Universidad Siglo 21

Proyecto Trabajo Final de Graduación

Ingeniería en Software

Proyecto de Investigación Aplicada (PIA)

Análisis de solución eficiente para escalabilidad de sistemas online mediante orquestación de contenedores

Franco Nelson Funes

SOF00522

Fecha: 20/07/2018

Docente: Mgter. Adriana Perez

Abstract

Currently, due to several reasons, some high availability web services accessed by a great number of users present denial of service failures, colloquially known as "downs". When a high availability system is down it can cause an important impact in the company's activities, with negative effects in its economy and public image. In this work we study different alternatives to solve this problem. The solutions presented here are based on Containers, Orchestrators and Cloud Computing, all of them growing technologies in the market. Moreover, we present our analysis of local restrictions to the application of these technologies.

Resumen

Actualmente se observa que ciertos sistemas que atienden a grandes cantidades de usuarios tienen falta de alta disponibilidad, o coloquialmente llamadas *caídas*, por distintos motivos. Estas caídas muchas veces causan impactos importantes, ya que pueden paralizar la actividad de una empresa, afectándola económicamente y a su imagen. Este trabajo estudia y da alternativas de solución basada en varios componentes tecnológicos que vienen ganando mucha fuerza en el mercado como lo son los contenedores, los orquestadores de contenedores y cloud computing. A su vez, este trabajo estudia las restricciones para las implementaciones de estas poderosas tecnologías.

Tabla de contenido

Tabla de imágenes

Índice de Tablas

Título

Análisis de solución eficiente para escalabilidad de sistemas online mediante orquestación de contenedores.

Introducción

Este trabajo final de graduación se enfoca a la problemática general de la no disponibilidad de sistemas por colapsos en peticiones, conocidos como sistemas de alta demanda. Esto nos lleva uno de los atributos de la calidad de un producto o servicio que es la de robustez. Los avances tecnológicos actuales nos ayudan, siguiendo algunas pautas, a prever y tener una estrategia de recuperación ante caídas o soporte a alta demanda de solicitudes por medio del uso de orquestadores de contenedores. En la actualidad, una pobre performance o caída total del sistema se traduce en varios problemas para la empresa relacionados con el costo de la no calidad de la implementación de nuestro sistema, como serían las pérdidas económicas y de la imagen de la empresa. Por otro lado, esto afecta directamente a los usuarios del sistema pudiendo ocasionar serios inconvenientes. Los contenedores como pieza fundamental de construcción de implementaciones se relacionan mucho con la utilización de almacenamiento en la nube (más comúnmente en inglés llamado *Cloud Computing*) que es una forma de almacenar toda nuestra infraestructura en algún servidor o varios en ubicaciones en distintas partes del mundo y accesibles por la internet.

Los antecedentes marcan un antes y un después de la existencia de internet. Luego de la creación de la gran red de redes, surgen sistemas de tipo web de alta demanda por la masividad que generan como páginas de comercio electrónico, plataforma de e-learning, plataformas de redes sociales. Esto en un principio se manejó con servidores físicos que conformaban clúster con balanceadores de carga que de acuerdo a ciertos criterios pueden tomar las decisiones sobre a cuál servidor se debería delegar la atención. Esta misma idea se evolucionó con la virtualización de servidores, utilizando los mismos clústeres de servidores físicos, se nos permite administrar los recursos como un todo y también usar balanceadores de carga a nivel software. Lo que el trabajo plantea es administrar de manera más eficiente los recursos disponibles con la utilización de contenedores, generando dinámicamente los mismos de acuerdo a la carga del sistema.

