UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

VICERRECTORADO

CENTRO DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
FACULTAD DE CIENCIAS Y TECNOLOGÍA

“DEVOPS EN EL DESARROLLO Y ENTREGA DE SOFTWARE BASADO EN CONTENEDORES”

TRABAJO EN OPCIÓN A DIPLOMADO EN DEVELOPEMENT

OPERATIONS “DEVOPS" V.1.

AUTOR: MARCO AUGUSTO RODAS CÁCERES

SUCRE-BOLIVIA

2024

CESIÓN DE DERECHOS

Al presentar este trabajo como requisito previo para la obtención del Diploma en Developement Operations "DEVOPS" V.1. de la Universidad Mayor, Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo al Centro de Estudios de Posgrado e Investigación o a la Biblioteca de la Universidad, para que se haga de este trabajo un documento disponible para su lectura según normas de la Universidad

También cedo a la Universidad Mayor, Real y Pontifica de San Francisco Xavier de Chuquisaca, los derechos de publicación de este trabajo o parte de él, manteniendo mis derechos de autor hasta un periodo de 30 meses posterior a su aprobación.

Marco Augusto Rodas Cáceres

……..

FIRMA:

DEDICATORIA

Dedico este trabajo de investigación a Dios, por ser un pilar fundamental en mi vida e inspirarme cada día para luchar continuamente en el proceso formativo profesional y personal. A mis padres Felipa y Augusto por ser mi ejemplo superación pese a las adversidades que se presenten. A mis hermanos adorados por ser las personas que me han apoyado incondicionalmente. A mi esposa Paola Rosario y mis hijos Sebastián, Adriana, Sofía por ser el motor de mi vida.

AGRADECIMIENTOS

Mi primer reconocimiento de gratitud es a Dios que su infinita misericordia me colmo de sabiduría e inteligencia y en medio de las adversidades me esforzó para cumplir mis metas.

A mis padres por su amor incondicional y apoyo desprendido en todos los momentos de mi vida.

A mi familia por su consideración y apoyo incondicional.

A mi esposa por nunca soltar mi mano, guiarme y acompañarme en las buenas y en las malas.

RESUMEN

En la última década, la tecnología de contenedores virtuales ha demostrado ser una gran opción al problema de portabilidad de las aplicaciones de software. Los contenedores virtuales facilitan el desarrollo rápido y ágil de aplicaciones, siendo Docker la plataforma más usada.

Bajo este contexto, en este documento se presentará los conceptos para implementar un esquema de ambientes ágiles mediante un proceso de virtualización automática en el entorno de desarrollo y entrega de aplicaciones.

El presente trabajo investiga la implementación de la metodología DevOps basado en contenedores para optimizar los procesos de desarrollo y entrega de software, destacando cómo este enfoque puede mejorar la eficiencia operativa y calidad del servicio. El estudio utiliza un enfoque de investigación mixto que integra métodos cualitativos y cuantitativos para abordar las necesidades específicas del entorno y evaluar la viabilidad de la implementación.

Megasis, una empresa nacional que cuenta con su departamento de desarrollo de software, considera beneficioso el estudio de este proceso de CI/CD basado en la tecnología de contenerización. Este estudio propone un modelo de infraestructura diseñada para trabajar con contenedores.

Palabras clave: DevOps, Docker, Kubernetes, Virtualización, CI/CD.

INDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

1. ANTECEDENTES Y JUSTIFICACIÓN

1.1 Antecedentes

A lo largo de los años el desarrollo de software ha evolucionado a pasos agigantados , es así que los entornos de desarrollo actual presentan nuevas características y desafíos al mismo tiempo, por citar tan solo unos cuantos, antaño, la infraestructura de servidores bajo los cuales se desarrollaba era monolítica ahora se tiende a los microservicios, o a la infraestructura como código, a ello se debe añadir características como mayor y más rápido despliegue (despliegue continuo), alta escalabilidad, tolerancia a fallos, empaquetamiento, migración de datos, el desarrollo en distintos ambientes, etc.

Un contenedor es un paquete estándar de software que agrupa el código de una aplicación junto con los archivos y bibliotecas de configuración relacionados y con las dependencias requeridas para que la aplicación se ejecute. (Microsoft Docs, 2020, parr. 2), los contenedores son ideales en entornos donde se requieran características como despliegue rápido, alta escalabilidad y tolerancia a fallos. Para ello se usan componentes de software diferentes y distribuidos; los contenedores son una de las mayores revoluciones de la industria del desarrollo en los últimos años y uno de los mecanismos comunes para desplegar software en cualquier servidor Empresas como Google, Microsoft, Amazon, Oracle, WMware, IBM, RedHat están apostando fuertemente por esta tecnología, ofreciendo todo tipo de herramientas y servicios relacionados con contenedores.

Por otro lado, en los últimos años, DevOps ha entrado en la etapa del desarrollo de software como una cultura que promueve la entrega de software rápida, frecuente y confiable. La cultura ha aparecido a medida que los procesos de integración y entrega continua han ocupado un papel importante en los procedimientos de desarrollo de software junto con metodologías ágiles para gestionar el desarrollo de software. CI/CD (Integración Continua/Distribución Continua) que es una parte importante de la etapa de desarrollo, en esta etapa es esencial poder replicar la

construcción del entorno con fiabilidad, y la tecnología de los contenedores contribuye a esta tarea. Es común que se tenga un sistema de integración continua en el que se realiza alguna de estas funciones: compilar el código, ejecutar pruebas de código o de seguridad, empaquetar el software, subirlo a un repositorio o servidor, desplegar la nueva versión en nuestros sistemas. Para cada uno de estos pasos se suele necesitar un entorno totalmente diferente. Para compilar se podría necesitar por ejemplo Maven, NodeJS y una variedad de librerías y código fuente. Para empaquetar quizás solo es necesario Docker o un compresor. Y para ejecutar pruebas end-to-end se tendría que instalar la infraestructura entera, un navegador y un framework como Selenium o algo parecido. El desarrollo sugiere claramente que las empresas que compiten en el mercado están analizando cómo la mejora de los procesos y la cultura y el cambio en el paradigma de desarrollo podrían dar una ventaja en la competencia con otras empresas. La energía que solía concentrarse en el desarrollo de la funcionalidad del software y su optimización ahora fluye hacia la mejora de los procedimientos de entrega de software. El cambio descrito puede dar a la empresa una "ventaja" en un mundo donde incluso un breve tiempo de inactividad significa grandes pérdidas financieras donde se esperan nuevas funciones en calendarios cada vez más ajustados.

1.1.2 Justificación metodológica

Hernández, Fernández y Baptista (2014), indican que un estudio se justifica metodológicamente cuando plantea una nueva metodología o forma que incluya otras formas de experimentar o desarrollar un trabajo o variables, las ventajas significativas que ofrecen los contenedores en comparación con los métodos tradicionales de despliegue de aplicaciones son las siguientes:

- **Portabilidad:** Los contenedores aseguran que las aplicaciones se ejecuten de manera consistente en cualquier entorno, lo que es crucial para el desarrollo y la entrega de software.
- **Aislamiento:** Cada contenedor se ejecuta de forma aislada, lo que mejora la seguridad y reduce los conflictos entre aplicaciones.
- **Eficiencia de recursos:** Los contenedores utilizan menos recursos que las máquinas

virtuales tradicionales, ya que comparten el mismo sistema operativo del host.

- **Desarrollo más rápido:** La capacidad de empaquetar una aplicación con todas sus dependencias acelera el desarrollo y las pruebas.
- **Escalabilidad:** Herramientas como Kubernetes facilitan la gestión de contenedores a gran escala, lo que es esencial para aplicaciones modernas.
- **Mantenimiento simplificado:** Los contenedores pueden ser actualizados o replicados de manera sencilla, lo que simplifica el mantenimiento y la implementación de cambios.

2.- SITUACIÓN PROBLÉMICA

Incompatibilidad: La incompatibilidad durante la etapa de desarrollo y especialmente despliegue puede ocurrir en distintos niveles:

Es un problema clásico las posibles diferencias entre las máquinas de los diferentes entornos de despliegue y/o los entornos locales entre los desarrolladores, lo cual provoca verdaderos dolores de cabeza por incompatibilidades o versionado no homogéneo de las dependencias, así como de su configuración Si se desarrolla un software y se desea pasar de un servidor instalado en un centro de datos a una máquina virtual que funciona en una nube pública, a veces el código no termina de funcionar del todo bien en su nuevo entorno. Para ejemplificar si se migra una aplicación del sistema operativo Debian a producción, en el sistema de Red Hat, puede pasar que no trabajen todas las funcionalidades.

- **Problemas portabilidad de la aplicación:** El empaquetado y la distribución de software son procesos esenciales para entregar aplicaciones a los usuarios finales. configuraciones, dependencias y metadatos que permiten la instalación, configuración y eliminación de software. El empaquetado y la distribución de software son procesos esenciales para entregar aplicaciones a los usuarios finales. configuraciones, dependencias y metadatos que permiten la instalación, configuración y eliminación de software, debido a que las plataformas de destino son distintas, es por ello que hay muchos tipos diferentes de formatos de empaquetado, como archivos ejecutables,

archivos comprimidos, instaladores, scripts, contenedores y administradores de paquetes. Cada formato tiene sus propias ventajas y desventajas, dependiendo de la complejidad, el tamaño, las dependencias y la compatibilidad de su software.

- **Migración de Sistemas antiguos a tecnologías Cloud:** Muchas empresas y organizaciones han comenzado a migrar sus aplicaciones a soluciones modernas, con el fin de modernizar y ofrecer nuevos servicios tecnológicos al usuario final. Iniciando de esa forma el camino hacia la adopción de tecnologías de Cloud, en ese transitar se útiles y ágiles para migrar cualquier desarrollo de una plataforma a otra. El empaquetado y la distribución de software son procesos esenciales para entregar aplicaciones a los usuarios finales. configuraciones, dependencias y metadatos que permiten la instalación, configuración y eliminación de software.
- Cada una de las perspectivas hacia la entrega de software descritas anteriormente resalta aspectos importantes del trabajo que la empresa debe realizar para mejorar el desempeño de entrega de software con el objetivo de crear un mayor valor para el cliente a un menor costo.

3.- FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN

¿En qué medida la implementación contenedores en la entrega de software basado en la metodología DevOps puede mejorar la eficiencia en el desarrollo de software?

4.- OBJETIVO GENERAL

Caracterizar la aplicación de contenedores como herramienta que mejorar la eficiencia de la entrega de software teniendo en cuenta el enfoque DevOps.

5.- OBJETIVOS ESPECÍFICOS

- Identificar características fundamentales de los contenedores presentes en la literatura teniendo en cuenta el enfoque DevOps.
- Determinar estrategias, herramientas, que deben ser aplicadas seguir para fomentar el uso de contenedores del desarrollo de software.

6.- DISEÑO METODOLÓGICO

6.1 Tipo de investigación

La presente investigación por su enfoque es cualitativa. El alcance de la investigación es descriptivo.

6.2. Métodos

6.2.1 Métodos Teóricos

6.2.1.1 Método Inducción – Deducción

Este método permitirá una comprensión más profunda y fundamentada de las mejores prácticas en la entrega de software, lo que facilita la toma de decisiones informadas y la mejora continua de los procesos.

6.2.1.2 Método Abstracto – Concreto

Este método asegurará que los conceptos teóricos se traduzcan en acciones prácticas que resulten en la entrega efectiva de software basado en contenedores.

6.2.1.3 Método Análisis Histórico – Lógico

Este método permitirá una comprensión profunda de cómo las prácticas de entrega de software basada en contenedores han llegado a su estado actual y cómo se pueden aplicar principios lógicos para su mejora continua y adaptación a nuevas tecnologías.

6.2.1.4 Método Análisis –Síntesis

Este método fue el que más utilizado, puesto que tuvo gran utilidad en la búsqueda y el procesamiento de la información (clasificación, descomposición, división), fue aplicado en las siguientes etapas: en la delimitación del tema, redacción de la situación problemática, la formulación del problema, redacción de los objetivos, así como en el marco teórico.

6.2.1.5 Método modelación

Este método es fundamental para la entrega de software basado en contenedores, ya que nos permitirá anticipar problemas y optimizar el rendimiento antes de la implementación real.

6.2.2 Métodos Empíricos

6.2.2.1 Método de la observación

Este método será fundamental para entender y mejorar los procesos de entrega de software, ya que permitió identificar áreas de mejora y validar prácticas efectivas a través de la contemplación directa y la experimentación.

6.2.2.2 Método de la medición

El método de la medición se centrará en la observación y análisis de datos concretos para obtener conocimientos a partir de la experiencia directa.

6.2.2.3 Método del experimento

El método del experimento se asocia a un enfoque sistemático y controlado para evaluar la eficacia de las tecnologías, métodos y herramientas utilizadas en este campo.

