema único, con el objetivo de reducir la complejidad del centro de datos, aumentar la escalabilidad y reducir el costo total de propiedad (TCO) [25]. A diferencia de la CI, la HCI es fundamentalmente una infraestructura definida por software que aprovecha hardware de uso general y una plataforma de software centralizada para gestionar los recursos de forma dinámica [24], [26]. Para comprender mejor la diferencia entre estos conceptos se comparan de acuerdo a sus características en la Tabla 5.

TABLA 5

COMPARATIVA ENTRE INFRAESTRUCTURA CONVERGENTE E HIPERCONVERGENTE [23], [24], [26].

Característica	Convergente (CI)	Hiperconvergente (HCI)
Definición	Componentes de hardware preintegrados en un paquete optimizado por el fabricante.	Recursos gestionados por una plataforma de software sobre hardware de uso general.
Arquitectura	Modular; componentes gestionados de forma independiente.	Definida por software; todos los componentes integrados en una sola capa.
Escalabilidad	Horizontal por módulos; cada componente escala de forma separada.	Escalado por nodos: al agregar un nodo se incrementan simultáneamente cómputo, almacenamiento y red.
Flexibilidad	Limitada; depende de configuraciones de hardware específicas.	Alta; diseñada para asignación dinámica de recursos.
Costo	Inversión inicial mayor por dependencia de hardware especializado.	Menor costo inicial al apoyarse en hardware commodity.
Recuperación ante desastres	Soportada, pero requiere hardware y configuración adicional.	Integrada de forma nativa con redundancia y replicación automática.

1.3 Modelos de servicio y despliegue en la nube

1.3.1 Computación en la nube

El National Institute of Standards And Technology (NIST) define a la computación en la nube como un modelo que permite el acceso generalizado, conveniente y bajo demanda a un conjunto compartido de recursos de cómputo configurables que pueden aprovisionarse y liberarse rápidamente con un mínimo esfuerzo de gestión o de interacción con el proveedor [27]. El NIST identifica cinco características esenciales:

Autoservicio bajo demanda: el consumidor puede aprovisionar capacidades de cómputo de forma unilateral según las necesite, sin requerir interacción humana con el proveedor.

Acceso amplio a la red: las capacidades están disponibles en la red y se accede a ellas mediante mecanismos estándar desde plataformas heterogéneas de cliente⁹.

Agrupación de recursos (*resource pooling o pool*): los recursos del proveedor se agrupan para servir a múltiples consumidores mediante un modelo multitenant, con recursos asignados y reasignados dinámicamente según la demanda.

Elasticidad rápida: las capacidades pueden aprovisionarse y liberarse elásticamente, en algunos casos de forma automática, para escalar según la demanda.

⁹En este contexto, un cliente se refiere a una aplicación o programa que solicita servicios o información a otro sistema

Servicio medido: los sistemas en la nube controlan y optimizan automáticamente el uso de recursos mediante una capacidad de medición apropiada al tipo de servicio.

1.3.2 Modelos de servicio

El NIST define tres modelos de servicio que describen el nivel de abstracción que el proveedor ofrece al consumidor [27]:

Infraestructura como Servicio (IaaS): al consumidor se le proporciona la capacidad de aprovisionar procesamiento, almacenamiento, redes y otros recursos de cómputo fundamentales de cómputo. El consumidor no gestiona la infraestructura física subyacente, pero sí controla los sistemas operativos, el almacenamiento y las aplicaciones desplegadas.

Plataforma como Servicio (PaaS): este modelo permite desplegar aplicaciones sobre la infraestructura de la nube usando lenguajes de programación, bibliotecas y herramientas compatibles con el proveedor. El consumidor no gestiona la infraestructura subyacente, pero sí controla las aplicaciones desplegadas [28].

Software como Servicio (SaaS): bajo este modelo se le entrega al consumidor las aplicaciones del proveedor que se ejecutan sobre una infraestructura de nube. El consumidor no gestiona ni controla la infraestructura subyacente ni las aplicaciones individuales.

Existen otros modelos derivados de estos o que entregan un recurso en específico, algunos de ellos son el BaaS [29], [30], CaaS, DBaaS, etcetera [31]. Para este trabajo es importante definir el CaaS

- **Contenedores como Servicio (CaaS)** es una extensión del modelo IaaS en la que el proveedor ofrece un entorno de orquestación de contenedores como superficie de consumo, permitiendo al usuario desplegar, gestionar y escalar aplicaciones contenedorizadas sin gestionar la infraestructura de orquestación subyacente [32].

1.3.3 Modelos de despliegue

El NIST define cuatro modelos de despliegue que describen la propiedad, administración y acceso a la infraestructura de la nube [27]:

Nube privada: la infraestructura es para uso exclusivo de una única organización. Esta puede ser administrada y operada por la organización, un tercero o una combinación de ambos.

Nube pública: la infraestructura es para uso abierto por el público en general. Un proveedor administra las instalaciones y ofrece el servicio.

Nube híbrida: composición de dos o más infraestructuras de nube distintas (privada, comunitaria o pública) unidas por tecnología que permite la portabilidad de datos y aplicaciones.

Nube comunitaria: la infraestructura es para uso exclusivo de una comunidad específica de organizaciones que comparten intereses comunes como lo pueden ser misión, requisitos de seguridad, política o consideraciones de cumplimiento normativo. Esta puede ser administrada y operada por una o más de las organizaciones de la comunidad, un tercero o una combinación de ambos.

1.4 Infraestructura programable

Dado que el modelo HCI es el actual estándar para gestionar a los grandes centros de datos modernos es indispensable entender como se habilita este marco de control y administración.

1.4.1 Virtualización

La virtualización es una tecnología que crea una capa de abstracción sobre el hardware físico mediante software, permitiendo que los recursos de un único sistema físico se dividan y presenten como múltiples entornos de ejecución independientes [33], [34]. El mecanismo que hace posible esta abstracción es el **hipervisor**, este asignando recursos a cada máquina virtual y garantizando su aislamiento [35].

Más allá de la virtualización de servidores, el mismo principio de abstracción se aplica a otros dominios de la infraestructura [36]. La virtualización de red desacopla los servicios de red del hardware físico, permitiendo crear segmentos lógicos independientes sobre una fabric compartida [37]. La virtualización de almacenamiento agrupa dispositivos físicos heterogéneos en un pool lógico gestionado centralmente [38].

1.4.1.1 Passthrough de dispositivos y VFIO

Una capacidad avanzada de la virtualización es la asignación directa de un dispositivo de hardware físico a una máquina virtual, denominada **device passthrough**. En la virtualización convencional el hipervisor emula los dispositivos en software, lo que introduce una capa que penaliza el rendimiento para cargas intensivas. El passthrough elimina esa penalización al hacer que el dispositivo PCI (una GPU, una FPGA o un acelerador de inferencia) se asigne exclusivamente a una VM específica, que lo accede como si fuera hardware local propio. [39] En Linux, el marco que implementa esta asignación es **VFIO** (Virtual Function I/O), que expone el dispositivo al espacio de usuario de forma segura apoyándose en la unidad de gestión de memoria de entrada/salida (IOMMU) para aislar los accesos DMA del dispositivo [40]. La virtualización y sus mecanismos se resumen en la Fig. 4.

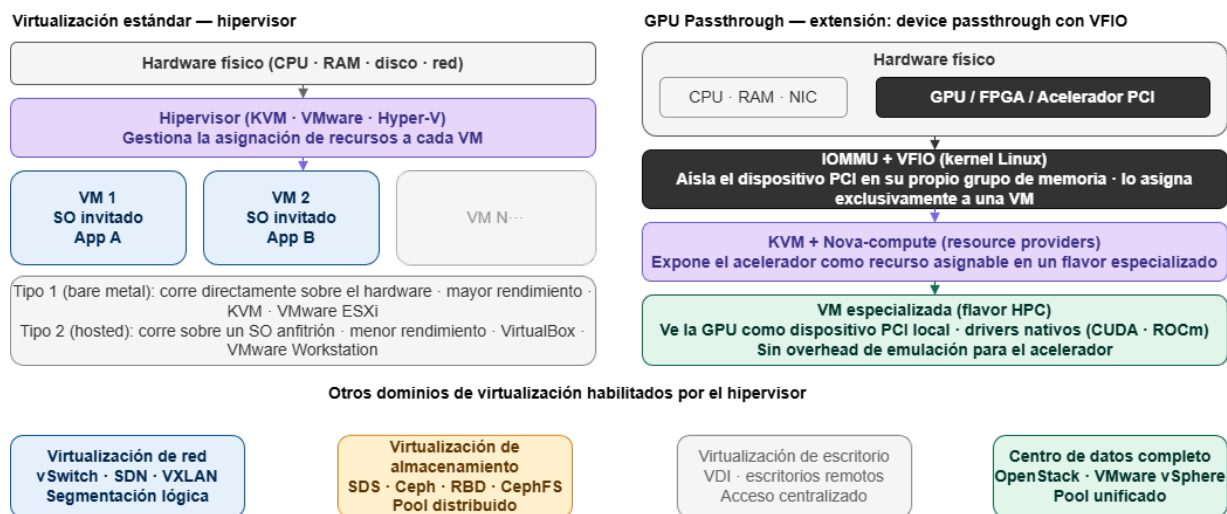


Fig. 4. Virtualización de recursos

1.4.2 Infraestructura definida por software

¿Cómo podemos aprovechar y automatizar la virtualización de recursos? La respuesta la encontramos en dos conceptos clave, el SDI e IaC.

1.4.2.1 SDI e IaC

Infraestructura Definida por Software (SDI): es un paradigma para el manejo de recursos de cómputo agrupados lógicamente y gestionados como si fueran software mediante APIs, lo que permite el aprovisionamiento de infraestructura basado en políticas y habilita la automatización del aprovisionamiento de cómputo [41], [42].

Infraestructura como Código (IaC): es una práctica que automatiza el aprovisionamiento y la gestión de la infraestructura de cómputo mediante archivos de configuración en lugar de procesos manuales, tratando a la infraestructura como software: los equipos la versionan, prueban y despliegan usando las mismas prácticas que aplican al código de aplicaciones [43], [44], [45].

Ambos conceptos operan en niveles distintos pero complementarios. La SDI abstrae y virtualiza los recursos físicos exponiéndolos como una superficie programable gestionada centralmente. IaC consume esa superficie programable mediante archivos de configuración almacenados en sistemas de control de versiones para su seguimiento, revisión y reversión. En otras palabras, la SDI es el paradigma que hace posible la programabilidad de la infraestructura, mientras que IaC es la metodología que la hace reproducible y automatizada. En la Fig. 5 se muestra la relación entre el SDI y la IaC de manera gráfica.



Fig. 5. SDI e IaC

1.4.2.2 Componentes del SDI

La SDI abstrae y virtualiza todos los recursos del centro de datos, haciendo que el hardware sea gestionable por software en tres dominios principales:

- **Cómputo:** Las máquinas virtuales (VM) son la unidad fundamental, ofreciendo un entorno aislado (CPU, Memoria, SO) cuyo ciclo de vida es totalmente controlado mediante APIs.

- Redes (SDN): Abstrae los elementos de hardware de red. Al separar el plano de control del plano de datos, permite gestionar la red de forma centralizada y programática, asignando segmentos virtuales a las cargas [37], [46].
- Almacenamiento (SDS): Combina diversos dispositivos físicos en un pool virtual. Desacopla la gestión de los datos del hardware físico, exponiendo capacidades estándar (bloque, objeto) como servicios consumibles [38]. Interacción: Los tres dominios no operan aislados. Una carga de trabajo necesita los tres recursos a la vez. La API programable es el puente esencial que permite a una capa superior (como IaC) describir, versionar y automatizar el aprovisionamiento conjunto de estos recursos sin intervención manual.

1.5 Contenedorización y orquestación

Esta tecnología se considera como la evolución de la virtualización y constituye la base para el CaaS [47]. Para este trabajo el CaaS es fundamental, ya que dentro de la arquitectura a plantear es este modelo el que satisface el objetivo específico «Estandarización del despliegue mediante contenedores».

1.5.1 Fundamentos de contenedorización

La contenedorización es el empaquetamiento del código de software junto con las bibliotecas del sistema operativo y las dependencias estrictamente necesarias para su ejecución, produciendo una unidad ligera y ejecutable que se conoce como **contenedor** y que opera de forma consistente sobre cualquier infraestructura [48], [49], [50].

Un entorno contenerizado presenta una arquitectura en capas que parte del hardware subyacente y asciende hasta la aplicación. La base la constituye la infraestructura física que provee CPU, memoria, almacenamiento y red. Sobre ella opera el sistema operativo anfitrión, cuyo kernel es el componente crítico compartido por todos los contenedores del host. Encima reside el motor de contenedores, responsable de crear, ejecutar y gestionar los contenedores. Finalmente, en la capa superior se encuentran los contenedores mismos, cada uno con los binarios y bibliotecas de su aplicación, pero sin sistema operativo propio. La abstracción de esta arquitectura como entorno de ejecución se puede visualizar en la Fig. 6.

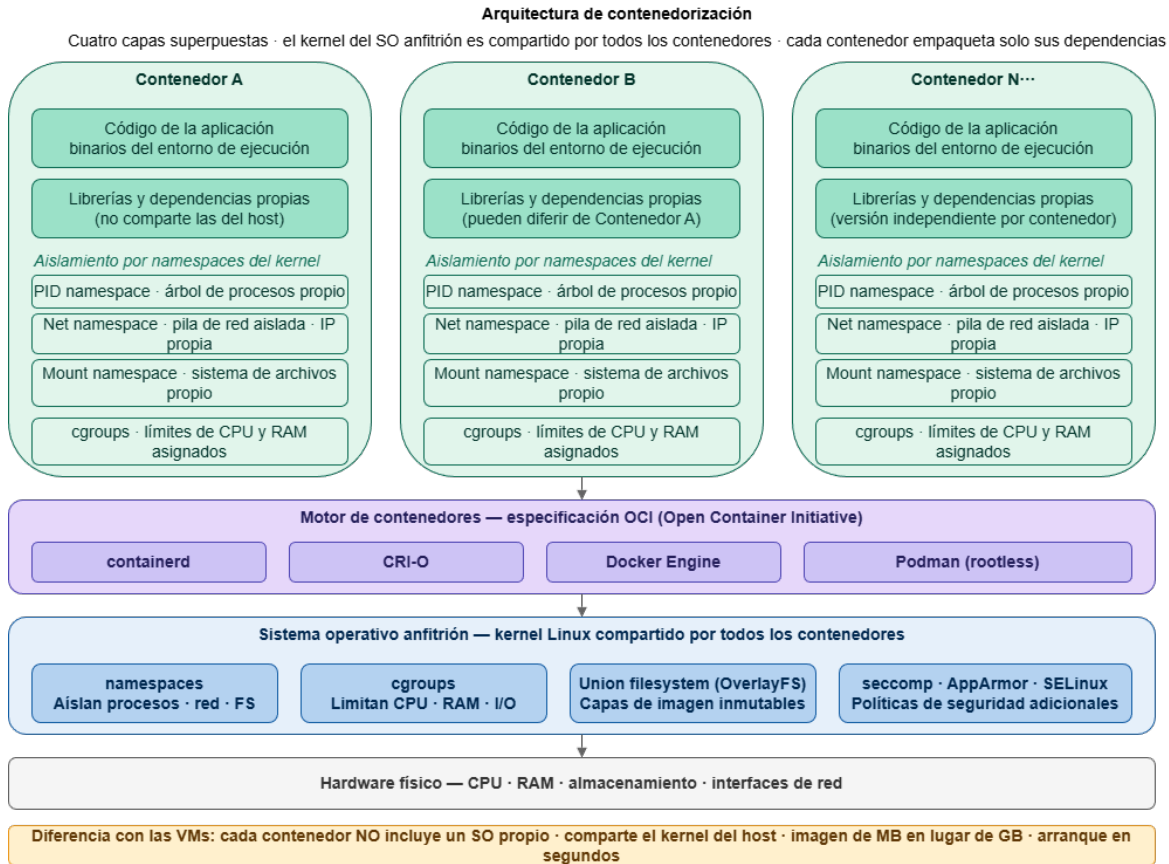


Fig. 6. Arquitectura de la contenedorización

El artefacto central del flujo de trabajo es la **imagen**: un archivo estático, inmutable y portable que actúa como plano de la aplicación, encapsulando el código y todas sus dependencias en un paquete de capas [51]. Las imágenes se construyen siguiendo la especificación de la **Open Container Initiative** (OCI) [52], garantizando interoperabilidad entre motores.

1.5.2 Diferencias entre máquinas virtuales y contenedores

Tanto las máquinas virtuales (VM) como los contenedores permiten ejecutar aplicaciones en entornos aislados, sin embargo, difieren fundamentalmente en el nivel en que aplican la virtualización [53], [54]. Mientras el hipervisor virtualiza el hardware para presentar un sistema operativo invitado completo por instancia, el contenedor comparte el kernel del anfitrión y logra su aislamiento mediante dos mecanismos nativos de Linux: los *namespaces*, que aíslan la visibilidad de recursos como procesos, red y puntos de montaje, y los *cgroups*, que limitan el consumo de CPU, memoria y E/S [55]. A continuación se presenta una tabla comparativa de estas tecnologías:

TABLA 6

COMPARATIVA ENTRE MÁQUINAS VIRTUALES Y CONTENEDORES [53], [54], [56].

Característica	Máquina Virtual	Contenedor
Nivel de virtualización	Hardware	Sistema operativo
Sistema operativo	SO invitado completo por instancia	Comparte el kernel del SO anfitrión
Tecnología de aislamiento	Hypervisor (VMware, KVM, Hyper-V)	Motor de contenedorización (<i>namespaces + cgroups</i>)
Tamaño de imagen	Gigabytes (GB)	Megabytes (MB)
Tiempo de arranque	Minutos	Segundos
Uso de recursos	Alto	Bajo
Densidad por servidor	Baja	Alta
Nivel de aislamiento	Hardware (fuerte)	Proceso (moderado)
Portabilidad	Limitada	Alta
Escalabilidad	Vertical (entorno completo)	Horizontal (por microservicio)
Casos de uso típicos	Aplicaciones legacy · entornos multi-SO · seguridad estricta	Microservicios · CI/CD · cloud native

1.5.3 Orquestación

La orquestación es la ejecución coordinada de múltiples tareas o procesos de automatización de TI a través de múltiples sistemas, aplicaciones y servicios, garantizando que el despliegue y la gestión de configuración se realicen en la secuencia correcta [57]. Es importante distinguirla de la automatización: la automatización ejecuta tareas sin intervención humana, mientras que la orquestación coordina esas tareas automatizadas en flujos de trabajo de orden superior.

1.5.3.1 Orquestación de contenedores

La orquestación de contenedores es el proceso de automatizar el despliegue, la gestión, el escalado y la conectividad de red de los contenedores a lo largo de su ciclo de vida [58]. En lugar de especificar cómo realizar cada tarea, el operador declara el **estado deseado** de la aplicación y el orquestador se encarga de alcanzarlo y mantenerlo [59].

Las funciones clave de un orquestador moderno incluyen el despliegue controlado de nuevas versiones sin tiempo de inactividad, el autoescalado basado en métricas, la **autocuración** que reinicia o reemplaza contenedores fallidos, el descubrimiento de servicios mediante nombres DNS estables, y la gestión de configuración y secretos [58], [59].

Kubernetes es la plataforma de orquestación de referencia, originalmente desarrollada por Google y donada a la Cloud Native Computing Foundation (CNCF) en 2015 [59]. Opera a través de tres componentes clave [58].:

- El **plano de control**, que toma decisiones globales sobre el clúster.
- Los **nodos**, que son las máquinas donde los contenedores se ejecutan.
- Los **pods**, la unidad mínima desplegable que puede contener uno o más contenedores con almacenamiento y red compartidos

1.6 Redes

La red es es la columna vertebral que hace funcionar a los centros de datos. Es el sistema nervioso central que permite que el resto de los sistemas interactúen y, por lo tanto, es el componente que más influye en la capacidad y escala del centro de datos.

1.6.1 Topologías de red para centros de datos

La topología de red de un centro de datos determina cómo se interconectan los nodos físicos y tiene implicaciones directas sobre el rendimiento, la resiliencia y la escalabilidad de estos sistemas [60].

La **topología de árbol** es la arquitectura tradicional de tres capas ampliamente utilizada en centros de datos heredados. Su principal limitación es que el tráfico este-oeste entre servidores del mismo nivel debe subir hasta la capa de núcleo antes de descender al destino, introduciendo latencia adicional y creando cuellos de botella a medida que aumenta la densidad de servidores.

La **topología Spine-Leaf**, variante moderna de la arquitectura Clos [60], [61], resuelve estas limitaciones mediante una fabric de dos capas donde cada switch de la capa Leaf se conecta a todos los switches de la capa Spine y nunca directamente a otro Leaf. Esta restricción garantiza que cualquier par de nodos esté separado por exactamente dos saltos, produciendo latencia uniforme y ancho de banda predecible entre cualquier par de nodos del centro de datos.

1.6.2 Redes superpuestas: el modelo overlay

Una red superpuesta (**overlay network**) es una red lógica construida sobre una red física existente. En lugar de depender de la configuración de los dispositivos físicos de red, una red overlay encapsula el tráfico de la red lógica dentro de paquetes de la red física, permitiendo que la topología lógica sea independiente de la topología física subyacente. Este mecanismo permite crear múltiples redes lógicas aisladas sobre la misma infraestructura física, habilitando la separación multitenant sin requerir hardware de red separado para cada usuario. [60]

1.6.3 VXLAN: encapsulamiento para redes escalables

VLAN (Virtual LAN) es el mecanismo tradicional de segmentación lógica de redes Ethernet, estandarizado en IEEE 802.1Q [62]. Su identificador de VLAN es un campo de 12 bits, lo que acota el espacio de direccionamiento a 4094 segmentos utilizables. En entornos de centro de datos con miles de inquilinos y decenas de miles de máquinas virtuales, este límite se agota con facilidad.

VXLAN (Virtual Extensible LAN) es un protocolo de encapsulamiento definido en el RFC 7348 [63] que extiende la segmentación de redes Ethernet sobre redes IP mediante un identificador de red (VNI) de 24 bits, ampliando el espacio de segmentación a más de 16 millones de redes lógicas. Su mecanismo de operación consiste en una arquitectura de Capa 2 sobre Capa 4 (L2 over L4): la trama Ethernet original (Capa 2) se encapsula íntegramente dentro de un datagrama UDP (Capa 4), el cual a su vez es transportado por un paquete IP (Capa 3). [60]

1.7 Seguridad

Esta sección profundiza en la seguridad como una propiedad fundamental de los sistemas distribuidos al hablar de tecnologías que actualmente constituyen el estándar de seguridad en el tráfico de datos a través de internet o cualquier otra red.

1.7.1 Criptografía de clave pública y certificados TLS

La confidencialidad e integridad del tráfico en redes distribuidas se garantiza mediante cifrado. El protocolo TLS (Transport Layer Security) es el estándar de facto para cifrar comunicaciones en redes TCP/IP [64]. TLS utiliza criptografía de clave pública: cada extremo posee un par de claves (una pública y una privada) y el protocolo garantiza que únicamente el poseedor de la clave privada puede descifrar los mensajes cifrados con la clave pública correspondiente.

Un **certificado digital** es un documento electrónico que vincula una clave pública con la identidad de su propietario, firmado por una entidad de confianza denominada Autoridad Certificadora (CA). El formato de uso universal en internet es X.509, cuyo perfil para PKI se especifica en el RFC 5280 [65]. La cadena de confianza funciona así: la CA firma el certificado del servidor con su propia clave privada; el cliente verifica esa firma usando la clave pública de la CA, conocida de antemano porque forma parte de su almacén de confianza [65].

1.7.2 Identidad y autenticación federada

En organizaciones con múltiples sistemas de información, la **autenticación federada** permite a un usuario autenticarse una única vez contra un proveedor de identidad centralizado y obtener acceso a múltiples sistemas sin repetir el proceso.

El modelo define dos roles: el **Proveedor de Identidad** (IdP) gestiona las credenciales del usuario y realiza la autenticación; el **Proveedor de Servicios** (SP) confía en el IdP para verificar la identidad sin gestionar credenciales propias.

SAML 2.0 (Security Assertion Markup Language) es el estándar basado en XML más utilizado para la federación de identidad en entornos empresariales [66]. **OIDC** (OpenID Connect) es un protocolo de identidad construido sobre OAuth 2.0 que provee autenticación mediante tokens JSON firmados [67], dominante en entornos cloud native por su compatibilidad nativa con APIs REST. Ambos protocolos persiguen el mismo objetivo: permitir que un sistema confíe en la identidad afirmada por otro sin compartir credenciales.

1.7.3 Gestión de secretos

En sistemas distribuidos, los secretos tales como contraseñas, claves de API, tokens de servicio, etcetera, representan uno de los vectores de ataque más frecuentes cuando se almacenan en archivos de configuración, variables de entorno o repositorios de código fuente. La **gestión centralizada de secretos** es la práctica de almacenar todos los secretos en un almacén seguro y distribuirlos a los servicios bajo demanda, con control de acceso granular y registro inmutable de cada acceso [68].

Un almacén de secretos provee tres garantías fundamentales: que los valores sensibles nunca se almacenan en texto plano fuera del almacén; que cada servicio recibe únicamente los secretos que necesita para operar¹⁰ y que toda operación sobre un secreto queda registrada para auditoría [68].

1.7.4 Automatización de certificados: ACME

La gestión manual del ciclo de vida de los certificados TLS es inviable en entornos distribuidos donde los servicios se crean y destruyen dinámicamente. El protocolo ACME (Automatic

¹⁰Es la aplicación directa del principio de mínimo privilegio formulado por Saltzer y Schroeder [69].

Certificate Management Environment), definido en el RFC 8555 [70], estandariza la interacción automatizada entre un cliente de gestión de certificados y una autoridad certificadora, permitiendo que el ciclo de vida completo de un certificado ocurra sin intervención humana.

El proceso de emisión mediante ACME requiere que el solicitante demuestre que controla el dominio para el que solicita el certificado. El método **DNS-01** consiste en crear un registro TXT específico en el DNS público del dominio; la CA verifica su presencia y emite el certificado si la verificación es exitosa. Este método tiene la ventaja de no requerir que el servicio sea accesible desde internet durante la validación, lo que la hace viable para servicios internos con dominios públicos.

1.7.5 Infraestructura de clave pública (PKI)

Una Infraestructura de Clave Pública (PKI) es el conjunto de políticas, procedimientos, hardware, software y personas necesarios para crear, gestionar, distribuir, usar, almacenar y revocar certificados digitales [65]. En entornos de centro de datos, una PKI interna permite emitir certificados de confianza para las comunicaciones internas del sistema sin depender de CAs externas, a costo operativo nulo y sin exposición de tráfico interno a validaciones externas.

La arquitectura típica de una PKI institucional define una **CA raíz** (cuya clave privada se protege con el máximo nivel de seguridad) que emite certificados a CAs intermedias, que a su vez emiten los certificados de uso final para los servicios [65]. Esta jerarquía permite revocar una CA intermedia comprometida sin afectar la CA raíz, limitando el radio de daño ante un incidente de seguridad.

CAPÍTULO

II

Principios de diseño arquitectónico y selección del stack

RESUMEN

Este capítulo establece los principios de diseño arquitectónico que orientan la propuesta y la selección del stack tecnológico que la materializa. El argumento se construye en tres etapas: primero, un análisis del caso de la Facultad de Ingeniería de la UNAM que identifica las necesidades concretas a satisfacer frente a un ecosistema fragmentado; segundo, la derivación de principios arquitectónicos organizados en dominios a partir de dicho diagnóstico y de marcos de referencia internacionales; y tercero, la selección, evaluación de interoperabilidad y justificación del stack tecnológico resultante.

CONCEPTOS CLAVE

Gobierno federado	Modelo de gestión donde diversas entidades tienen autonomía total para administrar sus propios recursos, ciclos de vida y despliegues, operando sobre una plataforma compartida que garantiza estándares de seguridad, cumplimiento y conectividad dictados por una gobernanza central.
Soberanía tecnológica	Principio estratégico que garantiza el control absoluto sobre los datos, la infraestructura y el software desarrollado.
Protoarquitectura	Modelo analítico intermedio que define las responsabilidades funcionales del sistema antes de vincularlas a tecnologías específicas.
Ecosistema CNCF	Universo de tecnologías probadas y estandarizadas utilizado como filtro de selección para garantizar madurez técnica e interoperabilidad.

Se sabe que la constitución y adopción de tecnologías obedece al contexto social y a las necesidades concretas de los grupos que las emplean [2]. Este principio no es ajeno al diseño de sistemas: Bass, Clements y Kazman [71] establecen que toda arquitectura de software se construye para satisfacer los requerimientos de una organización, y que son dichos requerimientos los que determinan las propiedades fundamentales del sistema. Lo anterior constituye para este trabajo el axioma principal para la definición de los principios de diseño arquitectónico y por extensión, también para la selección del stack tecnológico. Es importante mencionar que los principios de diseño arquitectónico que aquí se proponen parten del diagnóstico de las necesidades y áreas de oportunidad identificadas en la Facultad de Ingeniería de la UNAM como caso de estudio particular; sin embargo, el proceso metodológico que los sustenta es transferible a cualquier organización o institución que enfrente problemáticas similares.

2.1 Análisis de caso: Facultad de Ingeniería

El siguiente análisis parte de la observación directa del ecosistema tecnológico de la Facultad de Ingeniería de la UNAM. Si bien el alcance de este trabajo se centra en la gestión de infraestructura para la creación de un centro de datos, resulta necesario examinar las aplicaciones institucionales que los usuarios consumen, pues la calidad de cualquier implementación de software está directamente relacionada con una infraestructura flexible, de alta disponibilidad y alineada con marcos de trabajo estandarizados.

2.1.1 Ecosistema de software institucional

La Facultad de Ingeniería presenta un ecosistema tecnológico fragmentado, caracterizado por la escasa o nula interoperabilidad entre los sistemas que lo componen. Desde la perspectiva de los alumnos, esta fragmentación se manifiesta de la siguiente manera:

- Ausencia de un mecanismo de autenticación unificado.
- La duplicación de información entre sitios web independientes entre sí.
- La circulación de contenido desactualizado en fuentes secundarias y terciarias que no dependen de una única fuente de verdad.
- Escasa personalización de la información que fluye hacia el alumno.
- No existen mecanismos de suscripción temática a través de los cuales los alumnos puedan elegir la información que es de su interés y que desean conocer de manera oportuna.
- Algunos servicios web presentan intermitencias o tiempos de respuesta elevados. Esto es síntoma de sistemas que operan sin redundancia ni distribución de carga.

- El stack predominante en los sistemas institucionales se sustenta en soluciones maduras y probadas; sin embargo, se caracteriza por una baja frecuencia de actualización.

2.1.2 Gestión del conocimiento y equipo de desarrollo

Se observa que el equipo de desarrollo institucional presenta un perfil heterogéneo. Los sistemas críticos son atendidos por personal veterano con amplio conocimiento de los procesos institucionales y baja rotación, lo que garantiza continuidad operativa, pero concentra el conocimiento en personas específicas. La documentación técnica formal es escasa: en la práctica, el conocimiento sobre el funcionamiento, las dependencias y las decisiones de diseño de los sistemas reside en quienes los desarrollaron o han operado durante años, lo que representa un riesgo significativo ante ausencias o cambios de personal.

En paralelo, una fracción del desarrollo institucional recae en prestadores de servicio social, cuya rotación semestral introduce una tensión estructural: por un lado, este perfil tiende a incorporar prácticas y tecnologías más recientes; por otro, la brevedad de su participación impide que el conocimiento generado se consolide institucionalmente. El resultado es un ecosistema donde la innovación se introduce de forma episódica y frecuentemente enfrenta resistencia al cambio por parte de las estructuras establecidas.