Durante el desarrollo del presente trabajo de investigación, se encontraron las siguientes situaciones relacionadas:

- Caídas constantes de LexNet (Sistema de Justicia español) Fecha: 31/07/2017[1]: LexNet es un sistema de notificaciones español para comunicarse con procuradores, abogados y otros colectivos, donde por el aumento de actividad el sistema sufre caídas de varios minutos, lentitud de respuesta y errores en el ordenamiento de las notificaciones.
- Caída de servidor principal en Linea1 (sistema de metro de Chile) Fecha: 09/08/2017[2]. En este caso el servidor central que controla el tráfico de trenes cayó, provocando que los trenes realizaran sus recorridos más lento y esto causó retrasos generales en transporte de pasajeros ya que ocurrió en hora pico.
- Biotrén suspende todos sus servicios por caída general del sistema en Chile – Fecha: 16/08/2017[3]. En este caso, similar al anterior se cayó el sistema de control de trenes, con lo cual fue suspendido el servicio, donde se devolvió el monto de los pasajes, pero miles de personas fueron retrasadas en llegar a sus trabajos.
- Pueden encontrarse más ejemplos en "The Risk Digest"[4], que lista riegos y fallas en sistemas desde 1985.

Los ejemplos anteriores son solo un rápido extracto de noticias actuales de los problemas que pueden llegar a causar un mal diseño de la arquitectura de sistemas que requieren tener alto nivel de respuesta y robustez en horas pico, en sistemas que pueden considerarse de criticidad media - alta.

Como se demostró, existen casos de distintos sectores y de distintos países donde se visualiza las consecuencias por el problema de falta de servicio de plataformas informáticas, con lo cual se infiere que es un problema de amplio alcance. En el caso de los sistemas de transporte en Chile, resulta fácil dimensionar el problema relacionado a las pérdidas económicas que

[1] fuente: https://www.elconfidencial.com/tecnologia/2017-07-31/lexnet-justicia-sistema-informatico_1423267/, recuperado el 26/08/2017

[2] fuente: http://www.cooperativa.cl/noticias/pais/transportes/metro/caida-de-servidor-central-en-linea-1-provoco-colapso-en-el-metro/ , recuperado el 26/08/2017

[3] fuente: https: www.biobiochile.cl/noticias/nacional/region-del-bio-bio/2017/08/16/biotren-suspende-todos-sus-servicios-por-caida-general-del-sistema.shtml, recuperado el 26/08/2017

[4] http://catless.ncl.ac.uk/Risks/

supone dicha situación de la empresa y los problemas generados a los usuarios por no disponer de los servicios en horas pico.

Planteada la problemática, se formuló la siguiente pregunta:

¿Cómo se puede dar solución o mitigar un problema de falla generalizada de un sistema que es colapsado por alta demanda?

Justificación

Esta investigación tiene una aplicación práctica amplia porque es un problema más bien generalizado ya como se ejemplificó tiene directa implicancia en sectores disimiles y las consecuencias pueden ser graves y afectar muchas veces masivamente a los usuarios. Generalmente, el usuario no valora cuando un sistema funciona bien y es robusto, ya que supone que es una cualidad inherente del mismo, aunque para el profesional de sistemas no supone soluciones triviales.

Como resultado del estudio, se plantean alternativas que ayudan a solucionar/mitigar los problemas descriptos con utilización de tecnologías que han aparecido en los últimos años como lo son los contenedores, orquestadores de contenedores y cloud computing. El impacto de la implementación de estas tecnologías no solamente afecta al despliegue y mantenimiento de sistemas, sino que también afectará a la forma de desarrollo de los mismos.

Tanto como se plasmó en el presente trabajo de investigación y por parte de los defensores de estas nuevas tecnologías ven a los contenedores como el futuro del desarrollo de sistemas, ya que se alinea con cuestiones como cloud computing o microservicios y nuevos roles que han surgido como ingenieros DevOps. Los planteos realizados en este trabajo tienen amplio alcance en sistemas que son de uso masivo, entre los cuales tenemos sistemas de autogestión, sistemas de reservas de pasajes, compras de entradas, WebServices meteorológicos, entre otros.

