UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Migración de una aplicación on-premise a la nube con Amazon Web Services

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Montilla Tomás, Guillermo

Tutor/a: Sánchez Anguix, Víctor

Cotutor/a: Alberola Oltra, Juan Miguel

Cotutor/a externo: GIMENO BONO, ANTONIO

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo pretende documentar el proceso de migración de un proyecto empresarial *on-premise* (en local) a la nube, concretamente a Amazon Web Services y de esta forma poder utilizar el procedimiento para futuras migraciones.

Durante todo este proceso se identificarán las partes que conforman un proyecto real para saber qué servicios son necesarios, tales como bases de datos, servicios de aplicación, ya sea en clústeres de kubernetes, docker o en la propia máquina virtual, nombres de dominio, balanceadores, etc., además de cómo deben ser configurados en AWS de forma que se optimicen todos los recursos utilizados.

También se abordará qué ventajas tiene migrar un proyecto a la nube en vez de tenerlo localmente en un CPD y qué beneficios tiene AWS respecto a otros proveedores como pueden ser Azure o Google Cloud.

Palabras clave: AWS, cloud, *on-premise*, migración, aplicación web

Abstract

The current work pretends to document the migration process of an *on-premise* business project to the cloud, specifically to Amazon Web Services and use this document for future migrations.

Throughout this process, the parts that make up a real project will be identified to find out what services are necessary such as databases, applications services either in kubernetes clusters, docker or even in the virtual machine itself, domain names, balancers, etc., furthermore how they should be configured in AWS so all the resources used are optimized.

In addition, we will check the advantages of migrating a project to the cloud instead of having it locally in our CPD and which benefits have AWS compared to other providers such as Azure or Google Cloud.

Key words: AWS, cloud, *on-premise*, migration, web application

Resum

Aquest treball pretén documentar el procés de migració d'un projecte empresarial *on-premise* (en local) al núvol, concretament a Amazon Web Services i d'aquesta manera poder utilitzar el procediment per a futures migracions.

Durant tot aquest procés s'identificaran les parts que conformen un projecte real per a saber quins serveis són necessaris, com ara bases de dades, serveis d'aplicació allotjats en clústers de kubernetes, en docker o en la mateixa màquina virtual, noms de domini, balancejadors, etc., a més a més de com han de ser configurats en AWS de manera que tots els recursos estiguen optimitzats.

També s'abordarà quins avantatges té migrar un projecte al núvol en comptes de tindre'l localment al nostre CPD i quins beneficis té AWS respecte a altres proveïdors com poden ser Azure o Google Cloud.

Paraules clau: AWS, cloud, *on-premise*, migració, aplicació web

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Hoy en día, cualquier empresa que quiera prosperar, sea grande o pequeña, busca optimizar sus recursos de forma que sean eficientes y eficaces y de esta forma mejorar el rendimiento de su negocio, aumentando la producción, reduciendo costes e incluso disminuyendo el impacto medioambiental, en definitiva, quiere mejorar su desempeño y ser más competitiva.

Como nos explica Palos et al. (2019), el Instituto Nacional de Estándares y Tecnologías de los Estados Unidos de América define la computación en la nube como un modelo que permite acceder a un conjunto de servicios como servidores, redes o almacenamiento de manera conveniente cuando se necesita. Todos estos servicios son accesibles a través de una conexión a Internet, utilizados bajo demanda y permiten una gran escalabilidad de los recursos utilizados.

Por increíble que parezca, la principal fuente de ingresos de Amazon no es su mercado de compraventa, sino su servicio Amazon Web Services [1] (AWS), un proveedor de servicios en la nube que mezcla tanto infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS) y que ofrecen sobre todo seguridad, bajo costo y accesibilidad. Con muchos años de experiencia, AWS es hoy en día el proveedor de servicios en la nube más utilizado, tendencia que se cumple año tras año y que no deja de aumentar su uso.

En palabras de Guerola (2022), aunque puedan existir reticencias para adoptar el modelo en la nube como pueden ser la incertidumbre de la seguridad al no estar familiarizado, la conectividad y el acceso que depende de tener una alta velocidad de red, el valor económico o los cambios perpetrados en la propia organización que cambian su funcionamiento, las ventajas que ofrece la transformación digital como la reducción de coste a la hora crear nuevos entornos, el acceso casi inmediato al hardware sin inversiones de capital, la adaptabilidad de la carga de trabajo, la escalabilidad o la variedad de servicios ofrecidos superan con creces los inconvenientes.

Por estas razones es que las empresas desechan su infraestructura, proyectos y recursos *on-premise* a favor de servicios en la nube, sobre todo en su entorno de

[1] Portal AWS: https://aws.amazon.com/es/what-is-aws/

producción y es por eso que el perfil de administrador de sistemas, un puesto de trabajo más tradicional y centrado en los sistemas informáticos se está quedando obsoleto, dando paso al perfil de *cloud admin* o administrador de servicios en la nube.

Con todo esto en mente, este trabajo se enfocará en mostrar cómo migrar un proyecto real *on-premise* a AWS. En este primer punto se expondrán las causas y motivaciones por las que se ha elegido este trabajo de final de grado, así como también los objetivos y propósitos que se pretenden conseguir al realizarlo. También podremos ver en el tercer subapartado cómo estará estructurado y qué capítulos formarán este proyecto.

Como elemento distintivo, contaremos con datos reales de una empresa llamada ODEC cuyo negocio es la captura de datos, el tratamiento de información, la presentación de resultados, el desarrollo software y la externalización de servicios y que nos ayudarán a elaborar y ejemplificar esta guía y hacerla más veraz. Este negocio cuenta actualmente con una infraestructura mixta, tanto *on-premise* como en la nube, pero con el pasar de los años, la directiva se decanta cada vez más hacia un entorno *cloud* por lo que cuenta con los servicios que ofrece AWS, sobre todo por su fiabilidad y seguridad, que al final son los aspectos que más convencen a los clientes.

1.1 Motivación

El campo de trabajo de un informático puede abarcar multitud de áreas, la propia Escuela Técnica Superior de Ingeniería Informática (ETSINF) divide el grado en cinco ramas de especialización: computación, tecnologías de la información, sistemas de la información, ingeniería de computadores e ingeniería del software. Cada una de ellas nos encamina hacia el tipo de empleo que queremos o que nos gustaría ejercer.

Con lo expuesto anteriormente podemos decir que existen todo tipo de trabajos y áreas de la tecnología, como pueden ser, de programación, de ciberseguridad y auditorías, de *big data* y *data science*, planificación de recursos empresariales (*ERP*) y aplicaciones, de gerencia de las tecnologías de la información y las comunicaciones (TIC), de infraestructuras y sistemas, incluso de ventas de tecnologías de la información. Todas ellas son igual de importantes en la actualidad, pero en nuestra sociedad, incluso entre nuestros compañeros, se tiene la visión de que el perfil del informático es el que se pasa todo el día delante de una pantalla desarrollando código.

Por lo tanto, lo que pretende este proyecto, por una parte, es cambiar el enfoque empresarial, centrándose en identificar las deficiencias de quedarse estancado en viejas tecnologías de virtualización, malas optimizaciones de recursos, costes añadidos de personal y elementos externos al proyecto y en cómo solucionarlo ofreciendo a la empresa la posibilidad de migrar sus proyectos *on-premise* a un proveedor de servicios en la nube como es AWS mostrando sus ventajas, como la optimización de recursos, la reducción de costes, la fiabilidad y la escalabilidad además de dejarlo documentado para futuros proyectos que directamente se podrían crear directamente en AWS. Por otra parte, se pretende reivindicar el papel del informático en las empresas y demostrar que, en este mundo en constante evolución, su trabajo no se reduce a sentarse delante de un ordenador, sino que identificar, analizar y mejorar las necesidades de una organización y ayudarla a ir un paso más allá haciendo que pierda el miedo al cambio.

1.2 Objetivos

Este trabajo tiene como objetivo principal realizar una migración de un proyecto desde una infraestructura *on-premise* a un entorno en la nube, concretamente a AWS, además esto conlleva una serie de subobjetivos que se describen a continuación:

1. Exponer que la transformación digital hacia los servicios en la nube es una realidad cada vez más acentuada
2. Mostrar las ventajas que supone tener el proyecto en la nube con respecto a tenerlo *on-premise*.
3. Documentar el proceso de migración para futuras migraciones.

1.3 Estructura

El presente trabajo está formado por 7 puntos, y en este apartado se describirán brevemente cada uno de ellos.

1. **Introducción, motivación y objetivos:** En este primer punto se expone una pequeña introducción a la temática, los motivos que han llevado a la

realización de este trabajo, los objetivos que se pretenden conseguir y su estructura.

2. **Servicios en la nube:** En el segundo punto veremos qué problemas presenta tener un proyecto *on-premise* y por qué elegir AWS para migrarlo a la nube.
3. **Análisis del problema:** En este punto se evaluará el proyecto *on-premise* para identificar que lo conforma y poder elegir qué servicios de AWS son necesarios para su migración.
4. **Diseño de la solución:** En el cuarto capítulo se va a explicar que herramientas se van a utilizar para implantar la solución y cuál será el proceso.
5. **Implementación:** En este punto se describirán los pasos de la migración de la aplicación.
6. **Evaluación:** En el sexto capítulo se expondrán los resultados obtenidos tras la migración.
7. **Conclusiones:** Por último, se describirán las conclusiones que se extraigan de los resultados y valoraremos la migración con respecto a su eficiencia en la empresa.

1.4 Uso de la bibliografía

La mayor parte de la documentación consultada para la realización de este trabajo se encuentra en la propia web de AWS, tanto en inglés como en castellano, todos estos documentos están descritos en la bibliografía al final de este trabajo junto con otras fuentes relevantes.

Toda la información relacionada con el proyecto a migrar, junto con el desempeño en AWS se ha obtenido por experiencia propia al ser una herramienta de uso diario en el trabajo como administrador de sistemas y servicios en la nube en la empresa ODEC [2], que se dedica a la captura de datos, tratamiento de la información, presentación de resultados, desarrollo software y la externalización de servicios.

Puntualmente, se ha utilizado documentación proporcionada por Azure o Google Cloud para realizar la comparativa con AWS, documentos de la guía de Seguridad de las TIC del Centro Criptológico Nacional y documentación académica.

1.5 Objetivos de desarrollo sostenible

Los objetivos de desarrollo sostenible son 17 puntos que podemos visualizar en la **Figura 1.1** y que han sido planteados por la ONU para cumplir con la agenda 2030. Son unos principios que persiguen la igualdad entre personas, la protección del planeta y asegurar la prosperidad para no dejar a nadie atrás, podemos consultar estos puntos en la propia página del ministerio de derechos sociales y agenda 2030[3].

Este trabajo intenta cumplir en mayor o menor parte con todos los puntos, pero principalmente se centra en los siguientes objetivos:

- **Objetivo 8: Trabajo decente y crecimiento:** Un punto que recalca la importancia de formar a las nuevas generaciones en tecnologías emergentes para así disminuir el paro juvenil, aumentar la productividad, mejorar las condiciones laborales y fomentar el crecimiento económico sostenible.

[2] Portal de ODEC: https://www.odec.es/es/
[3] Portal del Ministerio de Derechos sociales y agenda 2030: https://www.mdsocialesa2030.gob.es/agenda2030/index.htm

- **Objetivo 9: Industria, innovación e infraestructura:** Este trabajo por su naturaleza intenta que las aplicaciones web se diversifiquen e innoven, dejando de estar alojadas localmente en los CPD, para así fomentar el crecimiento de la pequeña y mediana empresa, su producción y consumo eficiente y respetuoso.

- **Objetivo 10: Reducción de las desigualdades:** Una de las principales ventajas de la migración a la nube es que ya sea una empresa o un único usuario ya no se necesita de una infraestructura propia para alojar los servicios o aplicaciones, con esto se fomenta que cualquiera tenga acceso a las mismas oportunidades desde cualquier sitio.

- **Objetivo 12: Producción y consumo responsables:** La empresa o el usuario, al deshacerse de parte de su infraestructura, contribuye a la gestión y uso eficiente de sus recursos, reduciendo el consumo de energía y produciendo menos desechos y productos químicos.

Figura 1.1: Los 17 ODS que conforman la agenda 2030 – Fuente: https://www.mdsocialesa2030.gob.es/agenda2030/index.htm

1.6 Agradecimientos

Para finalizar con este capítulo me gustaría dar las gracias a varias personas que han hecho posible este trabajo, la excelente formación recibida en el Grado de Ingeniería Informática y el objetivo final de trabajar como responsable del departamento de sistemas y servicios en la nube en la empresa ODEC.

En primer lugar, quisiera agradecer a mis padres por ofrecerme la posibilidad de estudiar una carrera tan apasionante e innovadora como es la informática, y su apoyo y paciencia incondicional aunque haya habido traspiés.

En segundo lugar me gustaría nombrar como surgió la idea de este trabajo de final de grado, que fue durante un curso de formación permanente llamado "*Cloud computing* para el despliegue de aplicaciones" que impartía mi tutor Juan Miguel Alberola Oltra y que me permitió realizar la empresa ODEC.

Finalmente, sólo me quedaría agradecer a mis dos tutores de la UPV Juan Miguel Alberola Oltra y Víctor Sánchez Anguix y a mi tutor de empresa Antoni Gimeno Bono por prestarse a realizar este trabajo.

2. Servicios en la nube

En este capítulo vamos a centrarnos en los problemas que comporta tener un proyecto en la propia infraestructura del negocio, qué ventajas hay al migrarlo a la nube y cómo elegir el mejor proveedor para ello.

2.1 Proyecto *on-premise* y en la nube

Un proyecto *on-premise* según el Centro Criptológico Nacional (s.f.) es aquel que se implanta en la infraestructura tecnológica de la propia organización cliente, en modo local y, por tanto, las medidas de seguridad y toda la responsabilidad sobre el mismo recaen sobre la propia organización mientras que un proyecto en la nube es aquel en el que la mayor parte de sus servicios o su totalidad están conectados a servicios remotos a través de internet, por lo que la seguridad y la responsabilidad de mismo se reparten entre el proveedor que suministra los servicios y la organización cliente que contrata.

Normalmente, un proyecto *on-premise* suele contar con 3 servicios principales como podemos ver en la **Figura 2.1** más adelante:

- Servidor o servidores de aplicación
- Bases de datos
- Balanceadores de carga

Adicionalmente, los proyectos cuentan con otros servicios que conllevan diferentes elementos como un dominio adquirido junto con un servicio de resolución de nombres (DNS), una red de área local virtual (VLAN) para aislarlo de otros proyectos lo que comporta el uso de cortafuegos de red y switches de capa 2 y 3 cuya principal diferencia es el enrutamiento y posibilidad de comprender el etiquetado del tráfico de la VLAN, la virtualización mediante algún software como VMWare, la necesidad de algún tipo de software de copias de seguridad y por supuesto la parte física que es la adquisición y mantenimiento de servidores *host* y cabinas de disco. Podemos resumir que un proyecto *on-premise* genera costes relacionados con el hardware, el software, el personal que lo gestiona, la infraestructura de red donde se crea, su mantenimiento y soporte, las copias de seguridad asociadas y el respaldo como un plan de recuperación ante desastres o la energía que consume y refrigeración que comporta.

Todas estas variables se deben analizar detenidamente y compararlas con las que genera un proyecto en la nube para saber elegir qué estrategia seguir, todos estos factores quedan bien reflejados por Calderón i Mora (2020) en su comparativa entre un modelo *on-premise* y uno en la nube.

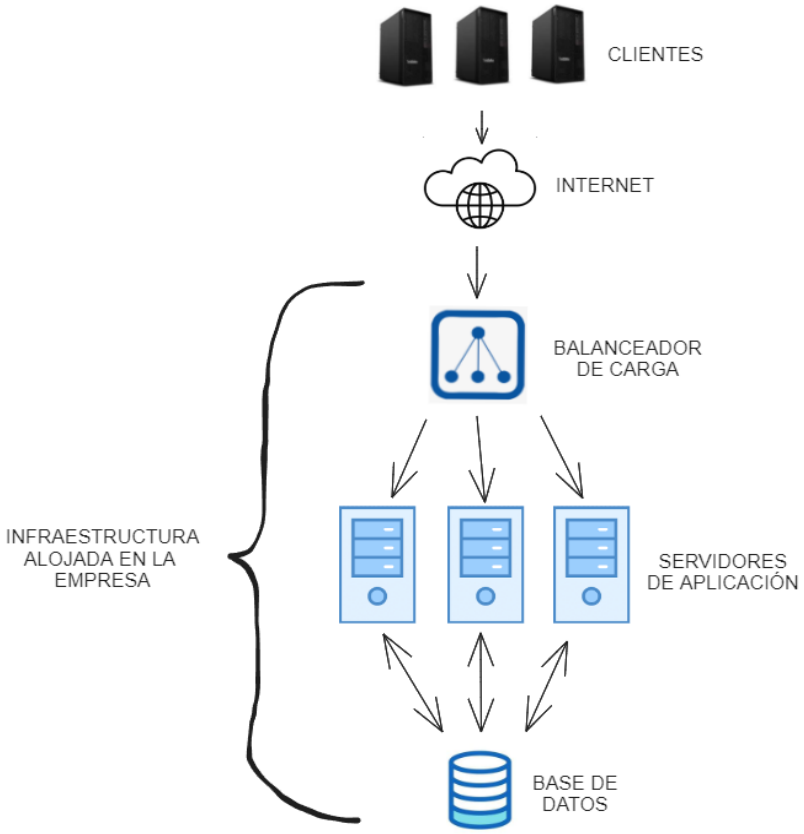


Figura 2.1: Esquema proyecto *on-premise* – Fuente propia

El *cloud computing* o computación en la nube según AWS (s.f.) es la distribución de recursos de TI bajo demanda a través de Internet mediante un esquema de pago por uso. Es decir, que el negocio no tiene que ocuparse de poseer ni mantener una infraestructura de servidores y almacenamiento, ya que puede obtenerlo a través de un proveedor como lo es AWS.

En nuestro caso, nos centraremos en el modelo IaaS que es la infraestructura como servicio, que, como podemos ver en la **Figura 2.2** es el proveedor el que nos ofrece la capa de virtualización, servidores, almacenamiento y red, lo que nos ofrece una gran flexibilidad y control de recursos de las tecnologías de la información, aunque cabe mencionar que AWS también ofrece PaaS o plataforma como servicio en la que el proveedor gestiona la administración de la infraestructura subyacente (hardware y

sistemas operativos) y SaaS o software como servicio en la que el cliente solo tiene que preocuparse de utilizar un software en particular.

Figura 2.2: Diferencias entre IaaS, PaaS y SaaS – Fuente: https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas

Pero ¿por qué utilizar la computación en la nube? Como nos cuenta Apostu et al. (2013), podemos enumerar varios puntos a favor y aunque todos son importantes, tenemos que saber cuáles son los fuertes para convencer al negocio de realizar la migración de un proyecto.

1. **Costos reducidos**: Es el punto más simple de ver, ya que al abstraer la capa física de la infraestructura, la empresa puede ahorrar en costes tanto de adquisición como de mantenimiento.
2. **Mayor seguridad y protección de datos**: Los proveedores de servicios en la nube, y en este caso AWS, implementan medidas de seguridad avanzadas y protección contra la pérdida de datos, lo que mejora mucho la seguridad en comparación a las soluciones locales que tienen que buscar y gestionar ellos mismos estas medidas.

3. **Alta disponibilidad y redundancia**: AWS ofrece múltiples ubicaciones para sus servicios en la nube, por lo que mejora la disponibilidad y la tolerancia a fallos en comparación a las configuraciones locales.
4. **Escalabilidad y flexibilidad**: La computación en la nube permite escalar recursos hacia arriba o hacia abajo según las necesidades de la aplicación, lo que ahorra al negocio tiempo de implementación y tiempos de inactividad.
5. **Acceso global y movilidad**: La computación en la nube, por su propia naturaleza, permite acceder a sus servicios desde cualquier lugar con acceso a internet, esto facilita el trabajo remoto que es tan demandado hoy en día.
6. **Mantenimiento y actualizaciones**: Es el propio proveedor de servicios el que se encarga del mantenimiento y de las actualizaciones tanto hardware como software de la infraestructura, garantizando así el rendimiento óptimo.
7. **Tiempo de implementación rápido**: Configurar y desplegar nuevos servicios o aplicaciones suele ser más rápido que hacerlo localmente.
8. **Enfoque del negocio**: La empresa, al delegar la administración y mantenimiento de la infraestructura a terceros, puede centrarse más en sus actividades principales.

En resumen, la computación en la nube ofrece ventajas significativas en términos de costes, escalabilidad y seguridad, lo que la convierte en una opción muy recomendable para que la empresa avance tecnológicamente en un mundo en constante cambio.

2.2 AWS

Actualmente, tal y como podemos ver en la **Figura 2.3** proporcionada por Freixas (2022), los proveedores de servicios en la nube más importantes son AWS, Microsoft Azure y Google Cloud y como indica el propio trabajo, en este caso vamos a utilizar AWS así que vamos a exponer algunos puntos del porqué va a ser así. Antes cabe mencionar que cada plataforma tiene sus fortalezas y debilidades y aunque muchas veces la elección viene dada por factores como la familiaridad con la plataforma, el tipo de aplicación o las necesidades de la empresa, las siguientes razones destacan en la elección de AWS.