6.2.2.4 Método del experimento social

Este método ayudará a identificar mejores prácticas y estrategias para fomentar un ambiente colaborativo y eficiente en los equipos que utilizan tecnologías de contenedores. Además, proporcionará información valiosa sobre cómo la cultura organizacional y las herramientas de DevOps, como la virtualización basada en contenedores, pueden influir en la calidad y rapidez de la entrega de software.

6.3 Técnicas

6.3.1 Análisis de Datos

Análisis cuantitativo: Utilizar técnicas estadísticas para analizar los datos recogidos a través de encuestas y cuestionarios. Esto incluirá el uso de software estadístico para realizar pruebas de significación, correlaciones y análisis de regresión, si es necesario.

Análisis cualitativo: Emplear técnicas como la codificación temática para evaluar las respuestas a preguntas abiertas en cuestionarios y entrevistas, así como para analizar los datos obtenidos en los estudios de caso.

6.3.2 Visualización de Datos

Herramientas de visualización: Utilizar herramientas como Microsoft Excel para crear visualizaciones de datos que ayuden a interpretar tendencias, patrones y anomalías en los datos recogidos.

6.4 Procedimientos e instrumentos de investigación

6.4.1 Procedimientos

Diseño de Cuestionarios: Desarrollar cuestionarios basados en la literatura revisada y los objetivos específicos del estudio. Estos cuestionarios serán pilotados inicialmente con un pequeño grupo de desarrolladores para garantizar su validez y ajustar cualquier elemento confuso o inadecuado antes de su implementación final.

Selección de Casos de Estudio: Identificar y seleccionar casos de estudio donde han sido implementados contenedores para el desarrollo y entrega de software. El criterio de selección incluirá diversidad en términos de tamaño del proyecto, tipo de aplicación y resultados.

6.4.2 Instrumentos de Investigación

Cuestionario: Diseñados para recoger información específica sobre la experiencia, percepciones y desafíos del equipo DevOps en el uso de contenedores.

Guías de Entrevista: Entrevistas semiestructuradas con miembros clave del equipo en Megasis, con el fin de obtener un entendimiento más profundo de las expectativas y experiencias.

CAPITULO I

MARCO TEORICO Y CONTEXTUAL

1.1 Marco Teórico

Los contenedores son conjuntos de procesos que se ejecutan de forma aislada en una partición de los recursos de un sistema, y proporcionan mejoras a la hora de la entrega de aplicaciones, la contención controlada del uso de recursos o el aislamiento de entornos. Por diseño, los contenedores forman parte del sistema operativo y, por tanto, están pensados para ser ejecutados en un entorno de solo un recurso, pero es precisamente por la cantidad de beneficios que ofrecen y por su popularización en el ámbito de los microservicios que surgió la necesidad de gestionar varios grupos de contenedores a lo largo de varios recursos distribuidos.

Es por eso que, a mitad de 2014, un grupo de ingenieros de Google decidieron poner en marcha el proyecto Kubernetes, con la idea de ser capaz de distribuir varios contenedores en diferentes máquinas y que se pudiesen comportar como un único sistema. Aunque Kubernetes no es la única tecnología que se usa actualmente para cubrir esta necesidad, sí no más bien es la que sugiere como orquestador y por tanto será nombrada más adelante. No obstante, cabe destacar que hay otras opciones también muy populares, como Mesos y Nomad, que ofrecen soluciones similares.

El proyecto Kubernetes rápidamente fue agarrando forma y se asoció con la Linux Foundation, y con todo el esfuerzo dedicado a su desarrollo se convirtió en uno de los proyectos con más commits de todo GitHub.

Previamente, lo más utilizado para manejar varios grupos de contenedores era Docker Swarm. Sin embargo, al llegar Kubernetes al mercado, con una propuesta mucho más robusta y con más capacidades, Docker Swarm se dejó de lado en pos de Kubernetes en muchos ámbitos, aunque sigue siendo una tecnología muy eficaz para gestionar unos pocos recursos.

1.2.1. DevOps

Para entender el concepto de DevOps debemos se debe hacer referencia del modelo de roles tradicional. Concretamente de las responsabilidades y funciones que toma cada rol.

Desarrolladores (Developers): responsables de diseñar y crear la aplicación propiamente dicha.

Control de calidad o Quality Assurance (QA): responsables de realizar las pruebas pertinentes para verificar la funcionalidad de la aplicación, que los requisitos se hayan cumplido, detectar fallos, etc.

Operaciones (Information Technology Operations): responsables de mantener y crear toda la infraestructura y el entorno para que la aplicación funcione correctamente. Este ámbito también es conocido comúnmente como administración de sistemas.

Jenkins

Jenkins es un producto del software que se encarga de automatizar y optimizar las tareas de DevOps (Development and Operations, Desarrollo y Operaciones), y facilitar el CD/CI (Continuous Development/Continuous Delivery, Desarrollo Continuo /Entrega Continua).

Es una herramienta utilizada por el equipo de QA para automatizar el despliegue y ejecución de tests en diferentes recursos, y permite organizar los resultados y ejecuciones de manera que los ingenieros tengan que gastar menos tiempo en estas tareas. Además, permite la conexión con otras herramientas de análisis y control del proyecto con funciones orientadas a metodologías ágiles como el SCRUM, lo cual facilita entender en qué estado están los diferentes proyectos que pueda estar abordando el equipo.

1.2.2 Virtualización

La virtualización surge de la necesidad de crear entornos software que recreen otro entorno como el real, con el objetivo de solo necesitar un entorno hardware capaz de soportar varios virtualizados. En este apartado se estudian diferentes formas de virtualizar.

Hipervisor

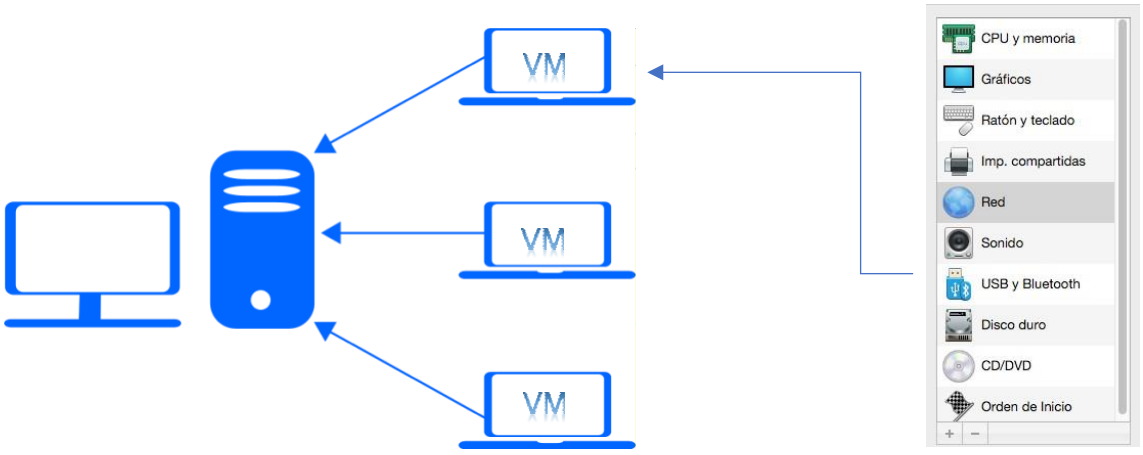
Otra forma de virtualizar es usando hipervisores, un entorno sobre el que se pueden ejecutar máquinas virtuales. Hay dos clases de hipervisores:

- Tipo 1: tanto el sistema operativo (SO) principal de la máquina física, como los sistemas virtuales se ejecutan sobre el hipervisor de forma aislada, es decir, independiente. Así el hipervisor proporciona acceso a los controladores y recursos.
- Tipo 2: El hipervisor se ejecuta sobre el sistema operativo, de esta forma los sistemas virtuales se ejecutan sobre el hipervisor. Así el hipervisor proporciona acceso de los sistemas virtuales al SO de la máquina física, y el aislamiento solo es entre los sistemas virtuales, no entre cada sistema virtual y el SO. Actualmente los programas de virtualización de máquinas se basan en hipervisores de este tipo.

Máquina Virtual

Para virtualizar entornos completos de SO se pueden usan máquinas virtuales (VM), ya que así se virtualiza un ordenador completo, incluso configurando la red en modo “bridge” la máquina virtual se ve en la red como un ordenador más.

Figura 1. Esquema de máquinas virtuales en máquina física



Fuente: Elaboración propia

Como se ve en la figura anterior se virtualizan máquinas en una máquina física, pudiéndose configurar todos los elementos hardware para virtualizar un ordenador de nuestra red. Donde como gran ventaja se tiene la posibilidad de ejecutar en una máquina física, varios SO. Pero esto conlleva, como gran desventaja, el consumo de los recursos, de forma no óptima, al disponer de instalaciones completas de SO por cada VM.

Virtualización ligera

Es un tipo de virtualización que se encarga de crear el entorno para el sistema virtual, usando un entorno lo más ligero posible compartiendo lo común del este y que no hace falta aislar en cada sistema virtual. Por ejemplo, para virtualizar máquinas Linux sobre un SO con el mismo kernel, no es necesario cargar todo el SO, ya se comparte.

Con esta idea surgen los contenedores, que en el siguiente apartado se estudian en más profundidad, y que llegan a ser tan ligeros que en algunos sitios llega a decirse que no es un sistema virtualizado, siendo solo un empaquetamiento de herramienta.

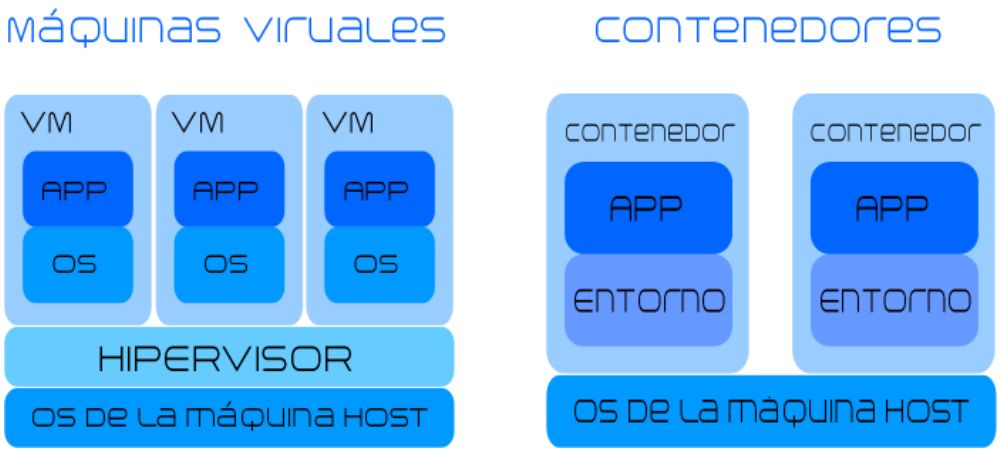
1.2.3. Contenedores

La tecnología de contenedores permite ejecutar aplicaciones de manera que no es necesario arrancar una VM con todo un SO, ya que utiliza las herramientas necesarias para la aplicación, siendo más eficiente y ligero que una VM.

Así se puede definir que un contenedor es un paquete que contiene una aplicación y las herramientas necesarias para que se ejecute. Por lo tanto, un contenedor está dotado de su propio entorno, y es independiente del entorno en el que se ejecute. Está cualidad permite que las aplicaciones empaquetadas en contenedores se puedan ejecutar en diferentes plataformas y SO.

En la siguiente figura se muestra las diferencias que supone ejecutar una aplicación empaquetada en un contenedor y esa misma aplicación ejecutada sobre una máquina virtual.

Figura 2. Imagen de la comparación de capas entre virtualización y contenedores



Fuente: Elaboración propia

- **Ventajas principales del uso de contenedores**

El uso de contenedores nos proporciona ventajas relacionadas con sus características. Las más destacadas son las mencionadas a continuación y que facilitan tanto el desarrollo de aplicaciones como su instalación.

Desarrollo:

Para el desarrollo de nuevas aplicaciones podemos necesitar instalar nuevas herramientas en el entorno donde queremos ejecutar la aplicación, que pueden ser un inconveniente para otra aplicación ya en ejecución.

Así con el uso de contenedores, que proporcionan aislamiento, podemos trabajar de forma segura y posteriormente, si no nos gusta el resultado, borrar el contenedor de nuestra máquina.

Pruebas:

El uso de contenedores ofrece varias ventajas importantes de cara a desarrollar o probar nuevas aplicaciones, ya que el contenedor al estar dotado de su propio entorno se ejecuta de forma aislada y, por lo tanto, permite ejecutar varias versiones de una aplicación simultáneamente, o probar nuevo código sin tener que dejar de ejecutar el anterior.

Distribución:

De la propiedad de aislamiento que tienen los contenedores, y al disponer de las herramientas necesarias para la ejecución de las aplicaciones empaquetadas, es fácil almacenarlas en un sitio accesible para su despliegue. Estos sitios se llaman repositorios, e incluso disponen de control de versiones de las aplicaciones que se actualizan. Así para desplegar una aplicación empaquetada en un contenedor se descarga del repositorio correcto.