Problema de Investigación

Cuando se refiere a orquestadores de contenedores, se infiere una temática de alta complejidad de IT en servidores propios de la empresa o en cloud, pero supone una ventaja en tiempos y costos a la larga en la etapa de mantenimiento y evolución del software.

Los orquestadores fueron creados para ayudar en la gestión de este sumamente nuevo concepto de virtualización llamado contenedor, se puede instalar cualquier aplicación. Varios aspectos del ciclo de desarrollo de software se ven afectados por este cambio de paradigma. Por un lado, incluyendo este concepto desde el comienzo de la codificación como así también, y donde más se va a orientar este trabajo, en las etapas posteriores a la implementación, como referimos en el párrafo anterior. El mayor interés de este trabajo es relacionar la generación automática/dinámica de contenedores con atender situaciones de estrés o sobrecarga de sistemas online con muchos usuarios concurrentes.

Dicho trabajo diferencia el concepto de máquina virtual y contenedores. En máquinas virtuales se conocen productos como VirtualBox® o VMWare® donde es virtualizada la totalidad del sistema operativo, y para iniciarlo supone iniciar un sistema completo, incluyendo con drivers virtualizados para acceder a los recursos de hardware del computador host. En contraposición la tecnología de contenedores utiliza una capa de servicio que provee APIs para acceder al sistema operativo como así también a los contenedores. Un contenedor puede no tener un sistema operativo dentro, por esto generalmente se los refiere como virtualización liviana, o lightweight por su término en inglés. En la actualidad, la mayoría de las empresas optan por máquinas virtuales por sobre servidores físicos dedicados, con lo cual el escenario común será desplegar contenedores dentro de las mismas.

En este punto del trabajo, se formularon las siguientes preguntas:

¿Serán reemplazados los despliegues en servidores físicos o en máquinas virtuales por orquestadores de contenedores?

¿Qué tipo de contenedores existen?

¿Cómo fue la evolución hacia la tecnología de orquestadores de contenedores?

¿Qué ventajas tienen?

¿Siempre es posible implementar contenedores? ¿Cuáles son las restricciones que tenemos?

¿Es más económico tener orquestadores en la nube que despliegues en servidores de hosting propio?

¿Internet ayudó al crecimiento de contenedores? ¿Por qué almacenar nuestra infraestructura en la nube? ¿Qué otros motivos generaron esta tendencia?

¿Cuáles son los usos de los contenedores/orquestadores? ¿Lo puede implementar cualquier empresa? ¿Existe una tendencia de mayor adopción en empresas?

¿Cómo podemos aprovechar esta tecnología para atacar los problemas de robustez y fiabilidad de los sistemas de alta demanda?

Objetivo general del proyecto de investigación

Analizar y proponer diferentes soluciones para la eficiencia de sistemas de información de alta demanda utilizando contenedores y orquestadores.

Objetivos específicos del proyecto

Los objetivos específicos del proyecto los podemos enumerar de la siguiente forma:

- Mostrar las tecnologías disponibles para alta disponibilidad basado en contenedores
- Comparar el funcionamiento de contenedores contra las máquinas virtuales tradicionales.
- Comparar las características de las distintas tecnologías de orquestadores del mercado.
- Definir cada una de las partes del ecosistema de contenedores
- Analizar la factibilidad de implementación de orquestadores y contenedores a distintos tipos de sistema de alta demanda.
- Explicar los cambios a nivel de ciclo de vida de software que implica la utilización de estas tecnologías.

- Explicar los beneficios del uso de esta solución.
- Analizar el factor de eficiencia en la aplicación de estas nuevas tecnologías.

Marco Teórico Referencial

Virtualización

De acuerdo a Tanenbaum, esta tecnología permite que una sola computadora contenga varias máquinas virtuales, cada una de las cuales puede llegar a ejecutar un sistema operativo distinto. La ventaja de este método es que una falla en una máquina virtual no ocasiona que las demás fallen de manera automática.