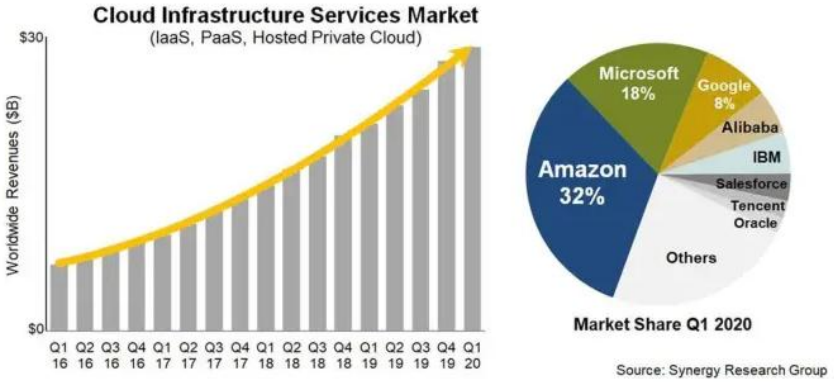


Figura 2.3: Ingresos por infraestructura en la nube

1. **Amplia gama de servicios de AWS**: Uno de los puntos fuertes de AWS es la gran cantidad de servicios que puede ofrecer, desde infraestructura, bases de datos y análisis hasta servicios de inteligencia artificial, resolución de dominios y servicio de correo electrónico para marketing, entre muchos otros ejemplos.
2. **Trayectoria**: AWS fue uno de los primeros proveedores de servicios en la nube y ha estado funcionando desde 2006, por lo que cuenta con una gran experiencia e infraestructura repartida por todo el mundo.
3. **Escalabilidad y rendimiento**: AWS también es conocido por optimizar de manera muy eficiente la escalabilidad de sus servicios para adaptarse a la carga de trabajo demandada por el cliente, lo que proporciona mucha flexibilidad.

Google Cloud y Azure por supuesto no se quedan atrás y ofrecen otras ventajas que hace falta mencionar cómo podría ser la integración de ciertos servicios en sus propios productos o el análisis de datos y como hemos dicho antes, la decisión final dependerá de diversos factores, por lo que muchas empresas optan por una estrategia híbrida para aprovechar las fortalezas de cada proveedor y cubrir sus debilidades.

Se puede ver en la **Figura 2.4** proporcionada por Gupta et al. (2021) la tasa de crecimiento de las tres grandes plataformas de servicios en la nube entre los años 2015 y 2020, y aunque los tres proveedores tienen en común una gran flexibilidad, escalabilidad, soporte y seguridad, el más completo para empresas grandes sigue siendo AWS por su gran variedad de servicios y trayectoria.

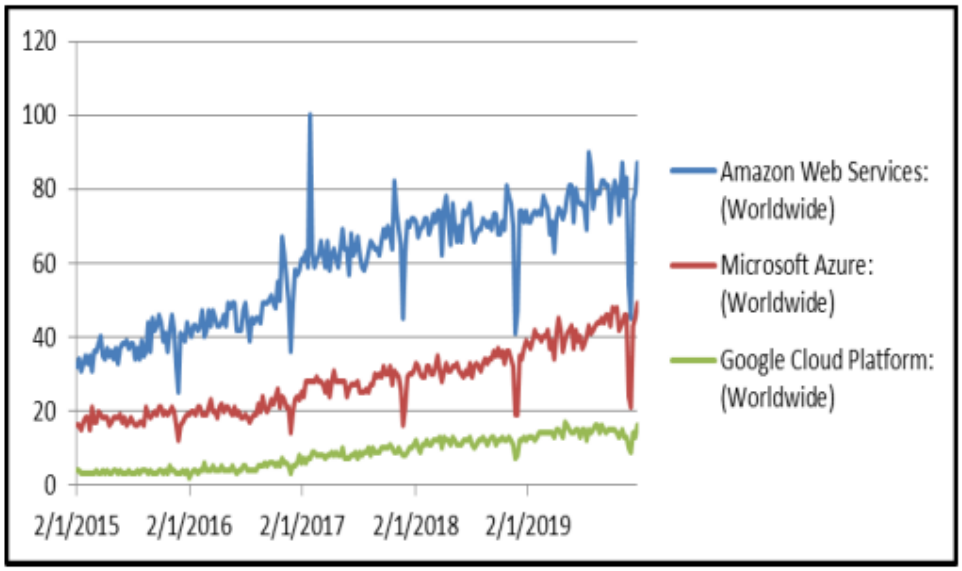


Figura 2.4: Cuota de mercado de AWS, Azure y Google Cloud

ODEC posee actualmente una cuenta de AWS con la mayoría de sus servicios habilitados y aunque Google Cloud o Azure también pueden ser opciones viables e igualmente útiles, incluso no descartables en un futuro de ser usados tras una reevaluación con la directiva si se diera el caso, este proyecto usará los servicios en la nube de Amazon para realizar la migración por estar ya familiarizados con esta plataforma. Entonces, AWS es un proveedor de servicios en la nube, actualmente líder del mercado que ofrece más de 200 servicios, con centros de datos repartidos por todo el mundo para mejorar la latencia según la zona de acceso y adoptado por todo tipo de tamaños de organizaciones debido a su flexibilidad, escalabilidad, seguridad y optimización de recursos. Para aquellos que no disponen de una cuenta empresarial y quieran usar los servicios de Amazon, AWS ofrece una prueba gratuita de 18 servicios durante 12 meses, entre los que están computación, almacenamiento, bases de datos y herramientas para desarrolladores y que podrían utilizarse por quienes quieran hacer pruebas.

2.3 ODEC

Hoy en día, ODEC[4] es una empresa dedicada a distintos servicios, captura de datos, tratamiento de la información, presentación de resultados, desarrollo software y *outsourcing* de servicios, actualmente consta de 4 oficinas repartidas por toda España y como le gusta definirse, ODEC es tu *data partner.* Es una empresa con más de 50 años de experiencia que nació en Gandia y que se dedicaba a la grabación de datos para países europeos, pero que poco a poco fue expandiendo sus objetivos y abarcando un nicho de mercado más amplio, y entre sus logros más importantes figuran acciones como la realización de informes para el Estudio General de Medios o EGM, realización de servicios electorales a nivel tanto comunitario como estatal, llevar a cabo censos para el Instituto Nacional de Estadística y otros países como Portugal, siendo pioneros en el sistema de voto electrónico, ofrecer servicios de marketing para el sector de la automoción, captura multicanal de datos CATI (captura telefónica), CAWI (captura web) y PAPI (captura en papel) y que además desarrolla su propio software para la captura y recogida de datos así como su publicación online. Como valor añadido cabe destacar que cuenta con las certificaciones ISO 9001, 27001 y la ENS (Escudo Nacional de Seguridad).

Para resumir un poco, se van a exponer los servicios que ofrece ODEC junto a una breve descripción y qué tecnologías usa:

- **Captura de datos**: ODEC ofrece una captura multicanal de datos garantizando calidad, fiabilidad y seguridad mediante CATI, CAWI, digitalización documental y software de desarrollo propio.
- **Tratamiento de la información**: Se usan tanto productos propios como de sistemas estándares que proporcionan flexibilidad a la hora de optimizar los procesos. Se utiliza programación web JEE, Java, JavaScript, AngularJS, también hacen uso de depuración y almacenaje con Oracle y TomServices.
- **Presentación de resultados**: ODEC ofrece un servicio de exposición de datos de forma sencilla y operativa con una gran versatilidad y difusión mediante automatización de gráficos, tablas y bases de datos. Informes y presentaciones personalizados para el cliente o proyecto y la difusión de resultados online mediante webs personalizadas.
- **Desarrollo software**: También es una empresa que genera soluciones tecnológicas e informáticas para sus clientes, por lo que son desarrolladas y

[4] Portal de ODEC: https://www.odec.es/es/

aplicadas a medida de sus necesidades, principalmente a negocios de la automoción, sanidad, sector público, telecomunicaciones y turismo. Cabe destacar que además es *partner* de AWS, Oracle, Microsoft, VMWare y Red Hat entre otros.

- **Outsourcing de servicios**: Por último, ODEC ofrece la externalización de múltiples servicios informáticos para optimizar y liberar recursos del cliente, como puede ser el hospedaje de datos o la administración de sistemas.

Figura 2.5: Logo ODEC – Fuente https://www.odec.es/es/

El proyecto que se pretende migrar se llama **COMPLET**[5], una plataforma software implementada por ODEC cuyo cometido es el diseño y la ejecución de encuestas a través de diferentes canales (web, Smartphone, tablet). A continuación, se describe la funcionalidad de Complet, y más adelante se identificarán qué servicios lo conforman para planificar su migración a la nube.

- Diseño de encuestas a medida, con soporte multilingüe.
- Recogida de las respuestas y los comentarios desde las distintas vías mencionadas anteriormente.
- Se adapta al formato móvil.
- Selección de una muestra de individuos y envío de invitaciones desde la propia plataforma.
- Gestión de cuotas globales e individuales.
- Seguimiento online del desarrollo del trabajo de campo.
- Diseño de plantillas a partir de CSS para optar a muchas opciones, dando mucha flexibilidad al diseño de cuestionarios.

[5] Portal de ODEC COMPLET: https://www.odec.es/es/captura-de-datos/#a_complet

En resumen, Complet es un producto en constante evolución que facilita las tareas de estudio gracias a su creación de encuestas en función de su metodología, su gestión centralizada desde una única plataforma, su gran cantidad de recursos y su capacidad de crear índices para crear bloques de preguntas en cualquier orden.

2.4 La transformación digital

ODEC siempre ha estado a la vanguardia en lo que a innovación se refiere y la transformación digital ha emergido como un proceso fundamental para las organizaciones en la era actual, la tecnología evoluciona rápidamente y cambia la forma en que operan los negocios, interactúan las personas y se desarrolla la sociedad en general. La transformación digital, como dice De Giusti (2023), se refiere a la integración de tecnologías digitales en todos los aspectos de una organización, lo que conlleva cambios significativos en la forma en que opera, interactúa con los clientes y colaboradores, y crea valor. Esto implica la adopción de tecnologías como la nube, la inteligencia artificial, el Internet de las cosas (IoT) y el análisis de datos para mejorar la eficiencia, la toma de decisiones y la innovación.

La computación en la nube ha estado siempre en constante evolución además de ir creciendo año tras año, pero las empresas siempre han sido reticentes a implementar estos cambios, ya sea por miedo a ser las primeras en hacerlo, por la desconfianza que pudiera generar o por la inversión de tiempo y dinero que hay que realizar en un primer momento. No fue hasta que ocurrió la pandemia del COVID-19 que se rompió con esta mentalidad. Delgado (2020) cuenta la necesidad que obligó al mundo a trabajar desde casa, sobre todo al sector de las TIC, y como la pandemia impulsó de forma increíble la computación en la nube, se aumentó la demanda de sus servicios, ya que las empresas buscaban migrar sus proyectos y operaciones a la nube para permitir el acceso remoto y hacerlo de forma segura. La exigencia de mantener la productividad propició que esta migración se realizara de forma acelerada, por lo que la seguridad se tuvo muy en cuenta y para garantizarla se tuvieron que seguir guías de buenas prácticas como la monitorización de los sistemas, actualización de políticas de seguridad y realizar talleres de formación entre otras cosas. Gracias a ello se logró un mayor énfasis en la colaboración y utilización de herramientas en línea para poder realizar videoconferencias, almacenar datos en la nube y gestionar los procesos y aunque muchas organizaciones tuvieron que aumentar su gasto en inversión tecnológica, pudieron comprobar que las ventajas en seguridad y privacidad y que la

flexibilidad que ofrecía con respecto a las necesidades de infraestructura para realizar cambios rápidos bajo demanda eran puntos muy importantes a tener en cuenta y que sacaron a relucir su criticidad.

¿Hubo diferencias entre las empresas que ya tenían infraestructura en la nube y las que no? Por supuesto, la revista UNIR (2021) nos cuenta que aquellas que estaban preparadas pudieron amortiguar en mayor o menor medida el impacto de la pandemia, el INE estima que las grandes empresas fueran las que más confiaron en utilizar estas tecnologías (62.1% de las grandes corporaciones), en cambio, a las medianas y pequeñas empresas les costó adaptarse a este cambio (42% y 24.4%).

Actualmente, ya no existe esa gran necesidad que había en aquellos días, pero marcó un punto de no retorno e impulsó que la computación en la nube evolucionara de forma exponencial y aunque muchas empresas optan por una estrategia híbrida el cambio de mentalidad que se produjo en la sociedad ya no se va a poder borrar. Hoy en día, por ejemplo, el trabajo desde casa es uno de los principales puntos que una persona tiene en cuenta para optar a un futuro trabajo, lo que ha obligado a las organizaciones a adaptarse a las necesidades de sus empleados y no al revés. Podemos ver gracias a López (2023) como España ha incrementado el uso de las TIC en las empresas y como ha madurado digitalmente tras la pandemia en la **Figura 2.6** y la **Figura 2.7**.

Presencia en empresas	2022.I	2021.I	2020.I	2019.I
Uso de ordenadores con fines empresariales	**66,1%**	65,7%	64,7%	60,4%
Conexión a internet	**98,3%**	99,0%	98,2%	98,4%
Empresas que emplean especialistas en TIC	**17,2%**	16,4%	18,4%	17,4%
Tienen sitio o página web	**78,5%**	78,3%	78,1%	78,2%
Usan medios sociales	**67,3%**	66,6%	63,0%	52,9%
Presencia en empresas	**2022**	**2021**	**2020**	**2019**
Realizan análisis *Big Data*	**15,1%**	11,1%	8,5%	8,3%
Realizan ventas por comercio electrónico	**31,6%**	26,9%	25,5%	20,4%
Realizan compras por comercio electrónico	**38,7%**	32,3%	34,9%	33,9%

Figura 2.6: Indicadores de equipamiento y uso de las TIC en las empresas

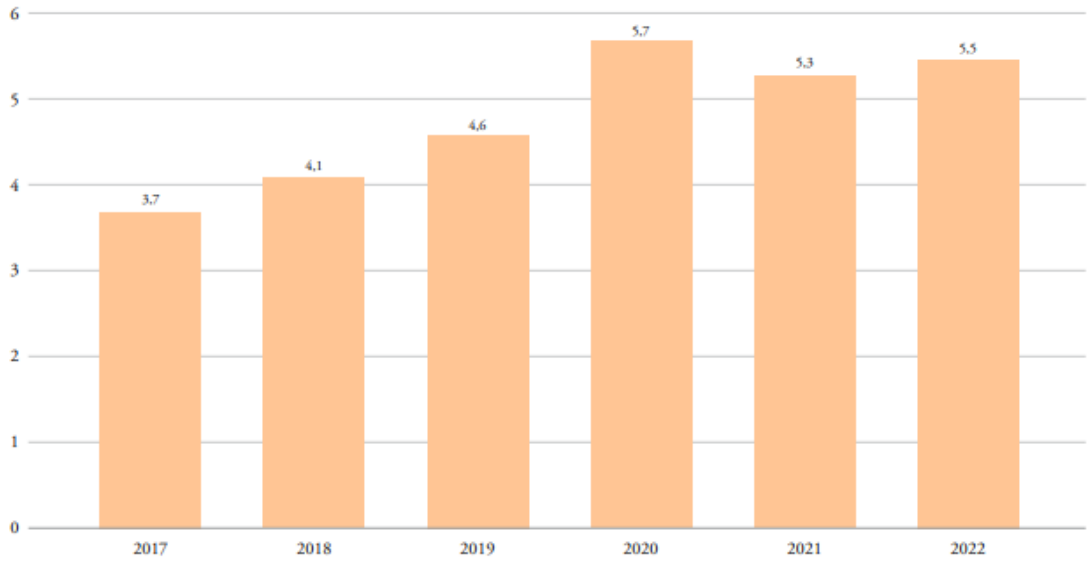


Figura 2.7: Índice de madurez digital de las empresas en España

Para ODEC, este cambio también se acentuó durante la pandemia y la imposibilidad de asistir a la oficina impulsó la migración hacia la nube de AWS que aún se encontraba en pañales. Este factor y el hecho de que la tecnología usada, es decir, la virtualización *on-premise* se estaba quedando obsoleta y que presentaba problemas, sobre todo con el tema de escalado de máquinas virtuales según la demanda de recursos, propició, por una parte, la intención de migrar los proyectos de producción a AWS y por otra empezar a usar Kubernetes.

3. Análisis del problema

AWS dispone de muchos servicios en la nube por lo que se ha de tener claro cuáles son necesarios para migrar nuestra aplicación, así que primero se explicará y desglosará cada una de las partes del proyecto *on-premise*. Como ya se ha indicado en los primeros capítulos, como valor añadido se dispone de un proyecto real que se va a migrar a AWS, aquí se va a explicar la infraestructura de la que dispone ODEC, y cómo funcionan sus aplicaciones.

3.1 La infraestructura en ODEC

La sede principal de ODEC se encuentra en Gandia y cuenta con uno de los centros de procesos de datos (CPD) más grandes de la zona, ubicado en la segunda planta del edificio, este CPD está formado por 9 racks distribuidos en 3 filas de 3 racks cada uno, y aquí se alojan servidores, cabinas de discos, firewall físico, switches, routers, etc. Podríamos decir que la infraestructura se asemeja a la **figura 3.1**. Esta sala es la más protegida de toda la empresa y solamente tienen acceso las personas con rol de administrador de sistemas. Por supuesto, el CPD tiene asignadas tareas de actualización documental y física de forma periódica con el objetivo de tener el máximo control sobre todos sus componentes. Además del CPD, la empresa cuenta con toda una red de fibra dedicada para su mayor fiabilidad coordinada mediante BGP o *Border Gateway Protocol* que garantiza que si cae una línea de internet, sale por la de respaldo. ODEC también tiene adquirido un rango de IP públicas que utiliza para publicar cada uno de los portales, ya sean propios o del cliente.

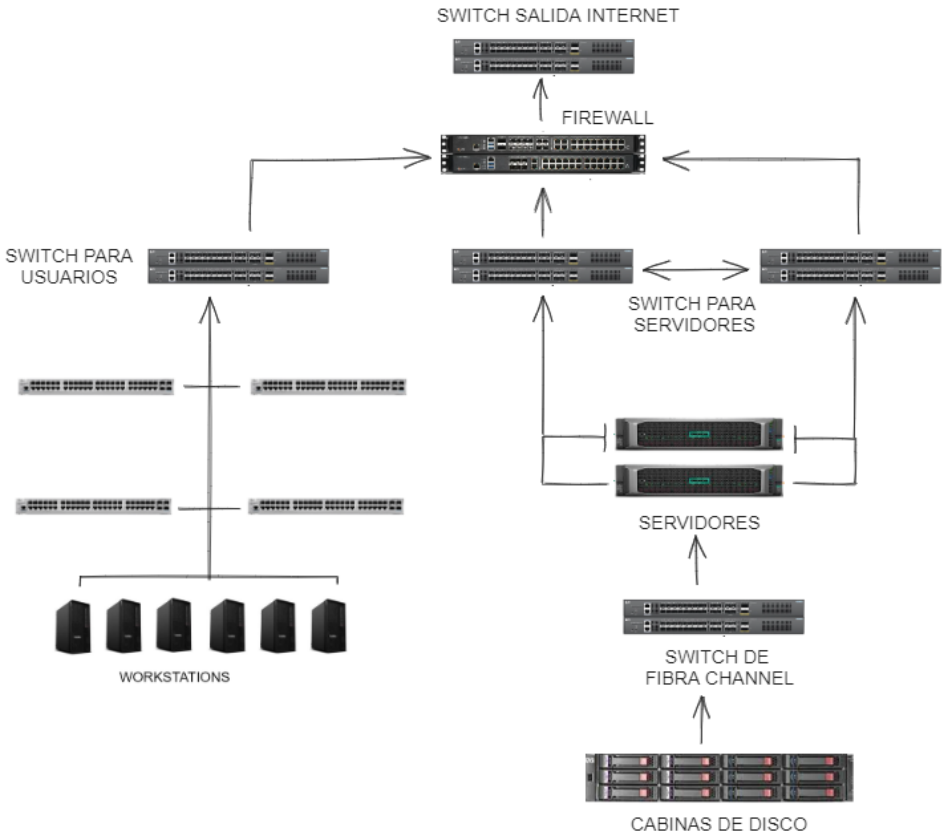


Figura 3.1: Infraestructura ODEC – Fuente propia

Como podemos ver, ODEC dispone de un firewall físico en alta disponibilidad con capacidad de crear diferentes subredes virtuales y del cual cuelgan unos switches, unos dedicados a los usuarios y otros a los servidores. Por una parte, en el CPD hay unos switches troncales que agrupan los switches repartidos por toda la empresa y a los que se conectan los usuarios según su zona de trabajo y, por otra parte, tenemos los switches dedicados para los servidores que también están en alta disponibilidad y que admiten 10Gb de conexión *Ethernet*. A su vez, estos servidores se conectan a distintas cabinas de discos a través de switches que admiten protocolo *fibre channel* o canal de fibra a 16Gbps. Además, el CPD de ODEC cuenta con todas las medidas de seguridad que exigen la ISO 27001 y la ENS, como medidas contra incendios, accesos de doble autentificación con registros, ventiladores, SAI, etc.

Toda esta infraestructura genera un coste anual muy elevado, el hardware ha de estar en mantenimiento por si se degrada de alguna forma, se hacen revisiones mensuales, se tiene que actualizar regularmente para que la seguridad no sea un problema y además hay que sumarle un coste que no se suele tener en cuenta a simple vista que es el consumo eléctrico de tenerlo todo en marcha las 24 horas del día, los 365 días del año.

3.2 Virtualización en ODEC

La virtualización es una tecnología que consiste en crear máquinas virtuales que corren un determinado sistema operativo con unos determinados recursos asignados según sus necesidades, esto optimiza y mejora los entornos de TI, ya que abstraen el hardware físico y los servicios que se ejecutan en él. ODEC es *partner* con VMWare y tiene adquiridas varias licencias ESXi y vSphere vCenter[6] que se utilizan de la siguiente manera:

1. Por una parte, VMware ESXi es una plataforma de virtualización, un hipervisor bare metal que se instala en los servidores físicos y que permite crear y administrar múltiples sistemas operativos en forma de máquinas virtuales.
2. Por otra parte, vSphere vCenter es una herramienta que permite gestionar de forma centralizada todos estos ESXi y las máquinas virtuales que contienen cada uno de ellos.