Para desarrollar contenedores podemos utilizar diferentes tecnologías que nos permiten crear aplicaciones y ejecutarlas, como Linux Containers y Docker.

- **Linux Containers (LXC)**

LXC es una interfaz para la creación y gestión de contenedores de Linux. Y presta el servicio de un SO de forma aislada compartiendo el kernel del SO Linux de la máquina sobre la que se ejecutan. Por lo que, aunque los contenedores pueden pertenecer a distribuciones de Linux diferentes deben de ser distribuciones que usen el mismo kernel, como ya se mencionó anteriormente.

Los contendores lanzados deben de ejecutarse de manera aislada, de forma que LXC usa Cgroups (grupos de control) y Namespaces (espacios de nombres) para manejar esta cualidad. Además, LXC usa librerías y otras herramientas necesarias para la administración de los contenedores.

- **Docker**

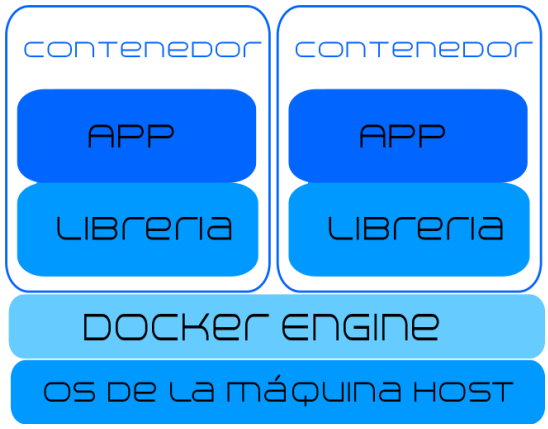
Docker es un proyecto de código abierto y una plataforma de contenedores, que permite automatizar el despliegue de aplicaciones empaquetadas en contenedores.

Docker permite independencia entre las aplicaciones y el entorno de ejecución, ya que como anteriormente se mencionó, un contenedor empaqueta tanto la aplicación como herramientas del sistema necesarias. Por esto, Docker ofrece abstracción de virtualización de las aplicaciones sobre el SO utilizado en la máquina que se ejecuta.

Esto dota a las aplicaciones de portabilidad, ya que un contenedor Docker funcionando en una máquina Linux, funciona también en una máquina con Windows o macOS.

A continuación, se muestra una imagen que representa los componentes que usa el motor de contenedores para Docker (Docker Engine) donde se ve cómo funciona como middleware ente los contenedores y el SO.

Figura 3. interfaces para acceder a las capacidades de virtualización del kernel Linux.



Fuente: Elaboración propia.

A continuación, se estudian los componentes necesarios para crear un contenedor Docker:

Imagen Docker

Una imagen Docker es la plantilla que se usa para crear el contenedor de una aplicación. Así una imagen Docker contiene una imagen base, además de la aplicación, que puede ser de un SO como Ubuntu o Alpine.

Linux Alpine

Es interesante destacar esta distribución de Linux, ya que para la creación de contenedores está muy optimizada, al tratarse de una distribución muy ligera y minimalista. Esto se muestra en la siguiente imagen que nos permiten tener contenedores más rápidos.

Alpine

Con un ahorro de tamaño tan considerable, y lo que conlleva, como mejor tiempo de descarga y despliegue, hay empresas que están cambiando las imágenes base de sus contenedores, para funcionar sobre Linux Alpine. Con esta presentación es lógico pensar que nos interesa crear nuestros contenedores sobre imágenes basadas en Linux Alpine, pero dado que conocemos mejor Ubuntu, a lo largo del proyecto también lo usaremos como imagen base para la creación de nuestros contenedores.

Figura 4: Imagen Ubuntu Vs Imagen Alpine

Fuente: Elaboración propia.

Fichero Dockerfile

Dockerfile es el documento que sirve para crear una imagen Docker, y por lo tanto contendrá la imagen base y la información para la composición de dicha imagen, es decir, un Dockerfile contiene las instrucciones necesarias para componer una imagen Docker.

Docker Hub

Se aloja en la nube y es útil para almacenar repositorios de los ficheros Dockerfile y disponer de las imágenes, que se han creado, en todo momento.

Docker Hub además nos permite acceder a los repositorios de otras personas, ya que se pueden crear tanto repositos públicos como privados.

Para acceder a Docker Hub se puede hacer identificándonos con nuestro identificador de Docker (Docker ID). Además, se puede visualizar si es una imagen oficial, como se muestra en la siguiente figura, tras la búsqueda del repositorio para Ubuntu.

1.2.4. Orquestación

Kubernetes: Kubernetes es una plataforma de código abierto desarrollada por Google, y que posteriormente fue mejorado a través de su comunidad, ya que fue donado a la "Cloud Native Computing Foundation". Es una evolución del proyecto "Borg" de Google. A nivel de funcionamiento Kubernetes es un orquestador de la ejecución de aplicaciones que se ejecutan en contenedores, y permite gestionar estas aplicaciones. Kubernetes significa timonel en griego, y provee de tecnología de despliegue, mantenimiento y escalado de aplicaciones entre sus funciones de orquestación de contenedores. En muchos sitios podemos ver que se refieren a Kubernetes como "K8s".

Admite varios motores de ejecución de contenedores entre los que tenemos Docker, el cual se usa para el desarrollo del proyecto, ya que Kubernetes fue diseñado, principalmente, para orquestar contenedores Docker al ser tecnología de Google.

Arquitectura de Kubernetes: Kubernetes distribuye los contenedores en pods, así estos pueden estar en varios nodos. A su vez los nodos forman un clúster, completando la estructura que tiene Kubernetes. A continuación, se explica de manera más detallada los elementos de Kubernetes.

Pods de Kubernetes

En Kubernetes los contenedores se agrupan en pods, por lo que todos los contenedores que se ejecutan en un pod lo harán en la misma máquina o host, ya que no se pueden separar. Debido a esto se agrupan en el mismo pod a contenedores que usarán y necesitarán los mismos recursos, por lo que puede decirse que forman un host lógico. Si entendemos que podemos decir que un pod es un host lógico dentro del clúster, es fácil entender que un pod tiene una dirección IP compartida por los contenedores que lo forman. Además, todos los Pods se alcanzan entre ellos, dado que comparten una red privada.

Así un nodo puede tener varios pods ejecutándose, y por lo tanto el encargado de administrarlos es máster. Hay que destacar que existen pods con un solo contendor "one-

container-per-Pod”, donde el pod es como un envoltorio del contendor.

Nodo de Kubernetes

Aunque se puede decir, a efectos prácticos, que las aplicaciones se ejecutan en el clúster (conjunto de nodos), realmente la aplicación se ejecuta en los nodos, siendo los contenedores distribuidos por Kubernetes de manera automática.

Estos nodos pueden ser un bien un ordenador, bien una máquina virtual, o una máquina en la nube. Y están administrados por un agente que se llama Kubelet, que más adelante veremos.

Los nodos pueden ser de dos tipos: los nodos esclavos y los nodos maestros. Por simplicidad a los nodos esclavo se les llama simplemente nodos y a los maestros, nodos máster, o simplemente máster.

Nodo máster de Kubernetes

Kubernetes utiliza el nodo máster para las labores de administrar y planificar los pods que se ejecutan en los nodos del clúster y por lo tanto de los contenedores que estos contienen. Esto se realiza a través de controladores de Kubernetes.

En función de la carga de trabajo, podríamos disponer de varios nodos máster, que dotan al sistema de más resistencia ante fallos.

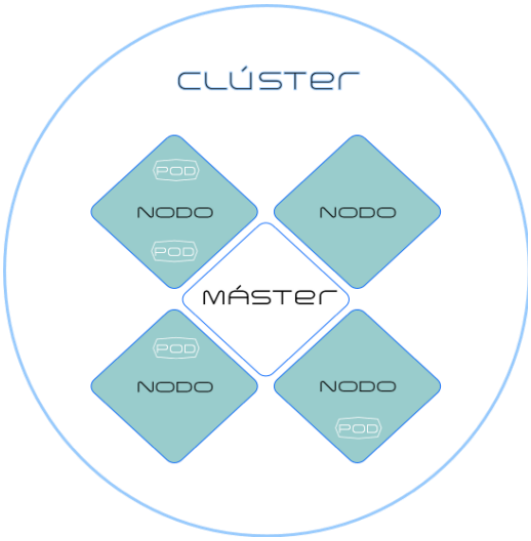
Clúster de Kubernetes

Un clúster se forma por los dos elementos explicados anteriormente, como también se puede deducir de la introducción de este apartado.

De esta forma Kubernetes tiene que coordinar contenedores en un sistema formado por varios nodos, haciendo que todo funcione como una sola unidad orquestada por el máster, logrando que las aplicaciones no estén vinculadas a una sola máquina.

Esto evita la necesidad de instalar una aplicación directamente en una máquina o host, realizándose sobre el clúster.

Figura 5. Clúster de Kubernetes



Fuente: Elaboración propia

En la figura anterior se ve la estructura de un clúster. Como mínimo un clúster tiene que estar formado por tres nodos, es decir, un máster y dos nodos esclavos, a excepción de Minikube que veremos más adelante y que usa uno para todo.

La comunicación de los nodos con el máster, y del usuario con el clúster se hace a través de la API de Kubernetes.

Componentes de Kubernetes

Para que la arquitectura descrita antes funcione correctamente Kubernetes utiliza una serie de componentes en cada elemento del sistema dotando al sistema de las funcionalidades necesarias.

Servidor de API (API server): El servidor de la API de Kubernetes, sirve para acceder a la API de Kubernetes en nuestro clúster. Por eso, es el front-end del plano de control de Kubernetes, es decir, es donde llegan las peticiones al clúster correspondientes a un servicio determinado. Estas pueden llegar desde un nodo o desde una petición por parte del administrador conectado al máster, y lo redirige a los componentes que correspondan.

Se encarga de preparar, validando y configurando, los datos de los objetos api (que más tarde explicaremos) que necesitamos para el clúster. Y se encarga de dar servicio de publicación de recursos del clúster. También prepara la interfaz a través la cual los componentes del clúster interactúan con los demás componentes.

Etcd: Es una base de datos que almacena y guarda la configuración del clúster, almacenando también información de los servicios que están disponibles. Etcd estará replicado en todos los nodos máster del clúster para asegurar una alta disponibilidad de la información. Esta información será también usada por el API server para que los servicios desplegados en los contenedores mantengan sus características de forma coherente.

Planificador (Scheduler): Asigna a los nuevos pods un nodo en el que ejecutar su trabajo, es decir, se encarga de repartir los recursos disponibles en el clúster, para la ejecución de los pods tras valorar los requisitos que necesitan para desarrollar su trabajo.

También es el responsable de monitorizar la utilización de recursos de cada host para asegurar que los pods no sobrepasen los recursos disponibles una vez ya estén en funcionamiento.

Gestor de controladores (Controller-manager): Es un componente que como su nombre indica se encarga de gestionar los controladores de los nodos. Estos controladores son:

Controlador de réplicas (Replication Controller): Se encarga de controlar la ejecución correcta de réplicas deseadas para una aplicación.

Controlador de nodo (Node Controller): Se encarga de controlar que los nodos pertenecientes a un clúster funcionan de manera correcta y detectar si un nodo ha dejado de funcionar.

Controlador de puntos finales (End-points Controller): Gestiona los puntos finales (end-points) de los servicios desplegados.

Controlador de la nube: Es un controlador más o menos moderno, es decir, de las últimas versiones de Kubernetes, y gestiona la conexión con el proveedor de servicios en la nube que se contrata para nuestro clúster.

Componentes de los nodos

Kubelet

Se ejecuta en cada nodo gestionando los pods y su contenido a través de los ficheros que describen cada pod (objeto YAML o JSON), y juntos forman las especificaciones de un pod.

Hay tener en cuenta que Kubelet no administra contenedores que no pertenecen a Kubernetes, es decir, que no hayan sido creados por Kubernetes.

Kube-proxy

Se ejecuta en cada nodo y proporciona abstracción de servicios al realizar el reenvío de conexión. Así, cuando llega una petición de servicio a un nodo por parte de un usuario desde el exterior, a un nodo donde no se está ejecutando algún pod de la aplicación que se encarga de ello, Kube- proxy se ocupa de hacer que la conexión se reenvíe al nodo correcto.

cAdvisor

Se dedica a recoger, información del uso de los recursos que se usan en los nodos vigilando la CPU, la memoria, sistemas de ficheros y el uso de la red. La información recogida se usa para informar al nodo maestro.