En un sistema virtualizado, se pueden ejecutar distintos servidores en diferentes máquinas virtuales, con lo cual se mantiene el modelo parcial de fallas que tiene una computadora, pero a un costo mucho menor y con una administración más sencilla. (Tanenbaum, 2009, p. 569)

Infraestructura como servicio (IaaS)

IaaS (Infrastructure as a Service) consiste en la modalidad de ciertos proveedores de ofrecer infraestructura IT (hardware, software, enlace de internet, entre otras cosas) por un costo por uso. Estos recursos son propiedad del proveedor y se rentan para uso privado/comercial. Generalmente estos costos están asociados a cantidades de utilización que corresponda, como tiempo de utilización, MegaHertz consumidos en procesadores, cantidad de RAM utilizada, cantidad de gigabytes transferidos. Ejemplos de proveedores que brindan este tipo de servicios son Azure de Microsoft, AWS de Amazon, Google Cloud de Google.

Escalabilidad

El concepto de *escalabilidad* es muy importante en la temática abordada por este trabajo y se define a continuación:

"[E]s la capacidad del sistema de hacer frente a un número creciente de usuarios sin reducir la QoS global que se entrega a cualquier usuario" (Sommerville, 2011, p. 504).

Contenedores e Imágenes

Según RedHat Corporation, "*[S]on tecnologías que le permiten empaquetar y aislar las aplicaciones con sus entornos de tiempo de ejecución completo, es decir, con todos los*

archivos necesarios para ejecutarse. Esto facilita mover entre entornos a la aplicación contenida (desarrollo, prueba, producción, etc.) mientras retiene la funcionalidad completa"[5].

Con este concepto se vislumbra que el mantenimiento de los sistemas será menos dependiente de su entorno de despliegue físico, optimizando tiempo y costos.

Los contenedores representan un entorno corriendo activamente que puede ejecutar casi cualquier tipo de software; puede ser una aplicación web como un demonio de servicio. (Gindi, J., 2016, p. 1).

Cabe destacar que se diferencia el termino contenedor como un proceso que puede ser iniciado, detenido, reiniciado, del concepto de imagen, que es el estado de no ejecución de un contenedor.

Orquestador de Contenedores

Tal como explica Smith (2017), los servicios usando contenedores pueden ser iniciados a través de múltiples servidores y hasta incluso distintos proveedores cloud (AWS, GoogleCloud, Azure). Una vez desplegados, los contenedores pueden ser iniciados en segundos sin la sobrecarga de una máquina virtual completa. Incluso mejor, aplicaciones desarrolladas en contenedores pueden ser desplegadas exactamente como fueron construidas, minimizando problemas de configuración y mantenimiento de dependencias. Todo esto es manejado por el orquestador (Smith, 2017, p. 31).

Un orquestador es una herramienta de clustering que permite a un operador interactuar con un único endpoint para operar y orquestar un conjunto de recursos, que en este caso es contenedores. En vez de manualmente distribuir la carga (contenedores) en el clúster, el orquestador automatiza esto y muchas cosas más. Es la herramienta que decide donde iniciar los trabajos, como almacenarlos y cuando eventualmente reiniciarlos, entre otras actividades. El operador necesita solamente configurar algunos comportamientos, decidir la topología y tamaño del clúster, configurar opciones de performance, y activar o desactivar funcionalidades avanzadas. (Sopelssa, 2016, p. 23)

[5] fuente: *https://www.redhat.com/es/topics/containers*, recuperado el día 28/09/2017

Tráfico Web

"El tráfico web viene siendo el tipo de tráfico dominante en la Internet, representando más bytes, paquetes y flujos que cualquier otro tipo de tráfico. Dada esta dominación, son sumamente importante los mecanismos/protocolos para analizar los problemas de performance por los efectos de los mismos." (Hernández-Campos, 2003, p. 1) – Traducción propia.

Cuando hablamos de sistemas con alta demanda justamente se quiere referir a sistemas que tienen un tráfico web elevado, es decir, que supera o está cerca del límite de capacidad de manejo del servidor. Esto afecta en la performance del sistema, afectando la experiencia de usuario, o en el caso extremo generando la caída del sistema conocida como Denial of Service (denegación de servicio).