Al final, como podemos ver en la **Figura 3.2**, queda un entorno de trabajo administrado por el vCenter en el que se gestiona que ESXi forma parte de este, si están distribuidos por clústeres de trabajo, por ejemplo producción y preproducción y también se gestionan las máquinas virtuales, permitiendo que las máquinas puedan moverse en caliente entre ESXi gracias al DRS o *Distributed Resource Scheduler* que optimiza los recursos y que además se pueden recuperar ante un fallo gracias al *High Availability* (HA) y al *vMotion.* Por último, el vCenter también es capaz de administrar los *datastores* o volúmenes de almacenamiento, que son visibles por todos los servidores y que gracias al *Storage vMotion*, las máquinas virtuales son capaces de migrar entre discos en caliente. Actualmente, el vSphere está sincronizado con el directorio activo de la empresa, por lo que solamente los usuarios autorizados tienen acceso a la herramienta para gestionarla. ODEC tiene creados dos clústeres, uno dedicado a desarrollo y otro dedicado a producción y que está conformado por 3 servidores. La migración en caliente de las máquinas virtualizadas no es posible entre clústeres, por lo que sería necesario apagar las máquinas si fuera necesario realizar esta acción (otros factores a tener en cuenta para la migración entre servidores es por ejemplo el procesador del propio servidor, si son incompatibles no se podría migrar las máquinas aunque formaran parte del mismo clúster).

[6] Portal VMware vSphere: https://docs.vmware.com/es/VMware-vSphere/index.html

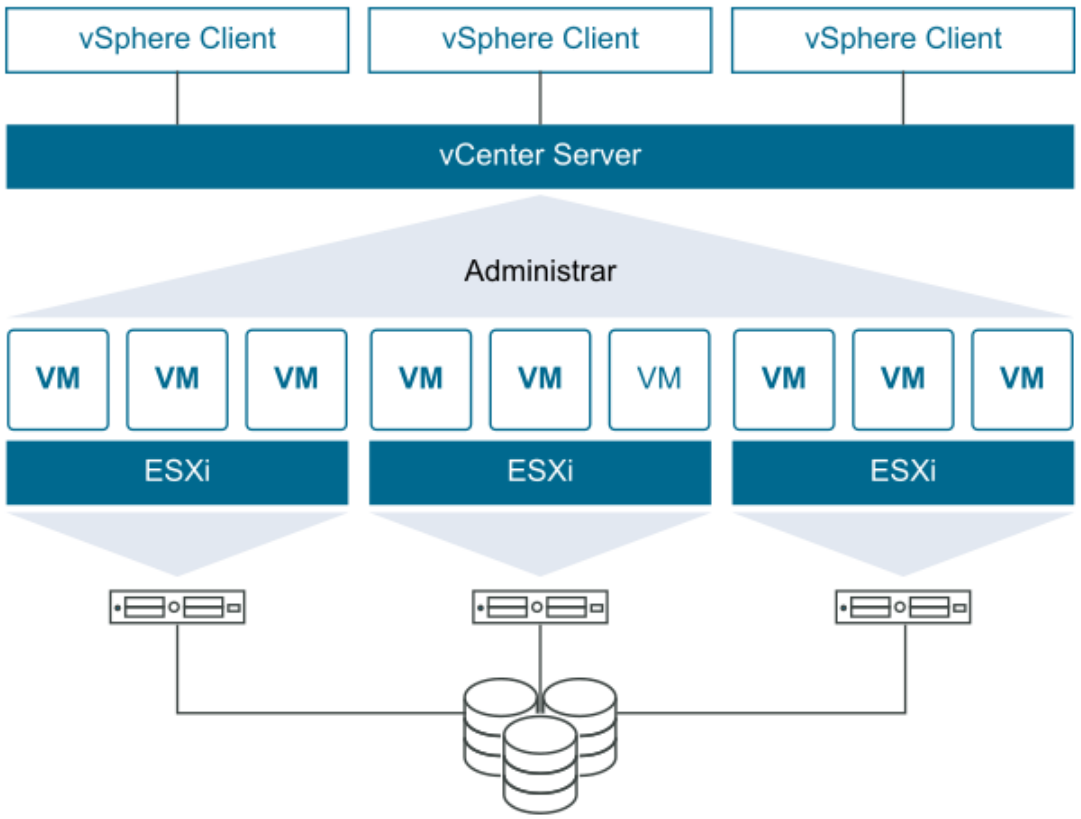


Figura 3.2 Estructura de VMware vSphere – Fuente: https://docs.vmware.com/es/VMware-vSphere/index.html

3.3 Proyecto COMPLET *on-premise*

Este proyecto está conformado por diversos servicios y máquinas y en este apartado se explicará cada uno de ellos. Como ya hemos dicho, el proyecto Complet se encuentra virtualizado en el clúster de producción gestionado por el vCenter de ODEC y a su vez, las máquinas virtuales están organizadas de dos formas dentro del vCenter, se puede observar en la **Figura 3.3**:

1. **Grupo de recursos:** Los grupos de recursos en vSphere como su nombre indica, permiten, de forma resumida, agrupar las máquinas para que usen solamente una cantidad de recursos de memoria y CPU disponibles. Un clúster en sí representa un grupo de recursos principal o raíz, por lo que no aparece como tal si solamente existe este grupo. Se deben crear grupos secundarios para que se muestre una jerarquía.

2. **Carpetas organizativas:** Esta forma de administrar las máquinas virtuales es meramente organizativa y sirve para visualizar de forma sencilla qué máquinas conforman un proyecto y no tiene un impacto real en el uso de recursos.

Figura 3.3 Jerarquía de elementos en vSphere – Fuente: https://docs.vmware.com/en/VMware-vSphere/8.0/vsphere-resource-management/GUID-60077B40-66FF-4625-934A-641703ED7601.html

El proyecto consta de 3 tipos de máquina virtuales y su estructura se asemeja a la **Figura 2.1** vista anteriormente y que generalmente es el estándar en servicios *on-premise*:

1. **Apaches**: Estas *VM* cumplen la función de balancear la carga hacia los nodos de trabajo según el servicio que se pida y la carga de trabajo. Tienen instalado un servicio **httpd** y se disponen de varios *virtualHost* (configuración que permite alojar varios portales web en una misma máquina) que redirigen las peticiones de los clientes al estudio equivalente y además existen al menos 2 apaches para garantizar alta disponibilidad.
2. **Nodos de trabajo**: Máquinas virtuales las cuales tienen instalado un servicio *jboss AS* (actualmente conocido como *wildfly,* un software libre que permite cumplir la función de servidor de aplicaciones Java EE) que tiene desplegado la aplicación desarrollada. Éstas conforman la mayoría de máquinas y cumplen los siguientes roles en este proyecto:
    a. **Motor de ejecución:** El motor de entrevistas es el encargado de interpretar las entrevistas para presentarlas a los usuarios en los diferentes canales (web, móvil) y recoger las respuestas. Una vez iniciada una entrevista se establece una interacción con el motor que va interpretando las respuestas y presentando nuevas preguntas al entrevistado. Todo ello, atendiendo al diseño del cuestionario realizado previamente.

b. **Aplicación de gestión:** Permite gestionar todo el ciclo de vida de un cuestionario:
    - Alta del cuestionario.
    - Gestión de la muestra y envío de emails.
    - Seguimiento en tiempo real.
    - Exportación de resultados para su posterior análisis.
c. **Diseñador de entrevistas:** Se utiliza para diseñar el cuestionario. El diseño consiste en:
    - Creación de preguntas de diferentes tipos (selección simple/múltiple, texto).
    - Diseño del flujo de secuencia del cuestionario en función de las respuestas obtenidas.
    - Simulación y pruebas.
d. **Nodo maestro de despliegues:** Este último es algo especial, ya que funciona como un orquestador encargado de desplegar los paquetes de aplicación en los distintos tipos de nodos.

3. **Base de datos:** Por último, tenemos la base de datos, una máquina virtual con un servicio ORACLE con todos los datos relacionados con el proyecto.

Estos serían los elementos principales de Complet *on-premise* en ODEC, pero no los únicos. De forma resumida se procederá a explicar cómo transcurre una petición de un cliente y qué partes además de las ya mencionadas entran en juego:

1. El cliente desde su casa realiza una petición a un estudio del portal Complet.
2. ODEC, como ya se ha explicado antes, tiene comprados un rango de IP públicas y gracias a Go Daddy (un registrador de dominios) se realiza la equivalencia del nombre DNS a una IP pública que recoge el cortafuegos.
3. El cortafuegos actúa como traductor de direcciones IP mediante traducción de direcciones de red (NAT) y transforma la IP pública en una IP privada de la red empresarial, en este caso la traduce a un grupo de objetos IP que equivalen a los apaches antes mencionados.
4. La petición que entra por uno de los dos apaches es procesada por el servicio httpd y que gracias al *virtualhost* se redirige al nodo que tiene que resolver la solicitud.

5. El nodo en cuestión se comunica con la base de datos y se realiza el camino inverso para contestar al cliente.

3.4 Problemas presentes

El proyecto que se pretende migrar a AWS se encuentra ubicado en el clúster de producción, que contiene unas 300 máquinas virtuales (de distintos proyectos) organizadas por grupos de recursos y carpetas organizativas. Esta forma de trabajar en la infraestructura local presenta un problema bastante importante y que la empresa puso en primer lugar a la hora de solucionar, la escalabilidad según la carga de trabajo en las encuestas. La escalabilidad es la capacidad de un sistema, o proceso de adaptarse a la carga de trabajo según las necesidades requeridas, aumentando o disminuyendo los recursos utilizados según se necesite, de forma que el proyecto sea eficiente sin perturbar la experiencia del usuario.

La escalabilidad vertical, que consiste en aumentar los recursos de un solo componente, se descartó por limitaciones de las propias máquinas virtuales y porque no había forma de automatizarlo de forma eficiente en vSphere así que la solución temporal dada fue la escalabilidad horizontal, que consistía en crear nuevos nodos para introducirlos en el clúster, distribuyendo la carga de trabajo entre todas las máquinas virtuales. Este método tampoco resultó del todo eficiente porque requería tiempo de creación de nuevas máquinas, añadirlas al grupo según el rol que iban a desempeñar o sacarlas del balanceador en caso de que la escalabilidad tuviera que bajar.

En conclusión, para solucionar este tema se ha propuesto migrar el proyecto a AWS y cambiar de tecnología, en general se van a migrar los servidores de aplicación, la base de datos y los balanceadores de carga.

4. Diseño de la solución

En el presente punto se procederá a mostrar que elementos del proyecto Complet se van a migrar y explicar qué herramientas se van a utilizar, tanto en el primer paso que consiste en contenerizarlo y probarlo en la propia infraestructura de ODEC como el segundo paso que consiste finalmente migrarlo a AWS, con una infraestructura más robusta y la planificación utilizada.

4.1 Elementos a migrar

Los elementos del proyecto Complet descritos antes que se van a migrar a AWS son los siguientes:

- **Nodos de trabajo:** Permiten ejecutar las diferentes aplicaciones que conforman Complet como ya se han descrito anteriormente. De los cuatro tipos que se han mencionado en el apartado anterior solamente se van a migrar tres, motor de ejecución, aplicación de gestión y diseñador de entrevistas. Cada uno de estos tipos representa actualmente en la arquitectura *on-premise* un tipo de máquina virtual distinta con sus bibliotecas, código y configuraciones, por lo que habrá que crear una imagen Docker distinta para cada una de ellas.
- **Base de datos:** Actualmente, la base de datos está en una máquina virtual con un servicio de ORACLE, pero debido al licenciamiento, cuando se migre a AWS hay que migrarla a PostgreSQL, por lo que se debe tener en cuenta ciertos formatos en las consultas o funciones que se utilizan. Aunque Amazon ofrece dos opciones de licencias para las bases de datos ORACLE, que son licencia incluida y *Bring-Your-Own-License* (BYOL), ODEC no se ha decantado por ninguna de ellas. La primera opción se ha descartado porque además de ser de pago, hay que adaptar la versión de ORACLE local a la que ofrece AWS. La segunda también se ha descartado porque la licencia de Oracle que tiene ODEC no puede migrarse a la nube, ya que esta licencia debe cubrir obligatoriamente todos los procesadores físicos de los servidores ESXi locales (sobre todo del clúster de producción) que conforman vSphere, ya que existen otras máquinas virtuales con servicios ORACLE instalados que van migrando de un servidor a otro según

el uso de recursos. Por estas razones, ODEC ha decidido utilizar PostgreSQL en AWS.

- **Balanceadores:** En la arquitectura local estos balanceadores equivalen a las máquinas virtuales con servicios httpd que redirigen el tráfico según el tipo de petición, pero en el clúster de Kubernetes que se pretende crear será una aplicación Nginx desplegada en el propio clúster.

Por último, se deben publicar en AWS los recursos estáticos de los que hace uso el proyecto Complet, como imágenes, ficheros html, ficheros de audio, video, css, de texto o de cualquier otro tipo.

Seguidamente, se van a presentar las imágenes de los dos primeros niveles de un diagrama de arquitectura tipo C4 del funcionamiento del proyecto Complet[7]. Este tipo de diagramas representa múltiples niveles del software.

1. **Contexto**: Es el más alto nivel y muestra una vista de contexto entre los actores externos y el sistema.
2. **Contenedor**: En este nivel se muestran las unidades lógicas que agrupan los componentes del software, aplicaciones, servidores, bases de datos, etc.
3. **Componente**: En este nivel se desglosan los contenedores en componentes individuales. En este proyecto no podremos mostrar a tan nivel de detalle los componentes de los contenedores, ya que tenemos 3 y se alargaría demasiado.
4. **Código**: En este último nivel se muestran los detalles a nivel de código fuente y que, por tema de protección de datos con ODEC, no mostraremos.

[7] Portal de Miro C4 Architecture: https://miro.com/miroverse/c4-architecture/

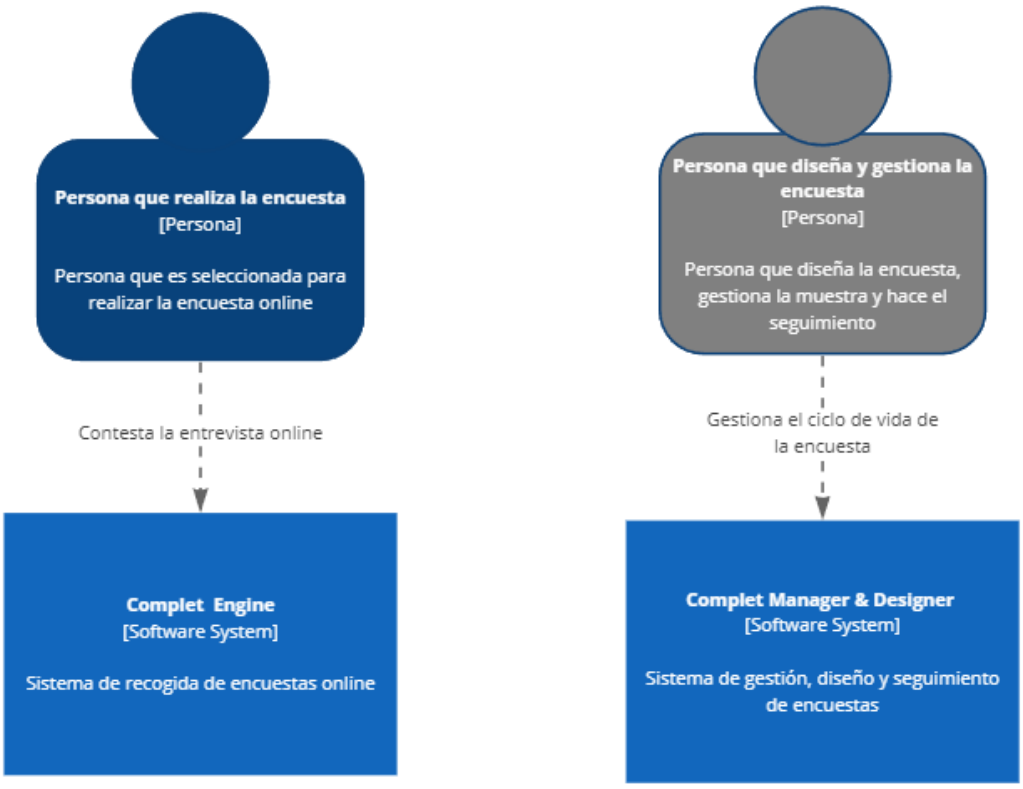


Figura 4.1 Complet C4 arquitectura – Contexto.

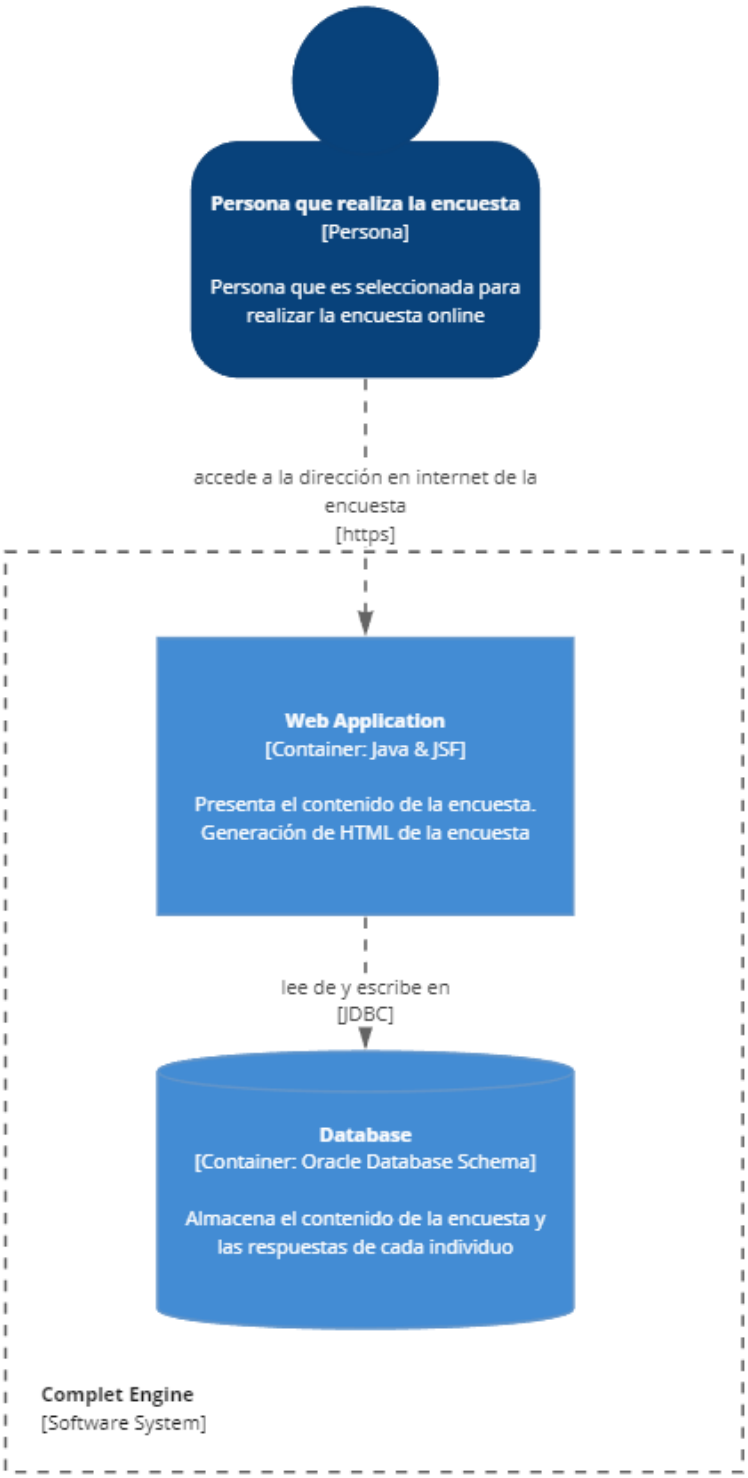


Figura 4.2 Complet C4 Arquitectura – Contenedor del motor de ejecución.