2.1.3 Infraestructura de cómputo

La infraestructura de cómputo de la Facultad se encuentra distribuida de forma no centralizada a lo largo de sus instalaciones. En prácticamente todas las áreas existen equipos destinados a alojar sistemas específicos: en algunos casos se trata de servidores dedicados con cuarto propio, en otros de estaciones de trabajo o equipos de escritorio que operan como servidores con dirección IP pública asignada. Ciertas áreas cuentan con capacidad de cómputo considerable que, sin embargo, permanece subutilizada durante la mayor parte del año al estar dimensionada para demandas puntuales. Esta heterogeneidad de hardware es una condición estructural del entorno y cualquier solución propuesta debe ser capaz de operarla como una fortaleza, no como una limitación.

Desde el punto de vista de la conectividad, la Facultad opera bajo la administración de red de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC), y cuenta con una red interna orientada principalmente a proveer conectividad inalámbrica en sus instalaciones, así como un rango de direcciones IP públicas para su uso. Los servidores operan en su mayoría de forma aislada, cada uno con su propia dirección IP pública y sin integración con los demás nodos institucionales.

Este punto tiene un impacto considerable en la calidad de los sistemas desarrollados en la universidad y, por ello, constituye el aspecto de mayor interés para su resolución mediante este trabajo.

2.1.4 Marco de gobierno y su impacto en la infraestructura

La UNAM articula su gobierno mediante un modelo federado: las facultades, escuelas e institutos que la integran mantienen autonomía administrativa en la gestión de sus recursos y procesos internos, pero operan bajo un marco normativo común que preserva la coherencia institucional. Este esquema tiene fundamento normativo explícito: la Ley Orgánica [72] confiere a la Universidad la facultad de gobernarse a sí misma y de organizarse como lo estime conveniente, mientras que el Estatuto General [73] distribuye esa facultad entre las entidades académicas, dotando a cada facultad, escuela e instituto de un director y un consejo técnico con atribuciones propias sobre su administración interna, subordinados a los órganos de gobierno centrales de la Universidad. Este esquema de gobierno contribuye, en mayor o menor medida, al aislamiento de los sistemas computacionales: las normas emitidas por las autoridades generales, si bien buscan atacar problemas orgánicos en la institución o mejorarla, no suelen acompañarse de esquemas de implementación bien definidos ni de soluciones de software interoperable, por lo que cada entidad resuelve de forma independiente los requerimientos normativos, provocando así más dispersión tecnológica.

La asignación de recursos de cómputo se obtienen en cada proyecto mediante su solicitud formal a través de oficios, proceso que si bien obedece a las normativas institucionales vigentes, introduce retrasos y puntos de fricción considerables en los ciclos de desarrollo de sistemas.

A pesar de que el marco de gobierno pareciera un obstáculo, es este mismo marco compartido el que define un conjunto de necesidades comunes que constituye la base para el diseño de soluciones conjuntas, soluciones que pueden ser solventadas a través de un modelo de nube comunitaria.

A partir de lo anterior se identifican las siguientes necesidades a satisfacer mediante la solución tecnológica propuesta:

- **Centralización con autonomía administrativa:** se requiere una plataforma que consolide los recursos de cómputo dispersos en un pool unificado y gestionable, pero que preserve la autonomía de cada entidad para administrar los recursos y completo control de los servicios que se ejecuten sobre la infraestructura subyacente.

- **Uso eficiente de los recursos existentes:** es indispensable evitar la subutilización de la capacidad de cómputo instalada mediante mecanismos de asignación dinámica que permitan redistribuir la capacidad ociosa entre las cargas de trabajo activas.
- **Compatibilidad con entornos heterogéneos:** la arquitectura debe operar en un entorno donde coexisten equipos de alta capacidad y equipos con recursos limitados, sin que la heterogeneidad impida la integración de los nodos al clúster.
- **Aprovisionamiento automatizado y autoservicio:** es necesario modernizar los procesos administrativos manuales mediante la automatización del aprovisionamiento de infraestructura y la habilitación de un mecanismo de autoservicio que permita a los equipos acceder a recursos de cómputo de forma autónoma y bajo demanda por las entidades que así lo requieran.
- **Interoperabilidad y fuente de verdad única:** la solución debe permitir la implementación de una capa de servicios compartidos que apunte a la eliminación de backends duplicados entre áreas y establezca una fuente de verdad centralizada.
- **Continuidad operativa y resiliencia:** la infraestructura debe incorporar mecanismos de alta disponibilidad que garanticen la continuidad del servicio ante fallos de hardware o mantenimiento de nodos individuales.
- **Base tecnológica de código abierto y costo cero de licenciamiento:** esto no es una necesidad en si misma, sin embargo, es una decisión estratégica para evitar caer en dependencias tecnológicas con proveedores comerciales.

2.2 Principios de diseño

Para estructurar estos principios de forma rigurosa se adoptó el enfoque de dominios propuesto por TOGAF [74], marco de arquitectura empresarial que distingue entre **principios de negocio**, **principios de datos**, **principios de aplicaciones** y **principios de tecnología**. Este modelo establece que las decisiones técnicas y operativas deben derivarse de las motivaciones del negocio, garantizando que la infraestructura no se defina de forma aislada, sino como consecuencia directa de las necesidades identificadas y buscando seguir las mejores prácticas consolidadas en la industria. Entre las buenas prácticas seleccionadas destacan: la **observabilidad**, que exige que el estado interno del sistema sea comprensible a partir de sus salidas externas mediante métricas, logs y trazas; la **auditabilidad**, que requiere que toda operación sobre recursos sea registrada con identidad, tiempo y resultado verificables; la **idempotencia**, que establece que aplicar la

misma operación de configuración múltiples veces debe producir siempre el mismo estado final [45]; la **resiliencia**, que demanda que el sistema continúe operando de forma degradada ante fallos parciales sin pérdida total del servicio [20]; la **soberanía tecnológica**, que exige que la organización mantenga control efectivo sobre sus datos, artefactos y plataformas sin dependencia exclusiva de proveedores externos [16]; y el **principio de mínimo privilegio**, formulado por Saltzer y Schroeder [69], que establece que cada componente y usuario debe operar únicamente con los permisos estrictamente necesarios para su función. Estas prácticas no son restricciones impuestas desde el exterior sino propiedades que la arquitectura debe satisfacer por diseño.

2.2.1 Dominio de negocio: visión, gobierno y operación

- **Economía de escala:** la consolidación de recursos dispersos en un pool unificado reduce el costo operativo por unidad de cómputo y elimina la subutilización del hardware existente.
- **Soberanía tecnológica:** control institucional sobre la ubicación, el ciclo de vida y el tratamiento de la información, sin dependencia de proveedores comerciales externos.
- **Rendición de cuentas:** el consumo de recursos de cómputo debe ser medible y atribuible a cada entidad, garantizando que su distribución responda a las prioridades institucionales.
- **Autonomía federada:** cada entidad preserva la administración de sus servicios y plataformas, mientras comparte la infraestructura subyacente bajo un marco normativo común.
- **Autonomía operativa:** cada entidad debe poder administrar sus recursos locales dentro de límites definidos centralmente, sin requerir intervención de niveles superiores para operaciones cotidianas.

2.2.2 Dominio de datos: gestión y protección de la información

- **Soberanía y posesión de datos:** es indispensable mantener control total sobre el ciclo de vida de sus datos sensibles, incluyendo su almacenamiento, acceso y cifrado, sin delegar esta responsabilidad a terceros.
- **Resiliencia de datos:** la arquitectura debe proteger la información institucional ante fallos geográficos o pérdida de nodos, garantizando la disponibilidad de los datos independientemente del estado de un campus específico.

- **Trazabilidad de acciones:** todo cambio sobre la infraestructura debe quedar registrado de forma inmutable y atribuible a un usuario real, garantizando la rendición de cuentas operativa.

2.2.3 Dominio de aplicaciones: interacción y entrega de servicios

- **Autoservicio y agilidad:** los equipos de desarrollo deben poder solicitar y recibir recursos de cómputo de forma autónoma, eliminando la dependencia de procesos administrativos manuales.
- **Interoperabilidad:** los sistemas deben exponerse mediante interfaces y estándares comunes que permitan la integración entre áreas y la eliminación de backends duplicados.

2.2.4 Dominio de tecnología: infraestructura y atributos del sistema

- **Mantenibilidad:** la arquitectura debe facilitar la detección, diagnóstico y reparación de fallos sin interrumpir la operación global, considerando la limitación de personal técnico disponible.
- **Transparencia:** ocultar la complejidad de distribución, replicación y localización de recursos, de modo que el consumidor del servicio perciba un sistema homogéneo independientemente de su distribución física.
- **Confiabilidad:** implementar mecanismos de failover y replicación que garanticen la continuidad operativa ante fallos de nodos individuales, conforme a las características de disponibilidad y fiabilidad descritas por Van Steen y Tanenbaum [20].
- **Apertura:** el stack tecnológico permitir que sus componentes sean integrados en otros sistemas y viceversa.
- **Escalabilidad:** la arquitectura debe permitir la incorporación progresiva de nodos y la constitución de clústeres especializados incluyendo cómputo de alto rendimiento e inferencia de modelos de lenguaje sin rediseño estructural. Así mismo se debe permitir que los servicios ejecutados sobre la infraestructura sean escalables para soportar más demanda cuando se requiera.
- **Seguridad:** se debe garantizar confidencialidad, integridad y autenticación como propiedades inherentes del sistema distribuido.
- **Observabilidad unificada:** el estado de todos los nodos y servicios del clúster debe ser visible en tiempo real, permitiendo la identificación proactiva de cuellos de botella y degradaciones de servicio.

- **Consistencia de configuración:** la configuración de todos los nodos debe gestionarse como código versionado, eliminando la divergencia entre entornos y garantizando que el estado del sistema sea reproducible y auditable.

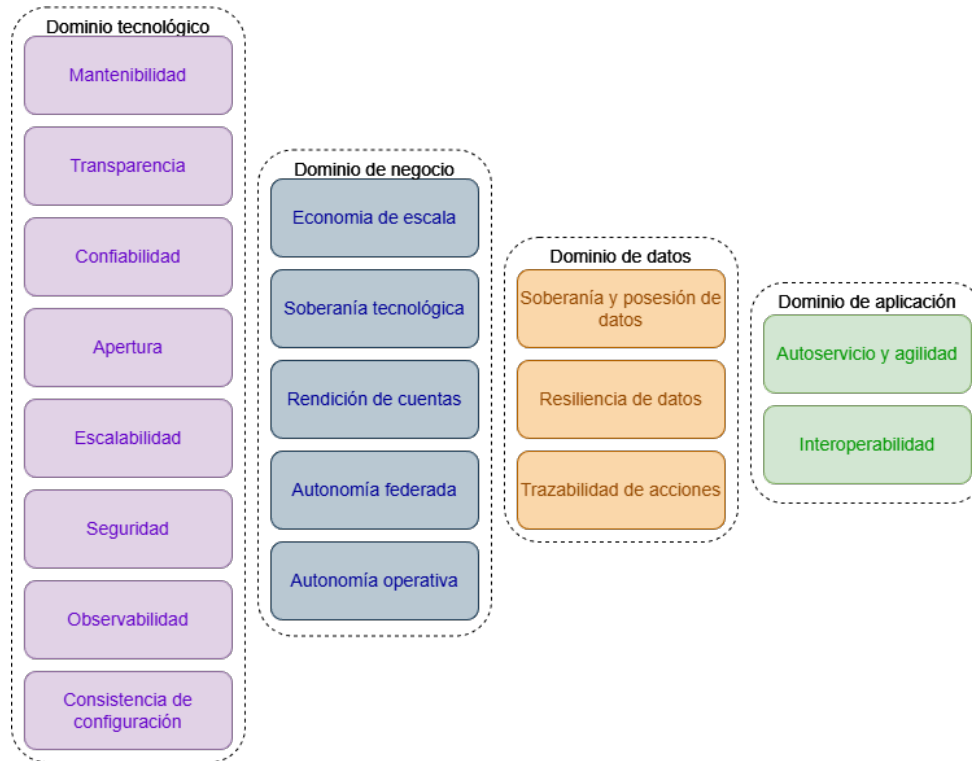


Fig. 7. Resumen de principios arquitectónicos por dominio

Se puede afirmar que la consolidación de cargas de trabajo sobre el hardware existente reduce la huella energética respecto a la operación de múltiples equipos subutilizados de forma aislada, sin embargo, la **sustentabilidad** no se abodará de manera directa en este trabajo pero constituye un principio de diseño transversal y que debe ser considerado para la implementación de esta o cualquier propuesta similar.

La validez de los principios definidos en las secciones anteriores no se valida únicamente a través del diagnóstico institucional, adicionalmente se encuentra respaldo en los marcos de referencia que los principales proveedores de infraestructura en la nube, marcos que han consolidado a partir de su experiencia operando sistemas a escala global. El AWS Well-Architected Framework [75] gestiona las mejores prácticas de diseño arquitectónico en torno a seis pilares: excelencia operativa, seguridad, confiabilidad, eficiencia de rendimiento, optimización de costos y sustentabilidad. El Google Cloud Well-Architected Framework [76] organiza sus recomendaciones en los mismos ejes no funcionales: seguridad, confiabilidad, rendimiento, costo, operaciones y sustentabilidad.

El Azure Well-Architected Framework [77] consta de cinco pilares equivalentes: confiabilidad, seguridad, optimización de costos, excelencia operativa y eficiencia de rendimiento.

La coincidencia entre estos tres marcos no es casual; evidencia que dichos principios atienden necesidades reales de la industria. El conjunto de principios definidos en este capítulo guarda correspondencia con el consenso observado entre AWS, Google y Microsoft; esta correspondencia se muestra en la tabla siguiente:

TABLA 7

CORRESPONDENCIA ENTRE LOS PRINCIPIOS PROPUESTOS Y LOS MARCOS WELL-ARCHITECTED DE LA INDUSTRIA.

Principio (este trabajo)	AWS WAF	Google Cloud WAF	Azure WAF
Economía de escala · Rendición de cuentas	Optimización de costos	Costo	Optimización de costos
Confiabilidad · Resiliencia de datos	Confiabilidad	Confiabilidad	Confiabilidad
Seguridad · Soberanía · Identidad federada	Seguridad	Seguridad	Seguridad
Observabilidad · Consistencia de configuración	Excelencia operativa	Operaciones	Excelencia operativa
Escalabilidad · Modularidad y expansión	Eficiencia de rendimiento	Rendimiento	Eficiencia de rendimiento
Sustentabilidad	Sustentabilidad	Sustentabilidad	—
Autonomía federada · Autoservicio	—	—	—
Soberanía tecnológica · Apertura	—	—	—

La mayoría de los principios definidos tiene correspondencia directa con alguno de los pilares de los tres marcos, lo que respalda la propuesta mostrando que está alineada con soluciones probadas y solidas, por lo tanto, además de los principios seleccionados se usará el marco AWS Well Architected Framework como un marco de respaldo para resolver cualquier escenario no contemplado durante la selección del stack y la definición de la arquitectura misma.

Es importante mencionar que los principios de autonomía federada, autoservicio, soberanía tecnológica y apertura no tienen un equivalente explícito en ninguno de los tres marcos comerciales, lo que es esperado ya estos principios (o necesidades) emergen de las particularidades del contexto institucional y no de un entorno de nube pública donde el proveedor gestiona la infraestructura.

Para seleccionar el stack tecnológico es necesario limitar el universo de opciones disponibles ya que en el mercado hay un sin número de tecnologías que permiten resolver cada una de las necesidades identificadas con una o varias aproximaciones de diseño. Para acotar este espacio de búsqueda de forma rigurosa, este trabajo se limitará al ecosistema definido por la Cloud

Native Computing Foundation (CNCF), organización neutral adscrita a la Linux Foundation que compila, categoriza y evalúa proyectos y productos del espacio cloud native mediante un proceso de madurez formal administrado por su Technical Oversight Committee (TOC). [78]

El proceso de evaluación del TOC establece tres niveles de madurez que son:

- Sandbox
- Incubating
- Graduated

Para alcanzar el nivel Graduated, el más exigente, un proyecto debe demostrar adopción en producción por organizaciones independientes, contar con committers de al menos dos organizaciones distintas, haber obtenido y mantenido el CII Best Practices Badge, haber completado una auditoría de seguridad independiente por un tercero con vulnerabilidades críticas resueltas, y haber definido explícitamente su modelo de gobernanza y el ciclo de vida de sus mantenedores en un documento público [79]. Este proceso garantiza que los proyectos seleccionados no son software experimental sino soluciones probadas en entornos de producción a escala, con comunidades activas y ciclos de vida predecibles que la institución puede adoptar sin incurrir en riesgos de abandono tecnológico. Conviene precisar el alcance de esta restricción: el Landscape cataloga proyectos y productos de múltiples fundaciones, por lo que figurar en él no implica ser un proyecto alojado por la CNCF: OpenStack, Ceph y Loki aparecen en el catálogo sin serlo. Lo que este trabajo acota es el ecosistema que el Landscape delimita, no la membresía de la fundación.

La restricción al ecosistema CNCF cumple así una doble función en este trabajo:

- Limita el universo de tecnologías candidatas a un conjunto manejable, auditado y que cumple estándares de calidad.
- Al filtrar únicamente proyectos de código adscritos a su ecosistema se garantiza la selección de componentes que se adhieran al principio de soberanía tecnológica.

2.3 Protoarquitectura funcional

Antes de presentar las tecnologías candidatas es necesario establecer un modelo intermedio que explique por qué se requieren componentes de cada categoría del CNCF Landscape para implementar un centro de datos distribuido. Este modelo no describe tecnologías específicas

sino responsabilidades funcionales: qué debe hacer el sistema en cada capa para satisfacer los principios definidos. Las tecnologías candidatas que se presentan en la sección siguiente son, por lo tanto la respuesta, concreta a cada una de estas responsabilidades.

La protoarquitectura se organiza siguiendo la jerarquía de capas del CNCF Landscape [78], que su guía oficial describe como una dependencia natural de abajo hacia arriba: cada capa superior solo puede operar si la capa inferior está correctamente resuelta. En términos de la guía, las capas se construyen una sobre otra partiendo de las herramientas para establecer la base de infraestructura, pasando por todo lo que un contenedor necesita para ejecutarse, hasta llegar a las herramientas para orquestar y gestionar contenedores y aplicaciones. Aplicado al contexto de este trabajo, esta dependencia no es una convención del CNCF sino una realidad operativa: un clúster de contenedores no puede existir sin un runtime que los ejecute, un runtime no puede operar sin infraestructura aprovisionada, y la infraestructura no puede ser confiable sin almacenamiento distribuido.

2.3.1 Provisioning: construir la base

El CNCF [78] describe esta capa como el conjunto de herramientas necesarias para crear y fortalecer la base de infraestructura sobre la que corren los contenedores, señalando que en los entornos tradicionales los ciclos de liberación de infraestructura requerían intervención humana durante meses, lo que es incompatible con el aprovisionamiento dinámico que exige el cómputo cloud native.

En el contexto específico de este trabajo, esa incompatibilidad adquiere una dimensión adicional: la infraestructura no solo es dinámica sino heterogénea y geográficamente dispersa. A partir de ese diagnóstico, se identifican tres problemas concretos que esta capa debe resolver y que determinan qué subcategorías son necesarias.

El primero es la **abstracción del hardware**: los servidores físicos deben transformarse en recursos de cómputo, red y almacenamiento homogéneos, accesibles mediante una API unificada. Sin esta abstracción, cada nodo es una isla que requiere administración independiente y no puede participar en un pool compartido. Este problema corresponde a la subcategoría **Automation & Configuration** que el CNCF define como el conjunto de herramientas que automatizan la creación y configuración de recursos de cómputo [78].

El segundo es la **reproducibilidad del estado**: la configuración de cada nodo debe describirse como código versionable, de modo que incorporar un nuevo nodo o recuperar uno fallido sea un proceso determinista y no una intervención artesanal. Esta necesidad también se satisface dentro de la subcategoría **Automation & Configuration**, y su importancia en este contexto se deriva directamente del principio de consistencia de configuración establecido en la sección anterior.

El tercero es la **soberanía sobre los artefactos**: las imágenes de contenedor que ejecutan los servicios institucionales deben almacenarse en un registro interno controlado por la institución, con análisis de vulnerabilidades y firma criptográfica que garanticen que ningún artefacto no verificado llega al entorno de ejecución. El CNCF identifica este problema en la subcategoría **Container Registry**, señalando que sin un registro centralizado ningún runtime puede acceder a las imágenes de forma controlada [78]. En el contexto institucional, depender de registros públicos externos contradice adicionalmente el principio de soberanía tecnológica.

A estos tres problemas se añade un cuarto que el CNCF agrupa en la subcategoría **Security & Compliance**: la gestión centralizada de secretos y certificados. La guía describe esta necesidad señalando que los entornos cloud native son altamente dinámicos y requieren distribución de secretos bajo demanda de forma completamente programática, sin intervención humana [78]. En un entorno multi-zona esta necesidad se amplifica: las credenciales tienden a proliferar en archivos de configuración y variables de entorno cuando no existe un almacén centralizado, introduciendo riesgos de exposición que contradicen el principio de seguridad de la arquitectura.

2.3.2 Runtime: ejecutar con garantías

El CNCF [78] describe esta capa como todo lo que un contenedor necesita para ejecutarse en un entorno cloud native, identificando tres subcategorías: el código para iniciarlo, herramientas para hacer disponible almacenamiento persistente, y herramientas que gestionan las redes del entorno de contenedores.

Aplicadas al contexto de un centro de datos distribuido multi-campus, estas tres subcategorías resuelven problemas que tienen implicaciones específicas más allá de la definición general del CNCF.

El **almacenamiento distribuido** es el componente más crítico de esta capa en el contexto propuesto. La guía señala que las arquitecturas cloud native son fluidas y elásticas, y que los contenedores se crean y destruyen continuamente cambiando su ubicación física, por lo que el almacenamiento debe proveerse de forma independiente al nodo [78]. En un entorno multi-campus, esta independencia del nodo tiene además una dimensión geográfica: el almacenamiento distribuido es el mecanismo que garantiza la resiliencia de datos ante la pérdida de una zona completa mediante replicación entre dominios de fallo.

El **runtime de contenedores** es descrito por el CNCF como el componente que ejecuta la aplicación dentro del contenedor, establece los límites de seguridad y asigna recursos como CPU, almacenamiento y memoria [78]. En el contexto de este trabajo, su estandarización mediante la Container Runtime Interface (CRI) es lo que permite que el orquestador de la capa superior sea independiente del mecanismo de ejecución subyacente, satisfaciendo el principio de apertura.

La **red de pods** es descrita por el CNCF como el mecanismo que crea una red virtual sobre la red física existente, denominada overlay network, para que los componentes de las aplicaciones modernas puedan comunicarse entre sí de forma privada [78]. En una arquitectura distribuida donde los pods pueden ubicarse en cualquier nodo del clúster, esta red debe operar de forma transparente sobre la infraestructura de red física subyacente, independientemente de si esa infraestructura proviene del hipervisor o del hardware bare metal.

2.3.3 Orchestration & Management: gobernar el sistema

El CNCF [78] describe esta capa señalando que en las arquitecturas cloud native las aplicaciones se descomponen en decenas o cientos de microservicios, y que gestionar manualmente el ciclo de vida de todos esos contenedores no es viable a escala, por lo que se necesitan procesos automatizados. La guía describe además cómo el orquestador compara continuamente el estado actual del clúster contra el estado deseado y actúa para corregir cualquier diferencia.

En el contexto de este trabajo, esta capa resuelve dos problemas que emergen directamente de la arquitectura multi-zona propuesta. El primero es la gestión del ciclo de vida de las cargas de trabajo: garantizar que el estado deseado del sistema se mantenga en todo momento, incluyendo la redistribución automática de pods ante la caída de un nodo o de una zona completa. El segundo es la exposición controlada de servicios: los servicios que corren dentro

del clúster deben ser accesibles desde el exterior de forma controlada, con terminación TLS y enrutamiento basado en reglas, sin exponer directamente los nodos al exterior.

2.3.4 Observability & Analysis: conocer el estado del sistema

El CNCF [78] define la observabilidad como una característica del sistema que describe el grado en que puede comprenderse a partir de sus salidas externas, señalando que operar sistemas cloud native a escala requiere que operadores y desarrolladores tengan acceso a telemetría de alta calidad.

La interpretación aplicada a este trabajo añade una dimensión que la guía describe en términos generales pero que en el contexto multi-zona adquiere especial relevancia: la observabilidad debe ser unificada. Cada zona puede operar sus propios recolectores de métricas y logs de forma autónoma ante pérdida de conectividad, satisfaciendo el principio de autonomía operativa; pero el ente gestor central necesita una vista consolidada de toda la Region para tomar decisiones de capacidad y detectar problemas inter-zona que ninguna zona individual podría identificar por sí sola.

A diferencia de las capas anteriores, Observability & Analysis no es una capa sobre la que se construye la siguiente: es una capa transversal que instrumenta a todas las demás simultáneamente. Esta propiedad es reconocida por el CNCF al definirla como una columna paralela a las capas, no como una capa más de la jerarquía [78].

A partir de la jerarquía de capas se construyó el siguiente diagrama para mostrar la protoarquitectura:

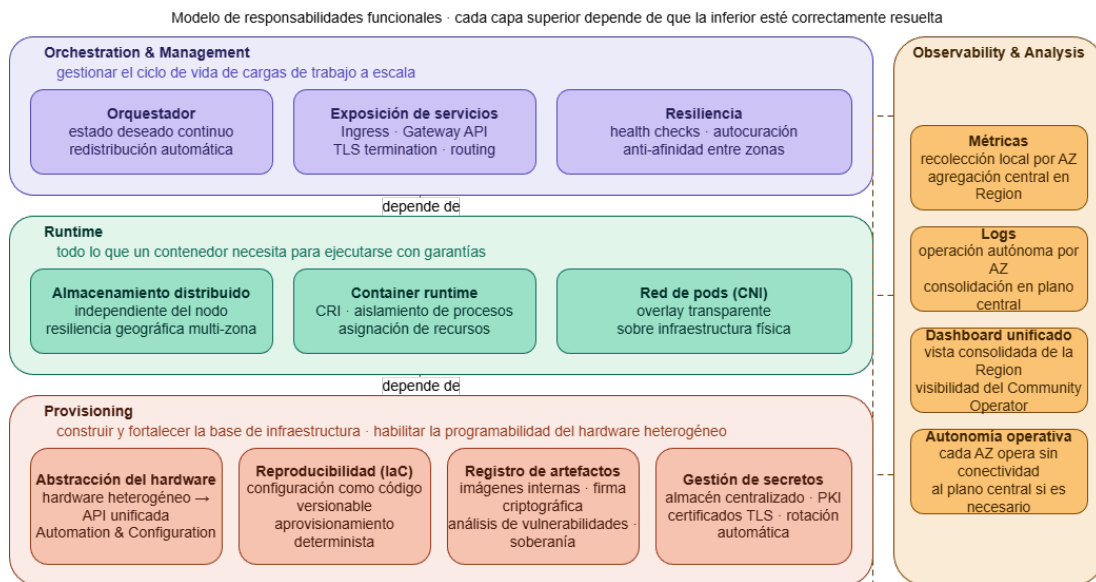


Fig. 8. Protoarquitectura funcional

2.4 Selección del stack

La protoarquitectura definida en la sección anterior establece qué debe resolver cada capa. Lo que sigue identifica las tecnologías concretas que cubren cada responsabilidad dentro del ecosistema CNCF, justificando su selección en términos de madurez, licencia y adecuación al contexto institucional.

2.4.1 Provisioning: Automation & Configuration

OpenStack es la plataforma de nube privada de referencia en entornos institucionales. Implementa los servicios fundamentales de IaaS, es decir, gestión de máquinas virtuales (Nova), redes virtuales (Neutron), almacenamiento en bloque (Cinder), aprovisionamiento bare metal (Ironic) e identidad (Keystone); exponiendo todo mediante una API unificada que abstrae la heterogeneidad del hardware subyacente [80], [81]. Su modelo de gobernanza abierta bajo la OpenInfra Foundation garantiza independencia respecto a proveedores comerciales.

OpenTofu describe el estado deseado de la infraestructura en archivos declarativos versionables y ejecuta los cambios necesarios para alcanzarlo. Es el sucesor open source de Terraform, mantenido bajo la Linux Foundation con licencia MPL 2.0, surgido tras el cambio de licencia de HashiCorp en 2023 [82].

Ansible configura los nodos una vez aprovisionados: instala dependencias, aplica políticas de seguridad y configura servicios del sistema operativo. Opera mediante SSH sin requerir agentes instalados, lo que lo hace adecuado para el hardware heterogéneo del entorno institucional [83].

2.4.2 Provisioning: Container Registry

Harbor es el único registro de contenedores con estado de proyecto graduado en la CNCF, distinción alcanzada en junio de 2020 [84]. Provee control de acceso basado en roles, análisis de vulnerabilidades mediante Trivy y firma de imágenes mediante Cosign [85]. Su capacidad de replicación entre instancias permite sincronizar imágenes entre zonas geográficamente separadas sin depender de conectividad externa [85].

2.4.3 Provisioning: Security & Compliance

OpenBao es un gestor de secretos de código abierto derivado de HashiCorp Vault, mantenido bajo la Linux Foundation con licencia MPL 2.0. Provee una API centralizada para el almacenamiento, rotación y distribución de secretos bajo demanda, con control de acceso granular por política y registro inmutable de cada acceso [68]. Actúa adicionalmente como autoridad certificadora raíz de la nube comunitaria. Es el único componente del stack ausente del CNCF Landscape¹¹, por lo que su selección no se sostiene en su pertenencia al catálogo sino en su licencia y en su gobernanza abierta bajo la Linux Foundation.

Por otra parte, **cert-manager** es un operador de Kubernetes graduado en la CNCF en noviembre de 2024 [86] que automatiza el ciclo de vida completo de los certificados TLS. Se integra con OpenBao como backend de CA, garantizando que todos los endpoints del clúster operen con certificados válidos y renovados automáticamente [87].

2.4.4 Runtime: Cloud Native Storage

Ceph es el sistema de almacenamiento distribuido de referencia en entornos de nube privada [88]. Su algoritmo CRUSH distribuye los datos entre los nodos del clúster sin un punto único de fallo [89] y expone tres interfaces complementarias: RBD (bloque), CephFS (sistema de archivos POSIX) y RGW (objeto compatible con la API S3) [90]. Esta triple interfaz le permite conectarse simultáneamente a OpenStack y a Kubernetes sin duplicar infraestructura.

¹¹Su antecesor, HashiCorp Vault, sí figura en la subcategoría Key Management [78].

2.4.5 Runtime: Container Runtime

Se identifican dos candidatos con estado de proyecto graduado en la CNCF, ambos compatibles con el estándar OCI [52] y con la Container Runtime Interface (CRI) que Kubernetes requiere [91].

Containerd es el runtime estándar de la industria, graduado en la CNCF desde febrero de 2019 [92]. Opera con un requisito mínimo de kernel Linux 4.x y es el runtime por defecto en las distribuciones de Kubernetes más utilizadas.

CRI-O fue diseñado específicamente para Kubernetes, graduado en la CNCF en julio de 2023 [93]. Implementa únicamente la superficie necesaria para servir como runtime de Kubernetes, reduciendo la superficie de ataque y las dependencias del sistema.

2.4.6 Runtime: Cloud Native Networkm, Orchestration, Scheduling & Orchestration

Cilium es proyecto graduado de la CNCF desde octubre de 2023 [94]. Implementa la red de pods mediante eBPF [95], operando directamente en el kernel de Linux sin requerir modificaciones en la configuración de red del hipervisor. En el stack propuesto cumple dos roles: como CNI gestiona la red de pods en todos los nodos del clúster; como implementación de la Gateway API de Kubernetes [96] gestiona el tráfico entrante desde la red externa, eliminando la necesidad de un controlador de Ingress¹² separado.

Kubernetes es el orquestador de contenedores de referencia y el primer proyecto graduado de la CNCF [97]. Implementa el modelo de reconciliación de estado deseado y constituye la capa CaaS de la arquitectura propuesta [98]. Opera sobre VMs aprovisionadas por Nova, consumiendo los recursos de la capa IaaS de OpenStack y presentando a las aplicaciones un clúster unificado independientemente de la distribución física de los nodos.