Integración continua

Es una estrategia de ciclo de vida de software usada para agilizar la velocidad del proceso de desarrollo. Esto es logrado mediante correr automáticamente los test automáticos cada vez que un cambio significativo es realizado al código base, de esta manera se obtienen más estables y más rápidamente ya que hay un control de estabilidad de la construcción del software (Miells, 2016, p. 146).

Delivery Continuo

Es la estrategia por la cual se logra una cierta confianza sobre los entregables construidos en los procesos de CI (Continous Integration) y se los disponibiliza a los usuarios. Esto es el objetivo de Delivery Continuo. (Miells, 2016, p. 169).

DevOps

El término de DevOps es un juego de palabras en inglés de "Developer" (desarrollador) y Operations (Operaciones), la cual nos da una pista detrás de la naturaleza de este rol. Es una

práctica donde la colaboración entre diferentes disciplinas del desarrollo de software es promovida.

Un ingeniero con la mentalidad de DevOps va a identificar rápidamente características similares en procesos de desarrollo de diferentes equipos, para brindar herramientas que puedan compartir entre estos equipos, satisfagan sus necesidades y se logre de esta forma reducción de costos de mantenimiento.

Otro objetivo de un DevOps es la automatización y el Delivery Continuo. Esto es simplemente, automatizar tareas tediosas y repetitivas dejando más tiempo para interacción humana, donde el valor es agregado (Verona, 2016, p. 1-3).

Protocolo

Los sistemas se comunican a través de protocolos, y hoy más aun por estar Internet como el medio que conecta a cada una de las partes. *"es un conjunto de reglas que rigen el formato y el significado de los paquetes, o mensajes, que se intercambiaron las entidades iguales en una capa. Las entidades utilizan protocolos para implementar sus definiciones del servicio. Son libres de cambiar sus protocolos cuando lo deseen, siempre y cuando no cambie el servicio visible a sus usuarios. De esta manera, el servicio y el protocolo no dependen uno del otro."* (Tanenmaun, 2003, p. 38)

Modelo de Referencia OSI

En este trabajo se encontrarán muchos conceptos de redes los cuales se basan en la especificación del modelo OSI, como por ejemplo la clasificación de balanceadores de carga.

"Este modelo está basado en una propuesta desarrollada por la ISO (Organización Internacional de Estándares) como un primer paso hacia la estandarización internacional de los protocolos utilizados en varias capas (Day y Zimmermann, 1983). Fue revisado en 1995 (Day, 1995). El modelo se llama ***OSI (Interconexión de Sistemas Abiertos****) de* ***ISO*** *porque tiene que ver con la conexión de sistemas abiertos, es decir, sistemas que están abiertos a la comunicación con otros sistemas."* (Tanenbaum, 2011, p. 37). La ilustración 1

representa el encapsulamiento en cada capa que dan origen a las distintas unidades, con el orden de las siete capas involucradas.