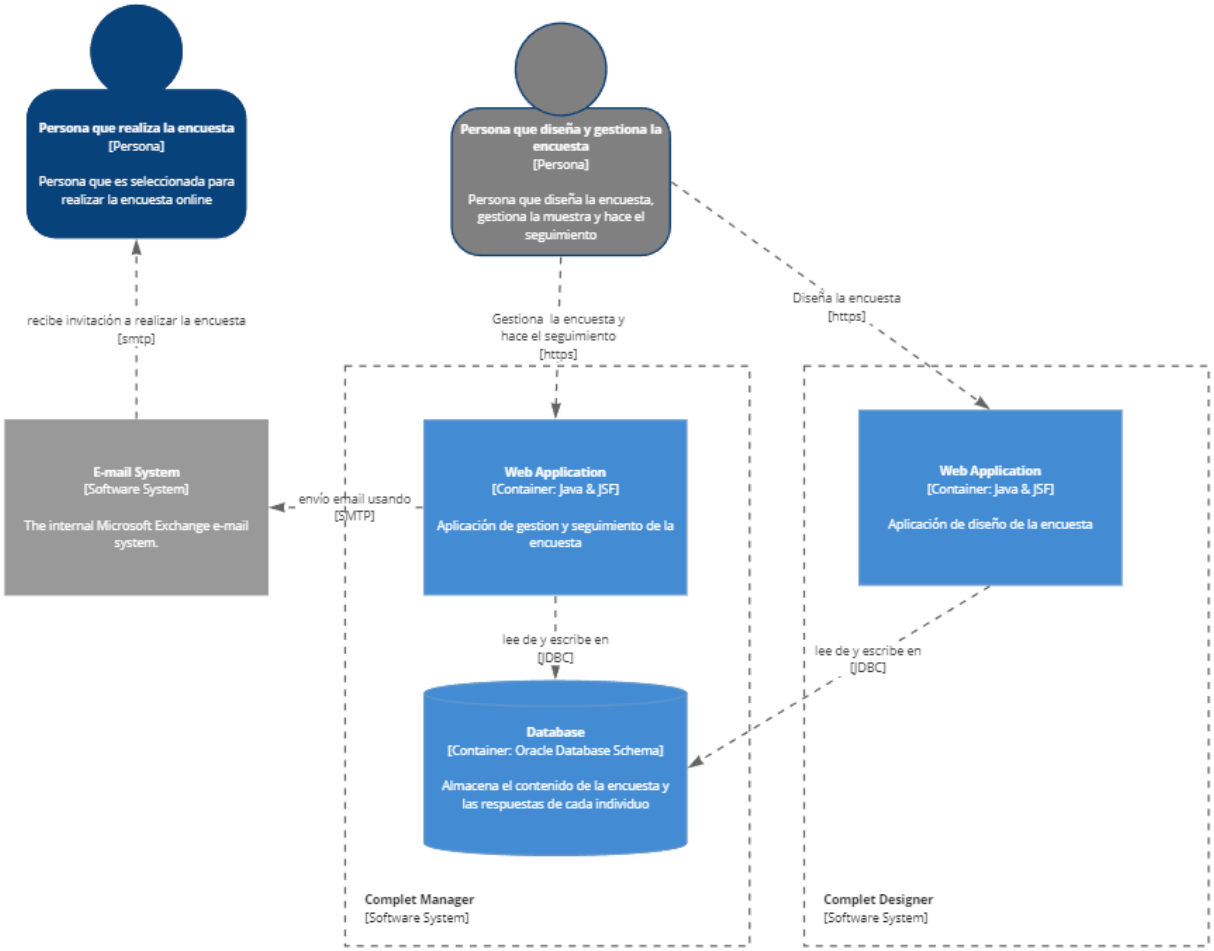


Figura 4.3 Complet C4 Arquitectura – Contenedor de la aplicación de gestión y del diseñador de entrevistas.

4.2 Docker

El primer paso es contenerizar las aplicaciones de Complet mediante Docker[8] y probarlas en la infraestructura *on-premise* antes de migrar el proyecto a AWS y asegurarnos de que funcionen correctamente.

Docker es una plataforma de código abierto diseñada para la creación, el despliegue y la ejecución de aplicaciones en contenedores. A diferencia de las máquinas virtuales tradicionales, que emulan sistemas operativos completos y requieren recursos significativos, los contenedores de Docker encapsulan aplicaciones junto con sus dependencias y bibliotecas en entornos aislados y ligeros. Esto permite

[8] Portal de Docker: https://docs.docker.com/get-started/overview/

que las aplicaciones se ejecuten de manera coherente y predecible en cualquier entorno que admita Docker, ya sea un equipo de desarrollo local, servidores de pruebas o nubes de producción. Entre los beneficios de utilizar Docker están:

- **Portabilidad**: Los contenedores de Docker son autocontenidos y encapsulan todos los componentes necesarios para que una aplicación funcione correctamente. Esto significa que una aplicación que funciona en un entorno de desarrollo también funcionará de la misma manera en un entorno de producción, evitando problemas de ``funciona en mi máquina´´.
- **Aislamiento**: Cada contenedor de Docker ejecuta aplicaciones en un entorno aislado. Esto garantiza que las dependencias y las configuraciones de una aplicación no interfieran con otras aplicaciones que se ejecuten en el mismo sistema. Además, el aislamiento proporciona una mayor seguridad al evitar que los posibles problemas en una aplicación afecten a todo el sistema.
- **Eficiencia**: Como los contenedores Docker comparten el kernel del sistema anfitrión, reduce drásticamente el uso de recursos, cosa que no ocurre con las máquinas virtuales tradicionales, que consumen muchos recursos al emular sistemas operativos completos. Esto permite ejecutar más aplicaciones en el mismo hardware, mejorando la utilización de los recursos.
- **Escalabilidad**: Docker facilita la creación de aplicaciones escalables. Puedes crear múltiples instancias idénticas de un contenedor y orquestar su gestión utilizando herramientas como Docker Compose o Kubernetes. Esto simplifica la administración de aplicaciones en entornos de gran escala.
- **Gestión de versiones**: Docker permite definir la configuración de una aplicación en un archivo llamado ``Dockerfile´´. Esto facilita la creación de imágenes de contenedor consistentes y permite el seguimiento y la gestión de versiones de la aplicación y sus dependencias.

Docker utiliza una arquitectura, **Figura 4.4**, cliente-servidor. El cliente se comunica con un demonio Docker, que realiza el trabajo de crear, ejecutar, y distribuir contenedores Docker. El cliente y el demonio se pueden ejecutar en el mismo sistema o también el cliente puede conectarse a un demonio remoto, estos dos se comunicarán mediante una API REST, a través de sockets UNIX o una interfaz de red.

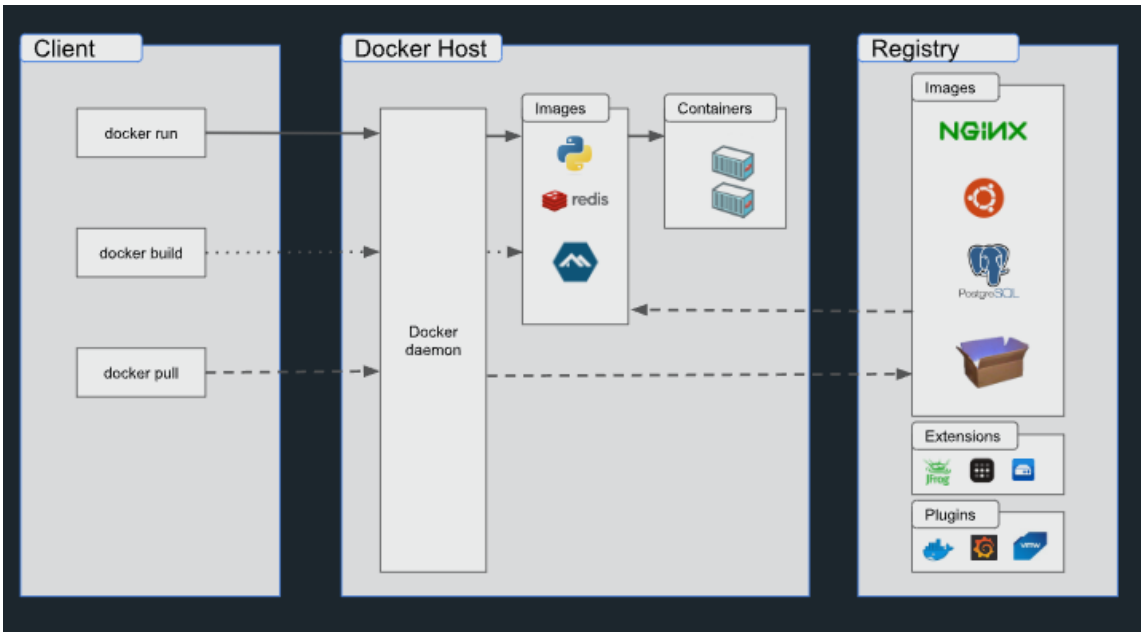


Figura 4.4 Ejemplo de la arquitectura Docker – Fuente: https://docs.docker.com/get-started/overview/

El primer paso, por lo tanto, será generar imágenes de los nodos de trabajo (Wildfly + aplicación) del proyecto Complet, esto implica crear un paquete autónomo y autocontenido que contenga la aplicación y todas sus dependencias, configuraciones y archivos necesarios para que la aplicación funcione en cualquier entorno Docker. Resumidamente, los pasos para generar una imagen son:

1. **Crear un Dockerfile**: El punto de partida para generar una imagen en Docker es crear un archivo llamado ``Dockerfile´´. Este archivo contiene instrucciones que Docker utilizará para construir la imagen. Las instrucciones en el Dockerfile describen cómo instalar dependencias, copiar archivos, configurar el entorno y más.

2. **Elegir una imagen base**: Cada imagen en Docker se basa en otra imagen existente, conocida como ``imagen base´´. Esta imagen base proporciona un sistema operativo y una configuración inicial. Se puede elegir una imagen base que coincida con el sistema operativo y el entorno que se necesite para la aplicación.
3. **Escribir las instrucciones del Dockerfile**: Dentro del Dockerfile, se escribirán una serie de instrucciones que Docker seguirá durante la construcción de la imagen, incluirá instrucciones como 'FROM', 'COPY', 'RUN', 'ENV', 'WORKDIR', 'CMD' y muchas más, todas ellas explicadas en la documentación en el portal de Docker.
4. **Construir la imagen**: Una vez que se ha creado el Dockerfile, se usa el comando `'docker build'` seguido de la ruta al directorio que contiene el Dockerfile para construir la imagen. Docker leerá el Dockerfile y seguirá las instrucciones para crear la imagen. Un paso opcional sería etiquetar y subir la imagen creada tanto a un repositorio privado como a uno público.
5. **Ejecutar un contenedor basado en la imagen**: Una vez que la imagen se ha construido y, opcionalmente, subido a un repositorio, puedes crear y ejecutar contenedores basados en esa imagen usando el comando `'docker run'`.

En resumen, generar una imagen en Docker implica definir un conjunto de instrucciones en un Dockerfile y utilizar el comando `'docker build'` para construir la imagen. Esta imagen puede luego ser utilizada para crear y ejecutar contenedores que contienen la aplicación y su entorno asociado.

4.3 Kubernetes

Para orquestar y administrar los contenedores Docker en un entorno escalable y automatizado se utilizará Kubernetes.

Kubernetes[9] es una plataforma de código abierto diseñada para automatizar, escalar y gestionar aplicaciones o servicios contenerizados (paquetes aislados entre sí) ya sea en entornos locales o en la nube, sobre todo en este último. Entre sus características podemos mencionar:

[9] Portal Kubernetes: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

- **Orquestación de contenedores**: Gestiona y orquesta contenedores como Docker, de manera eficiente y automatizando tareas como el aprovisionamiento, la escalabilidad, distribución de la carga, etc.
- **Abstracción de infraestructura**: Kubernetes proporciona una capa de abstracción entre las aplicaciones y la infraestructura física, permitiendo que las primeras sean independientes.
- **Escalabilidad**: Permite tanto la escalabilidad horizontal como la vertical, añadiendo o reduciendo instancias de contenedores según la demanda.
- **Automatización**: Es capaz de automatizar tareas repetitivas, como la implementación o actualización de las aplicaciones.
- **Balanceador**: Ofrece mecanismos para descubrir los servicios y distribuir el tráfico de red de manera equitativa.
- **Gestión de la configuración**: Permite gestionar de forma centralizada la configuración de las aplicaciones e instancias.
- **Actualizaciones sin tiempo de caída**: Facilita la implementación de actualizaciones sin cortar el servicio al cliente.
- **Portabilidad**: Kubernetes es compatible con muchos proveedores de Internet, entre ellos AWS.

Kubernetes se compone de maestros y nodos de trabajo, los primeros gestionan a los segundos y estos últimos se encargan de ejecutar los contenedores, podemos ver un ejemplo en la **Figura 4.5**. Los usuarios pueden gestionar Kubernetes mediante comandos (`kubectl`) o mediante interfaces gráficas[10]. Los elementos de un nodo maestro son:

1. **Kube-APIserver (API Server)**: Es el componente central de control del clúster, sirve como punto de entrada para administrarlo y para gestionar las solicitudes de los usuarios y las aplicaciones y se encarga de persistir el estado del clúster.
2. **Etcd:** Es la base de datos distribuida que se encarga de almacenar el estado del clúster, tanto los nodos maestros como los nodos del clúster consultan y actualizan el estado del clúster aquí.
3. **Kube-Scheduler**: Se encarga de asignar los Pods (contenedores) a los nodos específicos correspondientes según las necesidades.

[10] Administrar un clúster de Kubernetes con RKE: https://expleogroup.medium.com/managing-kubernetes-clusters-with-rancher-7804eba150f5

4. **Kube-Controller-Manager**: Incluye controladores (de despliegues o servicios) que se encargan de mantener el estado deseado del clúster.
5. **Cloud-Controller-Manager**: Este elemento se encuentra en entornos en la nube y se encarga de interactuar con los otros servicios que proporciona el proveedor de la nube.

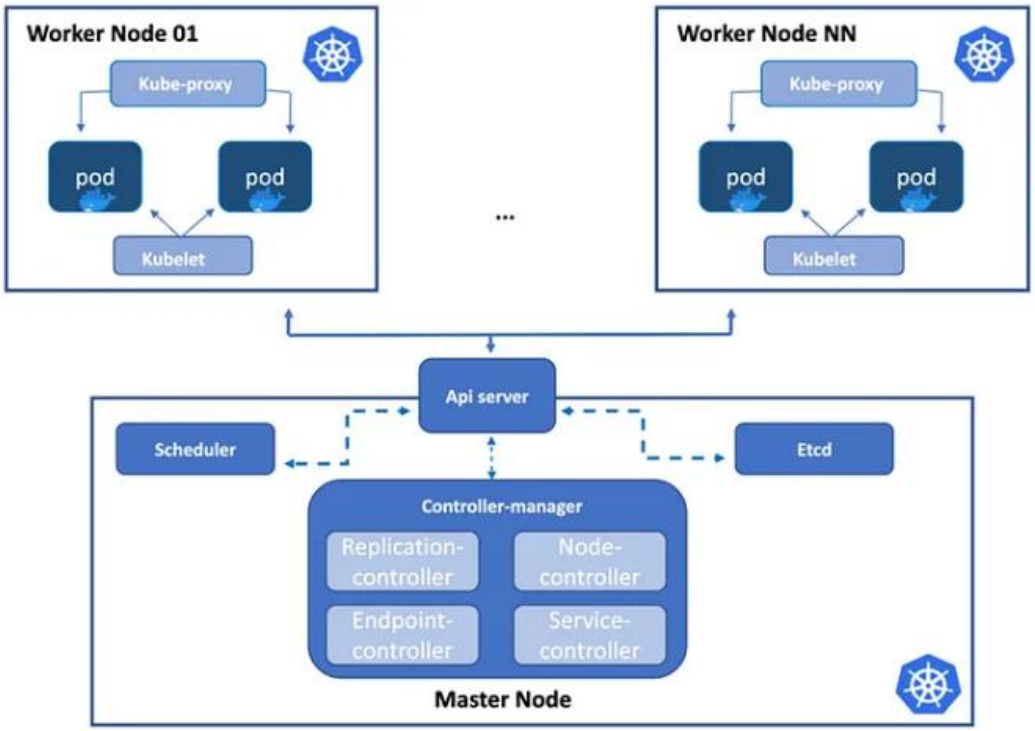


Figura 4.5 Ejemplo común de la arquitectura de un clúster de Kubernetes – Fuente: https://expleogroup.medium.com/managing-kubernetes-clusters-with-rancher-7804eba150f5

Los nodos de trabajo o máquinas virtuales, en cambio, ejecutan las aplicaciones y contenedores y tiene un conjunto de componentes que trabajan juntos para administrar sus elementos. Aquí algunos componentes:

1. **Kubelet**: Es el agente que se ejecuta en cada nodo y se comunica con los nodos maestros para garantizar su integridad. Se encarga de iniciar, detener y mantener los Pods en un nodo.
2. **Kube-Proxy**: Componente encargado de mantener las reglas de red del nodo y realizar el equilibrio de carga. Implementa Service Load Balancing y permite que los Pods se comuniquen entre sí.
3. **Container Runtime**: Es el software que se encarga de ejecutar los contenedores en el nodo, por ejemplo Docker.

4. **Pods**: Son la unidad básica de un despliegue en Kubernetes. Un Pod es un contenedor que se ejecuta en un nodo y que puede escalar la aplicación.
5. **Kube-DNS**: Proporciona la resolución de nombres de los servicios dentro del clúster. Los Pods se pueden comunicar entre sí utilizando nombres de servicio y no direcciones IP.
6. **CNI**: Se encarga de la configuración de red de los contenedores.
7. **Plugins de volumen**: Se encarga de que el almacenamiento de los contenedores sea persistente y permite montar volúmenes NFS, por ejemplo, ya que los Pods se crean y destruyen muy fácilmente.
8. **Administrador de recursos**: Es el encargado de asignar los recursos como son la CPU y la memoria a los Pods.

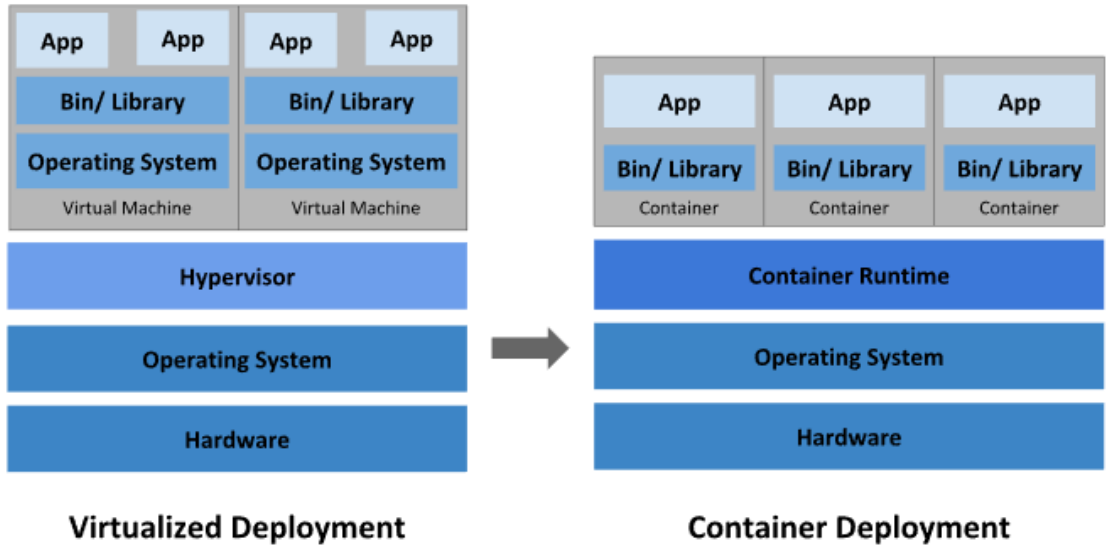


Figura 4.6 Despliegue virtualizado vs despliegue contenerizado – Fuente: https://kubernetes.io/docs/concepts/overview/

Hay que tener claro que Kubernetes no es una Plataforma como Servicio (PAAS), ya que opera nivel de contenedor y no a nivel de hardware, puede asemejarse, pero la gran ventaja de Kubernetes es que las soluciones que ofrece no son únicas y el usuario siempre podrá encontrar una forma no documentada de seguir adelante.

Por lo tanto, sabiendo ya el proceso de creación de imágenes Docker y cómo funciona Kubernetes, el despliegue de aplicaciones utilizando estas dos tecnologías es una estrategia poderosa para gestionar, orquestar y escalar las aplicaciones de manera eficiente y confiable. El proceso sería:

1. **Definición de Recursos de Kubernetes**: Tras la creación de las imágenes Docker y su registro en un repositorio, como se indica en el apartado de Docker, se define un conjunto de recursos en Kubernetes que describen cómo se debe ejecutar la aplicación. Esto se hace utilizando objetos como Deployments, que definen la cantidad de réplicas de la aplicación que deben estar en funcionamiento, y Services, que permiten la comunicación con los contenedores.
2. **Despliegue de la Aplicación**: Al aplicar los archivos de configuración al clúster de Kubernetes, se inicia el proceso de despliegue. Kubernetes lee las especificaciones y crea los contenedores basados en las imágenes Docker en los nodos del clúster.
3. **Orquestación y Escalabilidad**: Kubernetes es capaz de garantizar que el número especificado de réplicas de la aplicación esté en funcionamiento en todo momento. Si un contenedor falla o un nodo se vuelve inaccesible, Kubernetes automáticamente reinicia los contenedores o los migra a otros nodos saludables.
4. **Balanceo de Carga**: Si se utiliza un Service, Kubernetes también se encarga de distribuir el tráfico entre las diferentes réplicas de la aplicación, lo que garantiza una distribución equitativa de las cargas de trabajo entrantes.
5. **Actualizaciones y Rollbacks**: Cuando se necesita actualizar la aplicación, se puede cambiar la versión de la imagen Docker en la especificación del Deployment. Kubernetes implementará las actualizaciones de manera gradual, asegurando que siempre haya una cantidad mínima de réplicas disponibles durante el proceso. Si se presenta algún problema, Kubernetes permite revertir rápidamente a una versión anterior de la imagen.
6. **Monitorización y Escalabilidad Automática**: Kubernetes ofrece características de monitorización y escalabilidad automática. Puede ajustar automáticamente la cantidad de réplicas en función de la carga de trabajo y establecer umbrales para escalar verticalmente u horizontalmente.

En resumen, el despliegue de Kubernetes con imágenes Docker simplifica la administración de aplicaciones al proporcionar una plataforma robusta para gestionar la infraestructura, escalar las aplicaciones y mantener una alta disponibilidad. La combinación de estas tecnologías permite mantener aplicaciones confiables y escalables de manera más eficiente.

Teniendo claro el concepto de Kubernetes, el siguiente paso es saber qué utilizar para gestionar y administrar estos contenedores y es aquí donde entra AWS Elastic Kubernetes Service (EKS) y Elastic Kubernetes Service Anywhere (EKSA).