Complementos de los nodos

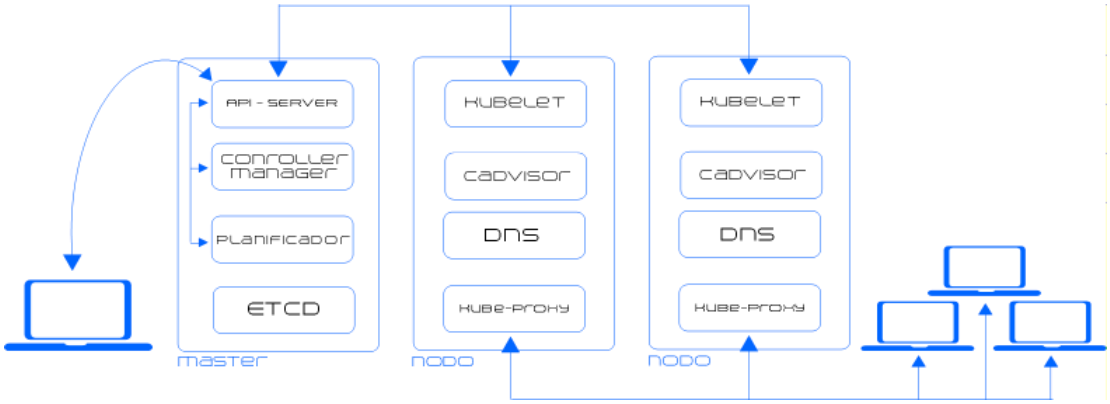
Los complementos dotan a la estructura del clúster de funcionalidades adicionales.

DNS

Pese a ser un complemento, su funcionalidad es necesaria para el correcto funcionamiento del clúster. Será usado por kube-proxy, para reenviar la información.

En la figura posterior se muestra la estructura completa de la arquitectura con los componentes en cada elemento del clúster. También la representación de las conexiones entre los nodos y el máster, los usuarios con el clúster y el administrador con el clúster.

Figura 6. Imagen de los componentes en la estructura K8s



Fuente: Elaboración propia.

- **Objetos API de Kubernetes**

Todos los elementos vistos hasta ahora, como nodos, controladores, pods, y otros que veremos más adelante, son considerados objetos API para ser administrados por Kubernetes.

Para esto se definen ficheros en formato JSON o YAML que se llaman ficheros de definición, para los objetos de nuestro clúster, con los que posteriormente podremos crear el objeto en el sistema. También se puede crear un objeto y configurarlo directamente en el clúster, utilizando Kubectl.

En la API de Kubernetes, donde cada objeto tiene su representación nos podemos encontrar tres tipos de objetos en función de su desarrollo:

Alpha: versiones de objetos API recién definidos, apenas probada y por lo tanto usarlos nos podrían dar problemas, así que si no es necesario es bueno evitar usarlos.

Beta: versiones ya más probadas, pero no definitivas. Hay que evitarlas para aplicaciones que se necesiten a niveles profesionales.

Estable: es una versión del objeto que ha sido probada con éxito.

- **Recursos en Kubernetes**

Las máquinas que se usan en un clúster de Kubernetes tienen recursos limitados y por lo tanto es importante conocer como son usados por las aplicaciones, es decir, como se asignan los recursos de los nodos en la ejecución de pods.

Los dos principales recursos por nodo son los recursos de memoria y los recursos de CPU o lo que es lo mismo, recursos de computación.

Asignación de recursos

Para el control de estos se pueden establecer límites de consumo en los contenedores y los pods. Estos límites se establecen en los objetos con la etiqueta "limits".

De igual modo que los límites de memoria y CPU, podríamos querer establecer unos requisitos de lanzamiento de los contenedores y pods. En este caso en el objeto se configura una etiqueta "request", que hace referencia a los recursos solicitados.

Para establecer el valor de los límites y de los recursos requeridos, en los pods y contenedores, se fija el valor de las etiquetas específicas de cada recurso. Para lo cual se utilizan unidades de memoria y CPU, usando etiquetas "memory" y "CPU" respectivamente.

Así para el recurso de memoria la unidad es el byte, y para el recurso de cómputo la unidad es la CPU. Por ejemplo, si se quiere asignar 1GB de memoria se establece el valor "1G" o "1Gi", y si queremos asignar la mitad de la CPU se establece el valor 0.

Comportamiento en caso de exceder recursos

Es muy importante conocer cómo se comporta Kubernetes en caso de que los recursos sean excedidos. Para ello vemos los siguientes casos:

Un contenedor o pod solicita más recursos que su límite: en este caso el contenedor o pod es finalizado por exceder los recursos.

Un contenedor o pod solicita más recursos de los disponibles: en este otro caso el contenedor o pod se queda a la espera de un nodo con los recursos suficientes para su ejecución, en estado "Pending", es decir, pendiente de asignación de nodo.

- **Espacio de nombres (Namespaces)**

El espacio de nombres sirve para dividir el clúster físico de manera virtual, y podría decirse que crea clústeres virtuales. Concretamente los recursos que son divididos, como los nodos, no pertenecen a ningún namespaces a diferencia del resto, por ejemplo, cualquier pod desplegado.

Además, esta división lógica crea aislamiento de nombres entre los objetos de diferentes espacios de nombres. Por ejemplo, podríamos tener un pod en un nodo con un nombre, y en el mismo nodo, pero en un namespace diferente otro pod con el mismo nombre. También se pueden asignar permisos de creación o modificación de recursos a cada espacio de nombres para diferentes usuarios.

Cuando se crea un clúster de Kubernetes, se crean por defecto tres namespaces en el sistema:

Default: cuando se crean objetos a los que no se le ha especificado un espacio de nombres concreto se le asigna el namespaces "default".

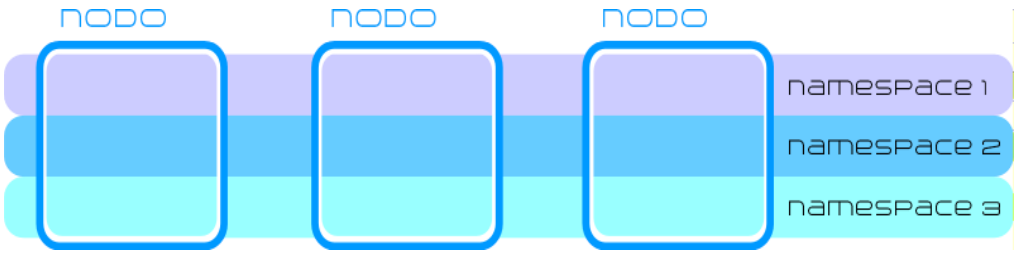
Kube-system: Kubernetes necesita objetos para funcionar y desempeñar su trabajo. Estos objetos se crean en este espacio de nombres

Kube-public: como su nombre indica es público y por lo tanto todos los usuarios pueden acceder a su contenido. Puede ser un espacio que esté vacío o que contenga información pública que interese mostrar.

Adicionalmente, a estos tres espacios de nombres, se pueden crear nuevos espacios para dividir el sistema de manera conveniente. Por ejemplo, sería interesante crear un nuevo espacio de nombres dedicado a pruebas de las aplicaciones que queramos testear previamente.

A continuación, se muestra una imagen que representa cómo sería esta división, de cada componente físico.

Figura 7. Separación del sistema físico, por el uso de namespaces



Fuente: Elaboración propia.

Las cuotas en los espacios de nombres

Los recursos también se pueden regular en los espacios de nombres para asegurarse de que un grupo de servicios no consuman todos los recursos. Por ejemplo, es muy útil configurar nuestro clúster para que un namespace dedicado a ejecutar aplicaciones y servicios en prueba no consuman recursos necesarios de los servicios que ya estén lanzados.

Para configurar cuotas se usa un objeto ResourceQuota y se pueden configurar no solo los recursos de memoria y CPU, también el número de pods, a través de etiquetas.

El objeto LimitRange

Otro recurso de control de recursos que se puede configurar en los espacios de nombres es un objeto LimitRange. Este objeto asigna valores de límites de recursos y recursos requeridos cuando no están especificados en los objetos contenedores.

- **Objetos controladores**

Al igual que los objetos que definen recursos del clúster, como los pods, también existen objetos de los controladores encargados de que este funcione correctamente para que pueda gestionar los pods y por lo tanto la orquestación de contenedores. A continuación, vemos los esenciales para el proyecto.

Deployment

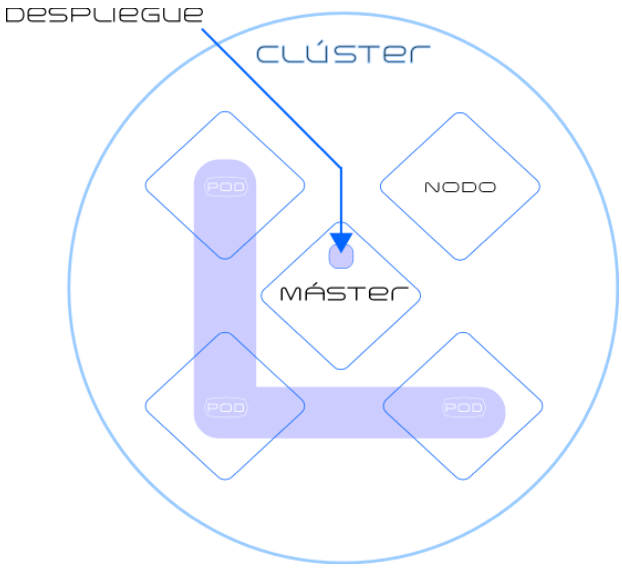
Es el controlador de despliegues de contenedores que necesitamos para nuestra aplicación, es decir, se encarga de que esta se ejecute en función de unas características concretas. Por ejemplo, el número de pods que queremos que se ejecuten.

ReplicaSet

Es un controlador de réplicas de pods de la aplicación desplegada y conlleva la creación de un objeto ReplicaSet. Por lo que es la propia implementación del despliegue la que se encarga de la administración de este objeto.

Estas cantidades especificadas de réplicas se vigilan y en caso de que no se cumplan ReplicaSet se encarga de recuperar el estado deseado del número de réplicas. Por ejemplo, si hemos especificado que queremos tres réplicas de un pod y uno de los nodos de nuestro clúster deja de funcionar, el ReplicaSet se encargará de solicitar la creación nuevos pods y así estos se alojaran en otros nodos manteniendo la configuración deseada.

Figura 8. Imagen de una implementación desplegada en un clúster.



Fuente: Elaboración propia.

- **Almacenamiento**

Las aplicaciones que se ejecutan en contenedores pueden necesitar usar algún tipo de almacenamiento para su correcta ejecución, ya sea solo durante la ejecución o recibiendo información del sistema de almacenamiento. Así Kubernetes especifica el uso de volúmenes capaces de almacenar datos.

Volúmenes

Para este almacenamiento se crea un objeto volumen, llamado "volumen", que se ejecuta en el pod y accesible por todos los contenedores ejecutados en el mismo pod. Este tipo de almacenamiento es temporal, es decir, lo que se almacena en un pod solo permanece durante la ejecución del pod. Por lo que, si tanto un pod como un nodo que ejecuta ese pod deja de funcionar, Kubernetes puede lanzar otro pod para reponer el servicio, pero el volumen vuelve a tener la información especificada en la configuración de la aplicación para ser desplegada.

EmptyDir

Monta un volumen vacío en el pod del contenedor en la ruta especificada en la configuración del contenedor con la etiqueta "emptyDir".

Volúmenes persistentes

Para usar almacenamiento persistente en un pod se utiliza un volumen de tipo persistente llamado persistentVolume. Este tipo de volumen cargará una ruta de la máquina física y con una petición de volumen llamada persistentVolumeClaim se monta en el contenedor.

Cuando se modifica algo en este volumen también se modifica en la máquina física. Así si un nodo se reinicia sigue teniendo la información que modificó.

HostPath

Es un tipo de volumen persistente que monta una ruta del sistema de ficheros del nodo en la ruta indicada en la configuración del contenedor. Esta información se da como valor de la etiqueta

“path” que estará anidada debajo de la etiqueta “hostPath”.

Volúmenes de proveedores de nube pública

Tanto Google como Amazon dispone de sus propios sistemas de almacenamiento persistente. Particularmente en el caso de Google el volumen se llama “gcePersistentDisk”.

Estos volúmenes solo se pueden usar en los contenedores de la misma zona geográfica. Por lo que, en caso de uso, es importante usar mecanismos como los selectores para que los pod que quieran usar estos volúmenes cumplan con este requisito de zona.

- **Exponer una la aplicación en Kubernetes**

Cuando la aplicación ya es desplegada en un clúster es importante saber cómo conectarse a la aplicación para que un cliente pueda realizar la petición de un servicio.

Una forma es a través del CLI de Kubernetes, es decir, usando comandos de Kubectl abriendo un proxy en el nodo donde tengamos el pod de la aplicación, exponiendo un puerto para redirigir el tráfico a ella. Esta forma es solo interesante para probar la aplicación, pero ya que se quiere tener prestando servicio para peticiones externas, hay otra forma de acceder a través de los servicios en Kubernetes.