2.4.7 Observability & Analysis

Prometheus es el sistema de monitoreo y alerta de referencia en el ecosistema cloud native, graduado en la CNCF desde agosto de 2018 [99]. Recolecta métricas de nodos, servicios y

¹²Ingress: objeto de la API de Kubernetes que define las reglas de enrutamiento del tráfico HTTP/HTTPS proveniente del exterior hacia los servicios del clúster. Por sí mismo es solo una declaración de reglas: requiere un controlador que las implemente efectivamente sobre el tráfico de red.

componentes de Kubernetes, OpenStack y Ceph mediante un modelo de scraping pull sobre sus endpoints nativos.

Loki agrega los logs del clúster indexando únicamente etiquetas por flujo en lugar del contenido completo, reduciendo significativamente los recursos de almacenamiento respecto a soluciones como Elasticsearch [100].

Grafana actúa como plano de visualización unificado, consumiendo simultáneamente Prometheus y Loki como fuentes de datos y permitiendo correlacionar métricas y logs en una única interfaz. Su licencia AGPLv3 garantiza costo de licenciamiento nulo para uso institucional interno.

2.5 Evaluación de interoperabilidad y coherencia del stack

La viabilidad de un stack tecnológico no se determina únicamente por las capacidades individuales de cada componente, sino por la existencia de interfaces estandarizadas que permitan su integración sin desarrollo adicional. Un conjunto de herramientas cuyas interfaces son incompatibles o requieren adaptadores propietarios introduce fragilidad arquitectónica y dependencias que contradicen el principio de apertura establecido en este capítulo. El análisis que sigue demuestra que las tecnologías candidatas seleccionadas forman un sistema integrado cuyas conexiones están definidas por estándares abiertos.

2.5.1 Capa de infraestructura: OpenStack y Ceph

La integración entre OpenStack y Ceph es la más documentada del ecosistema de nube privada y constituye el fundamento de la capa IaaS de esta propuesta [88]. Ceph expone sus tres interfaces directamente a los servicios de OpenStack mediante controladores nativos [80], [90]: el servicio de almacenamiento en bloque (Cinder) consume volúmenes RBD, el servicio de imágenes (Glance) almacena las imágenes de máquinas virtuales en el pool RBD, el servicio de cómputo (Nova) monta los discos raíz de las instancias directamente desde RBD sin pasar por el almacenamiento local del hipervisor, y el servicio de sistema de archivos compartidos (Manila) expone CephFS a las instancias. Esta integración elimina la dependencia del almacenamiento local en cada nodo de cómputo, lo que es precisamente la condición que hace posible la migración en vivo de máquinas virtuales entre nodos y la recuperación automática ante fallos.

OpenTofu gestiona el aprovisionamiento de todos estos recursos sobre la API de OpenStack de forma declarativa, mientras que Ansible configura los nodos físicos antes de que sean incorporados al clúster, instalando los paquetes de OpenStack, los clientes de Ceph y las dependencias del sistema operativo. Ironic se integra con Nova para incorporar los nodos bare metal al mismo pool de recursos que las máquinas virtuales, sin requerir modificaciones en el resto del stack.

2.5.2 Capa de runtime: containerd, CRI-O y Kubernetes

Kubernetes se comunica con el runtime de contenedores exclusivamente a través de la Container Runtime Interface (CRI), una API gRPC estandarizada que desacopla el orquestador del runtime subyacente [91]. Tanto containerd como CRI-O implementan esta interfaz, lo que significa que la elección entre uno y otro no modifica ningún otro componente del stack.

2.5.3 Almacenamiento persistente para contenedores: Ceph CSI

Kubernetes gestiona el almacenamiento persistente para contenedores a través de la Container Storage Interface (CSI) [101]. El proyecto Ceph mantiene un driver CSI oficial que permite a Kubernetes aprovisionar volúmenes RBD y sistemas de archivos CephFS de forma dinámica, sin intervención manual [102]. Esta integración cierra el ciclo: el mismo clúster Ceph que sirve almacenamiento a las máquinas virtuales de OpenStack sirve también almacenamiento persistente a los pods de Kubernetes, eliminando la necesidad de una solución de almacenamiento separada para cada capa.

2.5.4 Registro de imágenes: Harbor y Kubernetes

Harbor se integra con Kubernetes en dos niveles siguiendo la OCI Distribution Specification [103]. A nivel de autenticación, Kubernetes extrae imágenes de Harbor utilizando secretos de tipo `imagePullSecret`, de modo que solo los pods autorizados puedan acceder a los repositorios privados. A nivel de seguridad, Harbor puede configurarse para rechazar imágenes que no hayan superado el análisis de vulnerabilidades de Trivy o que no estén firmadas mediante Cosign [85]. Adicionalmente, Harbor almacena charts de Helm, convirtiéndose en el repositorio central de artefactos del clúster.

2.5.5 Cadena de seguridad: OpenBao, cert-manager y Cilium

Los tres componentes de seguridad forman una cadena de dependencias. OpenBao actúa como CA raíz y almacén de secretos [68]; cert-manager consume la API PKI de OpenBao para emitir y renovar automáticamente los certificados TLS de todos los servicios del clúster [87]; Cilium Gateway API termina el TLS entrante con los certificados provisionados por cert-manager [96]. Esta cadena garantiza que todo el tráfico interno y externo del clúster viaje cifrado con certificados emitidos por la CA institucional, sin intervención manual en ningún punto del ciclo de vida de los certificados.

2.5.6 Observabilidad: Prometheus, Loki y Grafana

El stack de observabilidad se integra con todos los componentes anteriores sin requerir modificaciones en ellos. Prometheus recolecta métricas de Kubernetes mediante el endpoint `/metrics` que exponen nativamente el API server, el scheduler, el controller manager y el kubelet; de OpenStack mediante exporters específicos por servicio; y de Ceph mediante el Ceph Manager Module de Prometheus. Loki recolecta los logs de todos los pods del clúster mediante Grafana Alloy, desplegado como DaemonSet en cada nodo. Grafana actúa como plano de visualización unificado, consumiendo simultáneamente Prometheus y Loki como fuentes de datos.

TABLA 8

INTERFACES DE INTEGRACIÓN ENTRE COMPONENTES DEL STACK.

Componente origen	Componente destino	Interfaz / Protocolo	Estándar
OpenTofu	OpenStack	API REST	OpenStack API
Ansible	Nodos físicos	SSH + módulos	—
IroniC	Nova	API REST	OpenStack API
OpenStack Nova	Ceph RBD	librbd	RADOS
OpenStack Cinder	Ceph RBD	Cinder volume driver	RADOS
OpenStack Glance	Ceph RBD	Glance store driver	RADOS
OpenStack Manila	CephFS	Manila share driver	POSIX / CephFS
Kubernetes	containerd / CRI-O	gRPC	CRI (OCI)
Kubernetes	Ceph	CSI driver	CSI v1
Kubernetes	Harbor	HTTPS	OCI Distribution Spec
cert-manager	OpenBao	API REST	PKI secrets engine
Cilium Gateway API	cert-manager	Kubernetes API	Gateway API
Prometheus	Kubernetes	HTTP scrape	/metrics endpoint
Prometheus	Ceph	HTTP scrape	Ceph MGR Prometheus
Prometheus	OpenStack	HTTP scrape	OpenStack exporters
Loki	Pods	Grafana Alloy DaemonSet	—
Grafana	Prometheus	HTTP / PromQL	—
Grafana	Loki	HTTP / LogQL	—

2.6 Justificación técnica de la selección final

La selección del stack tecnológico presentada en las secciones anteriores no constituye una enumeración de herramientas populares, sino la consecuencia directa de los principios arquitectónicos definidos en la sección anterior. El presente apartado establece esa correspondencia de forma explícita, demostrando que cada principio encuentra satisfacción concreta en uno o más componentes del stack y que ningún principio queda sin respuesta tecnológica.

2.6.1 Economía de escala

OpenStack materializa este principio al transformar el hardware heterogéneo existente en un pool unificado de recursos accesible mediante API, eliminando la subutilización característica de los servidores aislados. Ceph complementa este argumento en la dimensión del almacenamiento: al consolidar los discos de múltiples nodos en un único clúster distribuido, el costo por gigabyte se reduce respecto a mantener soluciones de almacenamiento independientes por área. Kubernetes extiende la misma lógica a las cargas de trabajo: al compartir los recursos del clúster entre múltiples servicios mediante bin packing, incrementa la densidad de utilización de cada nodo.

2.6.2 Soberanía tecnológica y apertura

Ambos principios convergen en un mismo criterio de selección: licencias libres y gobernanza abierta. OpenStack opera bajo la OpenInfra Foundation con licencia Apache 2.0 [80]. OpenTofu mantiene la licencia MPL 2.0 bajo la Linux Foundation [82]. Ansible se distribuye bajo GPL v3 [83]. Ceph bajo LGPL 2.1 [90]. Harbor, containerd, CRI-O, Cilium, Kubernetes y cert-manager bajo Apache 2.0 [79]. OpenBao bajo MPL 2.0 [68]. Prometheus bajo Apache 2.0. Grafana y Loki bajo AGPL v3 [100]. Ningún componente del stack impone restricciones de licenciamiento para uso institucional interno, y todos cuentan con gobernanza documentada y comunidades activas que garantizan continuidad independientemente de decisiones comerciales de un proveedor específico.

2.6.3 Modularidad y expansión

En la capa IaaS, OpenStack permite agregar nodos de cómputo al clúster sin interrumpir los servicios existentes; IroniC extiende esta capacidad al hardware bare metal. En la capa de almacenamiento, Ceph incorpora nuevos OSDs y redistribuye los datos automáticamente mediante el algoritmo CRUSH [89]. En la capa CaaS, Kubernetes incorpora nuevos nodos worker

mediante un único comando de adhesión al clúster. La estandarización de estas interfaces implica que la expansión futura hacia cómputo de alto rendimiento o inferencia de modelos de lenguaje no requiere sustituir ningún componente, sino agregar nodos especializados al clúster existente.

2.6.4 Resiliencia de datos y confiabilidad

Ceph implementa replicación configurable: por defecto mantiene tres réplicas de cada objeto distribuidas entre nodos y dominios de fallo distintos [90], [104], de modo que la pérdida de un nodo completo no provoca pérdida de datos ni interrupción del servicio de almacenamiento. OpenStack implementa alta disponibilidad en el plano de control mediante instancias redundantes de sus servicios críticos. Kubernetes reconcilia continuamente el estado deseado del clúster: ante la caída de un nodo, reprograma automáticamente los pods afectados en los nodos disponibles. La combinación de estos tres mecanismos satisface el principio de confiabilidad descrito por Van Steen y Tanenbaum [20] como la capacidad del sistema de seguir operando correctamente ante fallos parciales.

2.6.5 Escalabilidad

Kubernetes opera sobre VMs aprovisionadas por Nova, cuyo tamaño se define mediante flavors que permiten asignar recursos según las capacidades reales de cada nodo físico [105]. Ceph tolera discos de capacidad y velocidad dispares mediante el sistema de pesos por OSD [89]. Ironic incorpora hardware bare metal de cualquier generación al mismo pool de recursos [80]. containerd y CRI-O operan con un requisito mínimo de kernel Linux 4.x, compatible con hardware de generaciones anteriores. Cilium requiere kernel ≥ 4.9 [106], igualmente alcanzable en la mayoría del hardware institucional existente.

2.6.6 Seguridad

Harbor analiza vulnerabilidades mediante Trivy en cada imagen almacenada y puede configurarse para rechazar imágenes no firmadas mediante Cosign [85]. OpenStack centraliza la autenticación en Keystone [107]. Kubernetes implementa RBAC sobre todos sus recursos [98]. OpenBao centraliza el almacenamiento de secretos con registro inmutable de accesos y actúa como CA raíz [68]. cert-manager automatiza la renovación de certificados eliminando el

riesgo de expiración [87]. Cilium aplica políticas de red a nivel de kernel mediante eBPF [95], reduciendo la superficie de ataque respecto a soluciones basadas en iptables.

2.6.7 Observabilidad unificada y consistencia de configuración

Prometheus recolecta métricas de Kubernetes, OpenStack y Ceph mediante sus endpoints nativos; Loki agrega los logs de todos los pods del clúster; Grafana los correlaciona en una interfaz única. La consistencia de configuración está satisfecha por OpenTofu y Ansible: toda la infraestructura se describe como código versionable en repositorios Git, garantizando que el estado del sistema sea reproducible, auditable y recuperable ante cualquier fallo de configuración.

2.6.8 Autonomía federada, autoservicio, identidad federada y trazabilidad

OpenStack satisface la autonomía federada mediante su modelo de Domains y Projects: cada entidad de la federación recibe un Domain propio con cuotas configurables centralmente, dentro del cual administra sus recursos de forma autónoma [80]. Kubernetes extiende esta lógica mediante namespaces con ResourceQuotas y LimitRanges [108]. Keystone federado contra el directorio institucional mediante SAML/OIDC implementa la gestión de identidad federada [66], [67], [107], proveyendo un único punto de autenticación para toda la comunidad universitaria. OpenBao registra de forma inmutable cada acceso a secretos, y la consistencia de configuración vía OpenTofu y Ansible garantiza que todo cambio de infraestructura sea rastreable en el historial de versiones del repositorio.

TABLA 9

CORRESPONDENCIA ENTRE PRINCIPIOS ARQUITECTÓNICOS Y COMPONENTES DEL STACK SELECCIONADO.

Principio arquitectónico	Componentes que lo satisfacen
Economía de escala	OpenStack, Ceph, Kubernetes
Soberanía tecnológica · Apertura	Todo el stack (licencias libres · gobernanza abierta)
Modularidad y expansión	OpenStack, Ironic, Ceph (CRUSH), Kubernetes
Resiliencia de datos · Confiabilidad	Ceph (replicación ×3), OpenStack (HA), Kubernetes (reconciliación)
Escalabilidad	OpenStack (flavors), Ironic (bare metal), Ceph (pesos OSD), containerd/CRI-O
Seguridad	Harbor (Trivy · Cosign), Keystone (RBAC), OpenBao, cert-manager, Cilium (eBPF)
Observabilidad unificada	Prometheus, Loki, Grafana
Consistencia de configuración	OpenTofu, Ansible
Autonomía federada · Autoservicio	OpenStack (Domains · Projects), Kubernetes (namespaces · ResourceQuota)
Identidad federada · Trazabilidad	Keystone (SAML/OIDC), OpenBao (registro inmutable)

CAPÍTULO

III

Arquitectura del datacenter

RESUMEN

Este capítulo presenta la arquitectura de referencia del centro de datos distribuido propuesto. Siguiendo el C4 Model como marco de descripción, la arquitectura se desarrolla en tres niveles de abstracción progresivos: el nivel de Contexto establece el modelo de gobierno y la organización lógica de la nube comunitaria; el nivel de Contenedores describe la topología interna de cada zona de disponibilidad, sus planos de red y los requisitos de hardware por rol de nodo; y el nivel de Componentes detalla el apilamiento tecnológico dentro de cada tipo de nodo. Adicionalmente, dos elementos transversales se describen como prerequisites ineludibles que aplican a todos los niveles de la arquitectura simultáneamente.

CONCEPTOS CLAVE

Modelo C4

Marco de descripción arquitectónica utilizado para documentar el sistema en tres niveles de abstracción progresivos, adaptando su semántica a infraestructura.

Sitios de Control

Zonas de Disponibilidad (AZ) designadas estratégicamente para alojar el plano de control global y mantener el quórum de la base de datos distribuida, separándolas de las zonas exclusivas de ejecución.

Topología de Nodos

Separación estricta de responsabilidades a nivel de hardware dentro de una AZ, clasificando los equipos físicos en nodos de control, cómputo, almacenamiento y legacy.

Pila Tecnológica Superpuesta

Integración técnica a nivel de sistema operativo donde coexisten la capa de virtualización orquestación, redes y almacenamiento.

3.1 Marco de descripción arquitectónica

La descripción de la arquitectura propuesta en este capítulo se basa en el C4 Model [109], marco de descripción arquitectónica desarrollado por Simon Brown que organiza la documentación de sistemas en niveles de abstracción progresivamente más detallados. Si bien la propuesta no es un sistema de software puro, se considera pertinente el uso de estos niveles de abstracción para transmitir de manera efectiva la arquitectura propuesta dado que el C4 Model es aplicable a cualquier sistema cuya complejidad requiera mostrar diferentes niveles de abstracción a distintas audiencias.

El **nivel de Contexto** describe el sistema en su entorno organizacional: quiénes son sus usuarios, qué sistemas externos interactúan con él y cuál es su propósito general. Este nivel se aborda en este capítulo a través del modelo de gobierno de la nube comunitaria y a la jerarquía que define cómo se organiza la infraestructura entre las entidades participantes.

El **nivel de Contenedores**¹³ describe cómo se divide el sistema en bloques funcionales que se comunican entre sí. El análisis de este nivel dentro de este capítulo corresponde a la topología interna de una Availability Zone: los roles de nodo, los planos de red y la distribución de tecnologías entre ellos, incluyendo los requisitos de hardware de cada unidad de despliegue.

El **nivel de Componentes** describe el interior de cada bloque funcional. La descripción de estos componentes corresponde al apilamiento tecnológico dentro de cada tipo de nodo y las relaciones que establecen entre sí.

El cuarto nivel del C4, denominado Código, no se aborda en este trabajo, ya que la propuesta es una arquitectura de referencia y no una implementación que requiera describir la configuración de cada componente.

Adicionalmente, dos elementos son transversales a todos los niveles: la **topología de red**, que establece los prerequisites de conectividad sobre los que opera cada nivel, y la **cadena de seguridad e identidad**, que protege las comunicaciones en todos los niveles simultáneamente. Ambos se describen antes de descender al detalle de cada nivel C4.

¹³En la terminología del C4, unidades de ejecución o despliegue, no contenedores en el sentido de Docker o Kubernetes.

TABLA 10

CORRESPONDENCIA ENTRE LOS NIVELES DEL C4 MODEL Y LAS SECCIONES DEL CAPÍTULO 3.

Nivel C4	Pregunta que responde	Correspondencia en este capítulo	Audiencia
Contexto	¿Qué es el sistema y quién lo usa?	Modelo de gobierno · jerarquía Region- Domain-AZ · organización lógica	Dirección institucional · Community Operator
Contenedores	¿Cómo se divide el sistema en bloques funcionales?	Topología interna de la AZ · roles de nodo · planos de red · requisitos de hard- ware	Infrastructure Operators · administrado- res
Componentes	¿Qué hay dentro de cada bloque?	Pila tecnológica dentro de cada tipo de nodo	Ingenieros de sistemas
Código	¿Cómo está implementado?	Fuera del alcance de este trabajo	Desarrolladores
Transversal	¿Qué infraestructura y seguridad sostiene todo?	Topología de red · cadena de seguridad e identidad	Todos los perfiles

3.2 Elementos transversales

Establecen los prerequisites de conectividad e identidad sobre los que operan todos los niveles.

3.2.1 Topología de red

La arquitectura requiere una infraestructura de red que satisfaga requisitos específicos de segmentación, aislamiento y conectividad entre zonas. Esta sección describe la topología lógica esperada como prerequisite; el diseño físico detallado constituye un trabajo complementario fuera del alcance de esta tesis.

3.2.1.1 Topología interna de una AZ

Dentro de cada AZ la red se organiza siguiendo la topología **Spine-Leaf**, variante moderna de la arquitectura Clos [61] adoptada como estándar de facto para centros de datos cloud native [60]. Sus fundamentos se describen en la sección redes del Capítulo 1. Esta topología garantiza latencia uniforme entre cualquier par de nodos, condición crítica para Ceph cuyo rendimiento es sensible a la varianza de latencia entre nodos OSD.

La **capa Spine** se compone de switches de núcleo desplegados preferentemente en par redundante. El gateway de borde se conecta directamente a ambos spines garantizando que el fallo de uno no interrumpa la conectividad exterior de la AZ. La **capa Leaf** conecta los nodos físicos mediante bonding activo-pasivo en al menos dos interfaces.

Sobre esta fabric física se implementan los cuatro planos de red mediante overlays VXLAN [63].

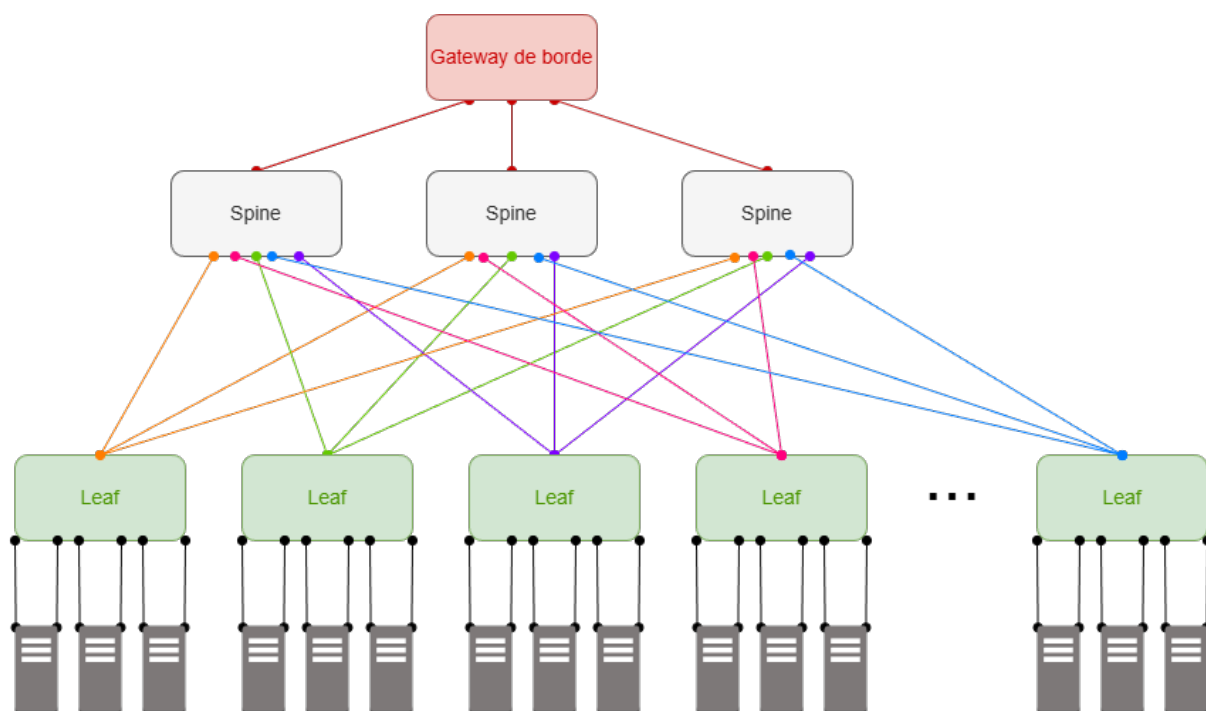


Fig. 9. Topología Spine-Leaf

3.2.1.2 Planos de red

La comunicación entre los distintos roles de nodo se segmenta en cuatro planos de red implementados como overlays VXLAN sobre la fabric Spine-Leaf. Esta segmentación cumple dos objetivos simultáneos: aislar el tráfico por función para optimizar el rendimiento, y reducir la superficie de ataque al limitar la visibilidad entre planos. Cada plano se identifica mediante un VNID distinto [63]; los cuatro planos de la arquitectura propuesta son:

La **red de gestión** transporta el tráfico de control entre todos los servicios de OpenStack, el plano de control de Kubernetes y los agentes de cada nodo (como llamadas a la API, mensajes de RabbitMQ, consultas a la base de datos, comunicación kubelet-API server, tráfico de OpenTofu y Ansible, etc.). Todos los nodos tienen presencia en esta red.

La **red de almacenamiento** transporta exclusivamente el tráfico entre nodos de cómputo y almacenamiento: acceso RBD desde Nova para los discos raíz de las VMs, acceso RBD desde el CSI driver de Kubernetes para los volúmenes persistentes de pods, acceso CephFS para sistemas de archivos compartidos, y tráfico de replicación entre OSDs. Su aislamiento evita que la replicación de Ceph degrade el rendimiento de las cargas de trabajo.

La **red de tenant** transporta el tráfico entre las VMs de los usuarios. El aislamiento por Domain y Project se implementa en dos niveles: VXLAN segmenta el plano de tenant del resto de los planos, mientras que Neutron mediante OVN crea switches lógicos independientes para cada Project dentro del mismo plano [110]. Dos Projects no pueden comunicarse a menos que el Community Operator o el Domain Owner configuren explícitamente una ruta entre sus routers lógicos. Sobre la interfaz de tenant de cada VM, Cilium implementa un tercer nivel de overlay para la red de pods mediante eBPF [95], extendiendo el aislamiento hasta el nivel de pod individual.

La **red externa** conecta la AZ con el exterior: otras AZs, internet y servicios institucionales. Todo el tráfico externo pasa por el gateway de borde, único punto de la AZ con visibilidad simultánea hacia el interior y el exterior.

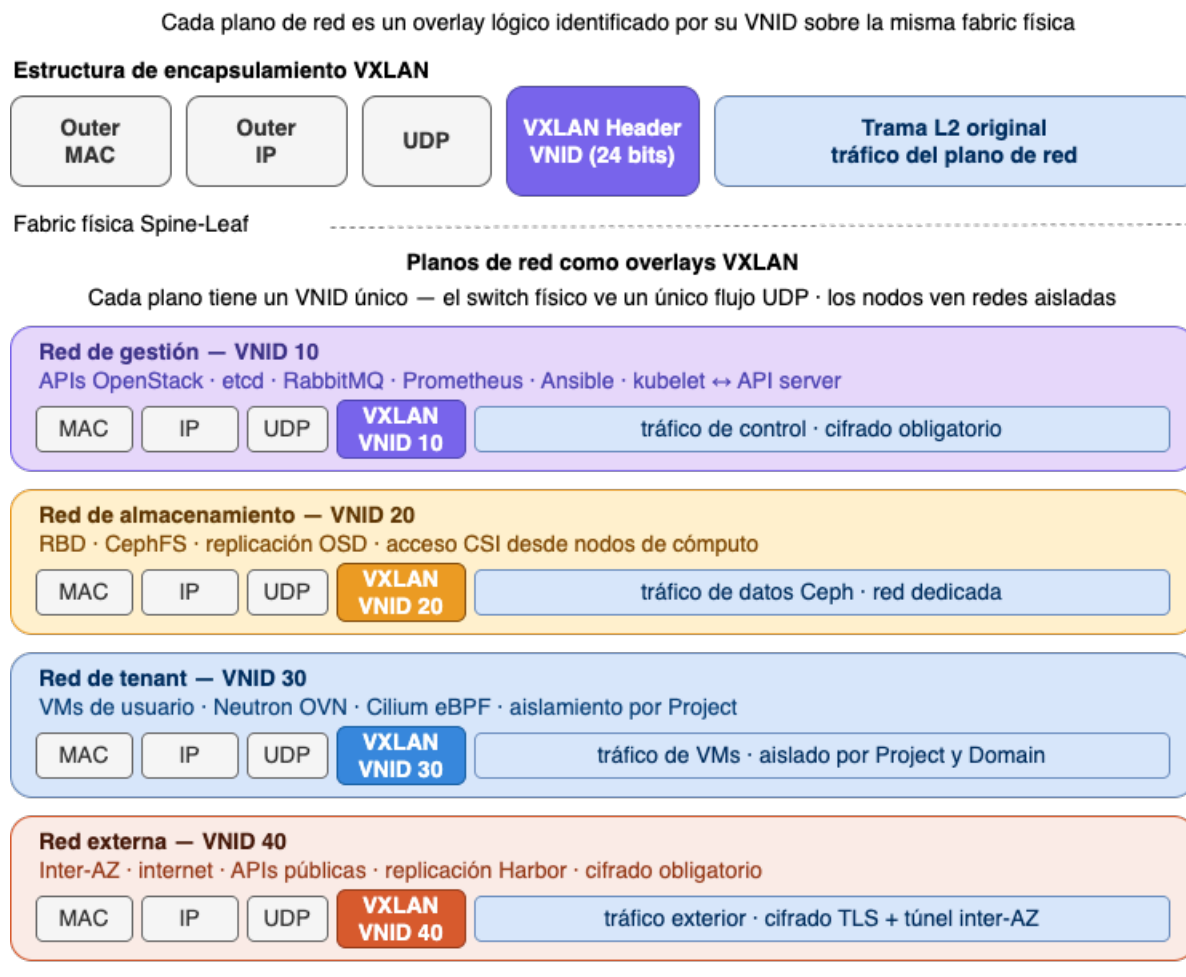


Fig. 10. Encapsulamiento VXLAN por plano de red

TABLA 11

PROPIEDADES REQUERIDAS POR CADA PLANO DE RED.

Plano de red	Alcance	Propiedad crítica	Cifrado requerido
Gestión	Intra-AZ	Baja latencia · alta disponibilidad	Sí
Almacenamiento	Intra-AZ	Alto ancho de banda · baja latencia	Opcional (red dedicada)
Tenant	Intra-AZ	Aislamiento entre Projects	Por política de tenant
Externa	Inter-AZ + internet	Tolerancia a latencia variable	Sí (obligatorio)

3.2.1.3 Conectividad entre AZs y flujos inter-zona

La interconexión entre AZs debe satisfacer tres propiedades. El primero es el **aislamiento del tráfico institucional**: el enlace debe separar el tráfico de la nube del tráfico de usuarios finales mediante un túnel cifrado entre los gateways de borde. Cuando la institución disponga de conectividad dedicada como un enlace MPLS [111], esta debe preferirse por su menor varianza de latencia. En ausencia de conectividad dedicada, un overlay cifrado mediante IPsec [112] o WireGuard [113] garantiza el aislamiento lógico necesario. El segundo es el **cifrado en tránsito** obligatorio para todo tráfico inter-zona. El tercero es la **tolerancia a latencia variable**, razón por la que se adoptó replicación asíncrona en Ceph y Harbor.

Tres tipos de tráfico atraviesan los enlaces entre AZs con comportamientos distintos respecto al plano de control. La **replicación de almacenamiento** opera de forma completamente autónoma: Ceph replica entre sus OSDs guiándose por el CRUSH Map sin requerir instrucción alguna del plano de control [89], [104]. La **replicación de imágenes** opera igualmente de forma autónoma mediante las políticas de Harbor [85]. El **tráfico del plano de control** es el único coordinado activamente. Esta distinción produce una asimetría de disponibilidad deliberada: las operaciones en curso continúan ante una caída del plano de control, mientras que el aprovisionamiento de nuevos recursos queda suspendido.

TABLA 12

ASIMETRÍA DE DISPONIBILIDAD ENTRE FLUJOS INTER-ZONA COORDINADOS Y AUTÓNOMOS.

Función	Coordinada por plano de control	Disponible si CP cae
Replicación Ceph entre AZs	No · autónoma (CRUSH Map)	✓
Replicación Harbor entre AZs	No · autónoma (políticas prev.)	✓
Operaciones locales de VMs en curso	No · gestionadas localmente	✓
Crear nuevas VMs en AZ remota	Sí · Nova scheduler	×
Distribuir nuevos pods entre AZs	Sí · K8s scheduler	×
Decisiones de scheduling global	Sí · plano de control central	×

3.2.2 Cadena de seguridad e identidad

La arquitectura incorpora cuatro componentes que forman una cadena de seguridad donde cada eslabón depende del anterior, gráficamente esta cadena de seguridad se observa en la Fig. 11.