4.4 AWS EKS y EKSA

Aunque puedan parecer lo mismo, ya que los dos permiten ejecutar Kubernetes, existe una simple diferencia entre EKS[11] y EKSA[12] y es que el primero es un servicio gestionado por AWS, mientras que el segundo es una extensión que permite ejecutar clústeres EKS en infraestructuras locales. Básicamente, la diferencia radica en la ubicación, mientras que EKS se ejecuta en la nube de AWS, EKSA se ejecuta en local y esta diferencia puede presentar diferentes formas en el control, la seguridad, la implementación o la escalabilidad del servicio. EKSA una herramienta de código abierto para Amazon EKS utilizada para implementar, administrar y operar clústeres de Kubernetes de manera sencilla y eficiente y que es compatible con VMWare vSphere[13] utilizando las mismas prácticas que en AWS y su servicio EKS. EKSA ofrece dos tipos de arquitecturas, clúster de gestión y clúster independiente.

- **Clúster de gestión (Management Cluster)**: Éste es un tipo de clúster dedicado a administrar y gestionar otros clústeres, como su tiempo de vida, aplicaciones, políticas o monitorización. Puede crear y actualizar varios clústeres EKSA independientemente
- **Clúster independiente (Standalone Cluster)**: Se trata de un clúster de Kubernetes autónomo que opera de forma independiente y contiene las aplicaciones. Tiene su propio control sobre los nodos y servicios y no es gestionado por ningún otro clúster. Se suele utilizar para entornos aislados.

Para la contenerización del proyecto Complet se va a utilizar este último tipo de arquitectura para el entorno de desarrollo, ya que tras comprobar su funcionamiento y posterior migración a AWS se destruirá.

[11] Portal AWS EKS: https://aws.amazon.com/es/eks/
[12] Portal Amazon EKS Anywhere: https://anywhere.eks.amazonaws.com/docs/
[13] EKSA en VMWare vSphere: https://anywhere.eks.amazonaws.com/docs/getting-started/vsphere/vsphere-prereq/

Como se ha explicado antes, EKSA es compatible con VMWare vSphere, pero tiene que cumplir algunos requisitos disponibles en la documentación oficial de EKSA[14]:

- vSphere vCenter tiene que estar como mínimo en la versión 7.
- Es necesario tener la capacidad de albergar al menos entre 6 y 10 máquinas virtuales.
- Debe existir un servicio DHCP que asigne las direcciones IP al grupo de máquinas virtuales, esta parte se soluciona mediante el servicio DHCP de los controladores de dominio en el directorio activo de la empresa.
- El clúster necesita una VLAN declarada en el vCenter. Como se ha mencionado en la infraestructura de la empresa, la subred virtual se crea desde el firewall de ODEC y se dedica exclusivamente a este clúster y tras su creación se debe declarar tanto en los switch de red como en el propio vCenter.
- Una OVA convertida en plantilla que esté en el vSphere para ser usada como punto de creación de las máquinas virtuales.
- Las credenciales necesarias para administrar máquinas virtuales.
- Una IP fuera del rango ofrecido por el DHCP para el *Control Plane Endpoint IP*, lo cual no es un problema, porque como ya se ha dicho, el clúster tendrá una VLAN dedicada.

Algunos requerimientos mínimos de las máquinas virtuales que se van a crear son:

- 2 vCPUs.
- 8 GB RAM.
- 25GB de disco.

4.5 Terraform

Para automatizar el proceso de creación del clúster en la infraestructura *on-premise* se utilizará Terraform[15], un software de infraestructura como código (*IAC*) que permite gestionar y aprovisionar los recursos de la plataforma de virtualización como las máquinas virtuales, las redes, el almacenamiento y otros componentes gracias a

[14]Requerimientos para EKS Anywhere en VMware vSphere: https://anywhere.eks.amazonaws.com/docs/getting-started/vsphere/vsphere-prereq/

[15] Portal de Terraform: https://www.terraform.io/

un proveedor dedicado, Terraform permite gestionar tanto infraestructuras en la nube como locales y existen proveedores para diversas tecnologías, VMware, Azure, AWS, etc.

Como ya hemos dicho, Terraform permite definir la infraestructura utilizando código, concretamente *HashiCorp Configuration Language* (HCL) o JSON, lo que permite que se comparta y que sea fácil de mantener, su sintaxis es declarativa, es decir, los archivos de configuración describen el estado de los recursos y elementos de la infraestructura. Además de haber proveedores para diversas plataformas (cada uno tiene su conjunto de APIs y recursos) también admiten tecnologías como Kubernetes o Docker. Parte de la configuración de Terraform consiste en definir los recursos o bloques de construcción, como las máquinas virtuales, redes, y otros elementos, y asimismo permite mantener el estado de la infraestructura actual gracias a un archivo de estado dedicado a ello, este archivo describe qué cambios aplica Terraform a la infraestructura y ayuda a evaluar el impacto que tendrán las modificaciones. Por último, gracias a los módulos, este software permite reutilizar configuraciones para facilitar los despliegues en el entorno de virtualización.

Como se ha explicado, Terraform cuenta con lo que se llama un plan de ejecución (archivo de estado), que es una descripción detallada de los cambios que el propio Terraform tiene pensado aplicar en la infraestructura, basándose en los cambios de los archivos de configuración y se ejecuta mediante el comando *"*`terraform plan`*"*. Si tras esto se está conforme con los cambios que se van a realizar, se puede aplicar mediante *"*`terraform apply`*"* o también se puede destruir con *"*`terraform destroy`*"*.

Este proceso es muy recomendable realizarlo antes de aplicar modificaciones, ya que ayuda a ser consciente de los cambios que se van a realizar y así evitar errores que conllevan el mal uso de recursos de la infraestructura o incluso su deterioro, además de la pérdida de tiempo.

Es necesario tener en cuenta, que en este caso para usar el proveedor[16] de vSphere para Terraform se debe configurar adecuadamente el acceso al entorno de vSphere, incluidos permisos, credenciales, etc. A continuación, se muestra un pequeño ejemplo simplificado en el que tras conectarse a vSphere, se crea una máquina virtual indicando nombre, grupo de recursos y discos de almacenamiento

[16] Proveedor vSphere para Terraform: https://registry.terraform.io/providers/hashicorp/vsphere/latest/docs

destino, número de CPU y memoria RAM, también se indica en que red va a estar y la cantidad de disco que va a ocupar.

```
provider "vsphere" {
  user     = "your_vcenter_username"
  password = "your_vcenter_password"
  vsphere_server = "vcenter_server_address"
  allow_unverified_ssl = true
}

resource "vsphere_virtual_machine" "example_vm" {
  name             = "my-vm"
  resource_pool_id = "your_resource_pool_id"
  datastore_id     = "your_datastore_id"
  num_cpus         = 2
  memory           = 2048

  network_interface {
    network_id = "your_network_id"
  }

  disk {
    label = "disk0"
    size  = 20
  }
}
```

Por resumirlo un poco, se pretende utilizar Terraform para desplegar un clúster de Kubernetes que use imágenes Docker de las aplicaciones del proyecto Complet (incluyendo imágenes Nginx para sustituir los balanceadores) gestionado por EKSA en la infraestructura de vSphere *on-premise*. Seguidamente, realizar una prueba de campo y comprobar el buen funcionamiento en un entorno de desarrollo y así, posteriormente, migrarlo a AWS junto a la base de datos. La guía general para este primer paso de transformación en la infraestructura local quedaría de la siguiente manera:

1. **Configuración inicial**: Primero es necesario asegurarse de tener Terraform y las herramientas de EKSA instaladas en una máquina local para posteriormente crear una carpeta para el proyecto.
2. **Configuración de EKSA CLI**: Configurar EKSA CLI con la información requerida para la creación y gestión de clústeres de Kubernetes en vSphere.
3. **Crear Archivos de Configuración de Terraform**: Crear un archivo main.tf para definir la configuración de Terraform para el clúster de Kubernetes y posteriormente definir los recursos necesarios para la infraestructura vSphere y el clúster de Kubernetes. Esto incluye máquinas virtuales, redes, almacenamiento, etc.
4. **Configurar el Proveedor de vSphere en Terraform**: En el archivo main.tf, configurar el proveedor de vSphere utilizando las credenciales y detalles de conexión apropiados, como se ha visto en el ejemplo de Terraform anteriormente mostrado.
5. **Crear Nodos de Trabajo**: Usar el recurso `vsphere_virtual_machine` en Terraform para crear las máquinas virtuales que servirán como nodos de trabajo del clúster de EKSA.
6. **Planificación y Aplicación**: Resumidamente, ejecutar `'terraform init'` para inicializar el proyecto, `'terraform plan'` para verificar la configuración y `'terraform apply'` para implementar la infraestructura y configuración definida.
7. **Verificación y Gestión**: Verificar que el clúster de Kubernetes gestionado por EKSA se haya desplegado correctamente. Utilizar las herramientas de Kubernetes para gestionar, implementar y administrar aplicaciones.

En resumen, Docker, Kubernetes, EKSA y Terraform son herramientas esenciales en un proyecto *on-premise*. Docker permite la creación de entornos de desarrollo consistentes, Kubernetes ofrece orquestación de contenedores y alta disponibilidad, EKSA facilita la implementación de Kubernetes localmente, y Terraform automatiza la gestión de la infraestructura. Juntas, estas herramientas brindan un marco sólido para desarrollar, implementar y administrar aplicaciones en un entorno local de manera eficiente y confiable.

4.6 Servicios de AWS

Tras comprobar que funcione en la infraestructura *on-premise*, el siguiente paso será proceder a migrar el proyecto Complet a AWS, los servicios que se utilizarán son:

- **EKS**[17]: Previamente explicado, es el servicio administrado que se utiliza para ejecutar Kubernetes en AWS. Las ventajas de utilizarlo ya se han explicado anteriormente y la forma en la que funciona se puede resumir de la siguiente manera:
    1. Crear un clúster en EKS en la AWS Management Console (consola gráfica web).
    2. Lanzar nodos de Amazon EC2 (otro servicio de AWS) administrados o autoadministrados.
    3. Cuando el clúster esté listo, comunicarse con él mediante `kubectl`.
    4. Implementar y administrar cargas de trabajo tal y como se haría en cualquier otro entorno de Kubernetes.

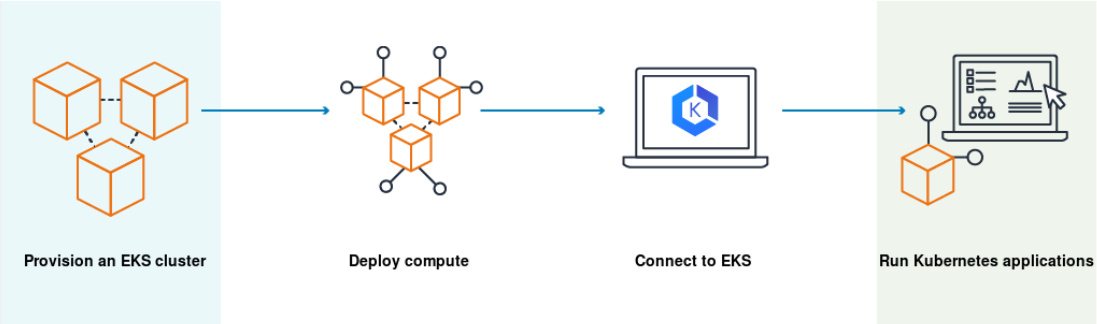


Figura 4.7 ¿Cómo funciona Amazon EKS? – Fuente: https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

Un clúster de Amazon EKS usa un plano de control y el servicio de Amazon EC2 (otro servicio de AWS) donde ejecuta los Pods. Existe documentación en la web de AWS sobre los planes y los precios, tanto del plano de control como de EC2 y que se pueden consultar para elegir cuál encaja mejor con la demanda de trabajo del proyecto a crear.

[17] Portal AWS EKS: https://docs.aws.amazon.com/es_es/eks/latest/userguide/what-is-eks.html

- ***Administrador de identidades* (IAM)[18]:** Es un servicio que especifica qué o quién puede acceder a los demás servicios y recursos de AWS y que se administra de forma centralizada. Es decir, se podrán especificar los roles o políticas para indicar qué personas y con qué permisos pueden acceder a los recursos de EKS.

- ***Virtual Private Cloud* (VPC)[19]:** El servicio VPC de Amazon permite el control sobre el entorno de redes virtuales aisladas en la nube de AWS. En estas redes se lanzan las instancias de EC2, bases de datos RDS y otros servicios que pueden albergarse en diversas zonas de disponibilidad o regiones, proporcionando protección y aislamiento de otras redes. Este servicio permite la creación de subredes, tablas de enrutamiento, redes virtuales privadas (VPN) con la red empresarial, etc. Esto es útil para separar aplicaciones o segmentar el tráfico y mantener un alto nivel de seguridad.

- ***Elastic Compute Cloud* (EC2) [20] :** EKS utiliza este servicio para crear instancias y ejecutar los nodos de trabajo del clúster. Estos nodos son máquinas virtuales que ejecutan los contenedores de las aplicaciones. Existen distintos tipos de instancias según las necesidades de CPU, memoria y almacenamiento y a través del servicio VPC se conectan a otros servicios como bases de datos.

- ***Elastic Block Store* (EBS)[21] y *Elastic File System* (EFS)[22]:** Para que EKS utilice almacenamiento persistente en los contenedores y las aplicaciones, se utiliza AWS EBS para almacenar datos a nivel de instancias y AWS EFS para almacenar datos compartidos entre nodos.

- ***Route 53*[23]:** Este servicio permite administrar registros DNS y así asignar nombres de dominio a los recursos de EKS, también se utilizará para los portales públicos del proyecto Complet que tendrán asignada una IP pública de las que dispone ODEC.

[18] Portal AWS IAM: https://aws.amazon.com/es/iam/
[19] Portal AWS VPC: https://aws.amazon.com/es/vpc/
[20] Portal AWS EC2: https://aws.amazon.com/es/ec2/
[21] Portal AWS EBS: https://aws.amazon.com/es/ebs/
[22] Portal AWS EFS: https://aws.amazon.com/es/efs/
[23] Portal AWS Route 53: https://aws.amazon.com/es/route53/

- ***CloudWatch***[24]**:** Servicio de AWS que recopila métricas y registros de clústeres, y nodos que ayudan con la monitorización y solución de problemas mediante reglas que se pueden definir.

- ***Secrets Manager***[25]**:** Se utiliza este servicio para administrar de forma segura las credenciales y secretos que las aplicaciones de los contenedores necesitan o para la renovación de certificado SSL para el portal web.

- ***Relational Database Service*** **(RDS)**[26]**:** RDS es el servicio de base de datos de AWS y permite configurar, operar o escalar las bases de datos relacionales en la nube sin preocuparte de la infraestructura. Como se decía al principio de este punto, se transformará la base de datos Oracle de la infraestructura local en ODEC a PostgreSQL en AWS gracias a este servicio.

- **AWS S3**[27]**:** S3 es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y que se utilizará para los ficheros estáticos de Complet.

- **AWS SES**[28]**:** Es un servicio de correo electrónico escalable y rentable proporcionado por AWS. Este servicio permite a los desarrolladores enviar correos electrónicos desde sus aplicaciones o servicios en la nube de manera eficiente y confiable. Este servicio se usará para el envío de correos de las encuestas de Complet utilizando un servidor simple de transferencia de correo (SMTP) que proporciona AWS y las credenciales para ello se crearán en el servicio IAM. La configuración de envío se implantará en las imágenes Docker de la aplicación.

Con esto se han explicado los principales servicios que se utilizarán en AWS para realizar la migración del proyecto Complet desde la infraestructura de ODEC a la nube de AWS.

[24] Portal AWS CloudWatch: https://aws.amazon.com/es/cloudwatch/
[25] Portal AWS Secrets Manager: https://aws.amazon.com/es/secrets-manager/
[26] Portal AWS RDS: https://aws.amazon.com/es/rds/
[27] Portal AWS S3: https://aws.amazon.com/es/s3/
[28] Portal AWS SES: https://aws.amazon.com/es/ses/

Resumidamente, el despliegue en AWS sería el siguiente:

1. Configurar un rol de IAM con los permisos necesarios para utilizar EKS y añadir políticas para que se comunique con lo demás servicios como EC2, EBS, RDS, S3, etc.
2. Configurar una VPC (en el caso de ODEC ya tenemos una definida).
3. Configurar las subredes necesarias y sus zonas de disponibilidad en la VPC junto a sus reglas de enrutamiento y la seguridad que se desee. Diferenciar entre subred pública para que los nodos de EKS tengan acceso a Internet y subred privada para gestionarlos de manera controlada.
4. Utilizar Terraform para crear un clúster EKS.
5. Configurar los nodos para que se ejecuten en las subredes adecuadas y tengan acceso a los servicios requeridos. Se puede configurar el tamaño de los recursos según las necesidades de los nodos.
6. Configurar el almacenamiento en los servicios EBS para almacenamiento persistente en los Pods y el servicio EFS para sistemas de archivos compartidos.
7. Configurar Route 53 para la gestión de nombres de dominio como el portal web y de los servicios de AWS que se utilicen. Hace falta asegurarse de que las políticas y los permisos sean los adecuados.
8. Crear y usar un bucket del servicio S3 para almacenar los archivos estáticos. También es necesario configurar los permisos adecuados para permitir el acceso público o no según las necesidades.
9. Implementar las aplicaciones en el clúster utilizando `kubectl` para desplegar las aplicaciones en los nodos de EKS.
10. Configurar el escalado y la administración utilizando las capacidades de las que dispone EKS para ajustar el número de nodos según la carga de trabajo. Monitorizar estas métricas con CloudWatch.

5. Implementación

El proceso de migración del proyecto Complet a la infraestructura de AWS marca un punto crucial en este trabajo. En este capítulo se detalla el riguroso trabajo de implementación, donde se trasladarán las aplicaciones, datos y recursos a la nube de AWS. La planificación estratégica diferenciará entre dos implementaciones como hemos comentando anteriormente, la implementación *on-premise* en la que utilizando EKSA y Terraform en nuestra infraestructura de vSphere desplegaremos un clúster de kubernetes que use imágenes Docker de las aplicaciones del proyecto y tras corroborar que su funcionamiento es correcto y realizar algunas pruebas se procederá al siguiente paso. La segunda implementación se trata de la migración propiamente dicha del proyecto Complet a la infraestructura de AWS junto a la puesta en marcha de todos los servicios complementarios necesarios para que el portal funcione correctamente.

5.1 Implementación *on-premise*

Este paso previo sirve para confirmar que el paso del proyecto a Kubernetes funciona correctamente, por eso vamos a hacer primero el despliegue en la propia infraestructura local de ODEC. La revisión de los pasos sería:

- Despliegue del clúster de Kubernetes con Terraform:
  - Utilizar Terraform para configurar el despliegue de un clúster de Kubernetes en la infraestructura local de ODEC.
- Gestión con EKSA en vSphere *on-premise*:
  - Utilizar EKSA para administrar y gestionar el clúster de Kubernetes en la infraestructura de vSphere *on-premise*.
  - Verificar la correcta configuración y el funcionamiento óptimo del clúster con las imágenes de las aplicaciones.
  - Implementar imágenes Docker de las aplicaciones del proyecto Complet, incluyendo Nginx para reemplazar balanceadores, dentro de este clúster.
- Prueba de campo en entorno de desarrollo:

    - Realizar pruebas exhaustivas en un entorno de desarrollo para garantizar el buen funcionamiento de las aplicaciones desplegadas en el clúster de Kubernetes.
    - Comprobar la interacción entre las diferentes aplicaciones, la conectividad y la estabilidad del sistema en este entorno.
- Evaluación y preparación para la migración a AWS:
    - Una vez se confirme el correcto funcionamiento en el entorno local, preparar el plan detallado para la migración a AWS, incluyendo la base de datos del proyecto Complet.

Esta fase inicial servirá como validación y preparación crucial antes de la migración a AWS, asegurando que las aplicaciones estén correctamente desplegadas y funcionando de manera óptima en un entorno controlado antes de la transición a la nube. La implementación quedaría:

1. **Configuración inicial**: El primer paso es asegurarse de tener Terraform y las herramientas de EKSA instaladas en una máquina local, por lo que es necesario descargarse la última versión de Terraform desde su web https://www.terraform.io/downloads.html, extraer el binario en el directorio deseado y añadir el directorio a la variable de entorno PATH para poder ejecutar Terraform desde cualquier ubicación de la máquina. Tras esto instalaremos las herramientas de EKSA desde https://docs.aws.amazon.com/es_es/eks/latest/userguide/getting-started-eksctl.html donde está la documentación oficial y nos puede resultar útil. Para verificar que tenemos las dos herramientas bien instaladas podemos ejecutar los comandos `terraform version` y `eksctl version`. Por último, tenemos que crear un directorio donde mantener todos los archivos del proyecto.
2. **Configuración de EKSA CLI**: Es necesario configurar EKSA CLI en el propio vSphere, para ello es necesario disponer de un usuario administrador que cree un usuario con unos roles definidos y los permisos necesarios para crear clústeres, este nuevo usuario es el que se encargará de la gestión las máquinas virtuales después de que Terraform cree y aprovisiones los recursos de vSphere. Una vez que el clúster está en funcionamiento, se puede usar EKSA para administrar sus operaciones. Esto incluye tareas como escalar nodos, y la monitorización del estado del clúster, y gestión de usuarios y permisos. La URL

https://anywhere.eks.amazonaws.com/docs/getting-started/vsphere/ explica el proceso muy claramente de los requerimientos necesarios para utilizar EKSA en vSphere, cómo preparar vSphere para usar EKSA que son las partes que nos interesan.