Servicios en Kubernetes

Los servicios exponen la aplicación de todas las réplicas existentes, en todo el clúster, de una aplicación de manera unificada. Para lo que se usan las etiquetas y selectores. Además, los servicios hacen posible redirigir las conexiones ofreciendo balance de red, es decir, distribuyendo la carga.

Tipos de servicios:

ClusterIP

Define el servicio asignando una IP interna de clúster y por lo tanto solo se puede acceder al servicio desde el propio clúster. De esta manera no hay que preocuparse de las variaciones de las direcciones IP de los nodos del clúster que se puedan dar por sustitución de nodos o reasignación de direcciones.

Es un tipo de servicio útil para aplicaciones dentro de nuestro clúster que quieren acceder a un servicio back-end.

NodePort

Asigna el mismo puerto en cada nodo para cada servicio, así con la dupla de NodeIP (IP del nodo) y NodePort (puerto asignado par el servicio) se puede acceder al servicio desde fuera del clúster. Kube-proxy se encargará de enrutar, el tráfico usando el servicio DNS de Kubernetes, al nodo correcto.

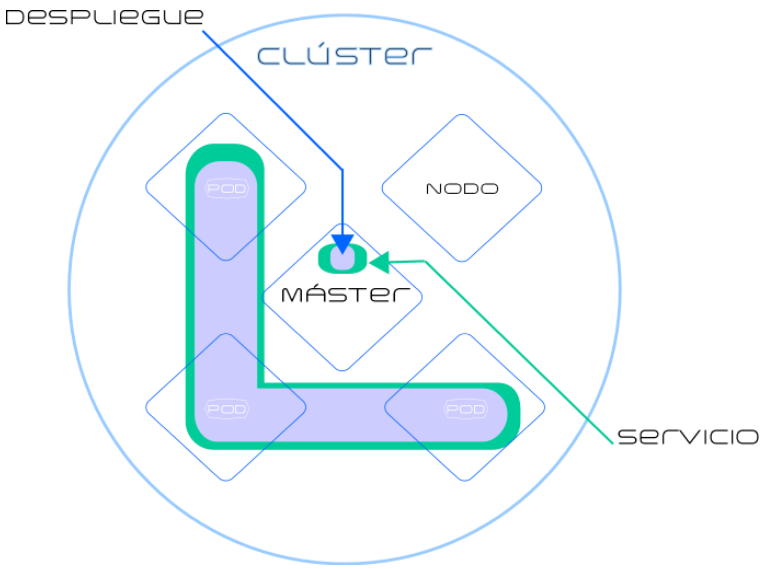
Para que el usuario que acceda a los servicios no tenga la necesidad de conocer un bloque de direcciones IP se puede crear un balanceador externo.

LoadBalancer

Si usamos un proveedor de servicios en nube para nuestro clúster de Kubernetes no necesitamos crear un balanceador de carga, ya que la mayoría de los proveedores tienen plugins para esta tarea. Así a estos servicios se denominan de tipo LoadBalancer y crea una IP fija y externa al servicio para que el balanceador se encargue de encaminar el tráfico.

En la siguiente figura se representa gráficamente como queda en servicio un despliegue de contenedores de una aplicación (deployment) que tiene tres pods distribuidos en tres nodos.

Figura 9. Estructura de un clúster con una implementación expuesta como un servicio.



Fuente: Elaboración propia.

- **Networking de Kubernetes**

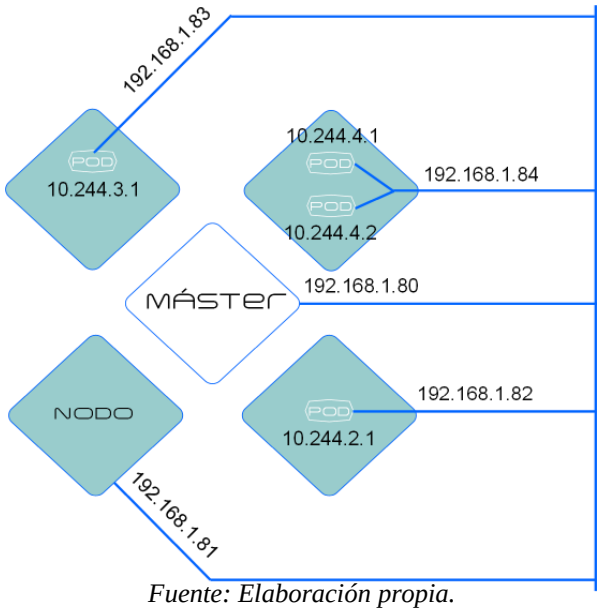
Un pod es la unidad más básica que nos encontramos en la arquitectura de Kubernetes. Por lo tanto, cada pod tiene una dirección IP perteneciente a una red privada del clúster. Así los contenedores que pertenecen a un mismo pod comparten esta dirección IP, siendo las conexiones diferenciables por los puertos que usa cada contenedor. Por lo que un end-point, en Kubernetes, hace referencia a un pod.

Para que la conectividad funcione correctamente en el clúster tiene que cumplirse:

- Que todos los pods de un mismo espacio de nombres puedan alcanzarse entre ellos.
- Que todos los nodos puedan alcanzarse entre ellos.
- Que todos los pods puedan alcanzar todos los nodos.

En la siguiente figura se muestran la configuración de la estructura de Kubernetes en una red privada (10.244.0.0/16) para los pod.

Figura 10. Direccionamiento de red de un clúster



Fuente: Elaboración propia.

Algunos plugins CNI:

- **Flannel:** por su simplicidad, y demostrada eficacia en su funcionamiento, le hacen el plugin a instalar en el clúster casi por defecto.
- **GCE plugin de enrutamiento avanzado:** es el plugin de Google, y se usa en su estructura de solución en nube.
- **Kube-router:** está pensado para Kubernetes y diseñado para dotar al sistema de alto rendimiento.
- **Calico:** es apropiado para redes escalables. Usa BGP distribuir las rutas.

➢ Clúster HA

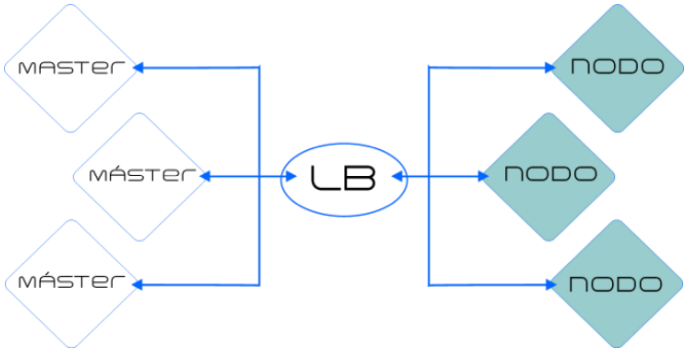
Kubernetes permite descentralizar un sistema para que se pueda evitar el principal problema de un sistema centralizado, es decir, la existencia de un único punto de fallo.

No obstante, al crear un clúster normal se desplaza este problema al coordinador, es decir, al

nodo máster encargado de orquestar los servicios. Para lo cual existe la solución de crear clústeres de alta disponibilidad (Hight Avality) de dos formas.

La primera es creando un clúster con más de un máster, para lo que es necesario un componente balanceador de carga entre los nodos y los nodos máster, como se muestra en la siguiente figura.

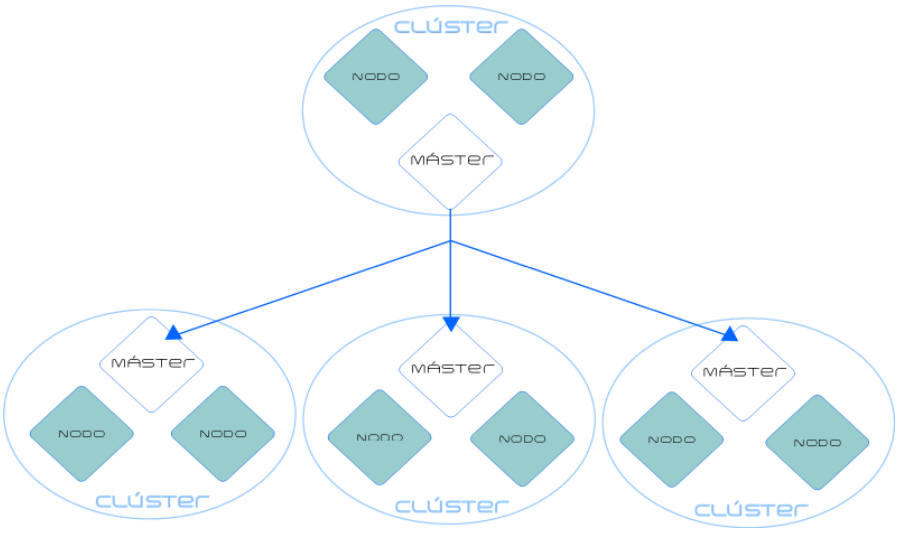
Figura 11. Clúster HA con múltiples nodos máster



Fuente: Elaboración propia.

Otra forma que hay de crear alta disponibilidad es usando federaciones de Kubernetes. Esto consiste en crear varios clústeres y añadirlos a la misma federación. La siguiente imagen muestra un esquema de esta configuración.

Figura 12. Federación de clústeres

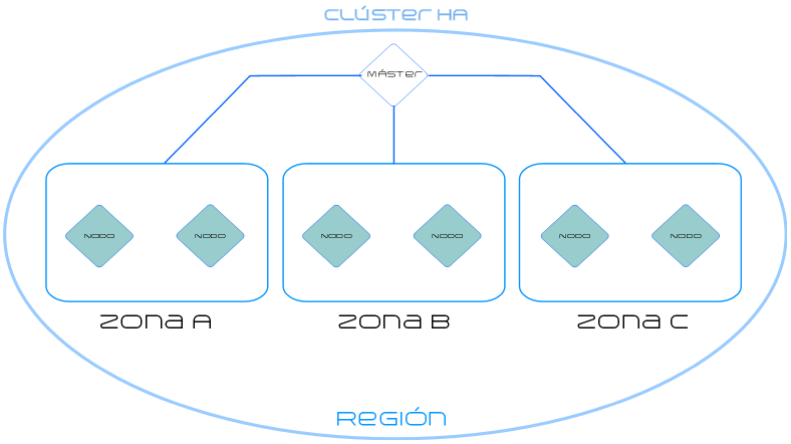


Fuente: Elaboración propia.

La alta disponibilidad de esta forma llega a ser una solución fiable si cada clúster se configura en diferentes zonas geográficas o regiones, ya que se entiende que dos zonas diferentes rara vez van a estar inaccesibles al mismo tiempo. Así una región se divide en zonas.

Google Cloud Platform permite administrar con un máster nodos de diferentes zonas, siendo también un clúster de alta disponibilidad. En la siguiente figura se muestra este concepto.

Figura 13. Clúster HA con nodos en diferentes zonas.



Fuente: Elaboración propia.

- **Características para estudiar de Kubernetes**

Kubernetes ofrece caracterizas que aportan las ventajas que buscamos para el proyecto y que cubren las motivaciones del proyecto. Así se identifican para poder estudiarlas en apartados posteriores, y ver su funcionamiento:

Despliegue automático: permite configurar un despliegue encargado de ejecutar los contenedores necesarios para un servicio.

Balanceo de carga: al configurar servicios "LoadBalancer" se distribuye la carga entre los end-point del despliegue optimizando el sistema al repartir las peticiones a un servicio.

Auto reparación: los controladores del clúster controlan que los contenedores que se han desplegado funcionan correctamente. En el caso de que no sea así se encargan de reemplazar

los contenedores que generan el problema. Esta auto reparación puede llegar a nivel de nodo, con algunos proveedores de Kubernetes en nube pública como Google Cloud Platform, donde si detecta que queremos tener tres nodos en un clúster y uno falla se genera un nuevo nodo.

Escalado: permite que una aplicación ya desplegada pueda aumentar o reducir el número de réplicas que tiene en ejecución.

Actualizar una aplicación con control de versiones: permite actualizar una aplicación sin parar el servicio, creando primero un nuevo contenedor actualizado de la aplicación y posteriormente cuando ya está funcionando, elimina el antiguo. Además, el control de versiones permite deshacer una actualización a una versión anterior.

Gestión de recursos: permite configuración de gestión de los recursos que se usa en el clúster para controlar y proteger el correcto funcionamiento.

- **CLI de Kubernetes: Kubectl**

Kubectl es una herramienta de línea de comandos que usando el API de Kubernetes ejecuta los comandos para administrar el clúster. Esto permite al usuario, ya sea administrador o desarrollador, comunicarse con las aplicaciones del clúster y gestionar tanto la tecnología de Kubernetes como las aplicaciones desplegadas a través de un terminal, en la máquina local, que sea el encargado de conectarse con el máster.

1.2.5. Tecnologías alternativas

Además de las tecnologías estudiadas en este capítulo, y suficientes para el desarrollo del objetivo, hay otras formas de empaquetar aplicaciones en contenedores que pueden orquestarse con Kubernetes. Y de igual forma hay otras de orquestar contenedores que no son Kubernetes.