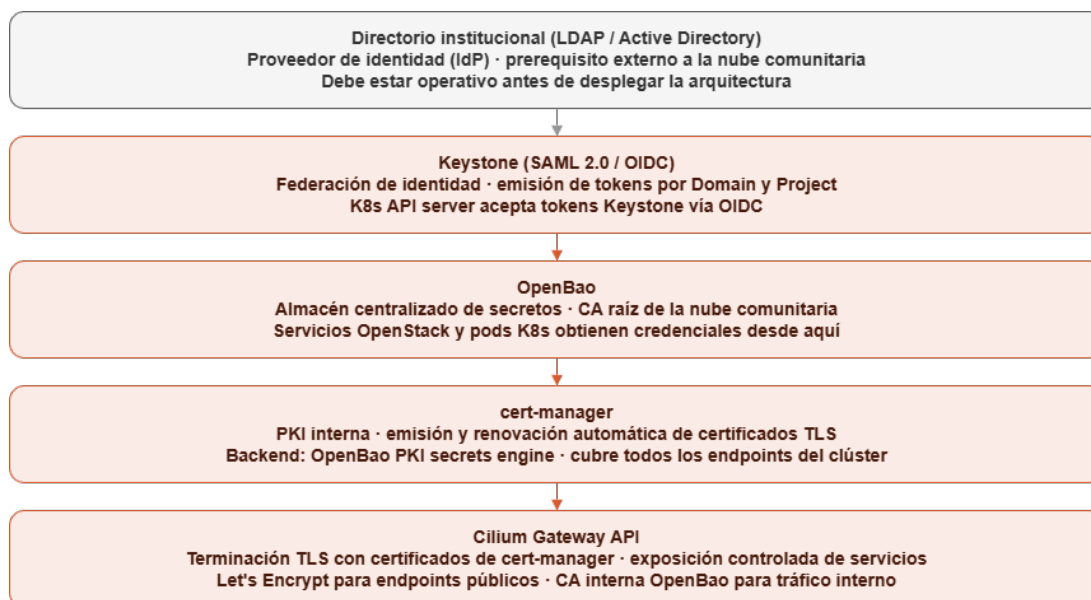


Fig. 11. Cadena de seguridad e identidad

El primer eslabón es **Keystone** configurado como proveedor de servicios (SP) federado contra el directorio institucional¹⁴, que actúa como proveedor de identidad (IdP) mediante SAML 2.0 [66] u OIDC [67]. Un usuario se autentica una única vez con sus credenciales universitarias y obtiene un token de Keystone con alcance limitado a su Domain y Project [107]. Kubernetes extiende este modelo mediante OIDC en su API server [98]. El directorio institucional debe estar operativo antes de desplegar la nube comunitaria; su ausencia no impide un despliegue inicial con usuarios locales en Keystone, pero sí impide satisfacer el principio de identidad federada.

El segundo eslabón es **OpenBao** como almacén centralizado de secretos y CA raíz de la nube comunitaria [68]. Los servicios de OpenStack obtienen credenciales desde OpenBao en el arranque sin que los valores queden en archivos de configuración en disco; los pods de Kubernetes inyectan secretos sin que queden almacenados en etcd.

El tercer eslabón es **cert-manager** [87], que gestiona dos categorías de certificados con cadenas de confianza distintas [65]. Los **certificados internos** (para la comunicación entre

¹⁴Típicamente LDAP [114] o Active Directory.

componentes del stack) son emitidos por la CA de OpenBao y son válidos únicamente dentro de la nube comunitaria. Los **certificados externos** (para los endpoints accesibles desde internet como Horizon, Harbor y Grafana) son emitidos por Let's Encrypt [115] mediante el protocolo ACME con reto DNS-01 [70], lo que requiere un dominio público institucional registrado y gestión programática de DNS.

TABLA 13

MODELO HÍBRIDO DE CERTIFICADOS: CA INTERNA PARA TRÁFICO INTERNO · LET'S ENCRYPT PARA ENDPOINTS PÚBLICOS.

Tipo de endpoint	Acceso	CA emisora	Confianza en navegadores
API server K8s · servicios OpenS-tack · métricas	Interna	OpenBao (CA raíz)	No requerida · tráfico interno
Horizon · Harbor · Grafana · APIs públicas	Internet	Let's Encrypt	Sí · sin advertencias

El cuarto eslabón es **Cilium Gateway API** [96], que termina las conexiones TLS [64] con certificados de cert-manager y enruta el tráfico hacia los servicios del clúster. Al concentrar la red de pods (CNI) y la exposición de servicios (Gateway API) en un único componente se reduce la superficie operativa.

3.3 Nivel de Contexto

Responde: ¿qué es el sistema, quién lo administra y bajo qué criterios?

3.3.1 Arquitectura de multitenencia y topología lógica

El primer nivel de abstracción de esta propuesta define la arquitectura de multitenencia y la topología lógica del centro de datos, que adopta el modelo de nube comunitaria como implementación concreta de un centro de datos distribuido. La selección de este modelo no es arbitraria: como se identificó en el análisis de caso del Capítulo 2, la UNAM articula su gobierno mediante un modelo federado en el que las facultades, escuelas e institutos mantienen autonomía administrativa sobre sus recursos mientras operan bajo un marco normativo común [72], [73]. Esta estructura es una condición de diseño fundamental: la nube comunitaria es estructuralmente isomorfa al modelo de gobierno universitario. Bajo este enfoque, la infraestructura no impone una nueva forma de organización sino que traduce la gobernanza institucional preexistente a una jerarquía de dominios de control y aislamiento tecnológico.

La topología lógica adoptada replica el consenso que los principales proveedores de nube han consolidado para operar sistemas distribuidos a escala global. La jerarquía **Region–Availability Zone–Nodo** emula el esquema de aislamiento de dominios de fallo y distribución geográfica utilizado por Amazon Web Services [75], Google Cloud [76] y Microsoft Azure [77].

A continuación se describen las unidades que componen esta jerarquía:

Region: representa la totalidad de la nube comunitaria y constituye el límite superior de gobernanza, administrado por un único ente gestor central. Técnicamente corresponde a una instalación de OpenStack con un plano de control unificado y un clúster de Kubernetes que orquesta la totalidad de los recursos federados.

Domain: representa a cada entidad participante y funciona como el primer nivel de multi-tenencia, administrado por un Domain Owner dentro de los límites definidos centralmente. Técnicamente equivale a un Domain de Keystone con gestión autónoma de usuarios y proyectos, y a un conjunto de namespaces de Kubernetes con cuotas de recursos asignadas.

Project: representa la unidad operativa final dentro de un Domain, proveyendo aislamiento operativo para cargas de trabajo específicas. Técnicamente corresponde a un Project de OpenStack con sus propias instancias, redes y volúmenes, y a namespaces específicos en la capa de orquestación.

Availability Zone (AZ): define la topología física y de resiliencia al agrupar nodos próximos bajo un mismo segmento de red troncal y autoridad administrativa. Funciona como el dominio de aislamiento ante fallos físicos locales y es operada por un Infrastructure Operator.

Nodo: es la unidad mínima de aportación física al pool de la Region. Los nodos se integran de dos formas: los servidores destinados a virtualización se gestionan mediante Nova-compute, mientras que los servidores físicos se incorporan mediante Ironic, presentándose como recursos computables indistinguibles de una instancia virtualizada.

En este sentido, la nube comunitaria propuesta materializa el paradigma de **Infraestructura Hiperconvergente (HCI)** definido en el Capítulo 1. Al consolidar el procesamiento (OpenStack y Kubernetes), el almacenamiento distribuido (Ceph) y las redes definidas por software (Neutron y Cilium) sobre nodos de hardware de propósito general aportados por las entidades participantes, la arquitectura elimina los silos de infraestructura tradicional. Esta hiperconver-

gencia es la característica estructural que permite abstraer la heterogeneidad del hardware universitario y gestionar toda la capacidad de la UNAM como un pool unificado, elástico y orquestado mediante una única capa de control por software.

3.3.2 Modelo de gobierno

Una infraestructura compartida entre entidades con autonomía administrativa propia requiere un modelo de gobierno que establezca quién toma qué decisiones, sobre qué recursos y bajo qué criterios. La traducción técnica del modelo de gobierno federado identificado en el análisis de caso [73] da origen a los siguientes roles:

El **Community Operator** es el ente gestor central con autoridad sobre la infraestructura compartida e independencia respecto a cualquiera de las entidades participantes. Administra el marco normativo y los planos de control de la Region, pero no opera los servicios ni posee necesariamente el hardware.

El **Domain Owner** es la entidad participante que administra su propio espacio de identidad y recursos dentro de la nube comunitaria. Es responsable de los servicios que operan sobre la infraestructura pero no administra la infraestructura física directamente.

El **Infrastructure Operator** es la entidad responsable de la operación técnica de la infraestructura: instala, configura y mantiene los nodos físicos que aportan recursos al pool. En la mayoría de los casos el Infrastructure Operator y el Domain Owner coinciden en la misma entidad, pero el modelo los distingue explícitamente para contemplar escenarios donde una entidad opera infraestructura que no posee, o donde una entidad arrienda recursos a terceros que adquieren la figura de Domain Owner.

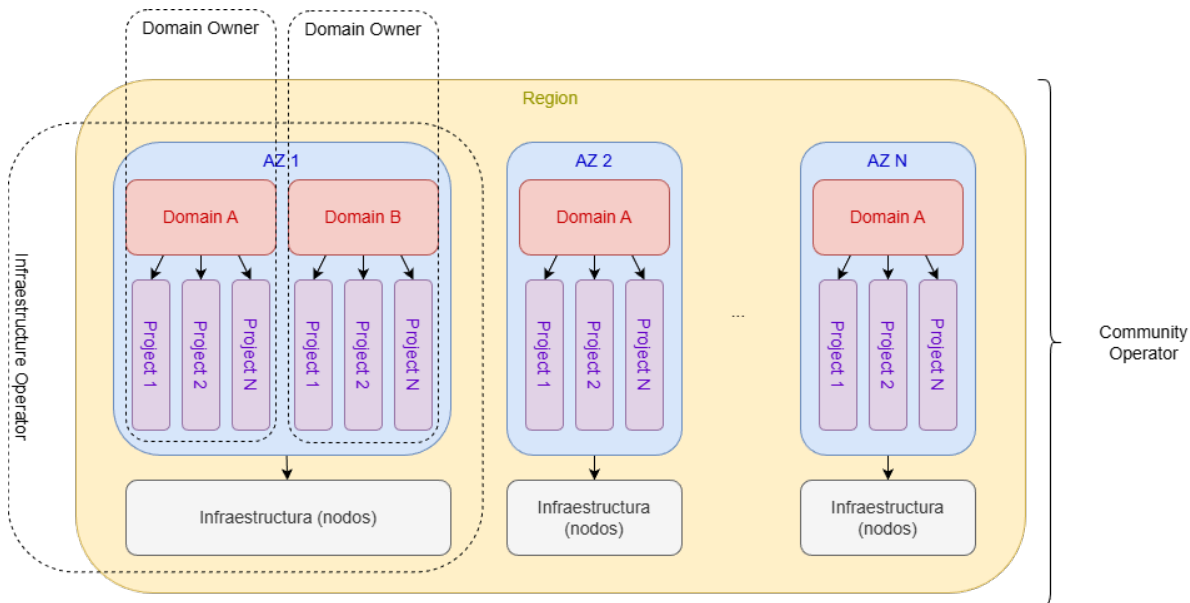


Fig. 12. Modelo de gobierno de la nube comunitaria.

Un referente institucional que ilustra el rol del Community Operator en el contexto mexicano es el Centro Nacional de Control de Energía (CENACE), organismo público descentralizado cuyo objeto es ejercer el control operativo del Sistema Eléctrico Nacional y garantizar imparcialidad en el acceso a la Red Nacional de Transmisión [116]: así como el CENACE no genera energía sino que gestiona las condiciones bajo las cuales los participantes acceden a la red, el Community Operator no opera servicios sino que gestiona las condiciones bajo las cuales los Domain Owners acceden al pool de recursos de cómputo. De manera análoga al Mercado Eléctrico Mayorista que opera el CENACE [116], los Domain Owners pueden ceder o arrendar su capacidad ociosa de cómputo a otras entidades bajo las políticas que el Community Operator define, utilizando un Domain de servicios compartidos como espacio de identidad neutral que garantiza trazabilidad completa sobre quién consume qué recursos.

3.3.2.1 Atribuciones del Community Operator

El Community Operator debe cubrir tres ámbitos no delegables.

El primero es la **gestión de políticas globales**: define los estándares mínimos de interoperabilidad como lo son las versiones de API, formatos de imagen, protocolos de red, requisitos mínimos de hardware, así como las políticas de seguridad que aplican a la totalidad de la Region.

El segundo es la **gestión de identidad federada**: administra el Domain raíz de Keystone desde el cual se crean y gobiernan los Domains de cada Domain Owner. Ninguna entidad puede incorporarse sin que el Community Operator provisione su Domain, asigne cuotas iniciales y establezca políticas de acceso.

El tercero es el **arbitraje de recursos y cuotas**: define la proporción de recursos que cada Infrastructure Operator aporta al pool y la fracción que cada Domain Owner puede consumir. En el plano técnico estas decisiones se traducen en cuotas por Domain en OpenStack [80] y en LimitRanges y ResourceQuotas por namespace en Kubernetes [108].

3.3.2.2 Planos de control del Community Operator

La autoridad del Community Operator está respaldada por control técnico efectivo sobre seis planos que convierten sus decisiones en restricciones técnicas aplicadas automáticamente por el sistema.

El **plano de identidad** comprende el Domain raíz de Keystone. Sin un token válido emitido por Keystone ningún usuario ni servicio puede acceder a ningún recurso de la Region. La capacidad de suspender o revocar un Domain hace ejecutable cualquier decisión disciplinaria sin intervención física sobre los nodos de la entidad afectada.

El **plano de cómputo** comprende la API de Nova y los aggregate groups que definen las AZs. El Community Operator determina qué nodos pertenecen a qué AZ y con qué cuotas opera cada Domain.

El **plano de red** comprende la configuración de Neutron a nivel de Region: redes externas, rangos de IP flotantes y políticas de enrutamiento inter-zona. Las redes de tenant de cada Project son autónomas dentro de su Domain, pero los recursos de red de los que dependen son definidos por el Community Operator.

El **plano de almacenamiento** comprende el CRUSH Map del clúster Ceph. El Community Operator es el único con autoridad para modificar la topología del clúster, incorporar nodos OSD¹⁵ o ajustar políticas de replicación.

¹⁵Object Storage Daemon: proceso responsable de gestionar un disco físico individual dentro del clúster Ceph.

El **plano de seguridad** comprende OpenBao como autoridad certificadora raíz de la nube comunitaria. Todo certificado TLS que opera en la Region deriva su confianza de esta CA raíz, cuya administración es responsabilidad exclusiva del Community Operator.

El **plano de observabilidad** comprende la instancia central de Grafana y los agregadores de Prometheus y Loki. El Community Operator tiene visibilidad sobre el estado de toda la Region; cada Domain Owner tiene visibilidad únicamente sobre su propio Domain.

TABLA 14

PLANOS DE CONTROL DEL COMMUNITY OPERATOR Y SUS MECANISMOS DE ENFORCEMENT.			
Plano de control	Tecnología	Facultad del Community Operator	Mecanismo de enforcement ¹⁶
Identidad	Keystone Domain raíz	Crear · suspender · revocar Domains	Token inválido → acceso denegado
Cómputo	Nova · aggregate groups	Definir AZs · cuotas · incorporar nodos	Cuota cero → sin capacidad de cómputo
Red	Neutron · rutas externas	Controlar acceso inter-zona e inter-net	Revocación de rutas → aislamiento
Almacenamiento	Ceph CRUSH Map	Topología · replicación · dominios de fallo	Sin OSD → sin almacenamiento
Seguridad	OpenBao CA raíz	Emitir · revocar certificados TLS	Revocación → tráfico interrumpido
Observabilidad	Prometheus · Loki · Grafana	Visibilidad global de la Region	Detección proactiva · sin exposición entre Domains

3.3.3 Plano de control regional y sitios de control designados

El plano de control regional convierte un conjunto de nodos distribuidos entre múltiples AZs en un sistema gestionable como unidad. Su función no es transportar datos entre zonas sino emitir las instrucciones que determinan qué debe ocurrir en cada AZ y mantener el estado global de la Region como fuente de verdad única. El plano de control opera sobre tres dimensiones: en **cómputo**, Nova API decide el placement de VMs e instruye a los agentes locales; en **red**, Neutron define las redes virtuales que los agentes OVN/OVS implementan localmente; en **almacenamiento**, Cinder gestiona el ciclo de vida de volúmenes instruyendo a Ceph, aunque la replicación entre AZs la gestiona Ceph de forma autónoma mediante su CRUSH Map.

Para garantizar la alta disponibilidad del plano de control sin exigir hardware uniforme en todas las AZs, la arquitectura adopta un modelo de **sitios de control designados**, análogo al esquema del CENACE con su centro principal (CENAL) y su Centro Nacional Alterno de

¹⁶ Aplicación efectiva y forzosa de una norma o política mediante mecanismos técnicos automatizados.

Control de Energía (CENALTE), diseñado para asumir en caso de emergencia las funciones en tiempo real del Sistema Eléctrico Nacional [116]. El Community Operator designa un subconjunto de AZs como **Control Sites** en función de su capacidad de hardware y estabilidad de conectividad. Estas AZs alojan el plano de control de OpenStack y los miembros del quórum de etcd de Kubernetes. Las demás AZs operan como **Execution Sites**: alojan nodos de cómputo y almacenamiento pero no participan en el quórum del plano de control.

Se propone que el número mínimo de Control Sites sea de tres, ya que satisface el requisito de mayoría impuesto por Raft, el algoritmo de consenso que implementa etcd [22]: con tres sitios la pérdida de uno mantiene quórum (2 de 3); con dos sitios la pérdida de uno suspende el plano de control hasta restaurar el sitio.

TABLA 15

TIPOS DE AZ SEGÚN SU PARTICIPACIÓN EN EL PLANO DE CONTROL REGIONAL.

Tipo de AZ	Rol en el plano de control	Comportamiento ante fallo del plano de control
Control Site	Aloja plano de control regional · participa en quórum etcd	Puede asumir liderazgo si otros Control Sites fallan
Execution Site	Solo nodos de cómputo y almacenamiento	Continúa operando recursos existentes · sin nuevas instrucciones

3.3.4 Rol de las tecnologías en el nivel de Contexto

En este nivel de abstracción cada tecnología del stack ocupa un rol funcional preciso dentro de la jerarquía Region–AZ–Nodo, operando como planos que el Community Operator controla para gobernar la Region completa.

OpenStack implementa el plano de control de la Region exponiendo una API unificada que abstrae la distribución física de los recursos entre AZs. Es la tecnología que materializa técnicamente la autoridad del Community Operator sobre cómputo, red e identidad.

Ceph implementa el plano de almacenamiento distribuido. Su algoritmo CRUSH mapea las AZs como dominios de fallo independientes y replica los datos de forma autónoma entre zonas, garantizando resiliencia de datos independientemente del estado del plano de control.

Kubernetes implementa el plano de orquestación de cargas de trabajo. Consume los recursos de OpenStack y los presenta a las aplicaciones como un clúster unificado. Las topology labels

permiten distribuir pods entre AZs siguiendo políticas de afinidad coherentes con las de OpenStack.

Cilium opera como CNI de la red de pods y como implementación de la Gateway API, gestionando la conectividad entre cargas de trabajo distribuidas entre AZs y controlando el acceso desde el exterior mediante su plano de datos eBPF.

Harbor opera como registro de imágenes por AZ. La replicación asíncrona entre instancias garantiza que las imágenes publicadas en una zona estén eventualmente disponibles en todas las demás, de forma autónoma respecto al plano de control.

OpenTofu y **Ansible** operan en el plano de aprovisionamiento: OpenTofu describe el estado deseado de la Region como código versionado; Ansible configura cada nodo al incorporarse, garantizando reproducibilidad y auditabilidad.

OpenBao opera como almacén de secretos y CA raíz de la Region. Todos los servicios obtienen credenciales y certificados desde OpenBao, eliminando la proliferación de secretos en archivos de configuración.

Aquí **cert-manager** automatiza el ciclo de vida de certificados TLS integrándose con OpenBao como backend de CA, siguiendo el protocolo ACME [70], [87].

Prometheus, **Loki** y **Grafana** implementan el plano de observabilidad: cada AZ recolecta métricas y logs de forma autónoma; un nivel central consolida los datos de toda la Region en un único panel.

TABLA 16

ROL DE CADA TECNOLOGÍA EN EL NIVEL DE CONTEXTO.

Tecnología	Nivel de operación	Rol en el nivel de Contexto
OpenStack	Region	Plano de control unificado · abstraer AZs tras una API única
IroniC	Region → Nodo	Incorporación de bare metal al pool de Nova
Ceph	Region	Almacenamiento distribuido · AZs como dominios de fallo CRUSH
Kubernetes	Region	Orquestación de cargas · topology labels por AZ
Cilium	Region	CNI de pods · Gateway API de servicios externos
Harbor	AZ	Registro local de imágenes · replicación asíncrona inter-zona
OpenTofu	Region	Aprovisionamiento declarativo de infraestructura
Ansible	Nodo	Configuración reproducible de cada nodo al incorporarse
OpenBao	Region	Almacén de secretos · CA raíz de la nube comunitaria
cert-manager	Clúster K8s	Emisión y renovación automática de certificados TLS
Prometheus	AZ + Region	Recolección local de métricas · agregación central
Loki	AZ + Region	Agregación local de logs · consolidación central
Grafana	Region	Visualización unificada de todas las AZs

3.4 Nivel de Contenedores

Responde: ¿cómo se divide el sistema en bloques funcionales que se comunican entre sí?

3.4.1 Topología interna de una Availability Zone

Este nivel desciende al interior de una AZ para describir cómo se organizan sus nodos, qué responsabilidades cumple cada tipo, qué perfil de hardware requiere y cómo se distribuyen las tecnologías entre ellos. La topología aquí descrita es normativa: toda AZ que se incorpore a la nube comunitaria debe adoptarla como condición de interoperabilidad.

La separación de responsabilidades dentro de una AZ se materializa en cuatro roles de nodo mutuamente excluyentes. Esta separación no es una preferencia de diseño sino un requisito de estabilidad operativa: mezclar el plano de control con cargas de trabajo introduce riesgos de interferencia que pueden comprometer la disponibilidad de toda la AZ. En todos los casos el disco del SO nunca forma parte del pool distribuido de Ceph.

3.4.1.1 Nodos de control

Los nodos de control se clasifican en dos tipos según su rol en la jerarquía de gobierno.

Los **nodos de control regional**, presentes exclusivamente en los Control Sites, alojan las APIs globales de OpenStack junto con RabbitMQ [117] y MariaDB/Galera [118], así como el API server, etcd, scheduler y controller manager de Kubernetes [98], y la instancia principal de

OpenBao. Deben distribuirse entre un mínimo de tres Control Sites para garantizar quórum de etcd [22]. Su pérdida suspende la gestión global de la Region pero no las operaciones en curso.

Los **nodos de control local**, presentes en todas las AZs, alojan los agentes de OpenStack que reciben instrucciones del plano regional y las ejecutan localmente, así como Harbor, Prometheus, Loki, Grafana y cert-manager. Garantizan operación autónoma de la AZ ante pérdida de conectividad con el plano regional.

Ningún nodo de control ejecuta cargas de trabajo de usuario bajo ninguna circunstancia.

Perfil de equipo típico: servidor dedicado de rack o torre con ECC RAM. No es adecuado ningún equipo de escritorio o workstation de consumo.

TABLA 17

REQUISITOS DE HARDWARE — NODOS DE CONTROL. ELABORACIÓN PROPIA A PARTIR DE LOS REQUISITOS MÍNIMOS DOCUMENTADOS PARA OPENSTACK [80] Y EL PLANO DE CONTROL DE KUBERNETES [98].

Componente	Requisito mínimo
CPU	8 núcleos físicos
RAM	32 GB
Disco SO	100 GB SSD (latencia crítica para etcd)
Disco datos	200 GB SSD (bases de datos · logs)
Red	2 × 1 GbE (bonding activo-pasivo) · gestión y externa
Cantidad mínima por AZ	3 nodos en Control Sites · 1+ en Execution Sites

3.4.1.2 Nodos de cómputo

Los nodos de cómputo son los hipervisores sobre los cuales Nova programa las VMs. Cada nodo ejecuta Nova-compute y Neutron-OVS, recibiendo instrucciones del plano regional. Los nodos bare metal aprovisionados vía Ironic son funcionalmente equivalentes desde la perspectiva del plano de control.

El nodo de cómputo no requiere almacenamiento local para las cargas de trabajo: los discos raíz de las VMs se montan desde Ceph vía RBD. La RAM es el recurso más determinante: cada GB adicional se traduce directamente en mayor capacidad de VMs para los Projects.

Perfil de equipo típico: servidor de rack, workstation de alto rendimiento o servidor torre con virtualización habilitada en BIOS/UEFI.

TABLA 18

REQUISITOS DE HARDWARE — NODOS DE CÓMPUTO. ELABORACIÓN PROPIA A PARTIR DE LOS REQUISITOS DE NOVA Y DEL HIPERVISOR KVM [105], [119].

Componente	Requisito mínimo
CPU	8 núcleos físicos con soporte de virtualización (Intel VT-x / AMD-V)
RAM	32 GB (determina el número de VMs simultáneas)
Disco SO	60 GB SSD o HDD (solo sistema operativo del hipervisor)
Disco datos	No requerido (almacenamiento de VMs vía Ceph RBD)
Red	2 × 10 GbE (bonding activo-pasivo) · gestión, almacenamiento y tenant
Cantidad mínima por AZ	2 nodos

3.4.1.3 Nodos de almacenamiento

Los nodos de almacenamiento alojan exclusivamente los daemons OSD de Ceph. Un subconjunto ejecuta adicionalmente Monitor, Manager y MDS en configuración redundante de al menos tres instancias. Cada disco OSD debe ser un dispositivo dedicado exclusivamente a Ceph: nunca compartido con el SO ni otros servicios [120].

Perfil de equipo típico: servidor de rack con bahías abundantes. Ceph puede mezclar SSDs y HDDs mediante tiering: SSDs para baja latencia, HDDs para alta capacidad [88].

TABLA 19

REQUISITOS DE HARDWARE — NODOS DE ALMACENAMIENTO. ELABORACIÓN PROPIA A PARTIR DE LAS RECOMENDACIONES OFICIALES DE HARDWARE DE CEPH [120].

Componente	Requisito mínimo
CPU	4 núcleos físicos
RAM	4 GB por OSD + 4 GB base (mínimo 16 GB recomendado)
Disco SO	60 GB SSD o HDD (exclusivo para el sistema operativo)
Discos OSD	≥ 3 discos dedicados por nodo · capacidad según necesidad del clúster
Red	2 × 10 GbE (bonding activo-pasivo) · gestión y almacenamiento
Cantidad mínima por AZ	3 nodos (replicación ×3 de Ceph)

3.4.1.4 Nodos legacy

Los nodos legacy no satisfacen los requisitos de los roles anteriores pero pueden incorporarse al clúster de Kubernetes como workers de segunda clase mediante K3s, una distribución ligera de Kubernetes diseñada para entornos con recursos limitados [121]. No participan en la capa IaaS de OpenStack ni contribuyen almacenamiento a Ceph. Su rol está restringido a cargas de baja criticidad mediante taints y tolerations [122].

Perfil de equipo típico: equipo de escritorio o servidor de generaciones anteriores.

TABLA 20

REQUISITOS DE HARDWARE — NODOS LEGACY. ELABORACIÓN PROPIA A PARTIR DE LOS REQUISITOS MÍNIMOS DE K3S [121] Y CILIUM [106].

Componente	Requisito mínimo
CPU	2 núcleos (arquitectura x86_64)
RAM	512 MB (K3s) · 1 GB recomendado para cargas útiles
Disco SO	10 GB (sistema operativo + K3s)
Disco datos	No requerido (almacenamiento vía Ceph CSI directo)
Red	1 × 100 MbE mínimo · gestión
Kernel Linux	≥ 4.9 (requisito de Cilium)
Cantidad mínima	No aplica (incorporación opcional)

3.4.1.5 Resumen comparativo

TABLA 21

RESUMEN DE REQUISITOS MÍNIMOS DE HARDWARE POR ROL DE NODO.

Rol	CPU mín.	RAM mín.	Disco SO	Discos OSD	Red	Mín. por AZ
Control	8 núcleos	32 GB	100 GB SSD	—	2 × 1 GbE	3 nodos
Cómputo	8 núcleos	32 GB	60 GB	—	2 × 10 GbE	2 nodos
Almacenamiento	4 núcleos	16 GB	60 GB	≥ 3 discos	2 × 10 GbE	3 nodos
Legacy	2 núcleos	512 MB	10 GB	—	1 × 100 MbE	Opcional

3.4.2 Rol de las tecnologías en el nivel de Contenedores

En este nivel las tecnologías se distribuyen entre los distintos roles de nodo según el servicio que prestan dentro de la AZ. La presencia en los planos de red determina qué tráfico puede recibir y emitir cada unidad de despliegue.

OpenStack (APIs globales) opera en los nodos de control regional de los Control Sites: es la interfaz entre el Community Operator y los recursos físicos de la Region.

Nova-compute y Neutron OVS/OVN operan en los nodos de cómputo: ejecutan localmente las instrucciones del plano de control regional.

Ceph (OSD) opera en los nodos de almacenamiento utilizando exclusivamente la red de almacenamiento. **Ceph (Mon/Mgr/MDS)** utiliza adicionalmente la red de gestión para coordinación del clúster.

Harbor, Prometheus, Loki, Grafana y cert-manager operan en los nodos de control local de todas las AZs, garantizando autonomía operativa local ante pérdida de conectividad con el plano regional.

Ironic opera en los nodos de control regional, gestionando el aprovisionamiento de hardware bare metal.

OpenTofu y Ansible operan desde fuera de la AZ, alcanzando todos los nodos a través de la red de gestión.

TABLA 22

PRESENCIA DE CADA TIPO DE NODO EN LOS PLANOS DE RED DE LA AZ.

Tipo de nodo	Gestión	Almacenamiento	Tenant	Externa
Control	✓	—	—	✓
Cómputo	✓	✓	✓	—
Almacenamiento	✓	✓	—	—
Legacy	✓	—	—	—

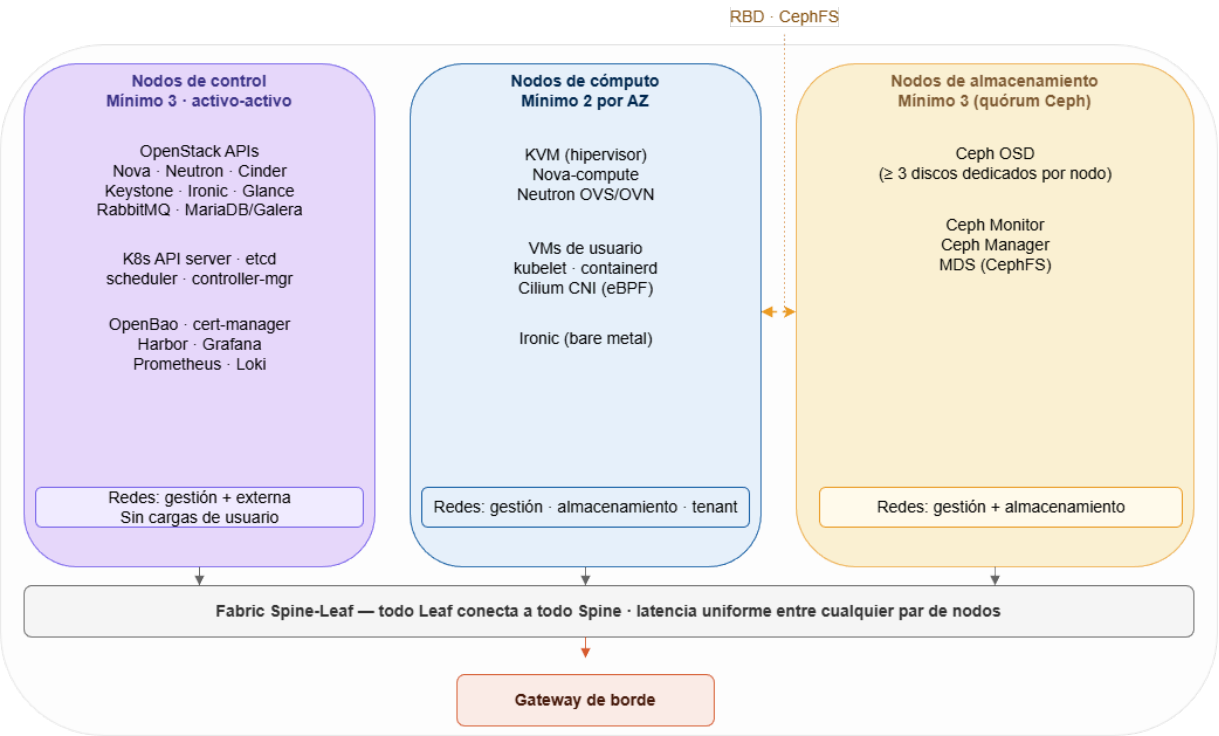


Fig. 13. Topología interna de una AZ

TABLA 23

DISTRIBUCIÓN DE TECNOLOGÍAS POR NODO Y PLANO DE RED DENTRO DE LA AZ.