3. **Crear archivos de configuración de Terraform**: En este punto es necesario crear varios archivos **main.tf** que defina la configuración de Terraform y posteriormente definir los recursos necesarios para la infraestructura vSphere y el clúster de Kubernetes. Esto incluye máquinas virtuales, redes, almacenamiento, etc. Aquí hemos diferenciado 4 secciones, cada una con su main.tf, variables y otros recursos necesarios, como se muestra en la **figura 5.1**.
    a. **Máquinas:** Indica las especificaciones técnicas para la creación de las máquinas virtuales.
    b. **Clúster:** Se define y especifica el número de *masters* y *workers* y sus IP.
    c. **Infraestructura:** Se crea la infraestructura para su monitorización, sus volúmenes estáticos y balanceadores.
    d. **Aplicación:** Se especifica qué tarea realizará cada *worker*, si será motor de ejecución, diseñador de entrevistas o aplicación de gestión.

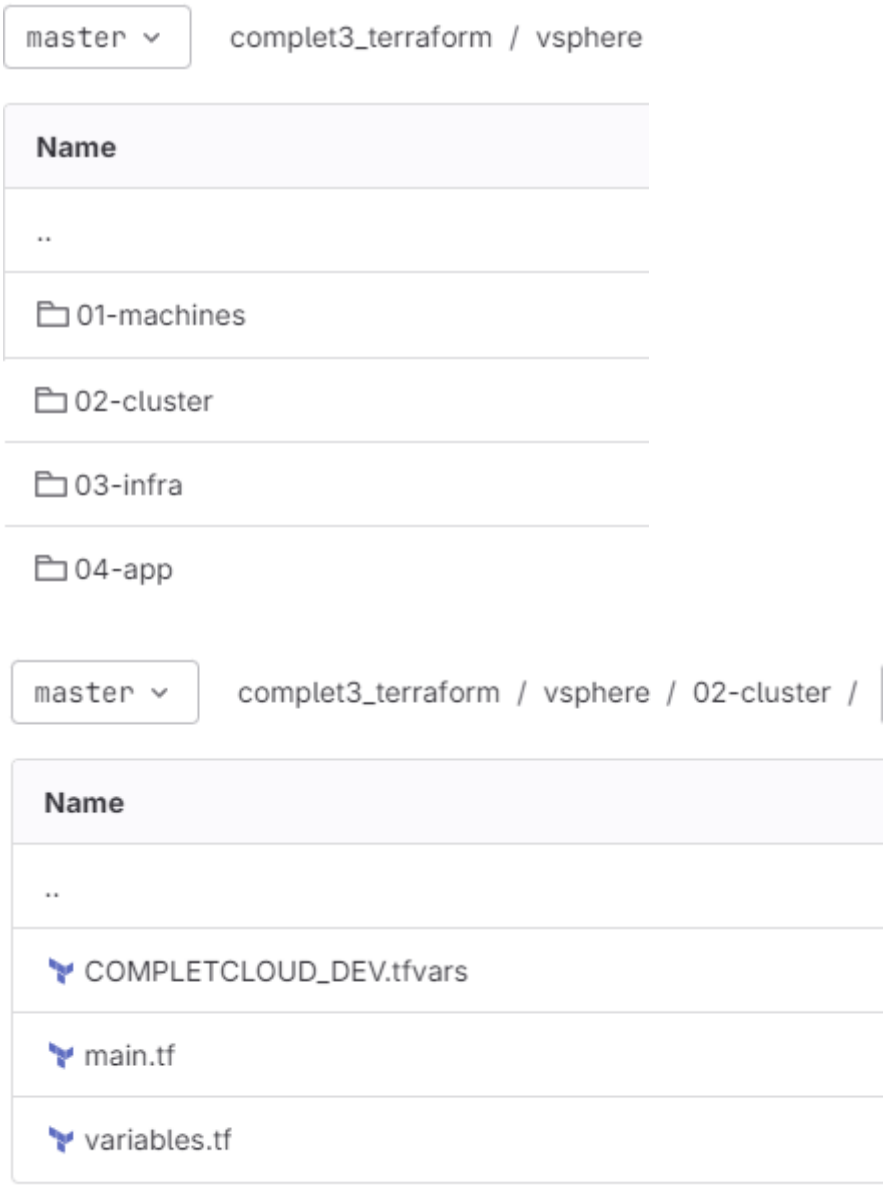


Figura 5.1 Elementos de Terraform del despliegue de Complet.

4. **Planificación y aplicación**: Para realizar la planificación y aplicación utilizando Terraform hay que seguir los siguientes pasos:
    a. **Terraform init**: Este comando inicializa tu entorno de trabajo con Terraform. Se ejecuta en el directorio que contiene tus archivos de configuración de Terraform, es decir, donde se encuentran los *main.tf*. Al ejecutar este comando, Terraform descarga los plugins necesarios para la ejecución de los providers que se han definido, en este caso vSphere, y prepara el estado de Terraform.
    b. **Terraform plan**: Este comando permite ver qué cambios Terraform realizará en la infraestructura sin aplicarlos. Es útil para revisar que

todo lo que se va a modificar es realmente lo que se pretende hacer. Muestra todas las acciones, tanto de creación, de modificación o de destrucción.

c. **Terraform apply**: Este comando aplica los cambios necesarios para alcanzar el estado deseado de la configuración, es decir, lleva a cabo las acciones determinadas en el paso de planificación. Al ejecutarlo, Terraform pide que se confirmen los cambios a aplicar.

5. **Verificación y gestión**: En este último paso es necesario verificar que el clúster de Kubernetes gestionado por EKSA se haya desplegado correctamente desde el vSphere, comprobando que las máquinas virtuales se han creado y que su funcionamiento es el correcto. Utilizaremos las herramientas de EKSA para gestionar, implementar y administrar aplicaciones.

Es necesario recordar que aquí no están todos los ficheros utilizados para la migración a Kubernetes en vSphere, ya que hay muchos datos sensibles como pueden ser secretos de AWS (ya que utilizamos EKSA), claves RSA, dominios internos o código propio de las aplicaciones, por lo que se ha optado por explicar los puntos generales de la migración *on-premise* sin poner en riesgo datos de la empresa o de terceros.

En los ficheros anexos, dentro del directorio *Complet_vsphere_terraform* podemos encontrar algunos *main.tf* que la empresa permite mostrarnos.

5.2 Implementación *en AWS*

El proceso de migración del proyecto Complet a la infraestructura de AWS implica el uso de herramientas tanto de automatización, como de servicios en AWS, algunos ya configurados previamente.

- Utilización de Terraform para automatizar procesos:
  Se empleará Terraform para orquestar y automatizar parte del proceso de migración. Esto abarcará la configuración y despliegue de recursos necesarios

en AWS, asegurando una implementación consistente y reproducible. Se definirán y gestionarán aspectos clave, como la infraestructura base, redes, seguridad, creación de la base datos y otros elementos necesarios para la migración del proyecto.

- Integración de servicios AWS preconfigurados:
  Se hará uso de los servicios de AWS que han sido previamente configurados, algunos de los cuales ya estaban en funcionamiento antes de la migración. Estos servicios preexistentes, como por ejemplo Amazon VPC para las redes y subredes, se integrarán en la migración para garantizar una transición fluida y sin interrupciones en las operaciones del proyecto.

- Adaptación de recursos a la nueva infraestructura:
  Se llevará a cabo la reconfiguración de los recursos y servicios del proyecto Complet para adaptarlos a la estructura y particularidades del entorno AWS. Se prestará especial atención a la optimización de recursos, ajustes de red y la gestión de la escalabilidad para aprovechar al máximo las capacidades y ventajas de la infraestructura en la nube.

- Verificación y validación post-migración:
  Una vez completada la migración, se realizarán exhaustivas pruebas de verificación para asegurar el correcto funcionamiento de las aplicaciones, la interacción entre los servicios y la integridad de los datos en el entorno de AWS. Se validarán las funcionalidades y se garantizará que todos los servicios estén operativos según lo planeado.

  Este proceso detallado de migración abarca tanto la automatización a través de Terraform como la integración cuidadosa de los servicios existentes en la infraestructura de AWS, todo con el objetivo de lograr una transición eficiente y exitosa del proyecto Complet a la nube de Amazon. Los puntos se detallan a continuación y en la **Figura 5.2.**

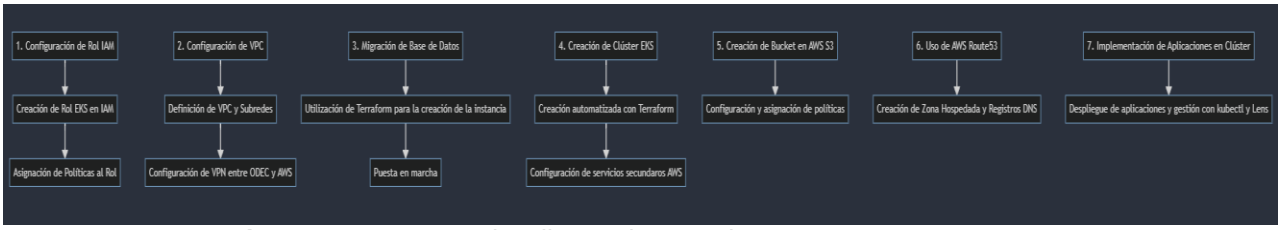


Figura 5.2 Esquema despliegue de Complet en AWS.

1. El primer paso es la configuración de un rol de IAM con los permisos esenciales para utilizar Amazon EKS, garantizando su interacción con otros servicios, como EC2, entre otros. Se accede al panel de IAM en la consola de AWS para crear un nuevo rol y se elige ``EKS´´ como el servicio que utilizará este rol.

   Posteriormente, se asignan las políticas necesarias al rol. Esto incluye políticas como AmazonEKSClusterPolicy, que permite la gestión de clústeres EKS, y AmazonEKSServicePolicy, que autoriza a los nodos de trabajo a unirse al clúster. Además, se asignan políticas específicas para otros servicios, como AmazonEC2FullAccess, AmazonRDSFullAccess, AmazonS3FullAccess, adaptándose a las necesidades particulares del entorno. Este rol también se asigna a los nodos de trabajo EC2 cuando EKS despliega las instancias, asegurando que estos nodos cuenten con los permisos necesarios para interactuar con otros servicios.

   Es fundamental recalcar que este punto debe ser refinado continuamente, buscando otorgar a los usuarios con este rol únicamente los permisos mínimos esenciales para sus operaciones, garantizando así la seguridad y la eficiencia en el uso de los recursos de la nube.

2. El segundo paso implica la configuración de una VPC, aunque en este caso ya está preexistente. No obstante, se proporcionará una explicación general del proceso y se presentará la configuración actual de la VPC en ODEC.

   Al acceder al panel de VPC en AWS, se inicia la creación proporcionando un nombre y definiendo un enrutamiento entre dominios sin clases (CIDR), que establece el rango de direcciones IP de la red. Luego, se definen subredes, tanto públicas como privadas, con la necesidad de configurar tablas de enrutamiento y puertas de enlace para permitir la comunicación de instancias en las subredes públicas con Internet. Los grupos de

seguridad se utilizan para controlar el tráfico de red hacia y desde las instancias. Además, se establece una VPN entre las subredes locales de ODEC y las subredes de AWS para permitir la comunicación entre ambas infraestructuras.

En la configuración actual de ODEC, representada en la **Figura 5.3**, se observa una VPC con dos CIDR, uno de los cuales está dedicado al proyecto Complet. Además, existen tablas de enrutamiento y puertas de enlace para la comunicación con Internet y las redes de la infraestructura local. Se utiliza una Elastic IP para asignar una dirección IP estática a una puerta de enlace NAT, permitiendo a las instancias de subredes privadas conectarse a servicios fuera de la VPC sin que estos servicios externos inicien conexiones con esas instancias. Por último, se ha establecido una VPN entre AWS y la red de ODEC para facilitar la comunicación segura entre ambas infraestructuras.

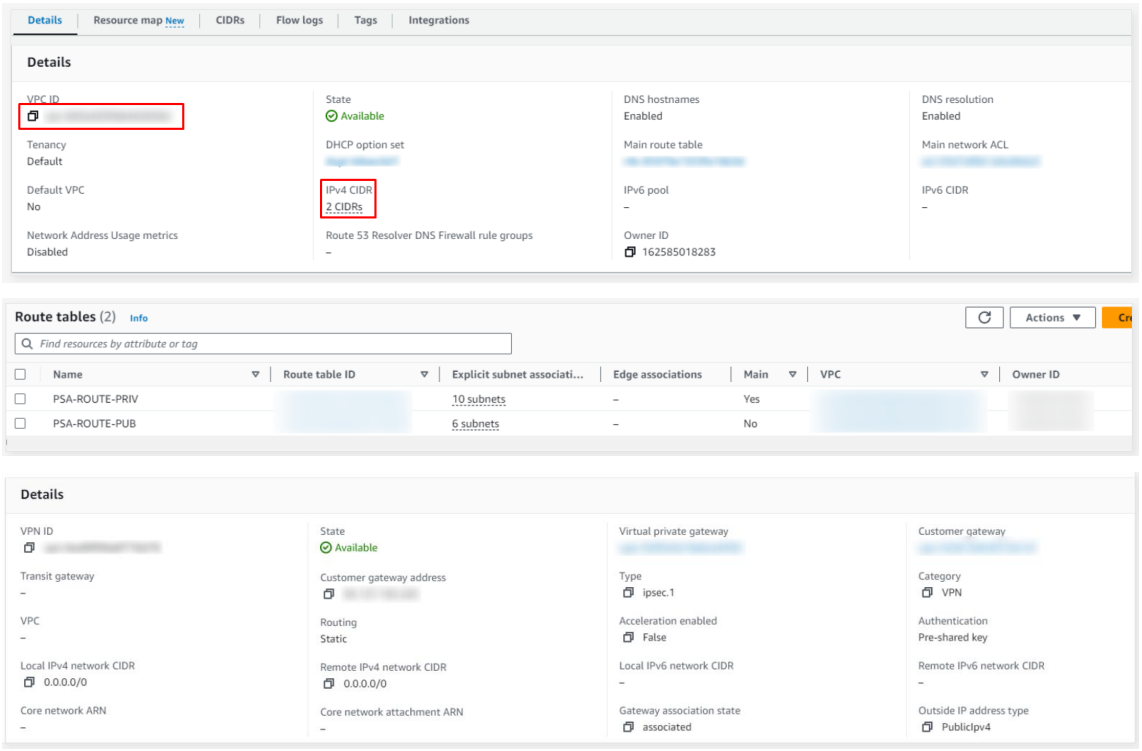


Figura 5.3 VPC, CIDR, tablas de enrutamiento y VPN en AWS.

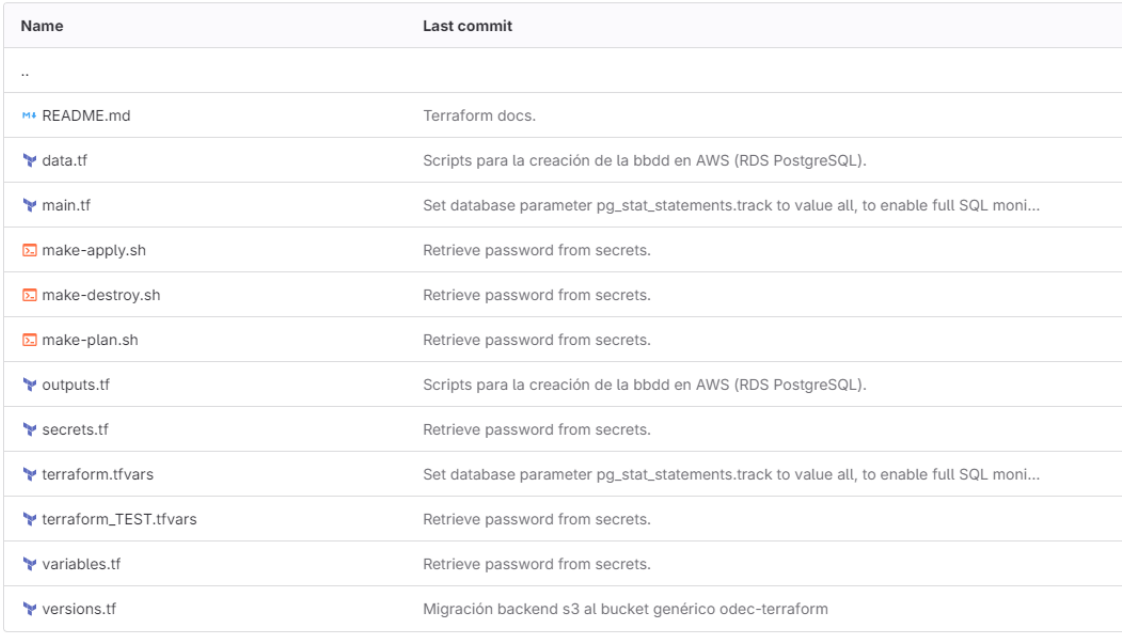
3. A continuación, nos adentraremos en la etapa de migración de la base de datos, como previamente se expuso en la sección 4.6 y se detallaron los servicios de AWS que se iban a utilizar para este propósito. Específicamente, en el contexto del servicio RDS, se delineó el plan para transformar la base de datos local Oracle a una base de datos PostgreSQL.

   El proceso de migración se ha automatizado mediante el uso de Terraform, lo que ha facilitado su ejecución y control. Se ofrecerá un vistazo general a este proceso, describiendo de manera resumida los pasos y la secuencia de acciones realizadas para llevar a cabo la conversión de la base de datos Oracle a PostgreSQL en el entorno de AWS.

   En el repositorio local de Git, específicamente en la **Figura 5.4**, se almacena el código de Terraform empleado para la creación de la base de datos. Este repositorio contiene múltiples archivos Terraform destinados a proveer variables, gestionar secretos y otros parámetros relevantes. Al final, el archivo principal *main.tf* coordina y ejecuta todo el proceso.

   Se puede acceder parcialmente a este código dentro del directorio denominado *ddbb_migration* que forma parte de los archivos anexos. Este

directorio contiene los archivos Terraform que encapsulan la configuración y la lógica necesaria para la migración de la base de datos.



Name	Last commit
..	
README.md	Terraform docs.
data.tf	Scripts para la creación de la bbdd en AWS (RDS PostgreSQL).
main.tf	Set database parameter pg_stat_statements.track to value all, to enable full SQL moni...
make-apply.sh	Retrieve password from secrets.
make-destroy.sh	Retrieve password from secrets.
make-plan.sh	Retrieve password from secrets.
outputs.tf	Scripts para la creación de la bbdd en AWS (RDS PostgreSQL).
secrets.tf	Retrieve password from secrets.
terraform.tfvars	Set database parameter pg_stat_statements.track to value all, to enable full SQL moni...
terraform_TEST.tfvars	Retrieve password from secrets.
variables.tf	Retrieve password from secrets.
versions.tf	Migración backend s3 al bucket genérico odec-terraform

Figura 5.4 Código de Terraform para la creación de la base de datos en AWS.

El siguiente pequeño código nos muestra como la base de datos se genera en la zona *eu-west3(Paris)*, se le asigna la VPC y las subredes deseadas, el tipo de instancia, almacenamiento, etc.

```
aws_region      = "eu-west-3"
platform_name   = "complet"
stage           = "pro"
aws_secret_name = "odec-pro-complet-project-secret"
# Networking
aws_vpc_name_tag                  = "PSA-VPC"
aws_subnets_public_name_tag_list  = ["PSA-SN-PUB-A", "PSA-SN-
PUB-B", "PSA-SN-PUB-C"]
aws_subnets_private_name_tag_list = ["PSA-SN-PRIV-A", "PSA-SN-
PRIV-B", "PSA-SN-PRIV-C"]
# DB properties
db_engine_version          = "13.7"
db_instance_class          = "db.m6g.xlarge"
db_username                = "complet_admin"
db_name                    = "postgres"
db_backup_retention_period = 7
db_storage_gb              = 20
db_max_storage_gb          = 2000
```

```
db_perf_insights_enabled   = true
db_deletion_protection     = true
# Tags
default_tags = {
  Proyecto = "complet"
  Env      = "pro"
}
```

Una vez establecida la infraestructura de la base de datos, el responsable del área procedió con la inicialización del sistema, ajustando los parámetros y configuraciones críticas para garantizar un entorno óptimo de ejecución. Asimismo, se encargó de la creación de usuarios con privilegios restringidos, asegurando la seguridad y el control de accesos a la base de datos.

En el proceso de migración de Oracle a PostgreSQL, se ha seguido la documentación de referencia proporcionada por la comunidad de PostgreSQL (disponible en el enlace: https://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion). Esta documentación ha sido fundamental para abordar las diferencias substanciales entre ambas plataformas, considerando variaciones en funciones, consultas y estructuras de datos. Estos ajustes se han realizado de manera minuciosa para asegurar la coherencia y funcionalidad óptima tras la conversión de la base de datos.

4. El cuarto paso implica la creación del clúster EKS en AWS, lo que engloba una serie de configuraciones detalladas que incluyen la configuración de los nodos, sus subredes, accesos a servicios, roles, recursos, almacenamiento persistente (utilizando AWS EBS) y compartido (mediante AWS EFS), balanceadores de carga, certificados, políticas de autoescalado, entre otros aspectos. Este proceso ha sido automatizado a través de Terraform, permitiendo una implementación consistente y escalable.

   En el repositorio de Git de ODEC, se puede encontrar una recopilación de todos los archivos de configuración de Terraform utilizados para la creación del clúster EKS. La **Figura 5.5** muestra la estructura y los componentes de estos archivos, los cuales abarcan la definición de recursos, la configuración de la red, la asignación de roles, las políticas de escalado, así como la provisión de almacenamiento tanto persistente como compartido,

entre otros aspectos relevantes para la creación y configuración integral del entorno del clúster EKS en AWS.