Alternativas de contenedores

Kubernetes nos permite gracias a una interface abstraer su funcionamiento del tipo de contenedores utilizados. Esta interfaz se llama “Container Runtime Interface” (CRI). Así tenemos:

- **RKT:** que se conoce como tecnología de “rocketnetes”, es tecnología estandarizada.
- **CRI-O:** son contenedores ligeros de RedHat, para aplicaciones ligeras.
- **Hypernetes:** basado en hipervisores.
- **Clear Containers:** contenedores ligeros y seguros de Intel.
- **LXD:** complemento hipervisor para los contenedores LXC, es decir, realmente es un administrador de contenedores LXC. Los LXD están enfocados para simular VM con contenedores LXC.

Alternativas de orquestación de contenedores

Docker Swarm

Es la aplicación nativa de Docker que permite la creación de un clúster, que en Docker Swarm se llama enjambre, añadiendo los nodos que queremos que formen parte de este.

El enjambre es la conversión de un conjunto de hosts, los que, deseados en el enjambre, en un host virtual para ejecutar los contenedores de Docker.

Apache Mesos

Desarrollado por la Universidad de Berkeley, para gestionar grandes cargas de cálculo y proporciona aplicaciones como Hadoop y Spask (para Big Data).

Conjure-up

No es realmente una alternativa de Kubernetes, es una alternativa a la ejecución de Kubernetes con contenedores Docker. Conjure-up usa contenedores LXD y permite el despliegue de Kubernetes guiado paso a paso.

1.3 Marco Contextual

1.3.1 Entorno Organizacional

Megasis, creada en agosto del 2014, se dedica a la instalación, desarrollo y administración de sistemas, páginas web para los mercados de: Agricultura, Construcción, Financiera Industria, minería, salud, seguridad pública y privada, transporte.. Actualmente, Megasis no cuenta con la virtualización y orquestación en contenedores en la nube, ambos conceptos son fundamentales para el desarrollo y la operación de aplicaciones modernas, permitiendo a la empresa desplegar y gestionar aplicaciones de manera más rápida, confiable y a escala. La adopción de entrega de software basada en contenedores en particular Docker (Contenedor) y Kubernetes (Orquestador), busca transformar este proceso permitiendo una mayor rapidez y escalabilidad.

1.3.2 Necesidades y Objetivos Específicos de la Organización

- Virtualizar los entornos de desarrollo y producción. Lo que permitiría actualizar las aplicaciones sin detener el servicio ofrecido a los clientes.
- Mejorar los tiempos de despliegue de las aplicaciones de la empresa.
- Prestar servicios en la nube, con las herramientas y tecnologías más ligeras y eficaces posibles.

1.3.3 Impacto y Beneficios Esperados

- Portabilidad: Las aplicaciones se montan en los contenedores una sola vez y, si su funcionamiento es el correcto, seguirán funcionando del mismo modo independientemente del entorno en el que se encuentren.
- Fiabilidad: Los contenedores, al ser entornos aislados, compactos y compatibles, son fácilmente migrables.
- Unificación: Los contenedores son mucho más ligeros que una máquina virtual completa, por tanto, en un mismo nodo se pueden llegar a ejecutar un número considerable de aplicaciones, cada una en su propio contenedor.

CAPITULO II

DIAGNOSTICO

2.1 Introducción

La tecnología de contenedores permite empaquetar aplicaciones de una manera sencilla y portable, facilitando la ejecución allá donde queremos que esté funcionando, ya sea una nube privada, nube pública, en el ordenador personal o en todas a la vez. Esta capacidad permite a los desarrolladores ser más autónomos en sus entornos de desarrollo, aumentando así la eficiencia de los equipos de desarrollo y de operaciones IT.

Un 70% de los encuestados consideran esta tecnología útil para la empresa. Al mismo tiempo, un 53% de ellos ya hacen uso de contenedores o tienen intención de hacerlo a corto plazo. Un 55% hacen o van a hacer uso de contenedores en despliegues ubicados en la nube, este porcentaje se prevé que aumente significativamente mediante el uso de las tecnologías propuestas, principalmente impulsado por la nube privada. De entre los despliegan en instalaciones propias cabe destacar que casi un 30% permiten la integración con soluciones de almacenamiento de otra nube. Por otro lado, la gran mayoría (80%) hace ya un back-up de sus procesos de contenedores, Además, la mayoría indica que menos de un 20% de sus aplicaciones son usadas en entorno de contenedores.

El propósito de este capítulo es realizar un diagnóstico exhaustivo de los métodos actuales del uso de contenedores en Megasis, identificando las limitaciones y evaluando cómo la integración de Docker y Kubernetes podrían abordar estos desafíos. Se analizarán las prácticas actuales del uso de contenedores para establecer una línea base de operaciones y se destacarán áreas específicas donde la contenerización podría introducir mejoras significativas. Este análisis servirá como fundamento para diseñar estrategias efectivas que no solo optimicen la contenerización de las aplicaciones, sino que también alineen estas mejoras con los objetivos estratégicos de la empresa.

2.1.1 Procesamiento y Análisis de Datos

2.1.2 Procesamiento y Análisis de Datos de la Encuesta

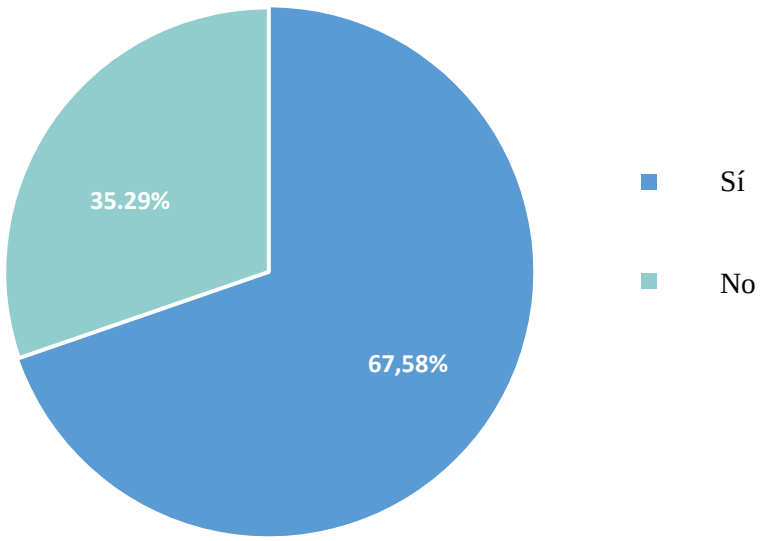
1. ¿Considera que la tecnología de contenedores es útil e interesante para la empresa?

Tabla 1

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Si	11	68,75%
No	5	31,25%
Total	17	100,00%

Fuente: Elaboración propia

Figura 16



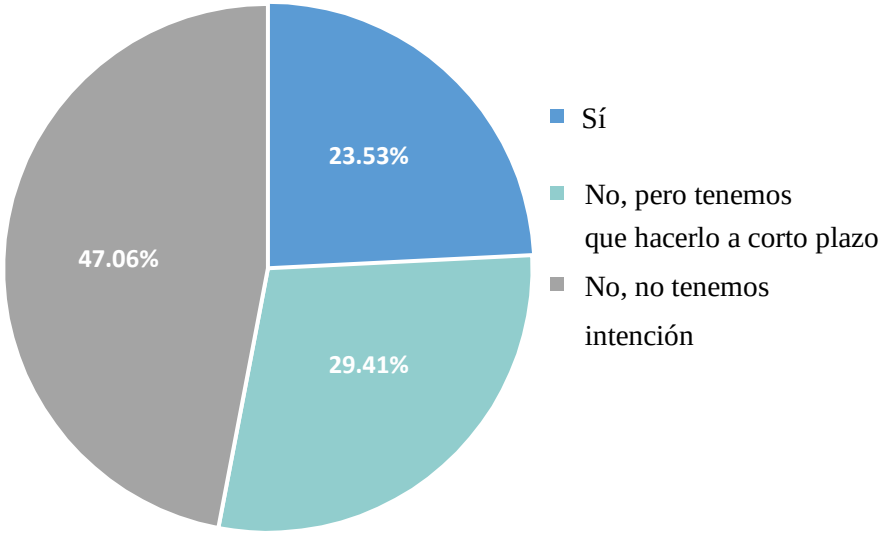
Fuente: Elaboración propia

2. ¿Están haciendo uso o tiene intención de hacer uso de tecnología de contenedores?

Tabla 2

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Si	8	47,06%
No, pero tengo intención que hacerlo a corto plazo.	4	23,53%
No, no tengo intención	5	29,41 %
Total	17	100,00%

Figura 17



Fuente: Elaboración propia

Interpretación: Se puede observar en la figura 1 que prácticamente un 70% de los empleados encuestados considera que la tecnología de contenedores es útil e interesante para su empresa. Además, en la figura 2, se puede apreciar cómo de extendida está la tecnología de contenedores entre los empleados y el uso que se hace de la misma. Aproximadamente, 1 de cada 4 empleados encuestados ya están haciendo uso de contenedores. Si a estos unimos las que tienen intención de hacer uso de contenedores a corto plazo, este porcentaje alcanza el 53%. Por tanto, el porcentaje de los desarrolladores con intención de hacer uso de contenedores en el corto plazo es superior a las que hacen uso en la actualidad, demostrando que es una tecnología sólida y en expansión. La principal causa por la cual el porcentaje restante afirma no tener intención de usarlo, es el desconocimiento acerca de la tecnología.

3. ¿Cuál es el interés o estado actual de la empresa en la adopción de contenedores?

Tabla 3

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Solo desarrollo	7	41,18%
Se usa o va a usar tanto para desarrollo como para producción	6	35,29%
Principalmente desarrollo con algún proyecto producción	4	23,53 %
Total	17	100,00%

Figura 18

Fuente: Elaboración propia

Interpretación: Tomando como base tanto los empleados que hacen uso de contenedores, como los que tienen intención de hacerlo, se analizó si el uso de esta tecnología se centra en el desarrollo de aplicaciones o si se trata de cargas de trabajo en entornos de producción. Podemos apreciar en la figura 3, que, si bien más de una tercera parte hace uso sólo para el desarrollo, siendo la opción más extendida, ya un 35% de los encuestados usa o va a usar contenedores tanto para desarrollo como para producción y un 24% hace uso o tiene intención de hacerlo principalmente para desarrollo, pero con algún proyecto en producción.

4. ¿Dónde preferiría que vayan ubicados estos proyectos?

Tabla 4

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Cloud	10	41,18%
Centro de datos propio	7	35,29%
Total	17	100,00%

Figura 19

Fuente: Elaboración propia

Interpretación: Si hacemos foco en el tipo de arquitectura utilizada para el despliegue de contenedores, vemos que casi un 60 % de los encuestados declaran hacer uso o tener intención de hacerlo en despliegues ubicados en la nube como se puede apreciar en la figura 4. Si analizamos los resultados diferenciando los empleados que hacen uso de contenedores en la actualidad en la nube, de los que tienen intención de hacerlo a corto plazo, este último caso aumenta hasta el 62,5%, lo que da idea de un mayor potencial de crecimiento de los contenedores en la nube frente a los instalados en el centro de datos propio.

5. ¿Qué tecnología de orquestación tiene intención de utilizar la empresa actualmente?

Tabla 5

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Cloud	10	41,18%
Centro de datos propio	7	35,29%
Total	17	100,00%

Figura 20

Fuente: Elaboración propia

Interpretación: Respecto a la tecnología de orquestación, cabe destacar que el uso de Kubernetes (incluyendo sus diferentes distribuciones como OpenShift o Rancher) se posiciona como la opción más habitual, con el 65% de los casos encuestados como se aprecia en la figura 5. La preferencia por los orquestadores en código abierto es aún mayor entre las grandes empresas y las que están en proceso de implementar contenedores.

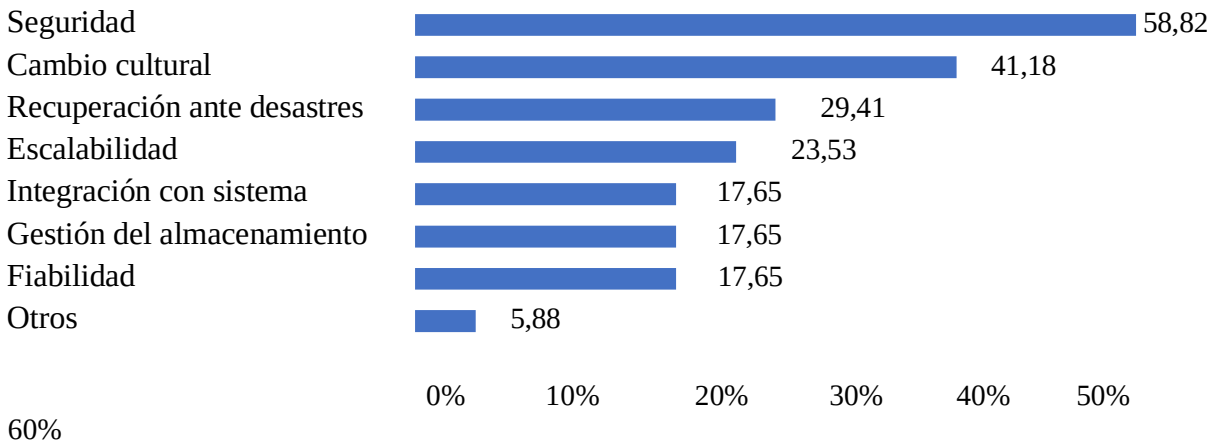
6. ¿De la siguiente lista, identifique los retos del uso de contenedores?