Tecnología	Nodo(s) donde opera	Red(es) que utiliza
OpenStack (APIs globales)	Control regional (Control Site)	Gestión · Externa
Keystone · OpenBao (CA)	Control regional (Control Site)	Gestión
K8s API server · etcd	Control regional (Control Site)	Gestión
Ironic	Control regional (Control Site)	Gestión
Harbor · Prometheus · Loki	Control local (todas las AZs)	Gestión · Externa
Grafana · cert-manager	Control local (todas las AZs)	Gestión · Externa
OpenStack (agentes)	Cómputo	Gestión · Tenant
Kubernetes (workers)	VMs en Cómputo	Gestión · Tenant
Cilium (CNI)	VMs en Cómputo	Tenant
Cilium (Gateway API)	Control local o dedicado	Gestión · Externa
Ceph (OSD)	Almacenamiento	Almacenamiento
Ceph (Mon/Mgr/MDS)	Almacenamiento	Gestión · Almacenamiento
OpenTofu / Ansible	Externo a la AZ	Gestión

3.4.3 Incorporación de nodos legacy

Extensión opcional de la arquitectura de referencia.

En el contexto institucional de la Facultad de Ingeniería es frecuente encontrar equipos cuyas características no satisfacen los requisitos mínimos de OpenStack pero que representan capacidad de cómputo aprovechable. Los nodos legacy se incorporan directamente al clúster de Kubernetes como workers de segunda clase mediante K3s [121], sin pasar por la virtualización de OpenStack. El almacenamiento persistente se accede directamente vía CSI driver de Ceph [102].

La heterogeneidad de hardware se gestiona mediante taints y tolerations [122]: cada nodo legacy recibe un taint que impide asignación de pods por defecto; solo los pods con toleration explícita pueden ejecutarse en ellos. Los nodos legacy están reservados para cargas de baja criticidad: desarrollo, pruebas, batch y CI/CD.

Esta modalidad introduce tres limitaciones. La primera es la **ausencia de aislamiento entre tenants**: los pods comparten el kernel del nodo sin la capa de aislamiento del hipervisor KVM, insuficiente para cargas de producción multitenancy. La segunda es la **exclusión del modelo de cuotas**: los recursos de nodos legacy no participan en el sistema de cuotas por Domain de OpenStack, introduciendo una asimetría en la rendición de cuentas. La tercera es la **mantenibilidad reducida**: operan fuera del flujo de aprovisionamiento de Ironic y Nova, requiriendo intervención manual ante fallos.

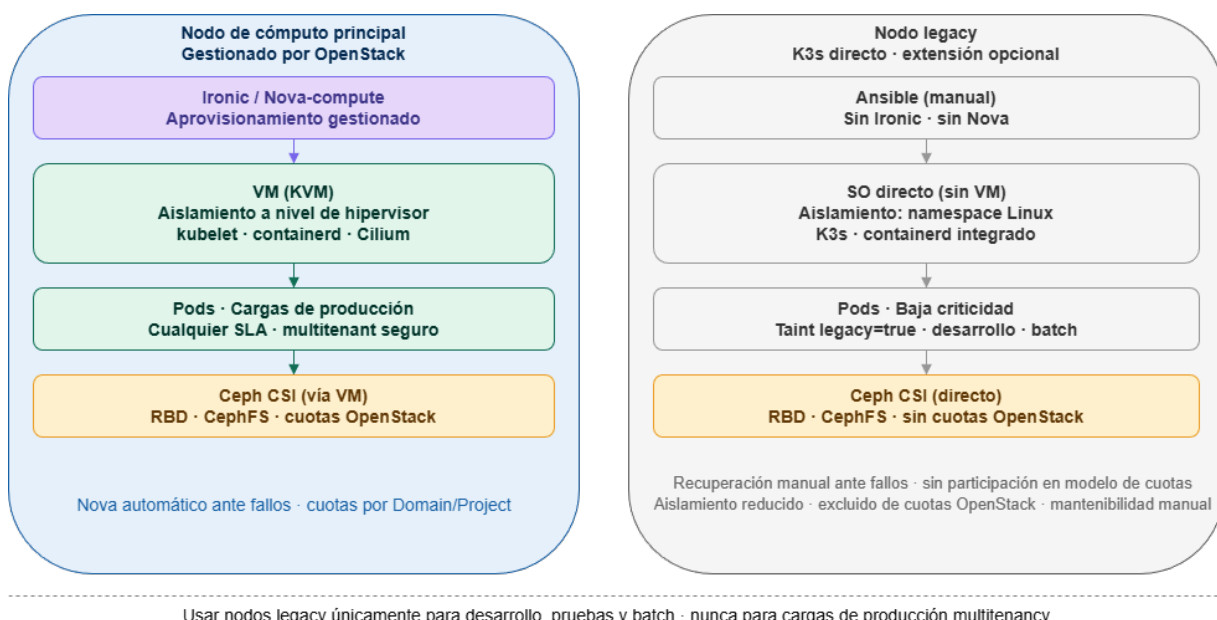


Fig. 14. Nodos legacy vs nodos principales

TABLA 24

COMPARACIÓN ENTRE NODOS PRINCIPALES Y NODOS LEGACY.

Dimensión	Nodo principal	Nodo legacy
Incorporación	Nova-compute / Ironic	K3s directo
Aislamiento	VM (hipervisor KVM)	Namespace Linux
Almacenamiento	Ceph CSI vía VM	Ceph CSI directo
Cuotas OpenStack	Sí (por Domain/Project)	No
Recuperación ante fallos	Nova (automática)	Manual
Cargas de trabajo	Producción · cualquier SLA	Baja criticidad · desarrollo
Aprovisionamiento	Ironic + Ansible	Ansible

3.5 Nivel de Componentes

Responde: ¿qué hay dentro de cada bloque funcional?

Este nivel describe el interior de cada tipo de nodo: cómo se apilan sus tecnologías, qué relaciones establecen entre sí y qué exponen al resto del sistema. Es el nivel más granular de la arquitectura y el que establece las relaciones precisas entre las capas IaaS, CaaS y de almacenamiento.

3.5.1 Nodo de control

El nodo de control aloja dos planos de control superpuestos que comparten el mismo hardware pero operan de forma independiente: el plano de control de OpenStack y el plano de control de Kubernetes. OpenBao actúa como componente transversal que provee identidad y secretos a ambos planos desde el arranque.

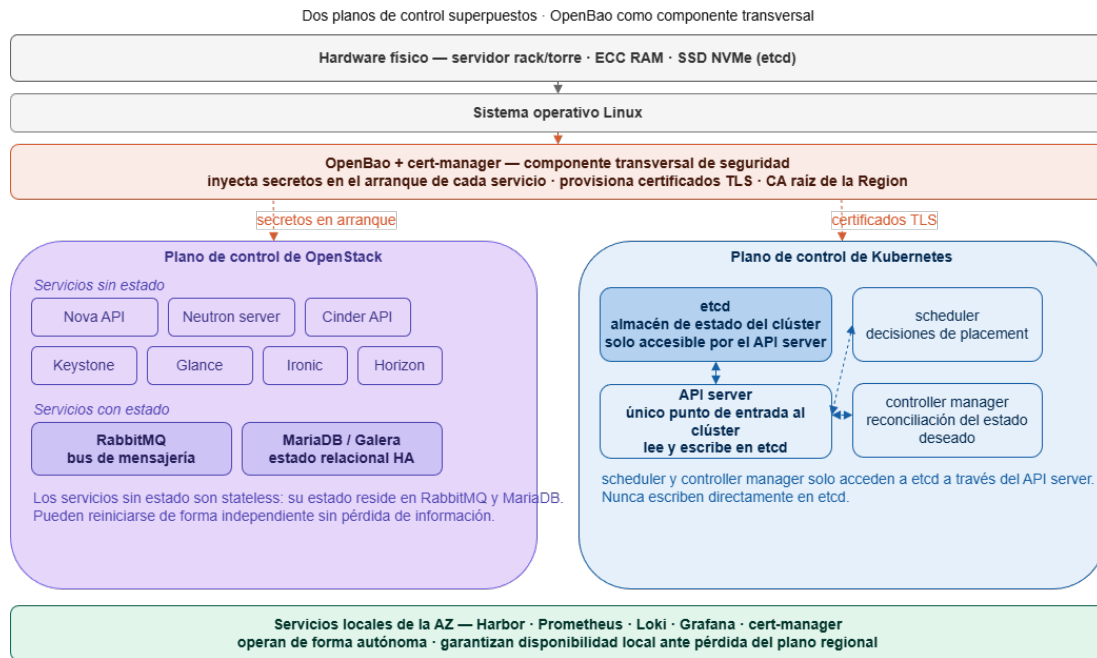


Fig. 15. Nodos de control

El **plano de control de OpenStack** se compone de servicios sin estado, es decir Nova API, Neutron server, Cinder API, Glance, Keystone, Ironi y Horizon [80]. Estos se comunican a través de dos dependencias con estado: RabbitMQ como bus de mensajería [117] y MariaDB con Galera como almacén relacional en replicación síncrona multi-maestro [118]. Los servicios sin estado pueden reiniciarse de forma independiente sin perder información; el estado reside exclusivamente en RabbitMQ y MariaDB. La configuración activo-activo entre los tres nodos de un Control Site garantiza que la pérdida de uno no interrumpa ningún servicio.

El **plano de control de Kubernetes** se compone de etcd como almacén de estado distribuido, el API server como punto de entrada único, el scheduler como componente de decisión de placement, y el controller manager como componente de reconciliación del estado deseado [98]. La relación entre estos componentes es estricta y unidireccional: el API server es el único que escribe en etcd; el scheduler y el controller manager acceden al estado únicamente a través del API server. Esta arquitectura garantiza que etcd nunca sea accedido directamente por componentes externos, reduciendo la superficie de corrupción del estado del clúster.

OpenBao inyecta las credenciales de base de datos y mensajería en cada servicio de OpenStack en el arranque, sin que los valores pasen por archivos de configuración en disco. Para los componentes de Kubernetes, cert-manager (integrado con OpenBao como CA) provisiona el certificado TLS que cada componente usa para autenticarse con el API server.

TABLA 25

CAPAS DE EJECUCIÓN Y TECNOLOGÍAS DENTRO DE UN NODO DE CONTROL.			
Capa	Tecnología	Gestiona	Visible para
OpenStack sin estado	Nova API · Neutron · Cinder · Keystone	APIs globales de la Region	Community Operator · Domain Owners
OpenStack con estado	RabbitMQ · MariaDB/Galera	Mensajería y estado relacional de OpenStack	Servicios de OpenStack
K8s estado	etcd	Estado del clúster Kubernetes	API server únicamente
K8s control	API server · scheduler · controller	Orquestación global del clúster	Kubernetes workers · operadores
Seguridad transversal	OpenBao · cert-manager	Secretos · certificados TLS de ambos planos	Todos los servicios del nodo

3.5.2 Nodo de cómputo

Un nodo de cómputo no ejecuta contenedores directamente sobre el sistema operativo del hipervisor. Opera en dos niveles de ejecución superpuestos que preservan el aislamiento entre cargas de trabajo de distintos Domains y Projects.

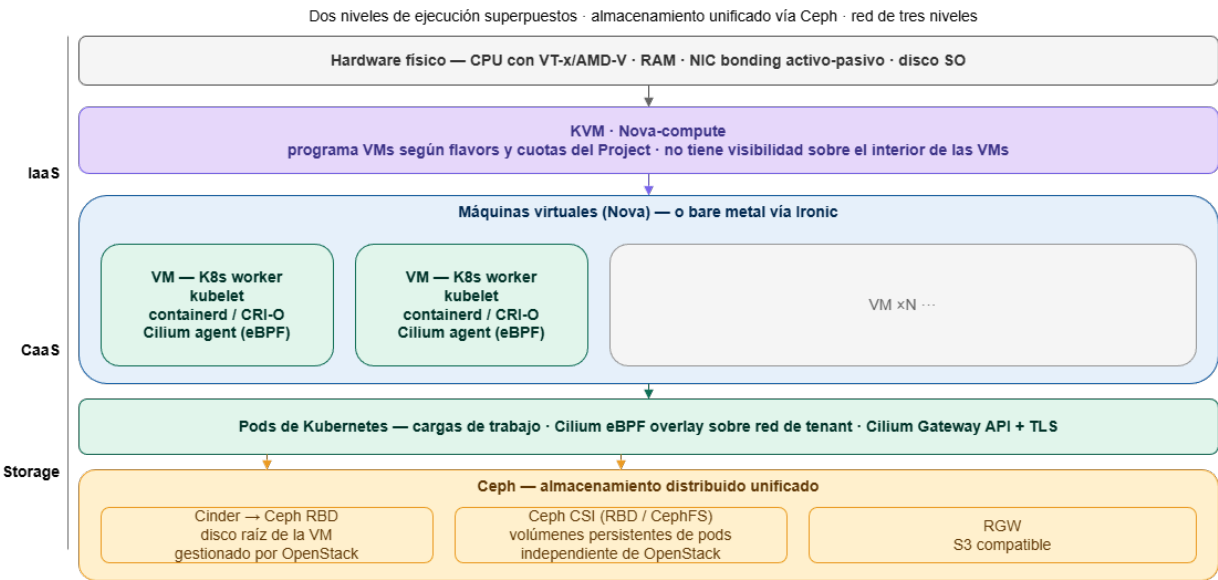


Fig. 16. Nodos de cómputo

El **nivel de hipervisor** es gestionado exclusivamente por OpenStack. KVM ejecuta las VMs [119] y Nova-compute las programa según los flavors y cuotas de cada Project [105]. El nodo no tiene visibilidad sobre qué corre dentro de sus VMs.

El **nivel de VM** es donde Kubernetes opera. Cada VM ejecuta kubelet, containerd o CRI-O, y el agente de Cilium. Kubernetes trata la VM como un nodo worker sin visibilidad sobre el hipervisor subyacente.

En el caso de bare metal aprovisionado vía Ironic, el nivel de hipervisor desaparece: el sistema operativo se instala directamente sobre el hardware y se presenta a Nova como un recurso indistinguible de una VM.

Cilium gestiona la red de pods mediante eBPF [95] tanto dentro de la VM como entre VMs distintas. El resultado es una arquitectura de red completamente transparente para las aplicaciones, organizada en tres niveles: física (Spine-Leaf), tenant (Neutron/VXLAN) y pods (Cilium/eBPF).

El almacenamiento llega al nodo de cómputo por dos rutas que convergen en el mismo clúster Ceph: **Cinder** → **Ceph RBD** para el disco raíz de la VM, gestionado por OpenStack [80]; y **Ceph CSI** para los volúmenes persistentes de pods, gestionado directamente por Kubernetes sin intermediación de OpenStack [101], [102]. Esta coexistencia es la razón por la que Ceph opera como capa de almacenamiento unificada para IaaS y CaaS simultáneamente.

TABLA 26

CAPAS DE EJECUCIÓN Y TECNOLOGÍAS DENTRO DE UN NODO DE CÓMPUTO.			
Nivel	Tecnología	Gestiona	Visible para
Hipervisor	KVM + Nova-compute	VMs sobre hardware físico	OpenStack
VM / bare metal	kubelet + containerd/CRI-O	Pods sobre sistema operativo	Kubernetes
Red de pods	Cilium (eBPF)	Conectividad entre pods	Aplicaciones
Almacenamiento	Ceph CSI (RBD/CephFS)	Volúmenes persistentes para pods	Kubernetes
Almacenamiento	Cinder → Ceph RBD	Disco raíz de la VM	OpenStack

3.5.3 Nodo de almacenamiento

El nodo de almacenamiento tiene un único propósito: exponer capacidad de disco al clúster Ceph de la forma más eficiente posible. A diferencia de los nodos de control y cómputo, opera en una única capa funcional con tres roles diferenciados según el subconjunto de servicios de Ceph que aloja.

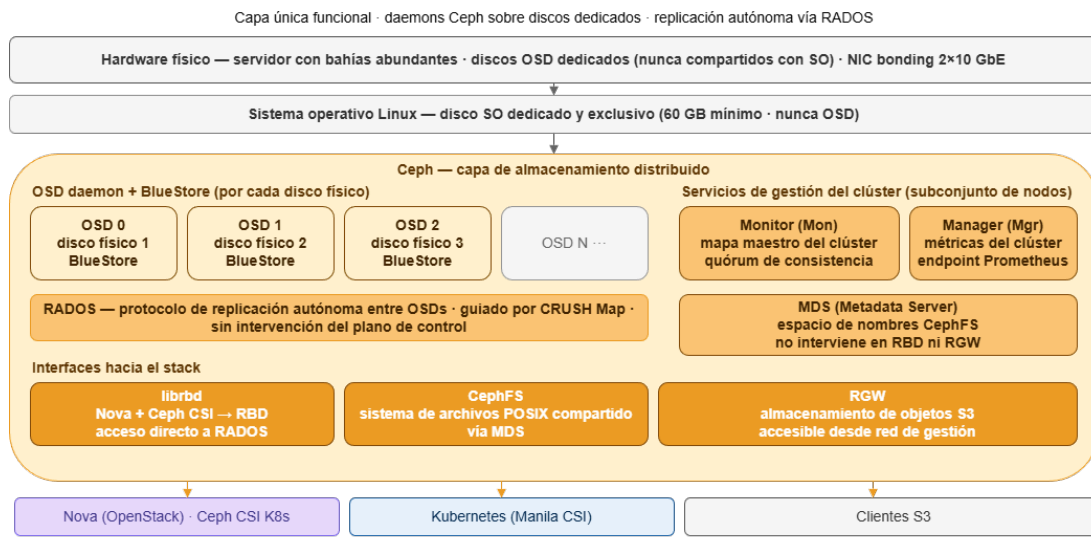


Fig. 17. Nodos de almacenamiento

El componente central es el **daemon OSD**, responsable de gestionar un disco físico individual. Cada OSD mantiene su propio almacén de objetos local sobre BlueStore (el backend de almacenamiento nativo de Ceph) y participa en el clúster distribuido mediante el protocolo RADOS (Reliable Autonomic Distributed Object Store) [104]. Cuando una escritura llega a un OSD primario, este coordina la replicación hacia los OSDs secundarios designados por el CRUSH Map antes de confirmar la operación al cliente [89]. Este mecanismo es completamente autónomo respecto al plano de control de OpenStack y Kubernetes.

Un subconjunto de nodos ejecuta adicionalmente los servicios de gestión del clúster [90]. El **Monitor** (Mon) mantiene el mapa maestro del clúster y participa en el consenso de quórum que garantiza la consistencia del estado. El **Manager** (Mgr) recolecta métricas y expone el endpoint de Prometheus sin requerir agentes adicionales. El **MDS** gestiona el espacio de nombres de CephFS cuando se requiere un sistema de archivos POSIX compartido; no interviene en las operaciones de bloque (RBD) ni de objeto (RGW).

La relación con el resto del stack se establece a través de tres interfaces [88], [90]: **librbd**, que conecta directamente Nova y el CSI driver de Kubernetes con los OSDs sin servicios intermedios; **CephFS**, que expone un sistema de archivos POSIX montable; y **RGW**, que expone almacenamiento de objetos compatible con la API S3.

TABLA 27

COMPONENTES INTERNOS DEL NODO DE ALMACENAMIENTO Y SUS INTERFACES HACIA EL STACK.			
Componente	Tecnología	Gestiona	Interfaz hacia el stack
OSD	Ceph OSD + BlueStore	Un disco físico · replicación autónoma via RADOS	librbd · CephFS · RGW
Monitor	Ceph Mon	Mapa maestro del clúster · quórum de consistencia	Internamente a Ceph
Manager	Ceph Mgr	Métricas del clúster · módulo Prometheus	Prometheus (scraping)
Metadata Server	Ceph MDS	Espacio de nombres de CephFS	CephFS (POSIX)

3.5.4 Extensión: nodos de alto rendimiento

La arquitectura contempla la incorporación futura de nodos especializados para cargas que requieren aceleración por hardware para tareas de inferencia de modelos de lenguaje, HPC, procesamiento de señales entre otros. Esto es una consecuencia directa del principio de modularidad definido en el Capítulo 2.

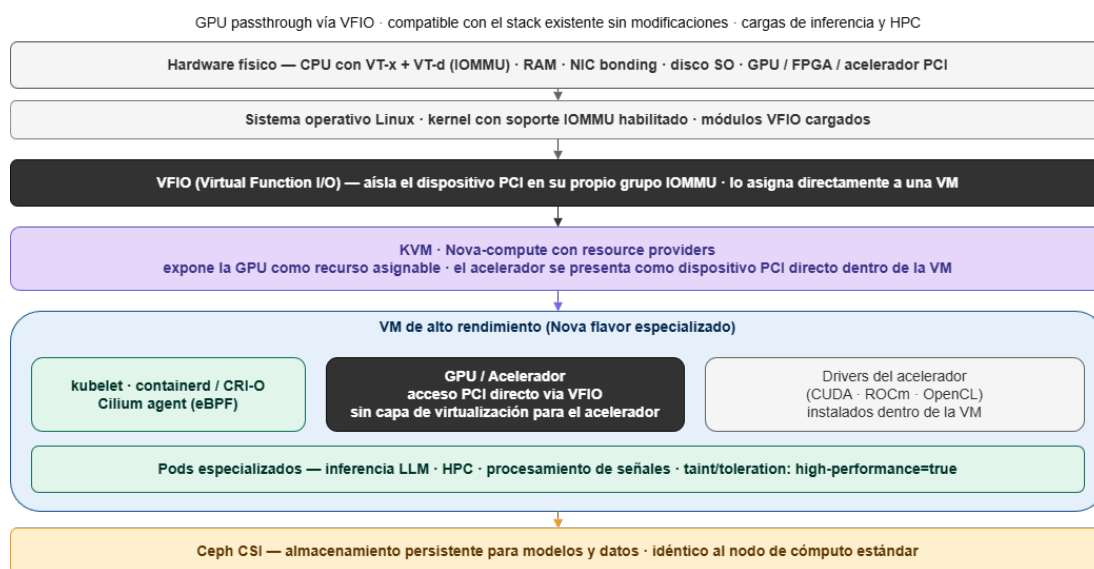


Fig. 18. Nodos de almacenamiento

El mecanismo de incorporación es GPU passthrough mediante VFIO [39], [40], cuyos fundamentos se describen en la sección 1.4.1. Desde la perspectiva de OpenStack, el nodo de alto rendimiento es un nodo de cómputo con un flavor especializado que incluye el acelerador como recurso asignable mediante el mecanismo de resource providers de Nova [105]. Desde la perspectiva de Kubernetes, las VMs que consumen el acelerador son nodos worker ordinarios distinguidos por sus labels de capacidad.

La incorporación de nodos de alto rendimiento no requiere modificaciones en ningún otro componente del stack: Ceph, Kubernetes, Cilium y la cadena de seguridad operan de forma idéntica. La definición detallada de requisitos de hardware y configuración de VFIO constituye una línea de trabajo futura identificada en el Capítulo 5.

3.6 Síntesis de la arquitectura propuesta

Las secciones anteriores describieron la arquitectura descendiendo por los niveles del C4 Model, desde el modelo de gobierno de la nube comunitaria hasta el interior de cada tipo de nodo. Esta sección resume la propuesta mediante cuatro figuras que muestran lo descrito a lo largo del capítulo.

3.6.1 El Community Operator como operador de un sistema compartido

El modelo de gobierno descrito en el nivel de «Contexto» proviene del propio modelo organizacional de la UNAM, cuya autonomía administrativa por entidad bajo un marco normativo común fue identificada en el análisis de caso del Capítulo 2. Ahora bien, la figura que ese modelo dio origen a un ente gestor central con autoridad sobre una infraestructura cuyos activos físicos pueden no le pertenecen, un ente gestor como el descrito no es inédito ni exclusivo a la propuesta de este trabajo ni a la industria del computo. El Centro Nacional de Control de Energía opera el Sistema Eléctrico Nacional bajo ese mismo arreglo [116], y la Fig. 19 establece la correspondencia término a término entre ambos sistemas.

En ambos casos un conjunto de participantes aporta capacidad heterogénea a una infraestructura común; un operador imparcial coordina recursos; el acceso al recurso compartido se rige por reglas que garantizan condiciones equitativas; y la continuidad del servicio descansa sobre un criterio explícito de tolerancia a fallos, N-1 en el sistema eléctrico y replicación entre dominios de fallo independientes en la nube comunitaria [89].

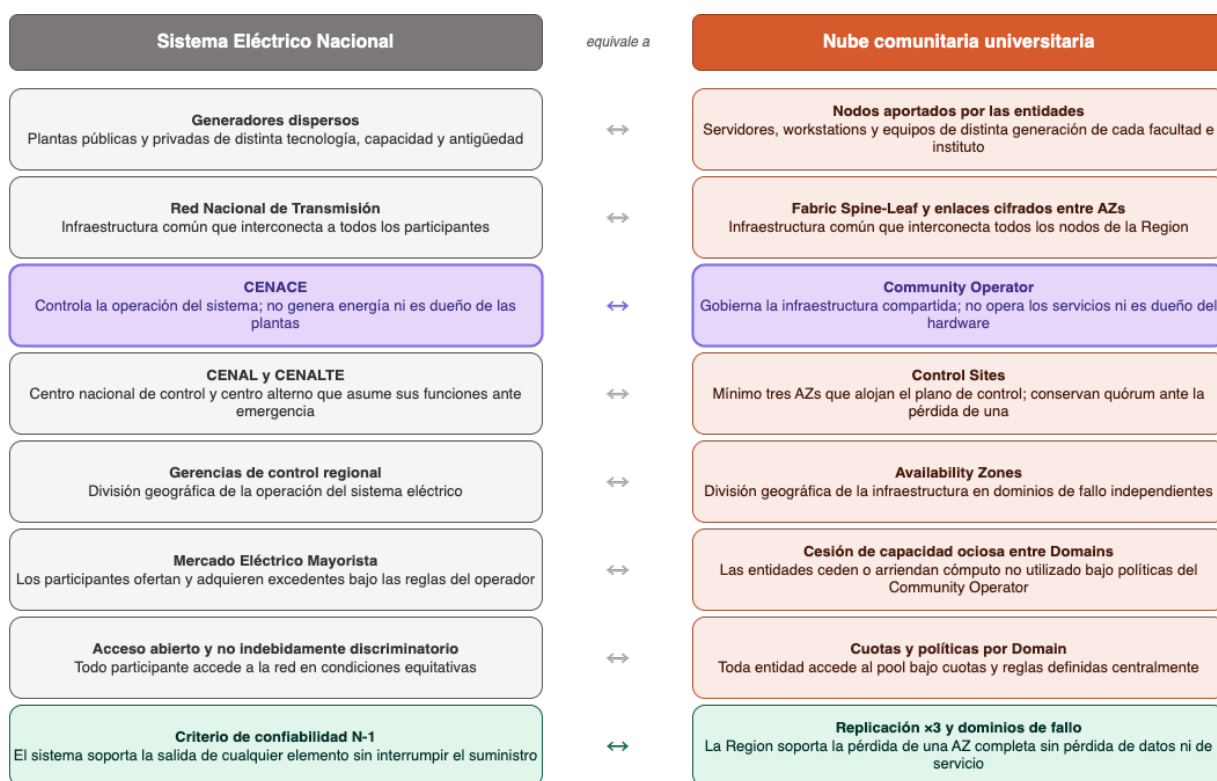


Fig. 19. Correspondencia estructural entre el Sistema Eléctrico Nacional y la nube comunitaria universitaria.

La coincidencia se extiende al detalle: los Control Sites cumplen la misma función que el centro nacional de control y su centro alterno; la cesión de capacidad ociosa entre Domains opera bajo la misma lógica que el Mercado Eléctrico Mayorista; y las cuotas por Domain equivalen al principio de acceso abierto y no indebidamente discriminatorio.

La correspondencia es estructural, pero conviene mencionar lo siguiente: la arquitectura no se diseñó a partir del CENACE, sino que ambos sistemas llegan a la misma forma organizativa porque enfrentan el mismo reto a resolver, es decir, coordinar recursos dispersos, heterogéneos y de propiedad ajena. El valor de la analogía para esta propuesta es, por lo tanto, demostrativo y no fundacional: evidencia que gestionar y operar un sistema de control de alta complejidad sin poseer sus activos físicos es un arreglo viable, sostenido a escala nacional en un sector crítico, y no una construcción teórica sin precedente.

3.6.2 Vista de conjunto de la Region

La Fig. 20 presenta la Region completa: la totalidad de la nube comunitaria bajo la autoridad de un único Community Operator, materializada como una instalación de OpenStack y un clúster de Kubernetes que abarcan todas las zonas de disponibilidad.

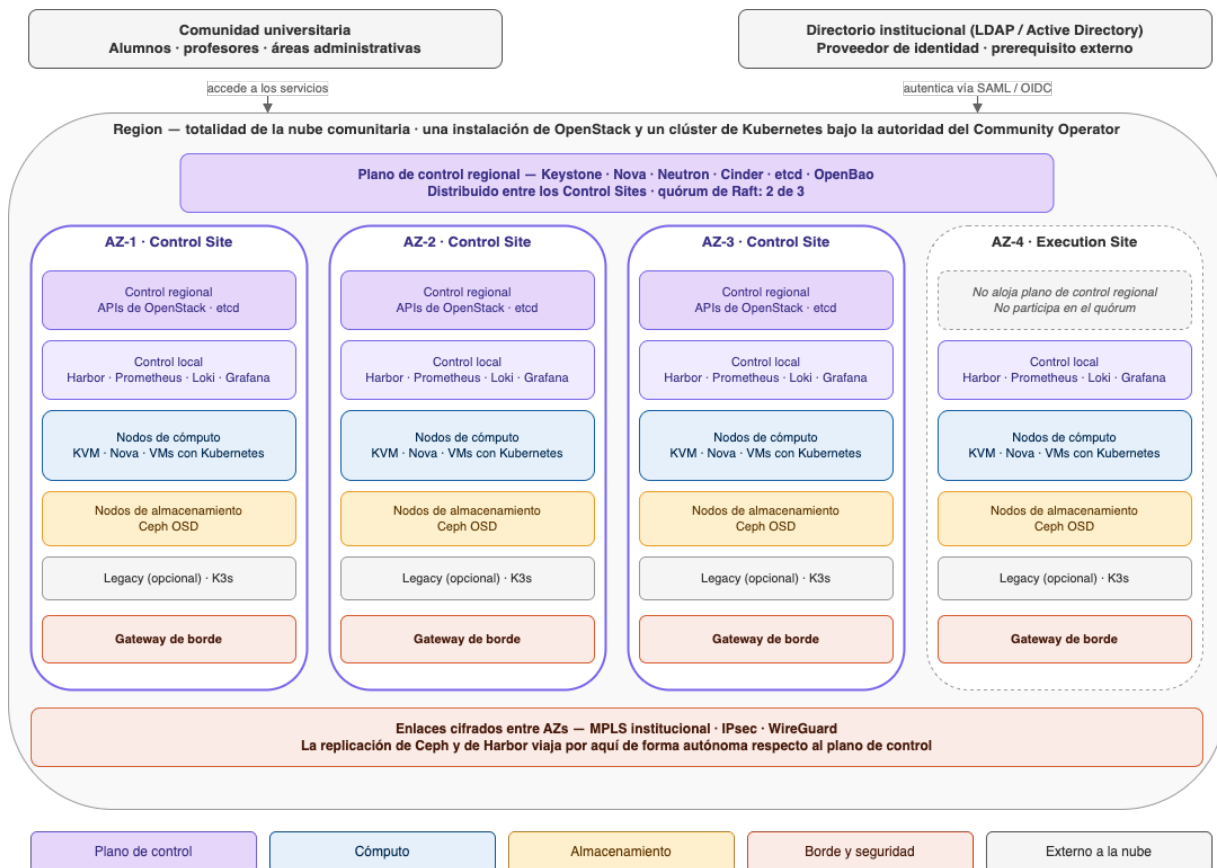


Fig. 20. Vista de conjunto de la Region: Control Sites, Execution Sites y enlaces cifrados entre AZs.