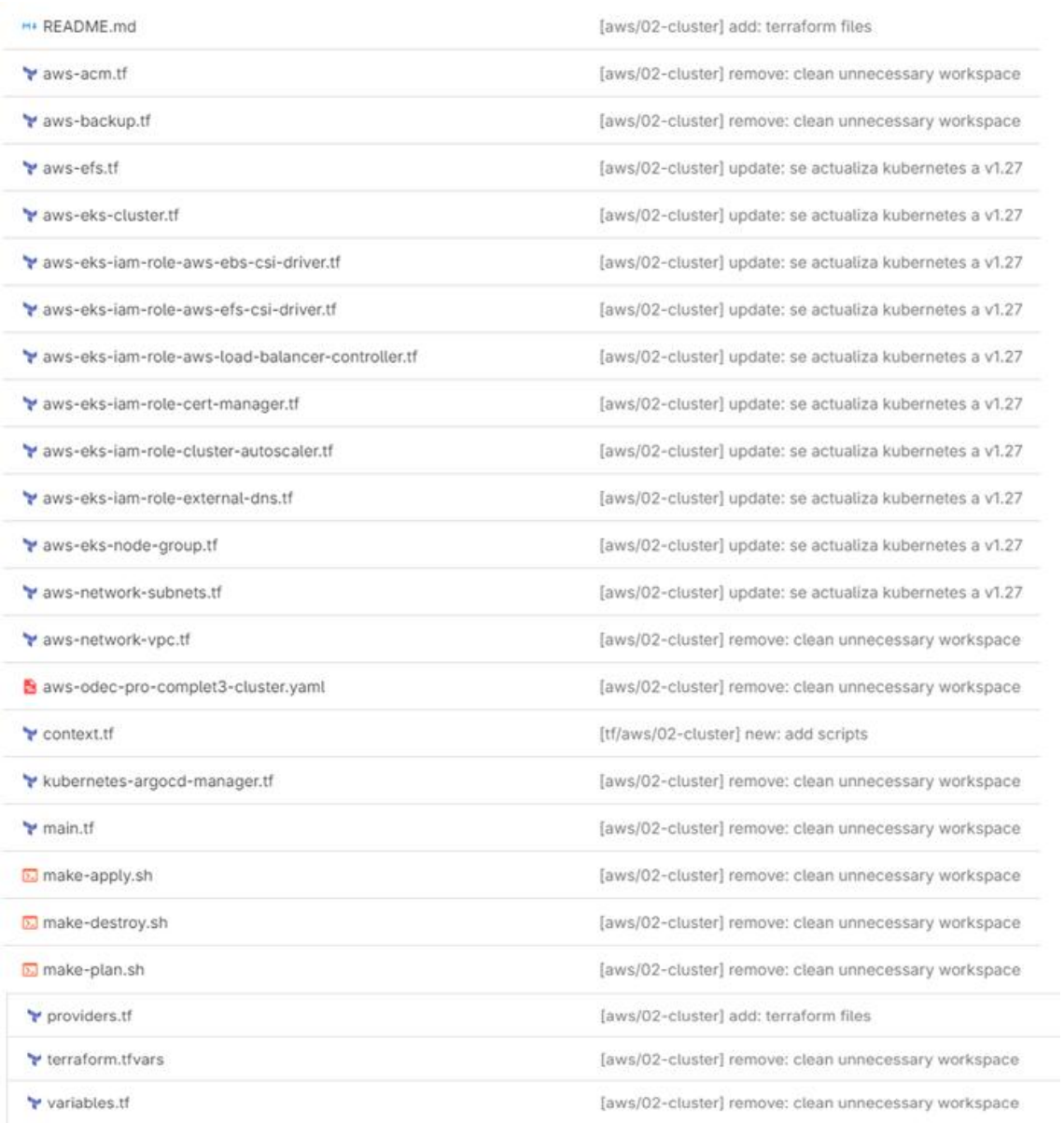


Figura 5.5 Código creación clúster en AWS.

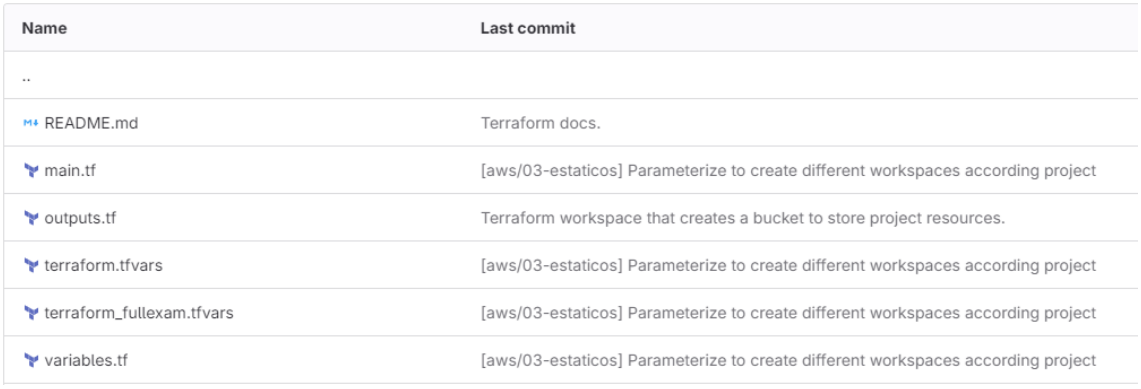
Después de la creacíon del clúster se genera un archivo llamado *aws-odec-pro-complet3-cluster.yaml*, éste es un fichero *kubeconfig* que *kubectl* utiliza para obtener la configuración de acceso al clúster de kubernetes.

También podemos ejecutar el siguiente comando para actualizarlo en nuestra máquina local.

```
aws eks --region eu-west-3 update-kubeconfig --name odec-pro-complet3-cluster
```

En el directorio *Complet_aws_migration* de los archivos anexos podemos ver parte del código anteriormente mencionado.

5. El siguiente punto fundamental implica la creación de un bucket utilizando el servicio AWS S3 para almacenar los archivos estáticos empleados por la aplicación Complet. Al igual que en los pasos anteriores, este proceso se ha automatizado utilizando Terraform, lo cual se refleja en la **Figura 5.6**.
   En resumen, el proceso consiste en la creación de un repositorio en AWS S3, asignándole un nombre específico y configurando los permisos necesarios. Parte de esta configuración implica definir niveles de acceso, incluyendo la habilitación de una porción para acceso público. Este enfoque detallado y automatizado asegura la correcta creación del bucket, garantizando la accesibilidad y gestión eficiente de los archivos estáticos empleados por la aplicación Complet.



Name	Last commit
..	
README.md	Terraform docs.
main.tf	[aws/03-estaticos] Parameterize to create different workspaces according project
outputs.tf	Terraform workspace that creates a bucket to store project resources.
terraform.tfvars	[aws/03-estaticos] Parameterize to create different workspaces according project
terraform_fullexam.tfvars	[aws/03-estaticos] Parameterize to create different workspaces according project
variables.tf	[aws/03-estaticos] Parameterize to create different workspaces according project

Figura 5.6 Código de creación del repositorio de estáticos en AWS S3.

Al igual que en lo demás puntos, hay un directorio en los archivos anexos llamado *Complet_S3_fileserver* dónde podemos ver parte del código.

6. Una de las últimas tareas cruciales es el empleo del servicio AWS Route53 para la creación de un dominio público que permitirá establecer un registro DNS para el portal Complet. Podemos ver el resultado en la **Figura 5.7**.

   Para llevar a cabo esta tarea, es primordial crear una ``zona hospedada´´ en Route53 con el dominio deseado. Dentro de esta zona, se definen registros de tipo A y CNAME, fundamentales para direccionar el tráfico hacia los pods Nginx alojados en el clúster de Complet. Cada tipo de registro DNS cumple funciones específicas, permitiendo redirigir el tráfico de acuerdo a las necesidades precisas del proyecto.

   Este paso reviste una importancia fundamental, ya que otorga visibilidad a nuestro proyecto hacia Internet. Al establecer los registros DNS en la zona hospedada, se habilita la accesibilidad del portal Complet desde la web, asegurando la correcta redirección del tráfico hacia los servicios y pods pertinentes alojados en el clúster. Este nivel de visibilidad resulta fundamental para la interacción y accesibilidad del proyecto desde el entorno público de Internet.

   En el contexto del proyecto Complet, a medida que experimenta un crecimiento en su base de clientes y, consecuentemente, aumenta la necesidad de enviar un mayor volumen de correos electrónicos personalizados, surge la importancia de gestionar adecuadamente las identidades de dominio asociadas al servicio Amazon SES de AWS.

   Para garantizar la entrega efectiva y confiable de los correos electrónicos, es esencial verificar las identidades de dominio que se darán de alta en Amazon SES. Este proceso implica la adición de una identidad desde el panel de ``identidades verificadas´´ de Amazon SES, seguido por la generación de registros CNAME asociados a dicha identidad.

   El procedimiento de verificación de identidades de dominio se inicia al añadir la identidad de dominio deseada desde la interfaz proporcionada por Amazon SES. Posteriormente, el sistema genera registros CNAME específicos que deben ser agregados a los proveedores de servicios DNS propietarios del dominio en cuestión.

Estos registros CNAME, al ser configurados en los servidores DNS del proveedor, actúan como mecanismos de verificación que establecen la validez y la autoridad del propietario del dominio para enviar correos electrónicos en nombre de dicho dominio a través de Amazon SES.

Este proceso no solo asegura la autenticidad del remitente, sino que también contribuye a mejorar la tasa de entrega y la reputación del remitente, aspectos críticos en entornos de envío de correos electrónicos a gran escala. Además, al validar las identidades de dominio, se fortalece la capacidad del proyecto Complet para gestionar eficientemente las comunicaciones por correo electrónico, especialmente en situaciones donde la personalización y la escalabilidad son requisitos fundamentales.

En resumen, la verificación de identidades de dominio en Amazon SES se erige como un paso estratégico y necesario a medida que el proyecto Complet expande su base de clientes y aumenta sus operaciones de envío de correos electrónicos, proporcionando una base sólida para la entrega segura y efectiva de mensajes personalizados.

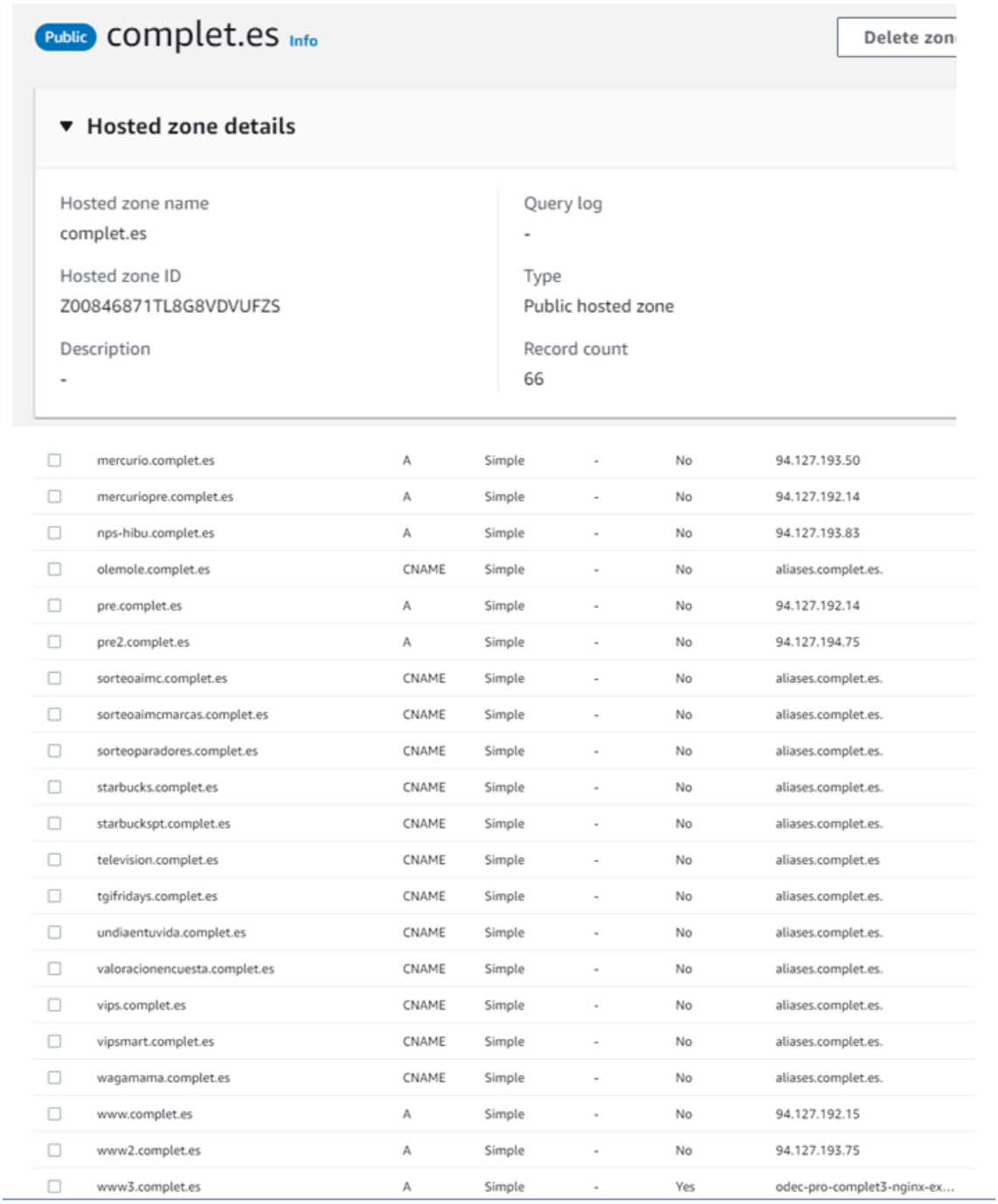


Figura 5.7 Dominio complet.es y algunos registros DNS en AWS Route53.

7. Para concluir, una vez establecido el clúster con todos sus recursos y servicios necesarios, se procede a la implementación de las aplicaciones en el clúster utilizando kubectl. Este proceso implica desplegar las aplicaciones en los nodos de AWS EKS, asegurando su correcto ciclo de vida en la infraestructura de los nodos. Esto incluye la definición de distintos tipos de pods y sus funciones, lo que es esencial para la gestión operativa y el despliegue de las aplicaciones en el entorno del clúster.

En ODEC, gracias al archivo kubeconfig, se gestiona el clúster a través de una interfaz gráfica mediante un programa llamado Lens[29].

La **Figura 5.8** refleja la visualización del clúster desde AWS EKS, donde se observan diversos tipos de recursos:

- Workloads: Aquí se encuentran los jobs, pods, despliegues, y otros elementos relacionados con las cargas de trabajo y su ejecución.
- Cluster: Muestra los nodos del clúster, el tipo de instancia EC2 que se utiliza y sus roles específicos en el sistema.
- Service and Networking: Detalla los servicios utilizados por los pods, sus direcciones IP, entre otros aspectos relacionados con la red y la comunicación entre servicios.
- Config and Secrets: Se encarga de la gestión de la información sensible, como contraseñas, tokens, claves SSH, y otros datos críticos.
- Storage: Hace referencia al almacenamiento, incluyendo volúmenes persistentes y otros recursos de almacenamiento utilizados por las aplicaciones.
- Authentication: Aquí se definen las cuentas de servicio que proveen identificación a los procesos que ejecutan los pods.
- Policy: Se especifican y gestionan políticas y cuotas para diferentes aspectos y recursos del clúster.

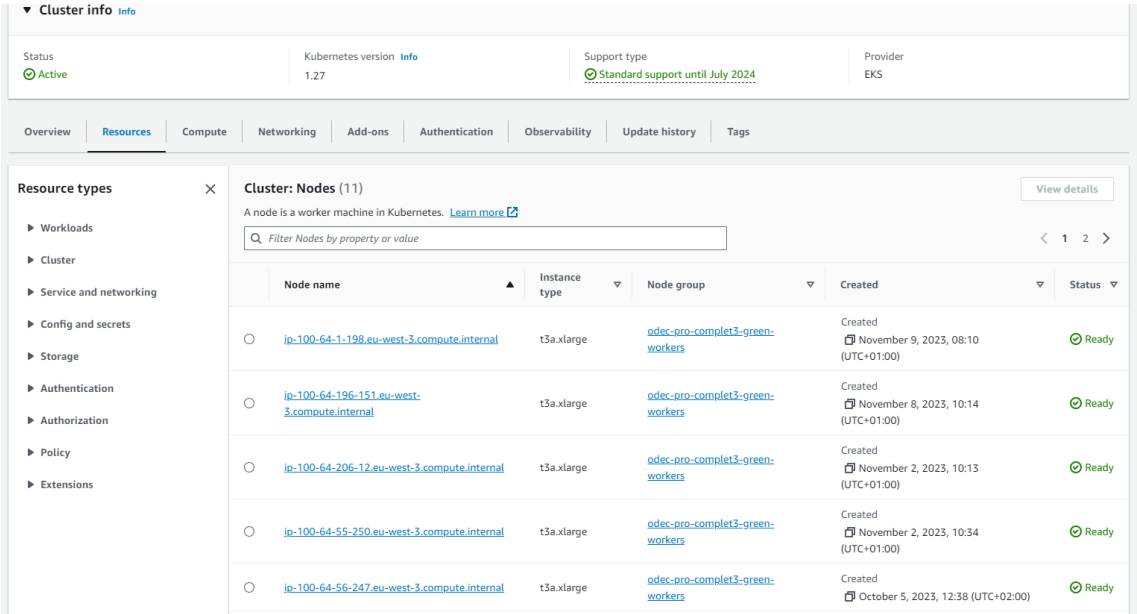


Figura 5.8 Recursos del clúster Complet en AWS EKS.

[29] Web Lens: https://k8slens.dev/

Tras la implementación exitosa del proyecto en la infraestructura de Amazon Web Services (AWS), se anticipa un análisis exhaustivo de los resultados en el próximo capítulo. Este análisis abarcará no solo la implementación en la nube, sino también la infraestructura *on-premise* existente.

6. Evaluación

En el presente capítulo, se abordará la fase crítica de evaluación posterior a la migración de nuestra aplicación desde un entorno *on-premise* hacia la infraestructura de AWS. Este análisis exhaustivo tiene como objetivo principal arrojar luz sobre el impacto y los resultados obtenidos tras el proceso de migración, proporcionando una visión integral de los beneficios, desafíos y oportunidades que han surgido a lo largo de este viaje tecnológico.

La migración de una aplicación empresarial hacia la nube representa un hito crucial en este trabajo de fin de grado, ya que implica un cambio fundamental en la gestión y entrega de servicios. Este capítulo se enfocará en la evaluación de factores críticos, tales como el rendimiento operativo, la escalabilidad, la seguridad y la alineación con los objetivos estratégicos de la organización, considerados como pilares fundamentales en la transición hacia la nube.

Para evaluar estos cuatro parámetros, se han identificado medidas específicas que permitirán medir de manera integral el impacto de la migración:

1. **Rendimiento Operativo:**

    - Se enfoca en la optimización significativa de los procesos de despliegue como elemento fundamental. Las pruebas incluirán mediciones de tiempo de respuesta, eficiencia en la entrega de servicios y minimización de tiempos de inactividad.

2. **Escalabilidad:**

    - El enfoque principal para medir la escalabilidad se centra en la escalabilidad horizontal. Se llevarán a cabo pruebas específicas que evaluarán la capacidad del sistema para escalar de manera efectiva con el aumento de la carga, utilizando clústeres y nodos adicionales como métricas de análisis y su tiempo de creación.

3. **Seguridad:**

    - El enfoque principal en el análisis de seguridad se centra en la facilidad de monitoreo y recolección de métricas del clúster para saber su estado. Se realizará una evaluación exhaustiva para determinar la

eficacia y facilidad con la que el equipo de seguridad puede supervisar el sistema, recolectar métricas relevantes y detectar posibles amenazas o vulnerabilidades en el entorno del clúster durante la migración a la nube.

4. **Alineación con Objetivos Estratégicos:**

    - Se centrará en la considerable autonomía de los equipos de desarrollo como indicador clave. Se evaluará cómo esta autonomía se alinea con los objetivos estratégicos de la organización, considerando la capacidad de los equipos para adaptarse y responder ágilmente a las necesidades del negocio.

Estas medidas específicas permitirán un análisis detallado y exhaustivo de los impactos de la migración a la nube en cada uno de los parámetros críticos identificados, proporcionando así una visión global de los efectos de este proceso en la organización.

Puntos a favor del despliegue *on-premise* kubernetizado y el despliegue en AWS kubernetizado con respecto al proyecto Complet original:

- **Optimización significativa en los procesos de despliegue**: Las pruebas incorporaron mediciones del tiempo de respuesta, el cual se ha evaluado mediante el registro de todos los despliegues a través de incidencias en la plataforma Jira[30]. Esta herramienta registra detalladamente los pasos ejecutados y el tiempo utilizado, destacándose como el factor más influyente en la medición del rendimiento operativo. Se destaca como un factor altamente positivo la notable disminución en los tiempos de despliegue de aplicaciones. Este aspecto crucial se ha traducido en una gestión más eficiente, simplificada y controlada de los procesos de implementación.

  La adopción de Kubernetes como plataforma orquestadora ha permitido optimizar significativamente la manera en que las aplicaciones son desplegadas y gestionadas en comparación con los métodos tradicionales. La infraestructura basada en contenedores facilita la portabilidad y escalabilidad, pero uno de los

[30] Web Jira: https://www.atlassian.com/es/software/jira

beneficios más tangibles se refleja en la agilidad y rapidez con la que las aplicaciones pueden ser implementadas en el entorno de producción.

Este aumento en la velocidad de despliegue no solo se traduce en una respuesta más ágil a las demandas del mercado, sino que también ofrece un mayor control sobre los procesos, facilitando la identificación y corrección proactiva de posibles problemas. La simplicidad añadida a los flujos de trabajo de despliegue contribuye no solo a una mayor eficiencia operativa, sino también a una reducción significativa de los posibles errores asociados a implementaciones manuales.