Respuesta múltiple

Tabla 6

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Seguridad	10	58,82%
Cambio cultural	7	41,18%
Recuperación ante desastres	5	29,41%
Escalabilidad	4	23,53%
Integración de sistemas legacy	3	17,65%
Gestión de almacenamiento	3	17,65%
Fiabilidad	3	17,65%
Otros	1	5,88 %

Figura 21



Fuente: Elaboración propia

Interpretación: En cuanto a los retos que se enfrentarían los empleados, destaca principalmente los relacionados con la seguridad, así como el cambio cultural como podemos apreciar en la figura 6. Este cambio cultural tiene que ver principalmente con la nueva manera de trabajar que supone el paso a uso de contenedores.

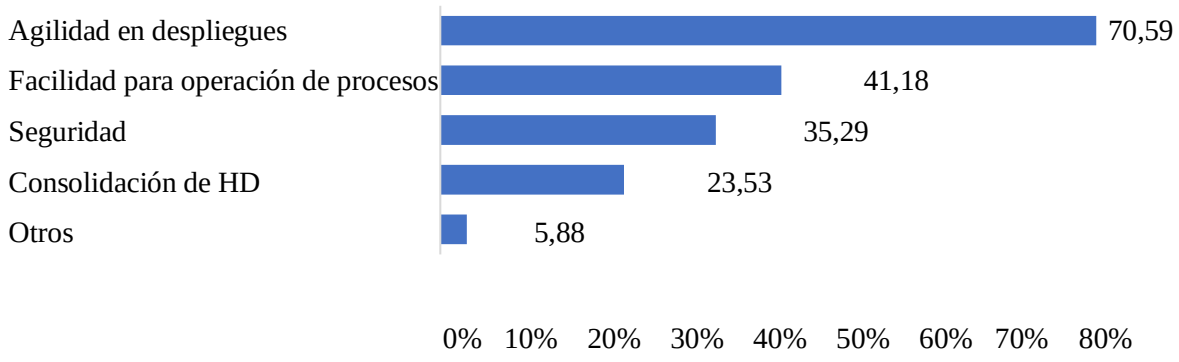
7. ¿Según sus conocimientos adquiridos sabe usted que ventajas aportaría a la empresa la tecnología de contenedores?

Respuesta múltiple.

Tabla 7

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Agilidad en despliegues	12	70,59%
Facilidad para procesos de operación	7	41,18%
Seguridad	6	35,29%
Consolidación de HD	4	23,53%
Otros	1	5,88 %

Figura 22



Fuente: Elaboración propia

Interpretación: Preguntados por las principales ventajas que supone el uso de tecnología de contenedores, los encuestados destacan como prioritaria la agilidad de los despliegues como se puede apreciar en la figura 7. Como vemos inicialmente en la investigación uno de los principales beneficios en el uso de contenedores es el hecho de que las aplicaciones se pueden abstraer del entorno en el que se ejecutan, lo que facilita el despliegue de las aplicaciones con independencia del entorno.

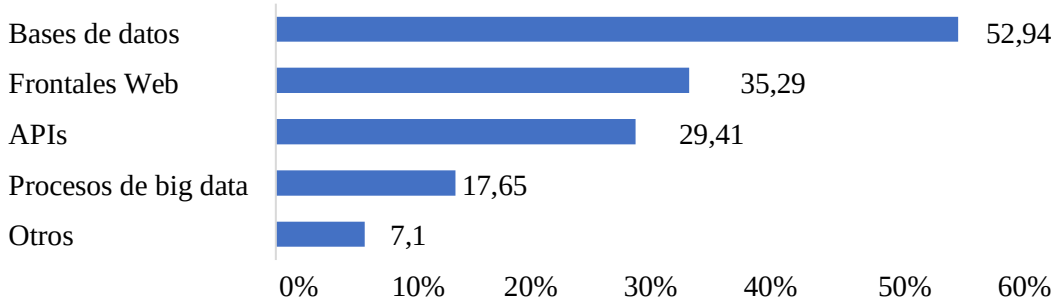
8. ¿Según su estructura qué tipo de cargas ejecutaría la empresa en contenedores?

Respuesta múltiple.

Tabla 8

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Bases de datos	9	52,94%
Frontales Web	6	35,29%
APIs	5	29,41%
Procesos de big data / Inteligencia Artificial	3	17,65%
Otros	1	5,88 %

Figura 23



Fuente: Elaboración propia

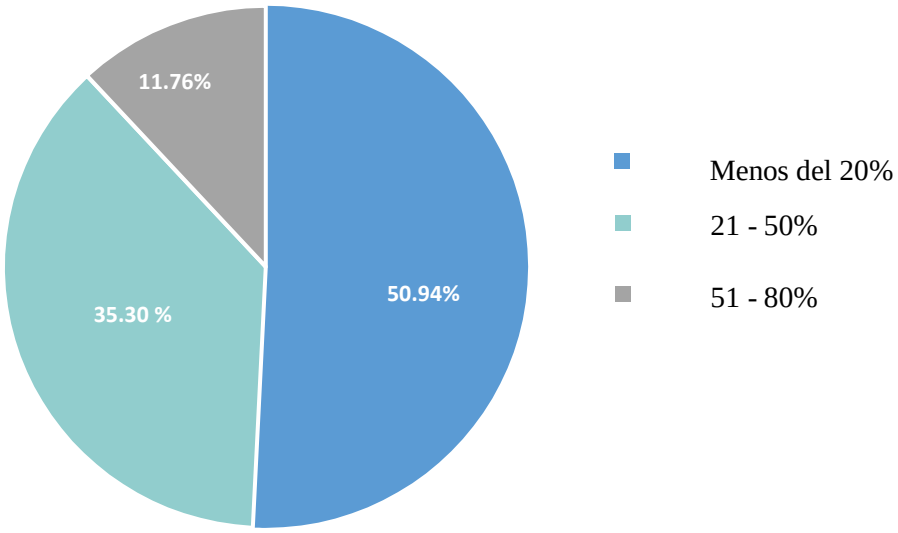
Interpretación: Si analizamos qué cargas de trabajo se ejecutan más frecuentemente sobre contenedores, cabe señalar para el caso de Megasis las bases de datos y los frontales web como las más destacadas por los encuestados con el 54% y el 33% respectivamente, como se aprecia en la figura 8. Este resultado contrasta, sin embargo, con los resultados a nivel global donde la infraestructura IT o a las aplicaciones empresariales tienen un peso mayor. Por otro lado, este aumento del consumo de contenedores para montar cargas como base de datos, nos indica un aumento de la madurez de la tecnología, así como de su percepción por parte de la empresa.

9. ¿Qué porcentaje de sus aplicaciones funciona o funcionaría en contenedores?

Tabla 9

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Menos de 20%	9	52,94%
21 – 50 %	6	35,30%
51 – 80 %	2	11,76 %
Total	17	100,00%

Figura 24



Fuente: Elaboración propia

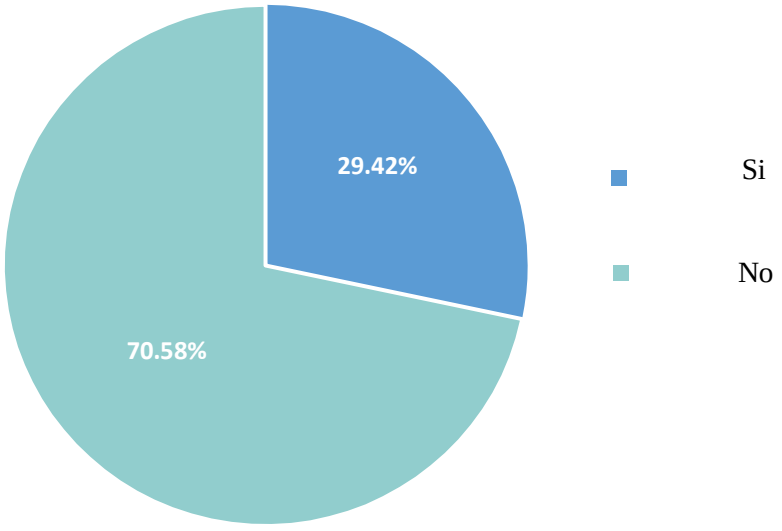
Interpretación: Podemos destacar que, al ser aún una tecnología en expansión en el ámbito empresarial mundial, la mayoría de los encuestados (51%) señalan que menos de un 20% de sus aplicaciones son usadas en entorno de contenedores como se muestra en la figura 9.

10. ¿Integra la solución de almacenamiento on-premise con otra nube?

Tabla 10

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Si	5	29,42%
No	12	70,58%
Total	17	100,00%

Figura 25



Fuente: Elaboración propia

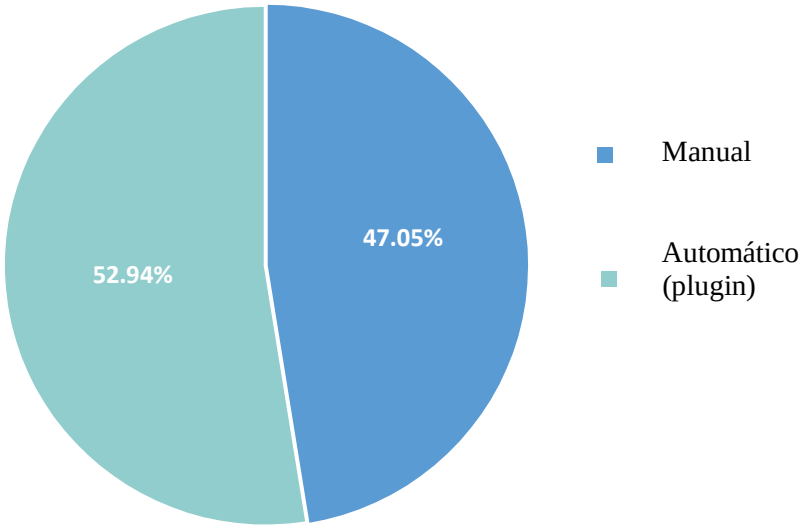
Interpretación: Como se destacaba anteriormente, la mitad de los encuestados efectúa el despliegue de contenedores en la nube. De entre los que despliegan contenedores en instalaciones propias cabe destacar que casi un 30% permiten la integración con soluciones de almacenamiento de otra nube como se aprecia en la figura 9.

11. ¿Cómo integra la solución de almacenamiento con su ecosistema de contenedores on-premise?

Tabla 11

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Manual	8	47,06%
Automático (Plugin)	9	52,94%
Total	17	100,00%

Figura 26



Fuente: Elaboración propia

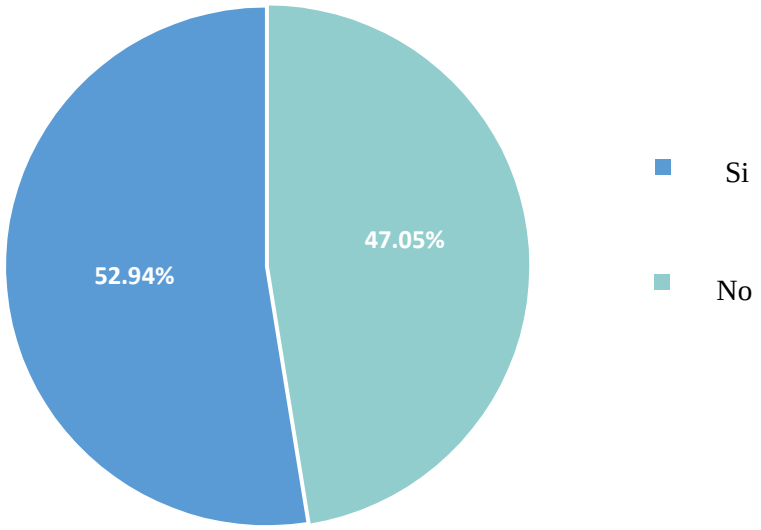
Interpretación: Otro aspecto que denota recorrido de mejora en la madurez tecnológica de la integración es el hecho de que la empresa tiene desplegado contenedores en sus instalaciones, un 48% de los encuestados afirman contar con una integración manual como se aprecia en la figura 10.

12. ¿Monitoriza sus cargas en los contenedores?