Tres propiedades de la arquitectura son visibles únicamente en esta vista. La primera es la **asimetría deliberada entre AZs**: los Control Sites alojan el plano de control regional y participan en el quórum de Raft, mientras que los Execution Sites aportan capacidad de cómputo y almacenamiento sin asumir responsabilidad sobre el estado global [22]. La segunda es la **homogeneidad del control local**: toda AZ, sea cual sea su tipo, opera su propia instancia de Harbor, Prometheus, Loki y Grafana, de modo que la pérdida de conectividad con el plano regional no interrumpe su operación. La tercera es la **autonomía de los flujos de datos entre zonas**: la replicación de Ceph y de Harbor viaja por los enlaces cifrados sin instrucción del plano de control, por lo que los datos permanecen protegidos aunque la gestión global quede suspendida.

3.6.3 Los tres niveles de abstracción en una sola vista

La Fig. 21 muestra los tres niveles del C4 Model empleados en este capítulo como un zoom progresivo sobre el mismo sistema [109].

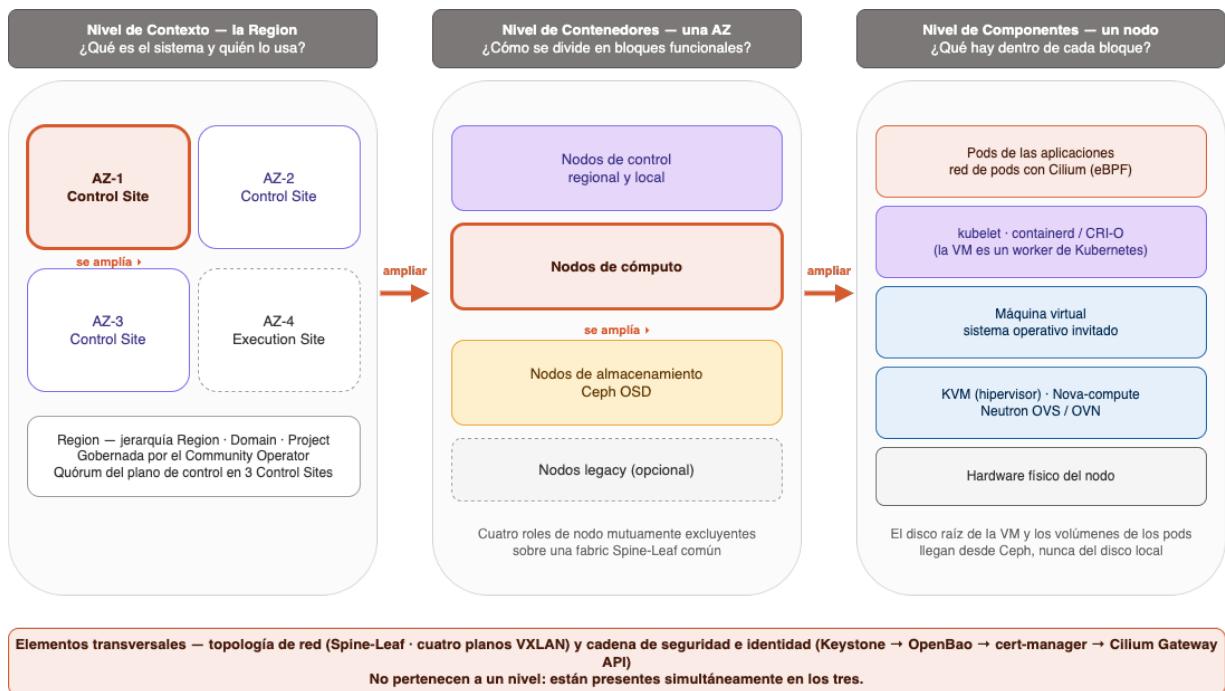


Fig. 21. Zoom progresivo por los tres niveles del C4 Model aplicados a la arquitectura.

Leída de izquierda a derecha, la figura recorre el capítulo completo: del nivel de Contexto, que responde qué es el sistema y quién lo gobierna, al nivel de Contenedores, que divide una AZ en cuatro roles de nodo mutuamente excluyentes, y de ahí al nivel de Componentes, que abre un nodo de cómputo para mostrar su apilamiento interno.

Cada ampliación selecciona un único elemento del nivel anterior: una AZ del conjunto de la Region y un nodo de cómputo del conjunto de roles de la AZ. Esta selección es la que hace manejable la descripción de un sistema distribuido sin perder rigor, dado que la topología interna de una AZ es normativa para todas: los elementos no ampliados quedan descritos por el mismo patrón.

La banda inferior recoge los dos elementos transversales. Ni la topología de red ni la cadena de seguridad e identidad pertenecen a un nivel determinado: están presentes en los tres simultáneamente, razón por la cual el capítulo los describe antes de descender al detalle de cada uno.

3.6.4 La pila tecnológica y su doble lectura

La Fig. 22 cierra la recapitulación descendiendo al interior de un nodo de cómputo y leyendo el mismo apilamiento de dos maneras que deliberadamente no coinciden entre sí.

Las mismas capas leídas de dos formas. Las dos lecturas no coinciden: el modelo de servicio corta la pila por superficie de consumo; el CNCF Landscape la corta por responsabilidad funcional.

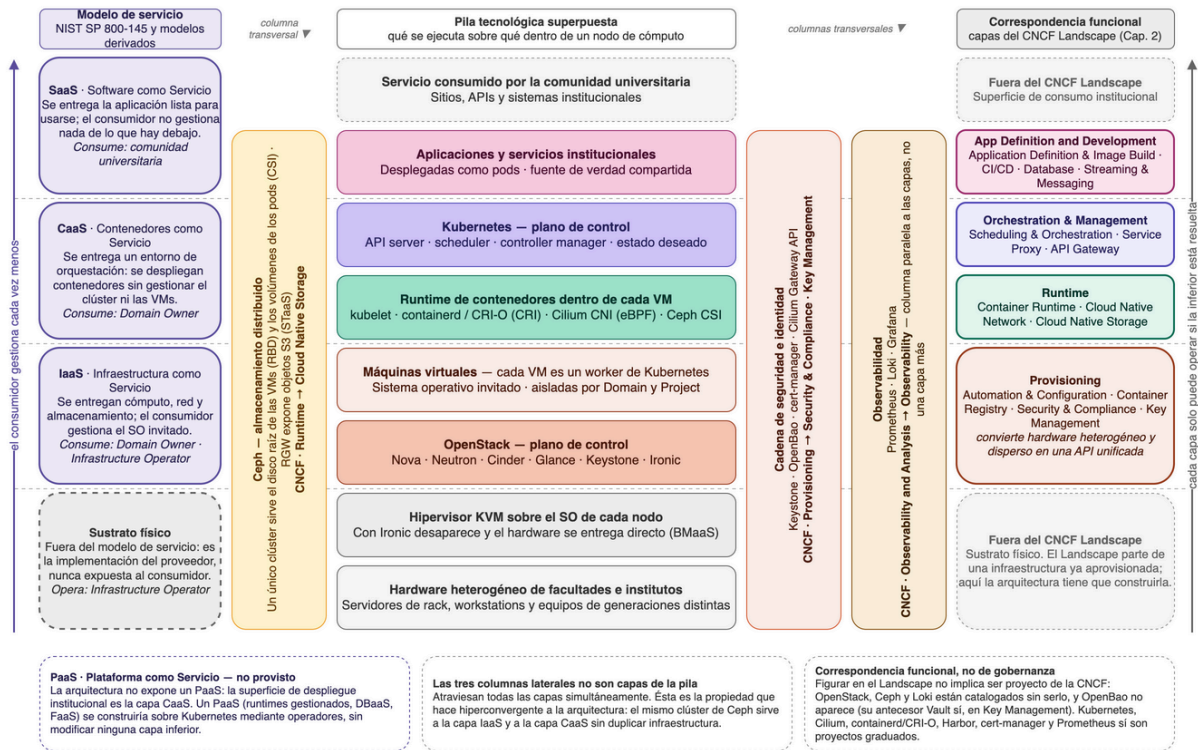


Fig. 22. Pila tecnológica superpuesta: modelos de servicio expuestos y correspondencia funcional con el CNCF Landscape.

La lectura de la izquierda corta la pila por **superficie de consumo** [27]. Las superficies no son alternativas entre sí sino que se apilan: OpenStack expone IaaS, Kubernetes se despliega sobre esa capa y expone CaaS [32], y los servicios institucionales que se ejecutan sobre el clúster se entregan como SaaS a la comunidad universitaria. Ironic y la interfaz RGW de Ceph añaden dos superficies más al mismo apilamiento sin infraestructura adicional [90]. La arquitectura no expone PaaS; de requerirse, se construiría sobre la capa CaaS sin modificar ninguna capa inferior.

La lectura de la derecha corta la misma pila por **responsabilidad funcional** y la vincula con la protoarquitectura definida en el Capítulo 2 [78]: OpenStack cubre la capa Provisioning, el runtime de contenedores junto con Cilium y el driver CSI de Ceph cubre la capa Runtime, Kubernetes cubre Orchestration & Management y las aplicaciones institucionales corresponden a App Definition & Development.

Que ambas lecturas no coincidan es el resultado más informativo de la figura. Los modelos de servicio describen qué se le entrega al consumidor; las capas del CNCF, qué responsabilidad cubre cada componente. Las tres columnas transversales atraviesan ambas lecturas de manera

simultánea y no pertenecen a ningún nivel de servicio en particular. Esa simultaneidad es la que hace hiperconvergente a la arquitectura: un único clúster de Ceph sirve el disco raíz de las máquinas virtuales de la capa IaaS y los volúmenes persistentes de los pods de la capa CaaS sin duplicar infraestructura.

3.6.5 Decisiones estructurales de la arquitectura

Las cuatro vistas anteriores descansan sobre un conjunto acotado de decisiones que determinan el comportamiento del sistema. La Tabla 28 las reúne junto con el criterio que las justifica y el nivel de abstracción en el que se manifiestan.

TABLA 28

DECISIONES ESTRUCTURALES DE LA ARQUITECTURA, SU JUSTIFICACIÓN Y EL NIVEL EN EL QUE SE MANIFIESTAN.

Decisión estructural	Criterio que la justifica	Nivel C4
Nube comunitaria como modelo de despliegue	Isomorfismo con el gobierno federado de la UNAM	Contexto
Jerarquía Region–Domain–Project–AZ–Nodo	Consenso de los proveedores de nube para aislar dominios de fallo	Contexto
Community Operator sin propiedad del hardware	Imparcialidad frente a las entidades participantes	Contexto
Mínimo de tres Control Sites	Requisito de mayoría del algoritmo Raft en etcd	Contexto
Cuatro roles de nodo mutuamente excluyentes	Evitar interferencia entre plano de control y cargas de trabajo	Contenedores
Cuatro planos de red segmentados sobre VXLAN	Aislamiento por función y reducción de la superficie de ataque	Transversal
Cadena Keystone → OpenBao → cert-manager → Gateway API	Cada eslabón deriva su confianza del anterior	Transversal
Ceph como capa única de almacenamiento	Hiperconvergencia: sirve a IaaS y CaaS sin duplicar infraestructura	Componentes
Kubernetes sobre VMs y no sobre bare metal	Preservar el aislamiento entre Domains que aporta el hipervisor	Componentes
Nodos legacy vía K3s con taints	Aprovechar hardware fuera de especificación sin exponer producción	Contenedores

Con esto queda descrita la arquitectura de referencia en sus tres niveles de abstracción y sus dos elementos transversales. Lo que resta por resolver no es de naturaleza técnica sino organizacional: bajo qué secuencia una institución con la estructura de la UNAM puede transitar desde su infraestructura actual hacia el modelo aquí propuesto, sin interrumpir los servicios en operación y sin exigir una inversión inicial que la haga inviable. Esa es la materia del Capítulo 4.

CAPÍTULO

IV

Estrategia de transición hacia el ecosistema distribuido

RESUMEN

Este capítulo define la estrategia metodológica para implementar la arquitectura de referencia, reconociendo que la adopción de un centro de datos distribuido es, fundamentalmente, un proceso de cambio organizacional.

CONCEPTOS CLAVE

Modelo de Madurez de Gartner	Herramienta de diagnóstico utilizada para evaluar el estado inicial de la organización y determinar las brechas que deben atenderse antes de iniciar cualquier despliegue técnico.
Gestión por Fases (PRINCE2)	Marco metodológico que estructura el proyecto en cinco etapas con criterios de salida estrictos, garantizando que cada capa de la arquitectura escale solo cuando los prerrequisitos de la etapa anterior están satisfechos.
Expansión Iterativa	Mecanismo de crecimiento del clúster donde la migración de un servicio a la nueva nube libera su servidor físico original, el cual se incorpora inmediatamente al pool de recursos.
Rehosting	Estrategia de modernización de bajo riesgo para consolidar hardware rápidamente sin modificar las aplicaciones.
Gestión del Cambio (Kotter)	Aplicación del modelo de 8 pasos para alinear la transformación tecnológica con la cultura institucional, garantizando el respaldo directivo y mitigando la resistencia organizacional.

4.1 Roadmap de implementación por fases

Para estructurar la secuencia de despliegue, este trabajo adopta los principios de gestión por etapas de PRINCE2 [123] como marco metodológico. PRINCE2 organiza los proyectos en etapas de gestión incrementales e iterativas, cada una con entregables definidos, criterios de calidad verificables y una compuerta de decisión (**stage gate**) al final que determina si el proyecto continúa hacia la siguiente etapa. Este enfoque es aplicable porque desacopla la complejidad total del proyecto en unidades de trabajo con alcance acotado, permitiendo que la organización valide cada capa de la arquitectura antes de comprometer recursos en la siguiente.

Tres principios de PRINCE2 articulan directamente la estructura del roadmap de modernización propuesto. El primero es **manage by stages**: cada fase del roadmap constituye una etapa de gestión con su propio plan, sus entregables y su criterio de salida; avanzar a la fase siguiente requiere que los criterios de la fase anterior estén satisfechos, no que haya transcurrido un plazo. El segundo es **focus on products**: cada fase se define por lo que produce y no por las actividades que ejecuta. El tercero es **manage by exception**: el ente gestor central establece las políticas y tolerancias globales, mientras el equipo técnico opera con autonomía dentro de esos límites, escalando únicamente cuando una decisión excede su autoridad delegada.

TABLA 29

RESUMEN DEL ROADMAP: PRODUCTOS, CRITERIOS DE SALIDA, TOLERANCIAS Y PRERREQUISITOS POR FASE.

Fase	Producto PRINCE2	Criterio de salida	Tolerancias	Prerequisito
Fase 0 — Gobierno y diagnóstico	Ente gestor constituido; diagnóstico de madurez documentado	Checklist de constitución completado y firmado; nivel de madurez identificado	Sin tolerancia: la fase no puede declararse completa con productos parciales	Decisión institucional de iniciar el proyecto
Fase 1 — Prerrequisitos de conectividad e identidad	Red Spine-Leaf verificada; directorio institucional operativo	Conectividad entre nodos validada; atributos de federación verificados	Sin tolerancia: prerrequisitos técnicos son condición binaria de entrada a Fase 2	Fase 0 completada
Fase 2 — Despliegue de la capa IaaS	OpenStack + Ceph operativos; portafolio de servicios rehostado	Primera AZ en producción; todos los servicios migrados; todos los nodos incorporados	±10% en plazo; cero tolerancia en integridad de datos durante migración	Fase 1 completada
Fase 3 — Activación de la capa CaaS	Kubernetes operativo; primeros servicios contenerizados	Clúster K8s con al menos un servicio en producción	±15% en plazo; el clúster IaaS subyacente no puede degradarse	Fase 2 estable ≥ 30 días
Fase 4 — Consolidación y madurez operativa	Observabilidad y gobierno maduros	Procesos operativos documentados, ejecutados y auditados al menos una vez	±20% en plazo; no se acepta regresión en disponibilidad de servicios	Fase 3 completada

En la Tabla 29 se muestra un resumen de las fases propuestas y sus criterios.

4.2 Fase 0 — Constitución del ente gestor y diagnóstico de madurez

4.2.1 Business Case

El **business case** de la Fase 0 es que ningún proyecto de modernización de infraestructura puede sostenerse sin dos condiciones previas de naturaleza organizacional: una instancia con autoridad para tomar decisiones y una comprensión objetiva del estado actual. Intentar desplegar tecnología sin gobierno constituido, reproduce los problemas de fragmentación que motivaron el proyecto. Intentarlo sin diagnóstico produce una estrategia desalineada con las capacidades reales de la organización. El costo de esta fase es principalmente tiempo de personal.

4.2.2 Constitución del ente gestor

El ente gestor central debe estar formalmente constituido como primer acto del proyecto. En términos PRINCE2, este ente cumple la función del **Project Board**: es quien aprueba el avance entre fases, establece las tolerancias globales y toma las decisiones que exceden la autoridad del equipo técnico.

Su constitución requiere tres acciones concretas:

- Definir su composición y línea de reporte institucional.
- Aprobar las políticas globales mínimas que regirán la operación de la nube (estándares de hardware, políticas de seguridad, modelo de cuotas).
- Designar al equipo técnico responsable del despliegue y operación inicial, que actuará como **Project Manager** en el lenguaje de PRINCE2.

La ausencia del ente gestor formalmente constituido convierte el despliegue en un proyecto técnico sin respaldo institucional, vulnerable a ser abandonado ante el primer obstáculo organizacional.

4.2.3 Diagnóstico de madurez

Una vez constituido el ente gestor, su primera actividad formal es realizar el diagnóstico de madurez de la organización. Este diagnóstico no es un prerrequisito técnico sino estratégico ya

que determina el punto de entrada real al roadmap, identifica las brechas que deben atenderse antes del despliegue tecnológico, y establece las expectativas organizacionales sobre el ritmo y el alcance del proyecto.

Es indispensable realizar un diagnóstico profundo de las necesidades de TI para determinar que un enfoque de modernización es adecuado en cada escenario, ya que las consecuencias de elegir el enfoque equivocado incluyen sobrecostos presupuestarios, retrasos en la entrega, interrupciones en los procesos institucionales y la finalización prematura del proyecto [124].

Como marco de referencia para estructurar este diagnóstico se retoma el **Gartner Infrastructure Maturity Model** [125], que organiza la evolución de la infraestructura de TI en seis niveles progresivos evaluados sobre tres ejes transversales: **People** (estructura organizacional y propiedad de los activos), **Process** (madurez de los procesos de gestión) y **Technology** (nivel de consolidación y automatización tecnológica).

TABLA 30

NIVELES DEL GARTNER INFRASTRUCTURE MATURITY MODEL EVALUADOS SOBRE LOS EJES PEOPLE, PROCESS Y TECHNOLOGY [125].

Nivel	People	Process	Technology
Basic	Sin estructura; operación individual y descoordinada	Sin procesos formales; gestión ad hoc e incidentes sin registro	Infraestructura no coordinada; configuración caótica
Standardized	TI posee los activos; procesos y herramientas se comparten	Estándares de ciclo de vida; SLAs básicos; gestión de eventos	Configuraciones estándar; herramientas básicas de monitoreo
Rationalized	Estructura organizacional y ownership consolidados en TI	Procesos maduros e integrados de gestión de sistemas	Herramientas integradas de gestión; activos consolidados
Virtualized	Organización alineada al uso de recursos en pool	Gestión holística de capacidad; chargeback flexible	Cómputo, almacenamiento y red virtualizados en pool
Service-Based	TI alineada a la entrega de servicios	Medición y garantía de servicios E2E; SLAs flexibles	Herramientas de gestión de servicios de extremo a extremo
Policy/Value-Based	TI influye proactivamente en el uso de tecnología para el negocio	Servicios E2E centralmente gestionados y balanceados	El negocio tiene interfaz directa a la priorización de servicios

El objetivo del roadmap es llevar a la organización desde su nivel actual hasta el nivel **Virtualized** al término de la Fase 2, y hacia **Service-Based** al concluir la Fase 3. El nivel **Policy/Value-Based** constituye el horizonte de madurez de largo plazo, fuera del alcance de este trabajo.

En el caso de estudio que motiva esta tesis, el diagnóstico sitúa a la organización en un nivel **Standardized**: TI posee activos de cómputo distribuidos entre dependencias, existen herramientas y configuraciones básicas compartidas, y hay procesos elementales de gestión de

eventos. A pesar de lo anterior, la propiedad de los activos no está consolidada¹⁷ entre áreas, los procesos de gestión no están integrados y la infraestructura no está unificada. Este nivel sitúa a la organización en condiciones de iniciar la Fase 1, con atención particular a los ejes **Process** y **People**, que presentan mayor rezago respecto a los requisitos de la arquitectura propuesta.

4.2.4 Stage gate

El **stage gate** de la Fase 0 requiere dos productos completos y verificados: el acta de constitución del ente gestor firmada por la autoridad institucional correspondiente, y el documento de diagnóstico de madurez con el nivel identificado en cada uno de los tres ejes. No existe tolerancia parcial en ninguno de los dos productos.

4.3 Fase 1 — Prerrequisitos de conectividad e identidad

4.3.1 Business Case

El **business case** de la Fase 1 está comprendida por los prerrequisitos técnicos de red e identidad, sin estos prerrequisitos el despliegue de la capa IaaS no puede iniciarse.

4.3.2 Infraestructura de red

El prerrequisito de red más crítico es la existencia de una fabric Spine-Leaf con soporte para VXLAN entre los edificios que alojarán los nodos de la primera AZ. Si la organización no cuenta con esta infraestructura, su despliegue debe completarse en esta fase antes de avanzar.

Los criterios de aceptación son tres: conectividad verificada entre todos los nodos candidatos mediante pruebas de latencia y ancho de banda; capacidad de segmentación en al menos cuatro VXLANs dedicadas (gestión, almacenamiento, tenant y externa); y un gateway de borde por AZ con capacidad de cifrado de tráfico.

4.3.3 Directorio institucional

El directorio institucional debe estar operativo y accesible desde la red de gestión de la primera AZ antes de desplegar Keystone. Esto implica verificar que el servidor LDAP o Active Directory

¹⁷En el contexto del modelo de Gartner, **racionalizado** o consolidado significa que la propiedad, el uso y la gestión de los activos de TI han sido revisados para eliminar redundancias, asignando responsabilidad clara a una instancia central de TI en lugar de distribuirla entre áreas que administran sus propios recursos de forma independiente.

existente expone los atributos de usuario necesarios para la federación (identificador único, grupos de pertenencia, correo institucional) y que su disponibilidad satisface los requisitos de un servicio de infraestructura crítica.

Si la organización no cuenta con directorio centralizado, su implementación es un proyecto independiente que debe completarse en paralelo a la preparación de red.

4.3.4 Stage gate

El **stage gate** de la Fase 1 es binario: los dos productos esperados deben estar completos antes de iniciar cualquier actividad de la Fase 2. No existe tolerancia parcial. Una red sin segmentación completa o un directorio con cobertura parcial de usuarios son prerequisites incumplidos, independientemente del porcentaje de avance.

4.4 Fase 2 — Despliegue de la capa IaaS

4.4.1 Business Case

El **business case** de la Fase 2 es la consolidación de la infraestructura de cómputo dispersa en un pool unificado, gestionable y accesible bajo demanda. Los beneficios esperados al término de esta fase son:

- Reducción del tiempo de aprovisionamiento de servidores de días a minutos
- Eliminación de la administración manual de hardware individual
- Primer nivel de continuidad operativa mediante la replicación de almacenamiento.

El indicador de éxito es que la totalidad del portafolio de servicios existente opere sobre la nueva infraestructura sin degradación de función, con todos los servidores físicos incorporados al clúster.

4.4.2 Productos de la fase

Los tres productos de la Fase 2 son:

- El clúster OpenStack con alta disponibilidad.
- El clúster Ceph.
- El inventario completo de servicios migrados mediante rehosting con su documentación de validación.

4.4.3 Incorporación de nodos y AZs

La incorporación de nodos al clúster debe seguir un proceso estandarizado y documentado. Ansible debe gestionar la configuración base de cada nodo de forma idempotente [45], [83], es decir:

- Sistema operativo
- Dependencias
- Políticas de seguridad

de modo que ejecutar el playbook sobre un nodo ya configurado no produzca cambios no deseados.

La incorporación de una segunda AZ es el primer hito de distribución real del centro de datos. Requiere verificar la conectividad cifrada entre los gateways de borde de ambas zonas, extender el clúster Ceph para incluir los nuevos nodos de almacenamiento como dominio de fallo independiente, y registrar la nueva AZ como aggregate group en Nova. La replicación asíncrona de Ceph entre zonas debe verificarse antes de declarar la AZ operativa.

4.4.4 Estrategias de modernización y criterios de selección

Antes de ejecutar la migración de servicios, la organización debe determinar qué estrategia de modernización aplicar a cada aplicación de su portafolio. La selección incorrecta de estrategia es una de las causas más frecuentes de fracaso en programas de modernización: produce sobrecostos, retrasos, baja generación de valor y, en casos extremos, la terminación anticipada del programa [124].

Gartner identifica cinco enfoques principales que difieren en su capacidad de intervenir sobre tres capas de la aplicación: tecnología, arquitectura y funcionalidad [124], [125].

El **rehost**, también denominado **lift and shift**, redistribuye la aplicación sobre nueva infraestructura sin modificar su código, sus características ni sus funciones. Interviene exclusivamente en la capa tecnológica y presenta el menor costo, riesgo e impacto sobre el negocio de todos los enfoques, aunque también el menor potencial de transformación arquitectónica o funcional.

El **replatform**, también denominado **lift and reshape**, migra la aplicación a una nueva plataforma de ejecución realizando cambios mínimos de compatibilidad en el código, sin modificar su estructura interna ni sus funciones. Interviene en la capa tecnológica y, de forma limitada, en la arquitectónica.

El **rearchitect** reestructura y optimiza el código existente sin cambiar su comportamiento externo, con el objetivo de eliminar deuda técnica y mejorar atributos no funcionales como escalabilidad y mantenibilidad. Interviene en las capas tecnológica y arquitectónica, con costo y riesgo significativamente mayores que los dos enfoques anteriores.

El **rebuild** rediseña o reescribe la aplicación desde cero preservando su alcance y especificaciones. Es el enfoque con mayor potencial de transformación en las tres capas, pero también el de mayor costo, riesgo y tiempo de entrega de valor.

El **replace** sustituye la aplicación por una solución COTS o SaaS. Tiene el mayor potencial de impacto sobre procesos de negocio y experiencia de usuario, y su perfil de riesgo depende de la madurez del mercado en el segmento de la aplicación a reemplazar.

TABLA 31

ESTRATEGIAS DE MODERNIZACIÓN: POTENCIAL DE CAMBIO POR CAPA Y PERFIL DE RIESGO-COSTO [124].

Estrategia	Potencial de cambio			Consideraciones		
	Tecnología	Arquitectura	Funcionalidad	Valor de negocio	Costo	Riesgo
Rehost	Alto	Ninguno	Ninguno	Bajo	Bajo	Bajo
Replatform	Alto	Parcial	Ninguno	Bajo	Medio	Bajo
Rearchitect	Alto	Alto	Parcial	Medio	Alto	Medio
Rebuild	Alto	Alto	Alto	Alto	Alto	Alto
Replace	Parcial	Parcial	Alto	Alto	Medio	Alto

La elección de estrategia para cada servicio debe basarse en un análisis de causa-efecto que identifique si el problema raíz reside en la tecnología, la arquitectura o la funcionalidad de la aplicación [124]. Los problemas de costo, complejidad y riesgo operativo tienen causa tecnológica y son resolubles mediante rehost o replatform. Los problemas de agilidad y deuda estructural tienen causa arquitectónica y requieren rearchitect. Los problemas de adecuación al negocio e innovación tienen causa funcional y requieren rebuild o replace.

El roadmap propuesto adopta el **rehost** como estrategia primaria de la Fase 2 por tres razones alineadas con los principios PRINCE2 de **manage by stages** y **manage by exception**. Primera, su perfil de riesgo y costo mínimos es compatible con los recursos disponibles en una organización académica sin experiencia previa en operación de nube. Segunda, consolida toda la infraestructura dispersa y produce así valor tangible en el corto plazo, sin requerir ninguna modificación sobre las aplicaciones existentes. Tercera, no es excluyente de enfoques más invasivos en fases posteriores: un servicio rehostado en la Fase 2 puede ser replatformado en la Fase 3 o rearquitecturado en el futuro cuando la organización haya acumulado la madurez operativa necesaria.

Los enfoques de rearchitect, rebuild y replace constituyen líneas de evolución futura del ecosistema, fuera del alcance de este trabajo, y son identificados como tal en el Capítulo 5.

4.4.5 Estrategia de expansión por liberación progresiva de nodos

La adopción del rehost como estrategia única de la Fase 2 responde además a una restricción característica del escenario institucional: la organización no dispone de infraestructura adicional para construir el clúster mientras los servicios existentes siguen operando sobre los servidores actuales. Esta restricción determina el modelo de expansión adoptado, que opera en dos momentos secuenciales y se retroalimenta de su propio avance.

El **primer momento** es la constitución del clúster base. Con el subconjunto mínimo de servidores necesario para desplegar un clúster OpenStack funcional con alta disponibilidad en el plano de control y replicación Ceph en al menos tres nodos de almacenamiento, se levanta la infraestructura inicial. Este clúster base no tiene capacidad suficiente para absorber toda la carga de trabajo existente, pero sí para alojar los primeros servicios migrados. Su constitución es el prerequisite técnico de todo lo que sigue.

El **segundo momento** es la migración iterativa con expansión continua del clúster. Una vez puesto en operación el clúster base, se inicia el proceso de rehosting servicio por servicio. Cada servicio migrado libera el servidor físico que lo alojaba; ese servidor es inmediatamente incorporado al clúster como nodo de cómputo o de almacenamiento adicional, expandiendo la capacidad disponible para alojar el siguiente servicio. El proceso se repite hasta que todos los servicios han sido migrados y todos los servidores han sido incorporados al clúster.

TABLA 32

MODELO DE EXPANSIÓN ITERATIVA DEL CLÚSTER MEDIANTE LIBERACIÓN PROGRESIVA DE NODOS.

Iteración	Acción	Resultado
0	Constitución del clúster base con nodos mínimos	Clúster operativo con capacidad inicial; primera AZ lista
1	Rehost del primer servicio; servidor físico liberado incorporado al clúster	Capacidad del clúster aumenta; siguiente servicio puede migrarse
2..n	Repetición: rehost del siguiente servicio; incorporación del nodo liberado	Expansión progresiva hasta absorber todo el portafolio
n+1	Último servicio migrado; último servidor incorporado	Clúster completo; infraestructura física totalmente consolidada

Este modelo tiene tres consecuencias directas sobre la planificación del proyecto que el equipo técnico y el ente gestor deben anticipar.