En resumen, la migración a Kubernetes ha demostrado ser un catalizador clave para mejorar la agilidad y eficacia en los procesos de despliegue de aplicaciones, lo que a su vez impacta positivamente en la capacidad de adaptación y competitividad de la infraestructura tecnológica de la organización.

Antes de la migración a Kubernetes, la aplicación en cuestión operaba en un entorno compuesto por 20 máquinas virtuales, cada una ejecutando instancias de JBoss, distribuidas y agrupadas según su función específica. Adicionalmente, existía una máquina orquestadora encargada de desplegar los archivos .war en los grupos de aplicación correspondientes.

Tras la exitosa migración a Kubernetes, se ha observado una mejora sustancial en los tiempos de despliegue de la aplicación. Antes del cambio, el proceso de implementación y distribución de los archivos .war era un procedimiento manual, sujeto a posibles errores y a una mayor complejidad operativa. Con la transición a Kubernetes, se introdujo una arquitectura basada en contenedores que transformó radicalmente este escenario.

Los resultados de las incidencias en Jira revelan que el tiempo promedio de despliegue de una nueva versión de la aplicación se redujo en un 40% como podemos ver en la **Figura 6.1** (tiempo en minutos). Anteriormente, este proceso implicaba una serie de pasos manuales, desde la preparación de las máquinas virtuales hasta la distribución de los archivos de aplicación. Con Kubernetes, la orquestación automatizada de contenedores ha simplificado drásticamente estos pasos, permitiendo una implementación más rápida y predecible.

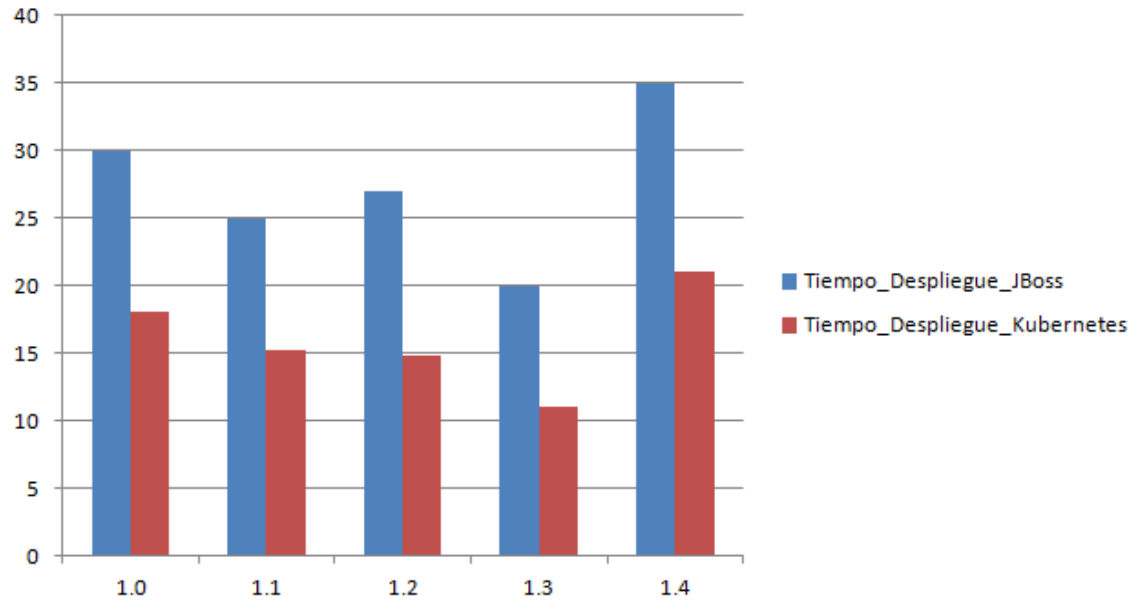


Figura 6.1 Mejora en el tiempo de despliegues.

- **Posibilidad de escalado horizontal (pods) de forma sencilla**: La adopción de la arquitectura basada en contenedores y la orquestación proporcionada por Kubernetes ha conferido a la aplicación una flexibilidad excepcional en términos de escalabilidad horizontal. Al enfrentar cargas de trabajo que exceden los recursos asignados en ambas infraestructuras (local y en la nube) y al recibir alertas a través del sistema de monitoreo, la adopción de una arquitectura basada en contenedores junto con la orquestación proporcionada por Kubernetes ha dotado a la aplicación de una flexibilidad excepcional en cuanto a su escalabilidad horizontal como podemos ver en los datos ilustrados más al final del punto. Esta característica ha marcado un cambio sustancial respecto a la infraestructura anterior, permitiendo gestionar la demanda creciente de recursos de manera dinámica y sin inconvenientes.

En el contexto de Kubernetes, el escalado horizontal se materializa mediante la replicación de pods de forma automática, las unidades fundamentales de despliegue en esta plataforma. La simplicidad con la que se puede lograr este escalado se traduce en la capacidad para aumentar la cantidad de instancias de la aplicación en respuesta a un incremento de la carga de trabajo, distribuyendo así eficientemente la demanda entre múltiples pods.

Este enfoque ofrece ventajas sustanciales, ya que no solo proporciona una respuesta ágil a los picos de tráfico, sino que también optimiza la utilización de recursos al adaptarse dinámicamente a las necesidades específicas de procesamiento en diferentes momentos. La gestión simplificada de múltiples instancias y su coordinación efectiva hacen que el escalado horizontal sea una herramienta valiosa para garantizar un rendimiento consistente y una disponibilidad óptima de la aplicación, al tiempo que se facilita una infraestructura más resistente y adaptable a las variaciones de la demanda del usuario.

En resumen, la capacidad de realizar un escalado horizontal de pods de manera sencilla, inherente a la adopción de Kubernetes, se erige como uno de los aspectos más destacados y beneficiosos derivados de la migración.

Después de la migración a Kubernetes, se ha experimentado una mejora significativa en la capacidad de escalado horizontal. La arquitectura basada en contenedores permite agregar y eliminar pods de manera dinámica y automática según la demanda de la aplicación. En comparación con la infraestructura original, donde el escalado estaba limitado a la disponibilidad de nuevas máquinas virtuales creadas manualmente gracias a plantillas predefinidas, Kubernetes ha demostrado ser mucho más flexible y eficiente.

Para ilustrar la mejora dejamos reflejados los siguientes datos:

- Escenario anterior (Máquinas virtuales):
  - Máquinas iniciales: 20
  - Tiempo para agregar nuevas máquinas: 30 minutos (manual)
  - Capacidad máxima del escalado horizontal: +5 máquinas.
- Escenario actual (Kubernetes):
  - Pods iniciales: 20
  - Tiempo para agregar nuevo pod: 5 minutos (automático)
  - Capacida máxima de escalado horizontal: +15pods

- **Ha facilitado considerablemente la autonomía de los equipos de desarrollo al fomentar el ``self-service´´:** Antes de la migración, la gestión de recursos y la implementación de aplicaciones estaban más centralizadas y dependían en gran medida de procesos manuales y recursos compartidos. Sin embargo, con la adopción de Kubernetes, se ha introducido un enfoque más descentralizado y orientado al "self-service" que ha empoderado a los equipos de desarrollo mejorando los siguientes puntos:
    - **Provisión Rápida de Recursos**
        - **Antes:** La provisión de recursos y la configuración asociada eran tareas centralizadas y requerían intervención manual.
        - **Después:** Con Kubernetes, los equipos pueden aprovisionar rápidamente los recursos que necesitan mediante la definición de configuraciones en archivos de manifiesto y su implementación automatizada.

    - **Despliegue Autónomo de Aplicaciones**
        - **Antes:** Los despliegues de aplicaciones estaban sujetos a procesos centralizados y dependían de la intervención de equipos específicos.
        - **Después:** Los equipos de desarrollo pueden implementar y gestionar sus propias aplicaciones de manera autónoma utilizando las capacidades de "self-service" proporcionadas por Kubernetes.

    - **Gestión Eficiente de Ciclos de Desarrollo**
        - **Antes:** Los ciclos de desarrollo podían experimentar demoras debido a la dependencia de recursos centralizados y procesos manuales.
        - **Después:** La autonomía facilitada por Kubernetes agiliza los ciclos de desarrollo al permitir a los equipos realizar implementaciones, actualizaciones y ajustes de escala de manera eficiente.

- **La facilidad en la que podemos definir y publicar métricas y alarmas:** Antes de la migración, el proceso de definición y recolección de métricas, así como la configuración de alarmas, podían ser tareas complejas y fragmentadas. Con la implementación de Kubernetes, se ha logrado una transición hacia un modelo más coherente y simplificado, aprovechando las funcionalidades inherentes a esta plataforma de orquestación de contenedores. Los principales beneficios son:

- **Integración con sistemas de monitoreo**
  - **Antes:** La integración con sistemas de monitoreo y la definición de métricas podían implicar múltiples herramientas y configuraciones dispersas.
  - **Después:** Kubernetes facilita la integración con sistemas de monitoreo, proporcionando interfaces estandarizadas para la recolección de métricas a nivel de clúster, nodos y servicios.

- **Automatización de Métricas y Alarmas**
  - **Antes:** La automatización de la recolección de métricas y la configuración de alarmas era un proceso menos centralizado y más propenso a errores.
  - **Después:** Kubernetes permite la automatización de estas tareas mediante la definición declarativa de políticas y reglas, mejorando la consistencia y reduciendo la posibilidad de errores humanos.

- **Visibilidad Mejorada**
  - **Antes:** La visibilidad en tiempo real sobre el estado y el rendimiento de la infraestructura podía ser limitada.
  - **Después:** La capacidad de definir y publicar métricas con facilidad en Kubernetes proporciona una visibilidad mejorada, permitiendo una monitorización más efectiva del rendimiento de las aplicaciones y la infraestructura.

- **Respuesta Rápida a Eventos**
  - **Antes:** La detección y respuesta a eventos críticos podían experimentar demoras debido a la complejidad en la configuración de alarmas.
  - **Después:** La agilidad en la definición y publicación de alarmas facilita una respuesta más rápida a eventos inesperados, mejorando la capacidad de mantenimiento y la disponibilidad del sistema.

En resumen, la facilidad en la definición y publicación de métricas y alarmas en un entorno kubernetizado representa un avance fundamental que contribuye a la fiabilidad y eficiencia operativa de las aplicaciones desplegadas.

En el contexto de la migración de la base de datos, se llevó a cabo una evaluación exhaustiva del rendimiento mediante la implementación de un test de carga utilizando JMeter[31]. JMeter, una herramienta de código abierto desarrollada por la Apache Software Foundation, se ha convertido en un estándar de la industria para la realización de pruebas de carga y desempeño en aplicaciones web. Esta herramienta, escrita en Java, se destaca por su capacidad para medir el rendimiento y analizar la carga funcional de diversos servicios, con especial énfasis en aplicaciones web.

El test de carga se diseñó específicamente para evaluar directamente la base de datos, prescindiendo de la interacción a través de las aplicaciones. Todos los recursos necesarios para la ejecución de estos tests, incluyendo el código y los scripts, se encuentran almacenados de forma accesible en el repositorio Git de ODEC.

Para garantizar una evaluación más precisa y eficiente del rendimiento a nivel de comunicaciones de red, los tests se ejecutaron desde una instancia de EC2 en AWS. Esta decisión se tomó con el objetivo de optimizar la eficacia de las pruebas y obtener resultados que reflejen de manera fiel el comportamiento del sistema.

El test de JMeter implementado abarca operaciones comunes en bases de datos, tales como inserciones, selecciones y actualizaciones, dirigidas hacia una tabla que tiene una estructura similar a la de Complet. La variación en el rendimiento se evaluó al modificar dos parámetros clave: el número de usuarios concurrentes y el número máximo de conexiones del *pool* de la base de datos.

Se realizaron pruebas comparativas utilizando dos instancias distintas de RDS, **Tabla 6.1** y **Tabla 6.2** con variaciones en el tamaño de la memoria asignada. Los resultados obtenidos se reflejan de manera gráfica en la **Figura 6.2.**

[31] Web de Jmeter: https://jmeter.apache.org/

Instance type	**db.m6g.xlarge**
vCPU	4
RAM (GB)	16
Storage (GB)	20
Max. Conn.	1705
Engine	PostgreSQL
Version	13.4
Region	eu-west-3

Tabla 6.1. Instancia RDS db.m6g.xlarge

Instance type	**db.r6g.xlarge**
vCPU	4
RAM (GB)	32
Storage (GB)	20
Max. Conn.	3479
Engine	PostgreSQL
Version	13.4
Region	eu-west-3

Tabla 6.2. Instancia RDS db.r6g.xlarge

Resultados

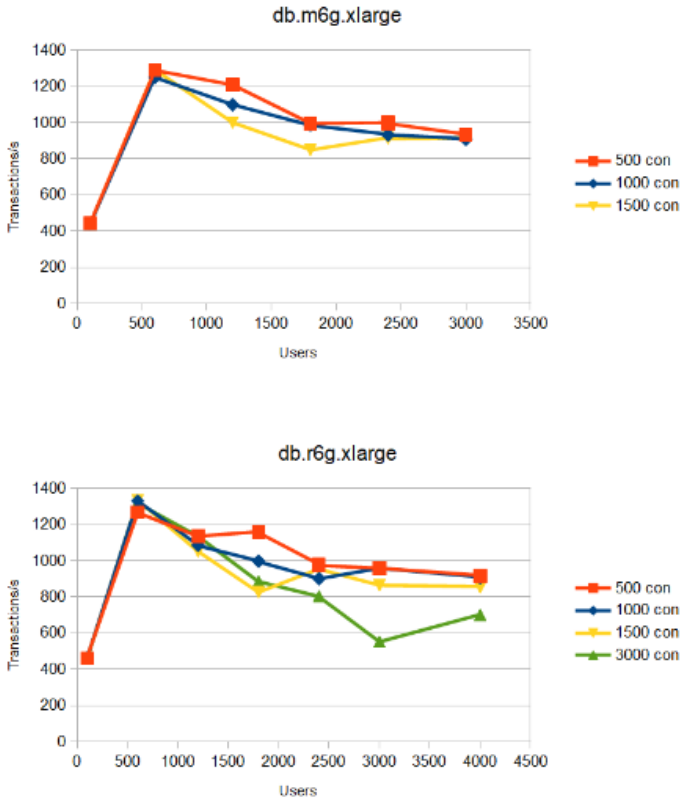


Figura 6.2 Resultados de los test de las bases de datos según instancia RDS

Es relevante destacar que, según los gráficos, la memoria RAM de las instancias no emergió como un factor determinante en el rendimiento observado. Por consiguiente, tras un análisis detenido, se tomó la decisión de mantener la instancia ***db.m6g.xlarge***, que se mostró como la opción más equilibrada y eficiente.

Estos resultados, respaldados por pruebas exhaustivas y análisis detallados, proporcionan una base sólida para la toma de decisiones informadas con respecto a la configuración y optimización de la base de datos en el entorno de AWS.

El plan de pruebas que se ha seguido para comprobar el buen funcionamiento del portal Complet ha sido:

- Acceso por login a Complet Manager a (https://app.complet.pro.odec.es/completManager, es una URL interna de ODEC y AWS.)
- Navegación por todas las opciones de menú.
- Creación de un estudio nuevo.
- Pruebas dentro del diseñador:
  - Crear nuevas pantallas y preguntas, editar existentes.
  - Copiar preguntas de otro estudio.
  - Pregunta con opciones recuperadas a partir de una consulta de la base de datos.
  - Lanzar link de demo.
  - Cuotas.
  - Exportar guión.
  - Envío de correo al provocar un error.
  - Correos en tiempo real.
- Publicar estudio de forma inmediata y programada.
- Cargar muestras.
- Realizar mailing.
- Reporting.
- Lanzar exportación inmediata y programada.
- Probar enlace de test.
- Probar enlace de producción.
- Exportar e importar traducciones.
- Crear un usuario y compartir un reporting.
- Debug de motor.

El despliegue en la nube, específicamente en AWS con la base de datos y otros servicios alojados en la misma nube, ofrece una serie de ventajas adicionales en comparación con un despliegue *on-premise*. A continuación, se detallan algunas de las mejoras clave:

- **Disponibilidad y tolerancia a fallos**: Factor que no dispone la infraestructura *on-premise* y que garantiza que el proyecto Complet siempre esté disponible.
- **Facilidad de integración con servicios gestionados de AWS y tiempo de respuesta**: Como la mayoría de servicios adicionales también se encuentran en AWS, nos facilita que la integración de la aplicación también esté en la nube.

7. Conclusiones

La migración de un proyecto *on-premise* a la nube de AWS ha representado una respuesta clave a los desafíos operativos que enfrentaba la empresa. Inicialmente, la infraestructura legada limitaba la escalabilidad, comprometía el rendimiento operativo, dos factores muy determinantes que apreciaba la empresa ODEC. También dificultaba la monitorización del sistema y conexiones simultáneas en el portal entre otras cosas. Adicionalmente, generaba unos costes extras de mantenimiento que repercutían en la empresa, ya que no son costes que se traspasen al cliente.

La estrategia adoptada para abordar este problema se centró en la kubernetización y posterior migración gradual y planificada hacia la infraestructura de la nube de AWS. Esta aproximación permitió mitigar riesgos y minimizar interrupciones en las operaciones diarias, mientras se maximizaba la utilización de los servicios nativos de la nube.

Las ventajas inherentes a esta propuesta se han evidenciado en los aspectos clave evaluados. En términos de rendimiento operativo, la agilidad y la disponibilidad mejorada han sido notables. La capacidad de escalar recursos de manera horizontal ha significado un cambio fundamental en la capacidad de adaptación a las demandas de cargas de trabajo. La implementación de sistemas de monitorización avanzados ha proporcionado una visibilidad sin precedentes sobre la infraestructura, mejorando la capacidad de respuesta y anticipación ante posibles problemas.

Asimismo, la alineación con los objetivos estratégicos se ha fortalecido considerablemente. La flexibilidad inherente a la nube de AWS ha permitido una asignación más eficiente de recursos y una optimización de costos, alineándose directamente con las metas de crecimiento y eficiencia de la empresa.

En cuanto a líneas de trabajo futuro, se plantea la exploración de tecnologías emergentes en la nube, como la implementación de servicios avanzados de inteligencia artificial y aprendizaje automático para mejorar aún más la eficiencia operativa y la toma de decisiones.

En conclusión, la migración exitosa hacia la nube de AWS no se ha limitado a resolver los desafíos operativos presentes en la infraestructura previa. Más bien, ha representado un punto de inflexión significativo que ha trascendido la mera solución de

problemas inmediatos. Esta transición ha marcado el comienzo de una transformación integral en la arquitectura tecnológica de la empresa de todos los proyectos del entorno de producción.

Al liberar nuestros sistemas de las limitaciones físicas de la infraestructura local, hemos sentado los cimientos para una estructura adaptable y altamente flexible. Este cambio fundamental no solo ha impulsado la resolución de los problemas operativos históricos, sino que ha abierto un abanico de posibilidades para la innovación y la evolución constante.

Bibliografía

Apostu, A., Puican, F., Ularu, G., Suciu, G., & Todoran, G. (2013). Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud. Recent advances in applied computer science and digital services, 2103. Disponible en: https://d1wqtxts1xzle7.cloudfront.net/

AWS, (s.f.) Disponible en: https://aws.amazon.com/es/what-is-cloud-computing/

Calderón, P. F. M., & Mora, M. G. Z. (2020). Computación en la nube: la infraestructura como servicio frente al modelo *On-Premise*. *Dominio de las Ciencias*, *6*(4), 1535-1549. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=8638170

Centro Criptológico Nacional, (s.f.) Guía de Seguridad de las TIC CCN-STIC 858. Disponible en: https://www.ccn-cert.cni.es/pdf/guias/series-ccn-stic/800-guia-esquema-nacional-de-seguridad/5099-ccn-stic-858-implantacion-de-sistemas-saas-en-modo-local-on-premise/file.html

De Giusti, A. E. (2023). Transformación digital. In *Plenario de la Academia de Ingeniería de la Provincia de Buenos Aires (Buenos Aires, 5 de junio de 2023)*. Disponible en: http://sedici.unlp.edu.ar/handle/10915/154207

Delgado Fernández, T. (2020). Influencia de la pandemia COVID-19 en la aceleración de la transformación digital. *Revista Cubana De Transformación Digital*, *1*(3), 01–05. Recuperado a partir de https://rctd.uic.cu/rctd/article/view/116

Freixas, J. (2022). La transformación digital de la mano del *cloud computing* y DevOps. Disponible en: https://marketing.onlinebschool.es/Prensa/Informe%20OBS%20La%20Tansformaci%C3%B3n%20Digital.pdf

Guerola Navarro, V. (2022). Impacto de *Cloud Computing* en los procesos de Transformación Digital. http://hdl.handle.net/10251/180717

Gupta, B., Mittal, P., & Mufti, T. (2021, March). A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services. In *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India*. Disponible en: http://dx.doi.org/10.4108/eai.27-2-2020.2303255

López, A. M. (2023). Digitalización, transformación digital y economía digital en España. Disponible en: https://www.cemad.es/wp-content/uploads/2023/04/48_Ana_M_Lopez-181.pdf

Palos-Sanchez, P., Reyes-Menendez, A., & Saura, J. R. (2019). Modelos de Adopción de Tecnologías de la Información y Cloud Computing en las Organizaciones. *Información tecnológica*, *30*(3), 3-12. Disponible en: http://dx.doi.org/10.4067/S0718-07642019000300003

UNIR REVISTA, (2021). El impacto del *cloud computing* en la era COVID. Disponible en: https://www.unir.net/ingenieria/revista/el-impacto-del-cloud-computing-en-la-era-covid/

ANEXOS

URL a ficheros anexos:

https://drive.google.com/file/d/1Mg5dQBZgSgVwFcfCYafUTOBGYXpmrCZG/view?usp=drive_link