Tabla 12

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Si	9	52,94%
No	8	47,05%
Total	17	100,00%

Figura 27



Fuente: Elaboración propia

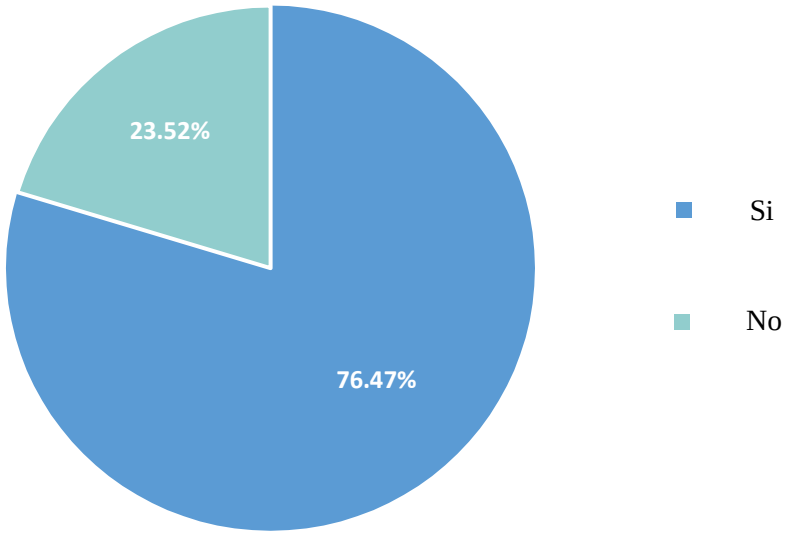
Interpretación: Otro aspecto abordado en las encuestas es la monitorización de las cargas en contenedores y la gestión del ecosistema que realiza la empresa de manera interna. En este sentido cabe destacar que aproximadamente la mitad de los encuestados no monitoriza las cargas de trabajo como se puede apreciar en la figura 12, lo que da idea del importante recorrido que existe en este ámbito, especialmente en entornos productivos.

13. ¿Hace back-up a sus procesos de contenedores?

Tabla 13

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Si	13	76,47%
No	4	23,52%
Total	17	100,00%

Figura 28



Fuente: Elaboración propia

Interpretación: En cuanto a temas como el back-up de procesos, la alta disponibilidad y los retos de seguridad a los que se enfrentan y que han dado el paso hacia la tecnología de contenedores, cabe destacar, que 8 de cada 10 empleados afirman hacer ya back up de sus procesos de contenedores.

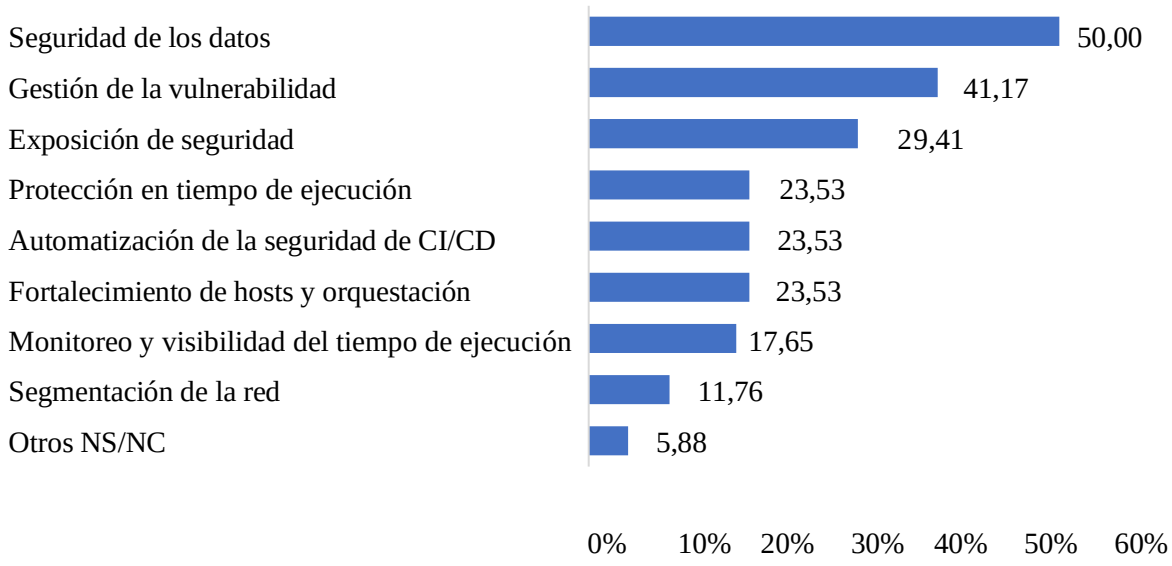
14. ¿Cuáles son sus principales desafíos de seguridad con los contenedores?

Respuesta múltiple.

Tabla 14

Respuestas	Frecuencia Absoluta N°	Frecuencia Relativa %
Seguridad de los datos	9	52,94%
Gestión de la vulnerabilidad	7	41,17%
Exposición de seguridad	6	29,41%
Protección en tiempo de ejecución	4	23,53%
Automatización de la seguridad de CI/CD	4	23,53%
Fortalecimiento de hosts y orquestación	4	23,53 %
Monitoreo y visibilidad del tiempo de ejecución	3	17,65 %
Segmentación de la red	2	11,76%
Otros NS/NC	1	5,88%

Figura 29



Fuente: Elaboración propia

Interpretación: Respecto a los principales desafíos de seguridad en contenedores, los encuestados destacan principalmente la seguridad de los datos y la gestión de la vulnerabilidad.

2.2 Conclusiones y Recomendaciones

2.2.1 Conclusiones

En este trabajo se caracterizó la aplicación de contenedores como herramienta que mejora la eficiencia de la entrega de software teniendo en cuenta el enfoque DevOps.

Lo más importante fue que la mayoría los desarrolladores de Megasis están familiarizados y capacitados para hacer uso en de contenedores y orquestadores, no precisamente de las herramientas propuestas, pero sí de tecnologías similares, Además muestran un alto interés de almacenar sus aplicaciones en la nube. Respecto a la tecnología de orquestación, cabe destacar que el uso de Kubernetes se posiciona como la opción más habitual entre los encuestados.

La combinación de Docker y Kubernetes permite a la empresa aprovechar lo mejor de ambas tecnologías: Docker para la creación y gestión de contenedores y Kubernetes para la orquestación a gran escala. Esta integración asegura que las aplicaciones se ejecuten de manera eficiente, con recursos optimizados y sin tiempos de inactividad, lo cual resulta ser una excelente opción en el enfoque DevOps ya que Docker proporciona la portabilidad y eficiencia necesarias para ejecutar contenedores en cualquier entorno, ya sea local, en la nube o en entornos híbridos.

En esta monografía se han estudiado las principales características que nos ofrecen los contenedores tanto para administrarlos y mantenerlos. Para lo cual se crea un escenario donde teóricamente se cumplen los objetivos a estudiar, y que son consecuencia de los motivos que llevan a desarrollar esta investigación.

Para concluir mencionar que se identifica a Docker y Kubernetes como las herramientas y tecnologías más ligeras y eficaces posibles, de cara a cubrir la demanda actual de servicios de la empresa Megasis.

2.2.2 Recomendaciones

Con este telón de fondo, la escena está preparada para iniciar la transición hacia nuevos métodos de entrega de aplicaciones y, finalmente, hacia soluciones totalmente nativas de la nube. A lo largo del proceso, hay que dar una serie de pasos, más aún con las herramientas que nos falta estudiar como adoptar prácticas de desarrollo ágiles y conseguir lo siguiente: arquitecturas basadas en microservicios. El objetivo final es ofrecer una mayor agilidad proporcionando una nueva clase de aplicaciones modernas de forma rápida, eficaz y a gran escala.

Para términos académicos donde estudiantes desean realizar microservicios con una herramienta que utilice las prácticas DevOps y sea gratuita se recomienda el servidor de automatización Jenkins.

REFERENCIAS BIBLIOGRÁFICAS

Alonso, R. 2017. Software Containerization with Docker, 3-4. Turku University of Applied Sciences. Degree Programme in Information Technology. Bachelor's thesis.

Andersson, J. y Norrman, F. (2020). Orquestación de contenedores: el camino de la migración B. Cessa, «¿Cómo escribir un buen Dockerfile?,»

Bajo, L. (2018). El arquitecto de software y DevOps. Software IEEE, 35 (1), 8–10.

Cloud, G. (n.d.). ¿Qué es DevOps? Investigación y soluciones | Google Cloud. Consultado Mayo, 2024,https://cloud.google.com/devops?hl=es

Docker. Infraestructure Optimization. [En línea] 2018. https://www.docker.com/use-cases/infrastructure-optimization.

DORA. (2018). Acelerar: estado de las estrategias de DevOps para una nueva economía. https://services.google.com/fh/files/misc/state-of-devops-2018.pdf

jPablo, «¿Por qué usar LXC?,» [En línea]. Consultado Mayo https://www.jpablo128.com/por-que-usar-lxc-linux-containers/.

Grigoryeva, S. 2016. Metodologías ágiles en proyectos de software a gran escala, 76. Lapland University of Applied Sciences. Degree Programme in Information Technology. Bachelor's thesis.

Gutta, S., Prasad, S. Y Angara, J. (2016). Ingeniería De Línea De Productos Devops (Dple): hacia Kubernetes [Tesis, Universidad Linnaeus, Departamento de Ciencias de la Computación y Tecnología de Medios (DM)]. Consultado abril 2024. http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-97744

J. M. Fiz, «Por qué todos apuestan por Kubernetes,» [En línea]. Consultado abril 2024: https://www.paradigmadigital.com/techbiz/por-que-todos-apuestan-por Kubernetes/.

Kubernetes. [En línea] Consultado mayo 2024 https://kubernetes.io/.

López-Fernández, D., Díaz, J., García, J., Pérez, J., & González-Prieto, Á. (2021). DevOps Team Structures: Characterization and Implications. IEEE Transactions on Software Engineering, 3716 - 3736. doi:10.1109/TSE.2021.3102982

Linux Containers. LXC. [En línea]. Consultado abril 2024. https://linuxcontainers.org/

Mercedes Brito, J. C. (2022). Integración y entrega continua (CI/CD) con Jenkins. Trabajo Fin de Grado, Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación. Universidad Oberta de Catalunya. Recuperado de la base de datos de la UOC.

Muñoz-Tenempaguay, C. D. (2023). DevOps en el desarrollo de software: integración y entrega continua. Polo del Conocimiento: Revista científico-profesional, 612.

Namespace manual. [en línea] Consultado marzo 2024.

Orozco, C., Pardo, C., Zúñiga, K., & Certuche, S.-C. (2022). Proceso para fomentar y apoyar la adopción de DevOps en PyMEs de software. Revista Científica, 45(3), 422-4371.

Preparación y evaluación de proyectos, segunda edición – Nassir Sapag Chain, Reinaldo Sapag Chain.

Real Academia Española. (2021). Diccionario de la Lengua Española.

redhat, «¿Qué es un contendor de Linux?,» [En línea]. Consultado abril 2024. https://www.redhat.com/es/topics/containers/whats-a-linux-container.

redhat, «¿Qué es Docker?,» [En línea]. Consultado abril 2024. https://www.redhat.com/es/topics/containers/what-is-docker.

Revista Internacional EPRA de Investigación y Desarrollo (IJRD), 376–382.

Y.-J. Hong, «CRI: the Container Runtime Interface, » [En línea]. Consultado mayo 2024. https://github.com/Kubernetes/Kubernetes/blob/242a97307b34076d5d8f5bbeb154fa4d97c9ef1d/docs/devel/container-runtime-interface.md

Zakharenkov, R. (2019). Devops In E-Commerce Software Development: Demand for Containerization.Consultado marzo 2024. https://www.theseus.fi/bitstream/handle/10024/171099/Zakharenkov_Roman.pdf

ANEXOS

ENCUESTA

1. ¿Considera que la tecnología de contenedores es útil e interesante para la empresa?
2. ¿Están haciendo uso o tiene intención de hacer uso de tecnología de contenedores?
3. ¿Cuál es el interés o estado actual de la empresa en la adopción de contenedores?
4. ¿Dónde preferiría que vayan ubicados estos proyectos?
5. ¿Qué tecnología de orquestación tiene intención de utilizar la empresa actualmente?
6. ¿De la siguiente lista, identifique los retos del uso de contenedores?
7. ¿Según sus conocimientos adquiridos sabe usted que ventajas aportaría a la empresa la tecnología de contenedores?
8. ¿Según su estructura qué tipo de cargas ejecutaría la empresa en contenedores?
9. ¿Qué porcentaje de sus aplicaciones funciona o funcionaría en contenedores?
10. ¿Integra la solución de almacenamiento on-premise con otra nube?
11. ¿Cómo integra la solución de almacenamiento con su ecosistema de contenedores on-primese ?
12. ¿Monitoriza sus cargas en los contenedores?
13. ¿Hace back-up a sus procesos de contenedores?
14. ¿Cuáles son sus principales desafíos de seguridad con los contenedores?