La primera es que **el clúster base debe dimensionarse con precisión**. Un clúster base sobredimensionado requiere inmovilizar demasiados servidores antes de poder iniciar las migraciones, prolongando el período en que los servicios siguen operando sobre infraestructura no gestionada. Un clúster base subdimensionado puede no tener capacidad suficiente para alojar el primer servicio migrado. El dimensionamiento correcto es el mínimo que satisface los requisitos de alta disponibilidad del plano de control y replicación del almacenamiento.

La segunda es que **el orden de migración determina la velocidad de expansión**. Los servicios que operan sobre servidores con más recursos deben migrarse antes que los que operan sobre servidores con menos recursos, de modo que cada iteración aporte la mayor capacidad posible al clúster. Este criterio de ordenamiento se superpone a los criterios de criticidad; cuando entren en conflicto, el equipo técnico debe elevar la decisión al ente gestor como **exception** en términos PRINCE2.

La tercera es que **la continuidad operativa de cada servicio durante su migración es no negociable**. El mecanismo que garantiza esta continuidad es el modelo de validación en paralelo, que consiste en operar el servicio simultáneamente en el servidor original y en la VM de destino durante un período de observación, antes de desactivar el original. Ningún servidor debe incorporarse al clúster antes de que el servicio que alojaba haya sido declarado estable en su nueva VM.

4.4.6 Proceso de rehosting por servicio

El proceso de rehosting sigue cuatro pasos para cada servicio, que en términos PRINCE2 constituyen un **work package** independiente con sus propios criterios de aceptación.

Primero, **inventario**: documentar los requisitos de cómputo, almacenamiento y red del servicio en su estado actual. Segundo, **aprovisionamiento**: crear la VM correspondiente en OpenStack con los recursos equivalentes, asignada al Project del área responsable del servicio. Tercero, **migración**: transferir los datos y configuraciones del servicio a la nueva VM, verificando integridad. Cuarto, **validación**: ejecutar el servicio en la nueva VM en paralelo al original durante un período de observación antes de desactivar el servicio original y liberar el servidor físico.

Los servicios migrados mediante rehosting quedan disponibles para ser contenerizados progresivamente en la Fase 3.

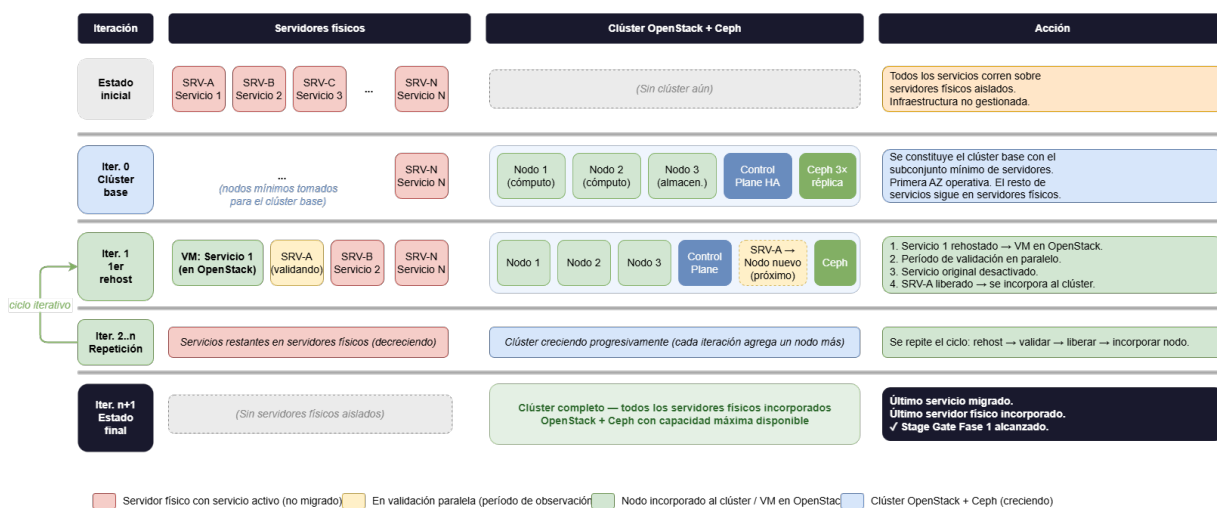


Fig. 23. Proceso de rehosting

4.4.7 Stage gate

El **stage gate** de la Fase 2 requiere que el plano de control de OpenStack esté en alta disponibilidad verificada, que el clúster Ceph tenga replicación activa entre al menos dos dominios de fallo, y que la totalidad del portafolio de servicios haya completado el proceso de rehosting con validación documentada y todos los servidores físicos incorporados al clúster. La tolerancia de plazo es $\pm 10\%$; la integridad de datos durante la migración no admite tolerancia.

4.5 Fase 3 — Activación de la capa CaaS

4.5.1 Business Case

El **business case** de la Fase 3 es la habilitación de un modelo de consumo de infraestructura orientado a servicios contenerizados, que reduce la fricción entre el desarrollo y el despliegue de aplicaciones. Al término de esta fase, las áreas participantes pueden desplegar servicios sin gestionar la infraestructura subyacente directamente, consumiendo la capa IaaS a través de abstracciones de Kubernetes. El indicador de éxito es que al menos un servicio opere en producción sobre el clúster con despliegue automatizado reproducible.

4.5.2 Productos de la fase

Los tres productos de la Fase 3 son: el clúster Kubernetes con plano de control en alta disponibilidad y cadena de seguridad completa, el registro interno Harbor operativo con al menos un flujo de imagen publicado, y el catálogo inicial de servicios contenerizados con sus manifiestos versionados en el repositorio institucional.

4.5.3 Despliegue de Kubernetes y la cadena de seguridad

El clúster de Kubernetes se despliega sobre VMs aprovisionadas por Nova en los nodos de cómputo de la primera AZ. El plano de control debe desplegarse en un mínimo de tres VMs dedicadas en configuración de alta disponibilidad. Los nodos worker se aprovisionan como VMs adicionales y pueden escalar horizontalmente incorporando nuevas instancias al clúster sin interrupción del servicio.

La cadena de seguridad debe desplegarse en el mismo orden en que fue descrita en el Capítulo 3: primero OpenBao como CA raíz y almacén de secretos [68], luego cert-manager integrado con OpenBao [87], y finalmente Cilium como CNI y Gateway API [96]. Este orden no es arbitrario: cada componente depende del anterior para operar correctamente.

Harbor debe desplegarse como registro interno antes de que cualquier imagen de contenedor sea utilizada en el clúster. Todo flujo de trabajo que reference imágenes de registros públicos externos debe redirigirse al registro interno desde el inicio, no como mejora posterior.

4.5.4 Contenerización progresiva mediante replatforming

El replatforming es la estrategia de modernización de la Fase 3: cada servicio migrado mediante rehosting en la Fase 2 es candidato a ser containerizado y red desplegado en Kubernetes. A diferencia del rehosting, el replatforming implica modificaciones en el empaquetado del servicio, como la construcción de una imagen de contenedor y la definición de manifiestos de Kubernetes, pero no en su arquitectura interna [124]. Cada servicio replatformeado constituye un **work package** independiente con sus propios criterios de aceptación.

La selección del orden en que los servicios son replatformeados debe guiarse por tres criterios. El primero es la **independencia de estado**: los servicios sin estado persistente complejo son los más simples de containerizar y deben abordarse primero para acumular experiencia. El segundo es la **críticidad**: los servicios de menor criticidad operativa ofrecen mayor tolerancia al error durante la containerización. El tercero es el **valor demostrativo**: elegir tempranamente un servicio visible para los usuarios finales genera evidencia concreta del valor de la plataforma ante la institución.

Los servicios que requieren rediseño profundo de su arquitectura, como la migración de bases de datos monolíticas o la separación de componentes acoplados, corresponden a estrategias de rearchitect o rebuild [124] que van más allá del alcance de este trabajo y constituyen una línea de investigación futura identificada en el Capítulo 5.

4.5.5 Stage gate

El **stage gate** de la Fase 3 requiere que el clúster Kubernetes esté operativo con la cadena de seguridad completa, que Harbor esté en producción con al menos un flujo de imagen automatizado, y que al menos un servicio opere en el clúster con su manifiesto versionado en el repositorio institucional. La tolerancia de plazo es $\pm 15\%$; el clúster IaaS subyacente no puede haber experimentado degradación de disponibilidad como consecuencia del despliegue de la capa CaaS.

4.6 Fase 4 — Consolidación y madurez operativa

4.6.1 Business Case

El **business case** de la Fase 4 es la sostenibilidad a largo plazo de la inversión realizada en las fases anteriores. Una infraestructura técnicamente correcta pero operada sin procesos formalizados es frágil ante la rotación de personal, el crecimiento de la demanda y los cambios

organizacionales. Al término de esta fase, el centro de datos distribuido deja de depender de personas individuales y pasa a depender de procesos institucionales documentados y auditables. El indicador de éxito es que un nuevo integrante del equipo técnico pueda ejecutar los procesos operativos siguiendo únicamente la documentación existente.

4.6.2 Productos de la fase

Los tres productos de la Fase 4 son: el stack de observabilidad con alertas calibradas y paneles por perfil de usuario, el manual de operaciones con los tres procesos formalizados del ente gestor, y el registro de lecciones aprendidas del proyecto completo.

4.6.3 Observabilidad unificada

El stack de observabilidad, compuesto por Prometheus, Loki y Grafana, debe estar completamente operativo al inicio de la Fase 4, idealmente activado desde la Fase 2 para que los datos históricos estén disponibles cuando se necesiten. Lo que caracteriza a la Fase 4 en esta dimensión no es la instalación sino la madurez: definición de alertas con umbrales calibrados a partir de datos históricos reales, construcción de paneles orientados a los distintos perfiles de usuario (operadores de infraestructura, administradores de Domain, usuarios finales), y establecimiento de procesos de respuesta ante incidentes basados en la telemetría disponible.

La observabilidad no es un componente técnico sino una práctica organizacional. Su valor real emerge cuando los equipos la consultan de forma habitual y cuando sus alertas son accionables, no cuando el software está instalado.

4.6.4 Gobernanza operativa

La madurez operativa requiere que el ente gestor central formalice tres procesos que en las fases anteriores pueden haberse gestionado de forma ad hoc. En términos PRINCE2, esta formalización corresponde al proceso de **Closing a Project** extendido: el proyecto de implementación concluye, pero la operación continua requiere estructuras permanentes.

El primero es el **proceso de incorporación de nuevos participantes**: definir los criterios técnicos y organizacionales que debe satisfacer una entidad para incorporarse a la nube comunitaria, el procedimiento de aprovisionamiento de su Domain y cuotas iniciales, y el período de observación antes de declarar la incorporación exitosa.

El segundo es el **proceso de gestión de cambios en la infraestructura**: toda modificación al stack tecnológico, sea una actualización de versión, un cambio de configuración o la incorporación de nuevos nodos, debe seguir un proceso documentado con evaluación de impacto, ventana de mantenimiento y plan de rollback.

El tercero es el **proceso de revisión periódica de cuotas y capacidad**: el ente gestor debe revisar regularmente la relación entre los recursos aportados por cada participante y los consumidos, ajustando cuotas y planificando expansión de capacidad antes de que la demanda supere la oferta disponible.

4.6.5 Registro de lecciones aprendidas

PRINCE2 establece el registro de lecciones aprendidas (**lessons learned**) como un producto de gestión obligatorio al cierre de cada etapa y del proyecto en su conjunto [123]. En el contexto de esta tesis, este registro tiene valor particular porque el escenario de implementación, una infraestructura distribuida en una organización académica con recursos limitados, es suficientemente específico para que las lecciones documentadas constituyan una contribución práctica independiente del resultado técnico.

El registro debe documentar al menos: las desviaciones respecto al plan original y sus causas, los prerequisites que resultaron más difíciles de satisfacer, las resistencias organizacionales encontradas y las estrategias que resultaron efectivas para resolverlas, y las estimaciones de plazo que resultaron más alejadas de la realidad.

4.6.6 Stage gate

El **stage gate** de la Fase 4 es el cierre formal del proyecto. Requiere que los tres procesos operativos estén documentados y hayan sido ejecutados al menos una vez con registro de resultado, que el stack de observabilidad tenga alertas activas con al menos un ciclo de calibración completado, y que el registro de lecciones aprendidas esté aprobado por el ente gestor. La tolerancia de plazo es $\pm 20\%$; no se acepta regresión en la disponibilidad de los servicios en producción como condición de cierre.

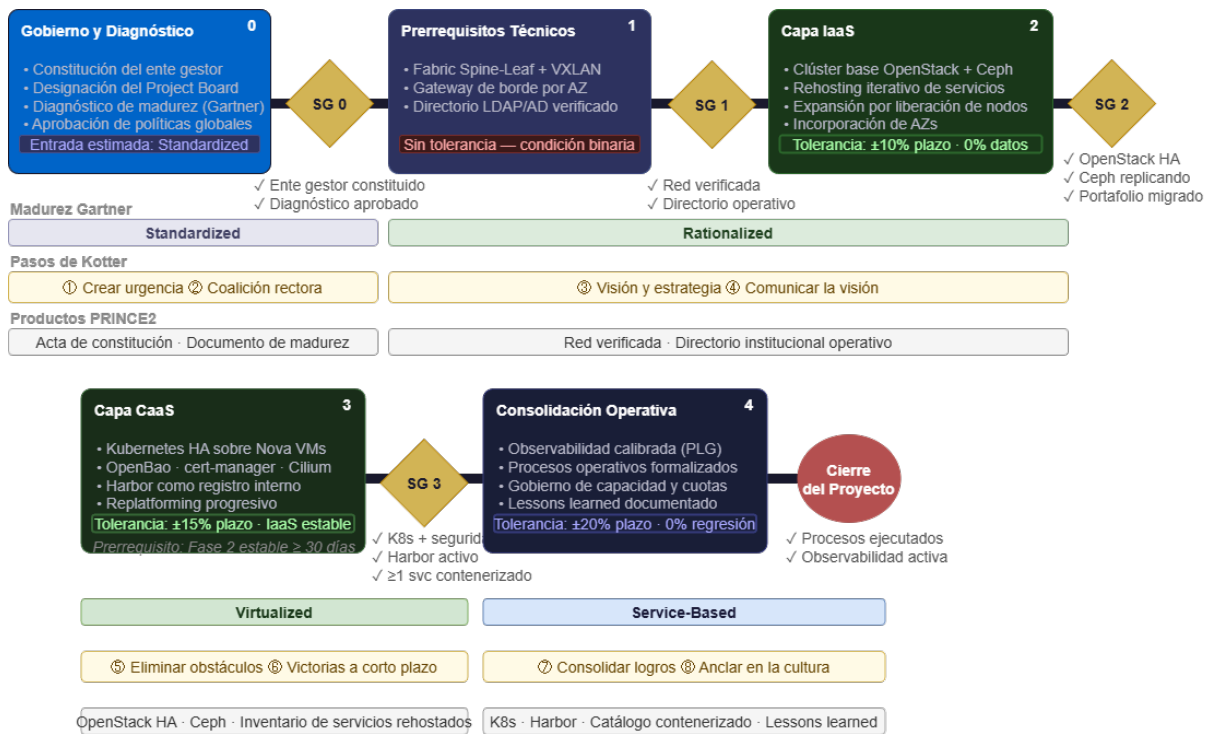


Fig. 24. Fases de modernización

4.7 Gestión del cambio organizacional

La implementación de un centro de datos distribuido es, antes que un proyecto tecnológico, un proceso de cambio organizacional. Las fases técnicas descritas en las secciones anteriores definen **qué** desplegar y en **qué** orden; esta sección define **cómo** conducir la organización a través de ese proceso. Para ello se adopta el modelo de ocho pasos de Kotter [126], que ofrece una secuencia estructurada para gestionar transformaciones en organizaciones complejas. Mientras PRINCE2 provee el andamiaje metodológico para el proyecto técnico, el modelo de Kotter provee el marco para la dimensión humana e institucional del mismo cambio.

4.7.1 Crear sentido de urgencia

El primer paso es construir una comprensión compartida de por qué la situación actual es insostenible. En el contexto de esta tesis, los argumentos son concretos: servidores dispersos sin gestión centralizada, ausencia de continuidad operativa, duplicación de esfuerzo administrativo entre áreas y dependencia de conocimiento tácito de personas individuales. Este diagnóstico debe comunicarse con datos propios de la organización, como el inventario de equipos, los incidentes documentados y los costos de operación por área, antes de proponer

alguna solución. En términos PRINCE2, esta actividad corresponde al proceso **Starting Up a Project**: justificar que el proyecto merece ser iniciado.

4.7.2 Formar una coalición rectora

El cambio no puede ser impulsado por el área de TI de forma unilateral. Se requiere una coalición que incluya autoridad institucional (dirección o rectoría), representantes técnicos de las áreas que aportarán infraestructura, y al menos un actor con credibilidad académica que legitime el proyecto ante la comunidad universitaria. Esta coalición es el núcleo del ente gestor central descrito en la Fase 0 y cumple en la práctica la función del **Project Board** de PRINCE2.

4.7.3 Desarrollar una visión y estrategia

La visión debe ser comunicable en términos no técnicos: **una infraestructura compartida que permite a cada área desplegar servicios con la misma agilidad que un proveedor de nube, sin ceder el control de sus datos ni depender de presupuestos externos**. La estrategia es el roadmap de cinco fases estructurado según PRINCE2. Tener ambos elementos elaborados antes de iniciar la Fase 0 evita que el proyecto se perciba como una iniciativa técnica sin dirección institucional clara.

4.7.4 Comunicar la visión

La visión debe comunicarse de forma consistente a través de los canales institucionales relevantes: sesiones informativas con las áreas participantes, presentaciones ante órganos de gobierno, y documentación accesible que describa el modelo de participación y sus beneficios concretos para cada área. La comunicación no es un evento único sino un proceso continuo a lo largo de todas las fases; cada **stage gate** aprobado es una oportunidad de comunicación.

4.7.5 Eliminar obstáculos

Los obstáculos más frecuentes en este contexto son de naturaleza organizacional, no técnica: áreas que perciben la consolidación como pérdida de autonomía, procesos de adquisición que no contemplan infraestructura compartida, y marcos normativos que no tienen categoría para una nube comunitaria interna. Identificar y gestionar estos obstáculos es responsabilidad de la coalición rectora. En términos PRINCE2, los obstáculos que superan la capacidad del equipo técnico para resolverlos constituyen **exceptions** que deben escalarse al **Project Board**.

4.7.6 Generar victorias a corto plazo

Cada fase del roadmap debe producir al menos una victoria visible antes de avanzar a la siguiente. En la Fase 2, el primer servidor físico incorporado al clúster tras una migración exitosa es una victoria concreta y medible. En la Fase 3, el primer servicio contenerizado con despliegue automatizado demuestra el valor de la plataforma a audiencias técnicas. Estas victorias deben documentarse y comunicarse activamente; son el argumento más efectivo contra la resistencia al cambio y coinciden con los productos verificables que PRINCE2 exige al cierre de cada etapa.

4.7.7 Consolidar logros e impulsar más cambio

Las victorias tempranas no deben interpretarse como señal de que el cambio está completo. Cada fase concluida debe usarse para ampliar la coalición incorporando nuevas áreas, para profundizar la automatización y para abordar los obstáculos que no pudieron eliminarse en etapas anteriores. La Fase 4 es precisamente este momento: no despliega nueva tecnología sino que consolida y extiende lo construido, en correspondencia con el proceso **Managing a Stage Boundary** de PRINCE2 aplicado al cierre del proyecto completo.

4.7.8 Anclar los cambios en la cultura

El cambio es sostenible cuando deja de depender de la voluntad de personas individuales y se institucionaliza en procesos, políticas y estructuras. Los mecanismos concretos son los descritos en la Fase 4: infraestructura como código versionada en repositorios institucionales, procesos operativos documentados, modelo de gobierno con procesos formalizados, y el registro de lecciones aprendidas que PRINCE2 establece como producto de cierre obligatorio [123]. Cuando un nuevo integrante del equipo técnico puede operar la infraestructura siguiendo la documentación existente, el cambio ha sido anclado.

TABLA 33

CORRESPONDENCIA ENTRE LOS OCHO PASOS DE KOTTER, LAS FASES DEL ROADMAP Y LOS PROCESOS DE PRINCE2.

#	Paso de Kotter	Expresión en el roadmap	Proceso PRINCE2 relacionado
1	Crear urgencia	Diagnóstico con datos propios; inventario de servidores e incidentes	Starting Up a Project
2	Coalición rectora	Constitución del ente gestor (Fase 0)	Project Board
3	Visión y estrategia	Visión institucional + roadmap de cinco fases	Initiating a Project
4	Comunicar la visión	Sesiones informativas y stage gates como hitos de comunicación	Directing a Project
5	Eliminar obstáculos	Escalación de exceptions al Project Board	Managing by Exception
6	Victorias a corto plazo	Productos verificables al cierre de cada etapa	Managing a Stage Boundary
7	Consolidar y ampliar	Incorporación de nuevas áreas en Fase 4; extensión de la automatización	Managing a Stage Boundary
8	Anclar en la cultura	IaC versionado, procesos documentados, lessons learned (Fase 4)	Closing a Project

CAPÍTULO

V

Conclusiones



5.1 Evaluación de la propuesta frente al problema planteado

Esta tesis partió de un problema concreto: la infraestructura de cómputo en organizaciones tiende a fragmentarse en servidores aislados, administrados de forma individual, sin continuidad operativa y sin capacidad de responder a la demanda creciente de recursos que se requieren en la actualidad acorde a la innovación tecnológica. Frente a ese problema, el objetivo general fue proponer una arquitectura de referencia basada en tecnologías de código abierto que permita consolidar esa infraestructura dispersa en un centro de datos distribuido, elástico y soberano. A partir de ese problema se atendieron objetivos específicos de la siguiente manera:

El primero, la fundamentación de la infraestructura base, se cumple mediante la selección y justificación de un stack compuesto por OpenStack como capa IaaS, Ceph como almacenamiento unificado definido por software, y un fabric de red virtual basada en SDN. La convergencia de recursos heterogéneos en un pool unificado se sustenta teóricamente en el concepto de Infraestructura Definida por Software y se implementa mediante herramientas de Infraestructura como Código que garantizan reproducibilidad y consistencia de configuración. La justificación del stack se contrasta con los marcos de referencia comerciales disponibles y la arquitectura resultante es coherente con los principios de transparencia de distribución descritos por Van Steen y Tanenbaum [20].

El segundo objetivo, la estandarización del despliegue mediante contenedores, se aborda con el diseño de la capa CaaS sobre Kubernetes, que transforma los servicios en unidades portables y escalables. La arquitectura de dos capas, es decir IaaS para los recursos físicos y CaaS para las cargas de trabajo, produce la separación de responsabilidades necesaria para que cada área de la organización consuma infraestructura sin gestionarla directamente.

El tercer objetivo, la metodología de transición y adopción, se materializa en el roadmap de cinco fases del Capítulo 4, estructurado bajo los principios de PRINCE2 [123], con criterios de salida verificables por fase, tolerancias explícitas y el mecanismo de expansión iterativa por liberación progresiva de nodos que permite iniciar la implementación sin infraestructura adicional.

Los tres objetivos se cumplen a nivel teórico y de diseño. La arquitectura propuesta es sólida al

compararse con marcos de referencia presentes en la industria, cabe aclarar que el *know how*¹⁸, es decir, la explicación técnica que sustenta las arquitecturas de la industria no se expone en dichos marcos.

A pesar de lo anterior, el ciclo de la propuesta no se cierra con el diseño: se cierra con la experimentación. Una prueba de concepto es el paso siguiente y necesario para validar si la arquitectura puede operar bajo las condiciones reales de una institución académica, y para determinar qué ajustes requiere antes de constituirse en una referencia replicable.

5.2 Verificación de la hipótesis

La hipótesis planteada en la introducción afirma que es posible diseñar una arquitectura de referencia para un centro de datos distribuido, construida exclusivamente con tecnologías de código abierto, integrada mediante interfaces estandarizadas y no mediante desarrollo a la medida, capaz de soportar cargas de cómputo de propósito general y especializado sin requerir rediseño estructural ante la incorporación de nuevos recursos. Su comprobación se sujetó a tres condiciones verificables, que se evalúan a continuación.

5.2.1 Suficiencia del stack

Condición satisfecha. La arquitectura descompuso el sistema en responsabilidades funcionales siguiendo la jerarquía de capas del CNCF Landscape, y para cada una de ellas se identificó al menos un componente de código abierto con licencia libre y gobernanza documentada: OpenStack, OpenTofu y Ansible en la capa de aprovisionamiento; Harbor y OpenBao en registro y gestión de secretos; Ceph, containerd o CRI-O y Cilium en la capa de ejecución; Kubernetes en orquestación; y Prometheus, Loki y Grafana en observabilidad. La tabla de correspondencia entre principios arquitectónicos y componentes del stack demuestra que ningún principio de diseño quedó sin respuesta tecnológica, y la restricción del universo de búsqueda al ecosistema CNCF garantiza que los componentes seleccionados no son software experimental sino proyectos sometidos a un proceso formal de madurez.

5.2.2 Coherencia de la integración

Condición satisfecha. La evaluación de interoperabilidad del Capítulo 2 estableció que las conexiones entre componentes se realizan mediante estándares abiertos y no mediante adaptadores

¹⁸El *know how* hace referencia al el conjunto de conocimientos, habilidades y herramientas necesarias para llevar a cabo algo.

propietarios o desarrollo a la medida: la Container Runtime Interface desacopla Kubernetes del runtime de contenedores, la Container Storage Interface lo desacopla del proveedor de almacenamiento, la OCI Distribution Specification normaliza el intercambio de imágenes con el registro, la Gateway API estandariza la exposición de servicios, SAML 2.0 y OIDC normalizan la federación de identidad, y el protocolo RADOS sostiene la integración de Ceph tanto con OpenStack como con Kubernetes. La consecuencia práctica de esta condición es la propiedad de sustituibilidad: la elección entre containerd y CRI-O, por ejemplo, no obliga a modificar ningún otro componente del stack. Esta es la evidencia más directa de que el conjunto seleccionado constituye un sistema integrado y no una agregación de herramientas.

5.2.3 Generalidad y extensibilidad de la arquitectura

Condición satisfecha. La arquitectura admite máquinas virtuales gestionadas por Nova, contenedores orquestados por Kubernetes, hardware bare metal incorporado mediante Ironic, equipo de generaciones anteriores integrado como nodos legacy vía K3s, y nodos de alto rendimiento con aceleradores asignados mediante VFIO; todo ello bajo una misma capa de control y sin que la incorporación de un tipo de nodo obligue a rediseñar los demás. La cobertura simultánea de cargas de propósito general y de cargas especializadas no exige capas de control paralelas: la capa CaaS consume los recursos que la capa IaaS abstrae, de modo que un nodo con aceleradores se incorpora por el mismo mecanismo que un nodo de cómputo convencional.

La extensibilidad se sostiene en el mismo principio. El mecanismo de expansión iterativa por liberación progresiva de nodos, descrito en el roadmap del Capítulo 4, incorpora recursos al pool unificado sin modificar la estructura de la arquitectura ni interrumpir las cargas en operación: crecer es agregar nodos a una jerarquía ya definida, no rediseñarla. Esta propiedad es la que permite iniciar la implementación con el hardware disponible y escalar conforme la organización libere infraestructura.

5.2.4 Alcance de la comprobación

Las tres condiciones se satisfacen dentro del alcance que la propia hipótesis delimitó: el de la viabilidad arquitectónica. La hipótesis queda sustentada en el sentido de que la arquitectura es derivable, internamente coherente y construible en su totalidad con componentes de código abierto interoperables; no queda sustentada en el sentido del desempeño del sistema bajo condiciones reales de operación. Mientras esa prueba no se ejecute, cualquier afirmación sobre el desempeño

real del sistema bajo carga o esfuerzo real de operación permanece fuera de lo que este trabajo puede sostener.

5.3 Implicaciones técnicas y organizacionales

La propuesta tiene implicaciones que van más allá de la infraestructura tecnológica.

En la dimensión técnica, el diseño de un centro de datos distribuido es un problema multifactorial. Esta tesis aborda la capa de cómputo pero, un centro de datos funcional requiere también el diseño de:

- Obra civil para las zonas de disponibilidad.
- Sistemas de potencia que busquen eficiencia energética.
- Sistemas de refrigeración.
- Cableado estructurado y automatización de la instalación física.

Cada una de estas dimensiones constituye un dominio de especialidad propio. La colaboración con profesionales de ingeniería civil, eléctrica y mecatrónica no es opcional para llevar la propuesta a un producto completamente funcional. Esta multidisciplinariedad es característica del diseño total de un centro de datos y no puede ser resuelta desde una sola área de conocimiento.

En la dimensión organizacional, el mayor reto de implementación no es técnico. La resistencia al cambio y las estructuras administrativas consolidadas representan el obstáculo más difícil de gestionar en proyectos de consolidación de infraestructura institucional. Las áreas que han administrado sus servidores de forma autónoma durante años pueden percibir la consolidación como una pérdida de control. Por esta razón, el modelo de gestión del cambio de Kotter [126], integrado en el Capítulo 4, tiene un valor que supera lo metodológico: es el reconocimiento explícito de que sin una estrategia deliberada para gestionar la dimensión humana del proyecto, la arquitectura más sólida puede fracasar antes de desplegarse.

En el contexto más específico, las universidades son responsables de formar a los profesionales del futuro. Para cumplir esa responsabilidad en un entorno tecnológico que evoluciona a la velocidad actual, las instituciones académicas deben mostrar madurez digital y ser capaces de sostener su infraestructura de cómputo bajo demanda. La tendencia global es clara: los índices de inversión en capacidad de cómputo muestran un crecimiento sostenido que refleja la importancia del cómputo en la economía y en la producción de conocimiento. Una universidad que no puede ofrecer a sus estudiantes, profesores e investigadores la potencia de cómputo que requieren en el momento en

que la requieren, pierde relevancia frente a las demandas de la sociedad que financia su existencia. En este sentido, la propuesta de esta tesis no es únicamente una solución técnica: es una apuesta por la soberanía tecnológica y la competitividad académica de las instituciones que la adopten.

Si la arquitectura propuesta logra demostrar su viabilidad mediante una prueba de concepto, tiene el potencial de constituir el know how para diseñar, desplegar y administrar centros de datos distribuidos en universidades públicas. Ese conocimiento, una vez documentado y validado, podría representar un cambio de paradigma en la forma en que las instituciones académicas conciben y gestionan su infraestructura tecnológica.

5.4 Limitaciones del estudio

Esta tesis tiene limitaciones que es necesario reconocer con claridad, la principal es la ausencia de una prueba de concepto. La validación de la arquitectura es exclusivamente teórica: la selección del stack, la justificación de las decisiones de diseño y el roadmap de implementación se sustentan en la literatura técnica y en los marcos de referencia de la industria, pero no han sido sometidos a la prueba de las condiciones reales de operación. Una instalación real expone restricciones que el diseño no anticipa: incompatibilidades de hardware específico, comportamientos no documentados de las herramientas bajo carga, resistencias organizacionales concretas, limitaciones de red que no aparecen en los diagramas, entre otros aspectos no contemplados. La documentación de la arquitectura, en particular, es un producto vivo que solo adquiere su forma definitiva cuando la nube está en operación y se ajusta al esquema de trabajo real.

La segunda limitación es el alcance del trabajo. Al escribir la tesis se hizo evidente que la magnitud del problema abordado es considerablemente mayor de lo que el formato de una tesis de licenciatura permite desarrollar con la profundidad necesaria. El diseño de la capa de red, los mecanismos de federación entre clústeres, el modelo de seguridad completo y la integración con los sistemas de identidad institucional son cada uno temas que merecen un tratamiento más extenso del que este trabajo ofrece. En retrospectiva, acotar el objetivo a una dimensión más específica como por ejemplo, concentrarse exclusivamente en el diseño de la capa IaaS con una prueba de concepto incluida, habría producido un resultado más profundo aunque de menor alcance. Es importante resaltar que este trabajo permite observar un panorama amplio para la implementación de este tipo de sistemas, dicho panorama representa un punto de partida para el diseño específico de los sistemas y subsistemas involucrados en un centro de datos.

La tercera limitación son las condiciones de entrada que la arquitectura presupone. Aunque la hipótesis no afirma que la propuesta sea aplicable de forma indiscriminada a cualquier organización, sí conviene explicitar que su instanciación exige un umbral mínimo que no toda organización satisface. Estas condiciones son cuatro.

La primera es de **infraestructura de red**: se requiere una fabric con soporte de segmentación mediante VXLAN entre los emplazamientos que alojarán los nodos. Una organización cuya red no admita esta segmentación debe resolverlo como proyecto previo, tal como establece la Fase 1 del roadmap.

La segunda es de **identidad**: se requiere un directorio institucional operativo que exponga los atributos necesarios para la federación. Su ausencia no impide un despliegue inicial con usuarios locales, pero sí impide satisfacer el principio de identidad federada, que es uno de los pilares de la propuesta.

La tercera es de **escala mínima de hardware**: la alta disponibilidad del plano de control exige tres sitios de control para sostener el quórum de consenso, y la replicación de almacenamiento exige al menos tres nodos por zona de disponibilidad. Una organización que no alcance ese umbral puede desplegar la arquitectura, pero no en la configuración de alta disponibilidad que este trabajo describe; obtendría una instalación funcional sin las garantías de continuidad operativa que justifican el diseño.

La cuarta, y la más determinante, es de **estructura de gobierno**. La nube comunitaria se propuso por ser estructuralmente isomorfa a un modelo de gobierno federado: entidades con autonomía administrativa sobre sus recursos operando bajo un marco normativo común. Una organización con gobierno centralizado, o una en la que no exista una instancia con autoridad reconocida para ejercer el rol de Community Operator, no obtiene de la jerarquía Region–Domain–Project el beneficio que justifica su complejidad; para esos casos, una nube privada de un solo tenant es una solución más adecuada y considerablemente más simple de operar.

Fuera de estas condiciones, lo que este trabajo ofrece a otras organizaciones no es la arquitectura en su forma concreta sino el proceso metodológico que la produjo: partir de un diagnóstico de necesidades propio, derivar principios de diseño organizados por dominios, traducirlos en una protoarquitectura funcional y seleccionar sobre ella un stack acotado a un ecosistema auditado. Ese proceso es reproducible, pero la arquitectura que resulte de aplicarlo no tiene por qué coincidir

con la aquí propuesta, y este trabajo no lo aplicó a un segundo caso que permitiera contrastarlo: el diagnóstico se realizó sobre la Facultad de Ingeniería de la UNAM como caso único.

5.5 Líneas de investigación futura

A partir de las limitaciones identificadas y del trabajo desarrollado, se proponen las siguientes líneas de investigación y desarrollo futuros.

La **prueba de concepto** es la continuación inmediata e indispensable de este trabajo. Su objetivo sería desplegar la arquitectura propuesta sobre un subconjunto real de servidores de la Facultad de Ingeniería, documentar el proceso, identificar las desviaciones respecto al diseño teórico y producir una guía de implementación validada experimentalmente. Una prueba de concepto exitosa constituiría la evidencia necesaria para proponer la adopción institucional de la arquitectura.

El **diseño de un rack plug-and-play** para la integración de nodos a la nube, representa una línea de trabajo en la intersección entre la ingeniería de hardware y la automatización de software. El objetivo sería diseñar una unidad física estandarizada que, al conectarse a la red de la instalación, se incorpore automáticamente al clúster sin intervención manual, reduciendo la fricción de la expansión del centro de datos y haciendo accesible la contribución de infraestructura para áreas sin personal técnico especializado.

El **diseño de red ampliado para el centro de datos** es una línea que requiere profundizar en los aspectos que esta tesis trata de forma general: el dimensionamiento de la fabric Spine-Leaf, los protocolos de enrutamiento entre zonas de disponibilidad, la segmentación de tráfico por tipo de carga y la integración con la red institucional existente. Este diseño requiere la colaboración con especialistas en telecomunicaciones y redes.

El **diseño de sistemas de potencia con eficiencia energética** aborda una dimensión crítica que esta tesis no trata: la alimentación eléctrica del centro de datos, los sistemas de respaldo, la distribución de potencia entre racks y la métrica de eficiencia energética (PUE). La eficiencia energética es un criterio de diseño cada vez más relevante tanto por razones económicas como por compromisos de sostenibilidad institucional.

Finalmente, el **diseño integral del centro de datos como proyecto multidisciplinario** es la línea que integra todas las anteriores o más. Un centro de datos de escala institucional requiere obra civil para las zonas de disponibilidad, diseño de sistemas de refrigeración, automatización de la instalación física y protocolos de seguridad perimetral. Ninguna de estas dimensiones puede

resolverse desde la ingeniería en computación de forma aislada. La constitución de equipos multidisciplinarios que integren ingeniería civil, eléctrica, mecánica y en computación es la condición necesaria para llevar la propuesta de esta tesis a su expresión completa como infraestructura física y digital en plena operación.

Índice de Figuras

Índice de Figuras

Fig. 1	Jerarquía de las tecnologías en la industria 4.0	8
Fig. 2	Abstracción de los recursos de cómputo en un centro de datos	16
Fig. 3	Características fundamentales de los sistemas distribuidos.	20
Fig. 4	Virtualización de recursos	24
Fig. 5	SDI e IaC	25
Fig. 6	Arquitectura de la contenedorización	27
Fig. 7	Resumen de principios arquitectónicos por dominio	41
Fig. 8	Protoarquitectura funcional	48
Fig. 9	Topología Spine-Leaf	61
Fig. 10	Encapsulamiento VXLAN por plano de red	62
Fig. 11	Cadena de seguridad e identidad	64
Fig. 12	Modelo de gobierno de la nube comunitaria.	68
Fig. 13	Topología interna de una AZ	77
Fig. 14	Nodos legacy vs nodos principales	79
Fig. 15	Nodos de control	80
Fig. 16	Nodos de cómputo	81
Fig. 17	Nodos de almacenamiento	83
Fig. 18	Nodos de almacenamiento	84
Fig. 19	Correspondencia estructural entre el Sistema Eléctrico Nacional y la nube comunitaria universitaria.	86
Fig. 20	Vista de conjunto de la Region: Control Sites, Execution Sites y enlaces cifrados entre AZs.	87
Fig. 21	Zoom progresivo por los tres niveles del C4 Model aplicados a la arquitectura.	88
Fig. 22	Pila tecnológica superpuesta: modelos de servicio expuestos y correspondencia funcional con el CNCF Landscape.	89
Fig. 23	Proceso de rehosting	102
Fig. 24	Fases de modernización	107

Índice de Tablas



Índice de Tablas

Tabla 1	Evolución y características de las Revoluciones Industriales.	7
Tabla 2	Diferentes formas de transparencia en un sistema distribuido [20].	17
Tabla 3	Características clave de la confiabilidad en sistemas distribuidos [20].	18
Tabla 4	Tipos de escalabilidad en sistemas distribuidos [20].	19
Tabla 5	Comparativa entre infraestructura convergente e hiperconvergente [23], [24], [26]. .	21
Tabla 6	Comparativa entre máquinas virtuales y contenedores [53], [54], [56].	28
Tabla 7	Correspondencia entre los principios propuestos y los marcos Well-Architected de la industria.	42
Tabla 8	Interfaces de integración entre componentes del stack.	53
Tabla 9	Correspondencia entre principios arquitectónicos y componentes del stack seleccionado.	56
Tabla 10	Correspondencia entre los niveles del C4 Model y las secciones del Capítulo 3.	60
Tabla 11	Propiedades requeridas por cada plano de red.	63
Tabla 12	Asimetría de disponibilidad entre flujos inter-zona coordinados y autónomos.	63
Tabla 13	Modelo híbrido de certificados: CA interna para tráfico interno · Let's Encrypt para endpoints públicos.	65
Tabla 14	Planos de control del Community Operator y sus mecanismos de enforcement.	70
Tabla 15	Tipos de AZ según su participación en el plano de control regional.	71
Tabla 16	Rol de cada tecnología en el nivel de Contexto.	73
Tabla 17	Requisitos de hardware — nodos de control. Elaboración propia a partir de los requisitos mínimos documentados para OpenStack [80] y el plano de control de Kubernetes [98].	74
Tabla 18	Requisitos de hardware — nodos de cómputo. Elaboración propia a partir de los requisitos de Nova y del hipervisor KVM [105], [119].	75
Tabla 19	Requisitos de hardware — nodos de almacenamiento. Elaboración propia a partir de las recomendaciones oficiales de hardware de Ceph [120].	75
Tabla 20	Requisitos de hardware — nodos legacy. Elaboración propia a partir de los requisitos mínimos de K3s [121] y Cilium [106].	76
Tabla 21	Resumen de requisitos mínimos de hardware por rol de nodo.	76

Tabla 22	Presencia de cada tipo de nodo en los planos de red de la AZ.	77
Tabla 23	Distribución de tecnologías por nodo y plano de red dentro de la AZ.	78
Tabla 24	Comparación entre nodos principales y nodos legacy.	79
Tabla 25	Capas de ejecución y tecnologías dentro de un nodo de control.	81
Tabla 26	Capas de ejecución y tecnologías dentro de un nodo de cómputo.	82
Tabla 27	Componentes internos del nodo de almacenamiento y sus interfaces hacia el stack. . .	84
Tabla 28	Decisiones estructurales de la arquitectura, su justificación y el nivel en el que se manifiestan.	90
Tabla 29	Resumen del roadmap: productos, criterios de salida, tolerancias y prerequisites por fase.	93
Tabla 30	Niveles del Gartner Infrastructure Maturity Model evaluados sobre los ejes People, Process y Technology [125].	95
Tabla 31	Estrategias de modernización: potencial de cambio por capa y perfil de riesgo-coste [124].	99
Tabla 32	Modelo de expansión iterativa del clúster mediante liberación progresiva de nodos. .	101
Tabla 33	Correspondencia entre los ocho pasos de Kotter, las fases del roadmap y los procesos de PRINCE2.	110

Referencias

Bibliografía

- [1] European Space Agency, «What Is Technology?». [En línea]. Disponible en: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/What_is_technology/
- [2] Trevor J. Pinch y Wiebe E. Bijker, «The Social Construction of Facts and Artifacts», en *The Social Construction of Technological Systems*, Wiebe E. Bijker, Thomas P. Hughes, y Trevor J. Pinch, Eds., Cambridge, MA: MIT Press, 1987, pp. 17-50.
- [3] Lebhar Ikram, Afaf Dadda, y Latifa Ezzine, *Beyond Industry 5.0: Tracing the Past, Present, and Future of Digital Industrial Transformations*. en 2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET). 2024, pp. 1-7.
- [4] Aeshita Mathur, Ameesha Dabas, y Nikhil Sharma, *Evolution from Industry 1.0 to Industry 5.0*. en 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). 2022, pp. 1390-1394.
- [5] Amazon Web Services, «¿Qué es la transformación digital?». [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/digital-transformation/>
- [6] Henning Kagermann, Wolfgang Wahlster, y Johannes Helbig, «Recommendations for Implementing the Strategic Initiative Industrie 4.0», acatech -- National Academy of Science and Engineering, 2013.
- [7] World Economic Forum, «What Is Industry 4.0 and What Will It Mean for Countries?». [En línea]. Disponible en: <https://www.weforum.org/stories/2022/04/what-is-industry-4-0-and-could-developing-countries-get-left-behind/>
- [8] IBM, «Industria 4.0». [En línea]. Disponible en: <https://www.ibm.com/mx-es/think/topics/industry-4-0>
- [9] American Chamber of Commerce Mexico, «Empresas mexicanas alcanzan 41.7% en transformación digital: Informe de Madurez Digital 2025 (IMD)». [En línea]. Disponible

en: <https://amcham.org.mx/empresas-mexicanas-alcanzan-41-7-en-transformacion-digital-informe-de-madurez-digital-2025-imd/>

- [10] Rodrigo Riquelme, «Sector privado de México es muy inmaduro en materia de digitalización». [En línea]. Disponible en: <https://www.eleconomista.com.mx/tecnologia/sector-privado-mexico-inmaduro-materia-digitalizacion-20250326-752158.html>
- [11] IT Masters Mag, «5G en México: cómo funciona y cuáles son sus beneficios». [En línea]. Disponible en: <https://www.itmastersmag.com/transformacion-digital/5g-en-mexico-como-functiona-y-cuales-son-sus-beneficios/>
- [12] C3ntro Telecom, «Mexico Holds Only 3% of the Fiber Optic Infrastructure Deployed in the U.S.». [En línea]. Disponible en: <https://www.c3ntro.com/news/mexico-holds-only-3-of-the-fiber-optic-infrastructure-deployed-in-the-u.s>
- [13] Mordor Intelligence, «Mexico Data Center Market: Growth, Trends, Forecasts». [En línea]. Disponible en: <https://www.mordorintelligence.com/es/industry-reports/mexico-data-center-market>
- [14] Cloudflare, «What is Vendor Lock-In?». [En línea]. Disponible en: <https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in/>
- [15] Javier López Casarín, «Independencia tecnológica». [En línea]. Disponible en: <https://www.eleconomista.com.mx/opinion/Independencia-tecnologica-20190918-0022.html>
- [16] Microsoft, «Independencia tecnológica - Microsoft Sovereign Cloud». [En línea]. Disponible en: <https://learn.microsoft.com/es-es/industry/sovereign-cloud/concepts/technological-independence>
- [17] Amazon Web Services, «What Is Compute?». [En línea]. Disponible en: <https://aws.amazon.com/what-is/compute/>
- [18] Lenovo, «¿Qué significa Cóputo?». [En línea]. Disponible en: <https://www.lenovo.com/mx/es/glosario/computo/>
- [19] Stephanie Susnjara y Ian Smalley, «What Is a Data Center?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/data-centers>

- [20] Maarten van Steen y Andrew S. Tanenbaum, *Distributed Systems*, 4.^a ed. Maarten van Steen, 2025.
- [21] Kate Gibson, «How to Leverage Economies of Scale to Grow Your Platform Business». [En línea]. Disponible en: <https://online.hbs.edu/blog/post/economies-of-scale>
- [22] Diego Ongaro y John Ousterhout, «In Search of an Understandable Consensus Algorithm», en *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, USENIX Association, 2014. [En línea]. Disponible en: <https://raft.github.io/raft.pdf>
- [23] Cisco Systems, «Unified Data Center: Efficiency and Flexibility to Improve Business Agility». [En línea]. Disponible en: https://www.cisco.com/c/dam/global/en_in/solutions/collateral/ns340/ns517/ns224/c11_663783_01_udc_wp.pdf
- [24] Sangfor Technologies, «Converged vs Hyperconverged: What's the Difference?». [En línea]. Disponible en: <https://www.sangfor.com/blog/cloud-and-infrastructure/converged-vs-hyperconverged-whats-difference>
- [25] Hewlett-Packard, «Five Things to Know About Hyper-Convergence». [En línea]. Disponible en: <https://ftp.ext.hp.com/pub/msc/8B110CBD-1838-4810-9495-0183D18A7FBB.pdf>
- [26] Craig Theriac, «What is Hyperconvergence? A Complete Guide to HCI». [En línea]. Disponible en: <https://www.scalecomputing.com/blog/what-is-hyperconvergence-a-complete-guide-to-hci>
- [27] Peter Mell y Timothy Grance, «The NIST Definition of Cloud Computing», National Institute of Standards and Technology, Special Publication 800–145, 2011. [En línea]. Disponible en: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [28] Red Hat, «What is Platform as a Service (PaaS)?». [En línea]. Disponible en: <https://www.redhat.com/en/topics/cloud-computing/what-is-paas>
- [29] IBM, «What is Backend as a Service (BaaS)?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/baas>
- [30] Cloudflare, «What is Backend-as-a-Service (BaaS)?». [En línea]. Disponible en: <https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/>

- [31] Douglas E. Comer, *The Cloud Computing Book: The Future of Computing Explained*. CRC Press / Chapman and Hall, 2022.
- [32] Stephanie Susnjara y Ian Smalley, «¿Qué son los contenedores como servicio (CaaS)?». [En línea]. Disponible en: <https://www.ibm.com/es-es/think/topics/containers-as-a-service>
- [33] Stephanie Susnjara y Ian Smalley, «¿Qué es la virtualización?». [En línea]. Disponible en: <https://www.ibm.com/mx-es/think/topics/virtualization>
- [34] Amazon Web Services, «¿Qué es la virtualización?». [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/virtualization/>
- [35] Amazon Web Services (AWS), «Diferencias entre hipervisores Tipo 1 y Tipo 2». [En línea]. Disponible en: <https://aws.amazon.com/es/compare/the-difference-between-type-1-and-type-2-hypervisors/>
- [36] Gustavo A. A. Santana, *Data Center Virtualization Fundamentals*. Cisco Press, 2014.
- [37] Thomas D. Nadeau y Ken Gray, *SDN: Software Defined Networks*. O'Reilly Media, 2013.
- [38] IBM, «What is Software-Defined Storage?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/software-defined-storage>
- [39] M. Tim Jones, «Linux virtualization and PCI passthrough», 13 de octubre de 2009, *IBM Developer*. [En línea]. Disponible en: <https://developer.ibm.com/tutorials/l-pci-passthrough/>
- [40] IBM Documentation, «What you should know about VFIO». [En línea]. Disponible en: <https://www.ibm.com/docs/en/linux-on-systems?topic=vfio->
- [41] SUSE, «Understanding Software-Defined Infrastructure: Benefits, Challenges, and Future Trends». [En línea]. Disponible en: <https://www.suse.com/topics/understanding-sd-infrastructure/>
- [42] Hewlett Packard Enterprise, «What is Software-Defined Infrastructure?». [En línea]. Disponible en: https://www.hpe.com/emea_europe/en/what-is/software-defined-infrastructure.html
- [43] Jim Holdsworth y Annie Badman, «What is Infrastructure as Code (IaC)?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/infrastructure-as-code>

- [44] Cisco, «What is Infrastructure as Code (IaC)?». [En línea]. Disponible en: <https://www.cisco.com/site/us/en/learn/topics/computing/what-is-iac.html>
- [45] Kief Morris, *Infrastructure as Code: Designing and Delivering Dynamic Systems for the Cloud Age*, 3.^a ed. O'Reilly Media, 2025.
- [46] Paul Goransson y Chuck Black, *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2014.
- [47] Claus Pahl, «Containerization and the PaaS Cloud», *IEEE Cloud Computing*, vol. 2, n.º 3, pp. 24-31, 2015.
- [48] Stephanie Susnjara y Ian Smalley, «What Is Containerization?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/containerization>
- [49] Google Cloud, «What is Containerization? How does it work». [En línea]. Disponible en: <https://cloud.google.com/discover/what-is-containerization>
- [50] James Turnbull, *The Docker Book: Containerization is the New Virtualization*. James Turnbull, 2014.
- [51] Amazon Web Services, «What is Containerization?». [En línea]. Disponible en: <https://aws.amazon.com/what-is/containerization/>
- [52] Open Container Initiative, «OCI Image Format Specification», Open Container Initiative (Linux Foundation), 2024. [En línea]. Disponible en: <https://github.com/opencontainers/image-spec/blob/main/spec.md>
- [53] IBM, «Containers Versus Virtual Machines (VMs): What's The Difference?». [En línea]. Disponible en: <https://www.ibm.com/think/topics/containers-vs-vms>
- [54] Google Cloud, «Containers versus VMs (virtual machines)». [En línea]. Disponible en: <https://cloud.google.com/discover/containers-vs-vms>
- [55] Michael Kerrisk, «namespaces(7) — Linux manual page». [En línea]. Disponible en: <https://man7.org/linux/man-pages/man7/namespaces.7.html>

- [56] Amazon Web Services, «What's the Difference Between Containers and Virtual Machines?». [En línea]. Disponible en: <https://aws.amazon.com/compare/the-difference-between-containers-and-virtual-machines/>
- [57] Red Hat, «What is orchestration?». [En línea]. Disponible en: <https://www.redhat.com/en/topics/automation/what-is-orchestration>
- [58] Red Hat, «What is container orchestration?». [En línea]. Disponible en: <https://www.redhat.com/en/topics/containers/what-is-container-orchestration>
- [59] Google Cloud, «What is container orchestration?». [En línea]. Disponible en: <https://cloud.google.com/discover/what-is-container-orchestration>
- [60] Dinesh G. Varma, *Cloud Native Data Center Networking*. O'Reilly Media, 2019.
- [61] Charles Clos, «A Study of Non-Blocking Switching Networks», *Bell System Technical Journal*, vol. 32, n.º 2, pp. 406-424, 1953.
- [62] IEEE Computer Society, «IEEE Standard for Local and Metropolitan Area Networks — Bridges and Bridged Networks», Institute of Electrical and Electronics Engineers (IEEE), 2022. [En línea]. Disponible en: <https://standards.ieee.org/ieee/802.1Q/10323/>
- [63] Mallik Mahalingam *et al.*, «Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks», Internet Engineering Task Force (IETF), 2014. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc7348>
- [64] Eric Rescorla, «The Transport Layer Security (TLS) Protocol Version 1.3», Internet Engineering Task Force (IETF), 2018. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc8446>
- [65] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, y Tim Polk, «Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile», Internet Engineering Task Force (IETF), 2008. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc5280>

- [66] OASIS Security Services Technical Committee, «Security Assertion Markup Language (SAML) V2.0 Technical Overview», OASIS, 2005. [En línea]. Disponible en: <https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html>
- [67] Nat Sakimura, John Bradley, Michael Jones, Breno de Medeiros, y Chuck Mortimore, «OpenID Connect Core 1.0», OpenID Foundation, 2014. [En línea]. Disponible en: https://openid.net/specs/openid-connect-core-1_0.html
- [68] OpenBao Project, «OpenBao Documentation». [En línea]. Disponible en: <https://openbao.org/docs/>
- [69] Jerome H. Saltzer y Michael D. Schroeder, «The Protection of Information in Computer Systems», *Proceedings of the IEEE*, vol. 63, n.º 9, pp. 1278-1308, 1975, [En línea]. Disponible en: <https://www.cs.virginia.edu/~evans/cs551/saltzer/>
- [70] Richard Barnes, Jacob Hoffman-Andrews, Daniel McCarney, y James Kasten, «Automatic Certificate Management Environment (ACME)», Internet Engineering Task Force (IETF), 2019. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc8555>
- [71] Len Bass, Paul Clements, y Rick Kazman, *Software Architecture in Practice*, 3.ª ed. Upper Saddle River, NJ: Addison-Wesley, 2013.
- [72] Congreso de los Estados Unidos Mexicanos, *Ley Orgánica de la Universidad Nacional Autónoma de México*. Diario Oficial de la Federación, 1945. [En línea]. Disponible en: <https://www.abogadogeneral.unam.mx/PDFS/COMPENDIO/53.pdf>
- [73] Consejo Universitario de la UNAM, *Estatuto General de la Universidad Nacional Autónoma de México*. Universidad Nacional Autónoma de México, Oficina de la Abogacía General, 2025. [En línea]. Disponible en: https://www.abogadogeneral.unam.mx/sites/default/files/archivos/LegUniv/EstatutoGeneral_UNAM_070425.pdf
- [74] The Open Group, *TOGAF Standard, Version 9.2*. The Open Group, 2018.
- [75] Amazon Web Services, «AWS Well-Architected Framework». [En línea]. Disponible en: <https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html>
- [76] Google Cloud, «Google Cloud Well-Architected Framework». [En línea]. Disponible en: <https://docs.cloud.google.com/architecture/framework>

- [77] Microsoft, «Azure Well-Architected Framework». [En línea]. Disponible en: <https://learn.microsoft.com/en-us/azure/well-architected/>
- [78] Cloud Native Computing Foundation, «CNCf Cloud Native Landscape». [En línea]. Disponible en: <https://landscape.cncf.io/>
- [79] CNCf Technical Oversight Committee, «Graduation Criteria». [En línea]. Disponible en: <https://www.cncf.io/projects/graduation-criteria/>
- [80] OpenInfra Foundation, «OpenStack Documentation». [En línea]. Disponible en: <https://docs.openstack.org/>
- [81] Kevin Jackson, Cody Bunch, Egle Sigler, y James Denton, *OpenStack Cloud Computing Cookbook*, 4.^a ed. Packt Publishing, 2018.
- [82] OpenTofu Project, «The OpenTofu Manifesto». [En línea]. Disponible en: <https://opentofu.org/manifesto/>
- [83] I. Red Hat, «Ansible Documentation». [En línea]. Disponible en: <https://docs.ansible.com/ansible/latest/>
- [84] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces Harbor Graduation». [En línea]. Disponible en: <https://www.cncf.io/announcements/2020/06/23/cloud-native-computing-foundation-announces-harbor-graduation/>
- [85] Harbor Project, «Harbor Documentation». [En línea]. Disponible en: <https://goharbor.io/docs/>
- [86] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces cert-manager Graduation». [En línea]. Disponible en: <https://www.cncf.io/announcements/2024/11/12/cloud-native-computing-foundation-announces-cert-manager-graduation/>
- [87] cert-manager Project, «cert-manager Documentation». [En línea]. Disponible en: <https://cert-manager.io/docs/>
- [88] Nick Fisk, *Mastering Ceph: Redefine Your Storage System*. Packt Publishing, 2017.
- [89] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, y Carlos Maltzahn, «CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data», en *Proceedings of the 2006 ACM/IEEE*

Conference on Supercomputing (SC '06), ACM, 2006. [En línea]. Disponible en: <https://ceph.io/assets/pdfs/weil-crush-sc06.pdf>

- [90] Ceph Project, «Ceph Architecture». [En línea]. Disponible en: <https://docs.ceph.com/en/latest/architecture/>
- [91] The Kubernetes Authors, «Container Runtime Interface (CRI)». [En línea]. Disponible en: <https://kubernetes.io/docs/concepts/architecture/cri/>
- [92] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces containerd Graduation». [En línea]. Disponible en: <https://www.cncf.io/announcements/2019/02/28/cncf-announces-containerd-graduation/>
- [93] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces Graduation of CRI-O». [En línea]. Disponible en: <https://www.cncf.io/announcements/2023/07/19/cloud-native-computing-foundation-announces-graduation-of-cri-o/>
- [94] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces Cilium Graduation». [En línea]. Disponible en: <https://www.cncf.io/announcements/2023/10/11/cloud-native-computing-foundation-announces-cilium-graduation/>
- [95] eBPF Foundation, «What is eBPF? An Introduction and Deep Dive into the eBPF Technology». [En línea]. Disponible en: <https://ebpf.io/what-is-ebpf/>
- [96] Kubernetes SIG Network, «Gateway API». [En línea]. Disponible en: <https://gateway-api.sigs.k8s.io/>
- [97] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces Kubernetes as First Graduated Project». [En línea]. Disponible en: <https://www.cncf.io/announcements/2018/03/06/cloud-native-computing-foundation-announces-kubernetes-first-graduated-project/>
- [98] Brendan Burns, Joe Beda, y Kelsey Hightower, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, 3.^a ed. O'Reilly Media, 2022.
- [99] Cloud Native Computing Foundation, «Cloud Native Computing Foundation Announces Prometheus Graduation». [En línea]. Disponible en: <https://www.cncf.io/announcements/2018/08/09/prometheus-graduates/>

- [100] Grafana Labs, «Grafana Loki Documentation». [En línea]. Disponible en: <https://grafana.com/docs/loki/latest/>
- [101] Kubernetes CSI Community, «Container Storage Interface (CSI) Documentation». [En línea]. Disponible en: <https://kubernetes-csi.github.io/docs/>
- [102] Ceph Project, «Ceph CSI Driver Documentation». [En línea]. Disponible en: <https://github.com/ceph/ceph-csi>
- [103] Open Container Initiative, «OCI Distribution Specification», Open Container Initiative (Linux Foundation), 2024. [En línea]. Disponible en: <https://github.com/opencontainers/distribution-spec/blob/main/spec.md>
- [104] Sage A. Weil, Andrew W. Leung, Scott A. Brandt, y Carlos Maltzahn, «RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters», en *Proceedings of the 2nd International Workshop on Petascale Data Storage (PDSW '07)*, ACM, 2007. [En línea]. Disponible en: <https://ceph.io/assets/pdfs/weil-rados-pdsw07.pdf>
- [105] OpenInfra Foundation, «Compute Service (nova) Documentation». [En línea]. Disponible en: <https://docs.openstack.org/nova/latest/>
- [106] Cilium Project, «System Requirements». [En línea]. Disponible en: https://docs.cilium.io/en/stable/operations/system_requirements/
- [107] OpenInfra Foundation, «Keystone Federation Configuration Guide». [En línea]. Disponible en: <https://docs.openstack.org/keystone/latest/admin/federation/introduction.html>
- [108] The Kubernetes Authors, «Resource Quotas». [En línea]. Disponible en: <https://kubernetes.io/docs/concepts/policy/resource-quotas/>
- [109] S. Brown, *Software Architecture for Developers*. Leanpub, 2018. [En línea]. Disponible en: <https://leanpub.com/software-architecture-for-developers>
- [110] Open vSwitch / OVN Community, «Open Virtual Network (OVN) Documentation». [En línea]. Disponible en: <https://docs.ovn.org/en/latest/>
- [111] Eric Rosen, Arun Viswanathan, y Ross Callon, «Multiprotocol Label Switching Architecture», Internet Engineering Task Force (IETF), 2001. [En línea]. Disponible en: <https://www.ietf.org/rfc/rfc3031>

- [112] Stephen Kent y Karen Seo, «Security Architecture for the Internet Protocol», Internet Engineering Task Force (IETF), 2005. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc4301>
- [113] Jason A. Donenfeld, «WireGuard: Fast, Modern, Secure VPN Tunnel», en *Network and Distributed System Security Symposium (NDSS 2017)*, Internet Society, 2017. [En línea]. Disponible en: <https://www.wireguard.com/papers/wireguard.pdf>
- [114] Jim Sermersheim, «Lightweight Directory Access Protocol (LDAP): The Protocol», Internet Engineering Task Force (IETF), 2006. [En línea]. Disponible en: <https://datatracker.ietf.org/doc/html/rfc4511>
- [115] Internet Security Research Group, «How It Works». [En línea]. Disponible en: <https://letsencrypt.org/how-it-works/>
- [116] Centro Nacional de Control de Energía, «Quiénes Somos». [En línea]. Disponible en: <https://www.cenace.gob.mx/paginas/publicas/cenace/quienessomos.aspx>
- [117] RabbitMQ / Broadcom, «RabbitMQ Documentation». [En línea]. Disponible en: <https://www.rabbitmq.com/docs>
- [118] Codership Oy, «Galera Cluster Documentation». [En línea]. Disponible en: <https://galeracluster.com/library/documentation/>
- [119] KVM Project, «Kernel Virtual Machine (KVM)». [En línea]. Disponible en: https://linux-kvm.org/page/Main_Page
- [120] Ceph Project, «Hardware Recommendations». [En línea]. Disponible en: <https://docs.ceph.com/en/latest/start/hardware-recommendations/>
- [121] K3s Project, «K3s Documentation». [En línea]. Disponible en: <https://docs.k3s.io/>
- [122] The Kubernetes Authors, «Taints and Tolerations». [En línea]. Disponible en: <https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>
- [123] Kris Hermans y Cybellium Ltd, *Mastering PRINCE2: A Comprehensive Guide to Learn PRINCE2*. Cybellium Ltd, 2023.

- [124] Koan, «Modernization Strategy Paper», Koan. [En línea]. Disponible en: https://koan.co.nz/files/Modernization%20Strategy%20Paper%20V0_9_2.pdf
- [125] Peter Hidas, «Roadmap for Your Infrastructure: The Gartner Infrastructure Maturity Model». [En línea]. Disponible en: <https://www.unicom-systems.com/Blog/Roadmap%20for%20Your%20Infrastructure%20-%20The%20Gartner%20Infrastructure%20Maturity%20Model.pdf>
- [126] John P. Kotter, *Leading Change*. Harvard Business School Press, 1996.
- [127] Michael J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. Wiley, 2014.