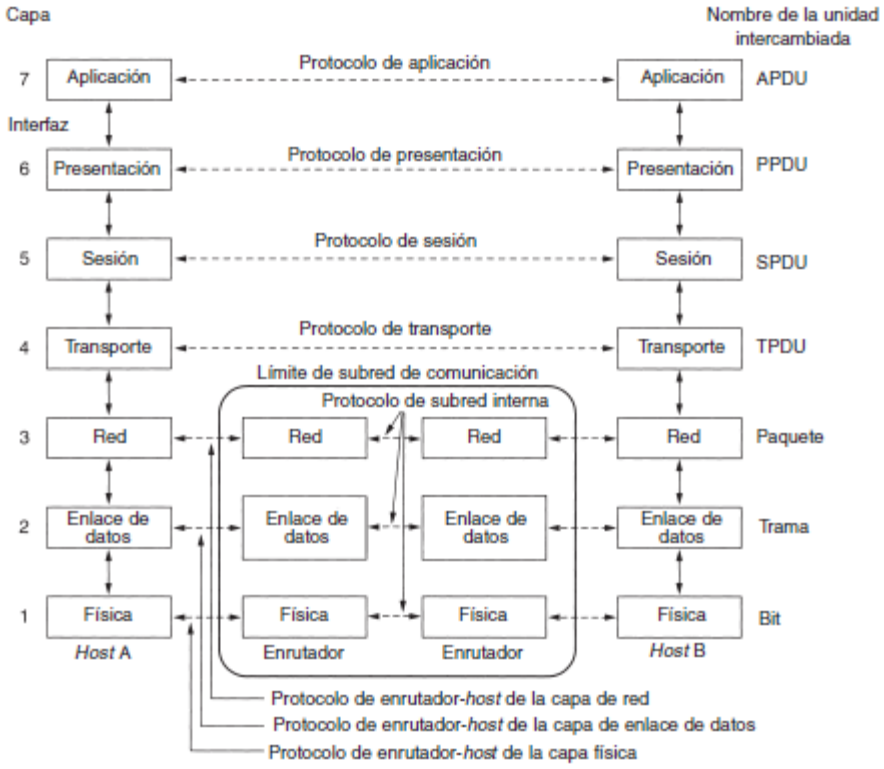


Ilustración 1: Modelo de Referencia OSI. Fuente: (Tanenbaum, 2003, p. 39)

Diseño Metodológico

De acuerdo a la naturaleza del trabajo se eligió una metodología de investigación de carácter cualitativa, tomando de la misma el tipo de investigación descriptivo y exploratorio.

Según Vieytes, los métodos cualitativos serían "*relativistas, holistas, descriptivos/exploratorios, subjetivos, inductivos, ilustrativos, interpretivistas y orientados para exponer el significado para los actores*" (2014, p. 43).

Por ser tópicos poco desarrollados y sobre todo en nuestro idioma, sumamente nuevos y de cierta complejidad, se optó por una investigación exploratoria. Vieytes lo explica de la siguiente manera: "*A veces ocurre que se conoce poco sobre la cuestión de interés, cuando se decide comenzar a investigar*". "*Las investigaciones exploratorias se inician, entonces, cuando hemos revisado los antecedentes de nuestro problema y encontramos que hay muy poco conocimiento acumulado del mismo*". (2004, p. 90)

Fuentes bibliográficas, documentales y datos secundarios

Las fuentes utilizadas están relacionadas sobre todos a artículos de Internet, libros, white papers en su mayoría en inglés que fueron la base de esta investigación. Como bien dice Vieytes, "*Los libros, revistas, documentos oficiales, empresariales o privados (de empresas, cámaras comerciales, resultados de investigaciones de mercados, artículos escritos por directivos, economistas, periodistas de medios masivos, etc) y los documentos de Internet entre otros, constituyen la forma más simple y económica de obtener información tanto teórica como investigaciones ya realizadas acerca de nuestro tema de interés.*

Con el término datos secundarios nos referimos a los que obtenemos de investigaciones realizadas con otros objetivos, por otros investigadores, pero que por su relación o cercanía pueden resultarnos útiles." (2004, p. 91)

Entrevistas en la investigación cualitativa

Por el tipo de investigación llevada adelante, se utilizarán entrevistas no estructuradas para no limitar las respuestas del entrevistado. Vieytes explica la importancia de la entrevista de la siguiente manera: "*Las entrevistas constituyen un recurso privilegiado para acceder a la*

información desde la perspectiva del actor. El objetivo central es captar lo que es importante en la mente de los informantes: sus significados, perspectivas y definiciones; en suma, el modo en que ellos ven, clasifican y experimentan el mundo." (2004, p. 661).

Con esta información se podrá entender lo que existe actualmente para solucionar o intentar mitigar el problema, sus opiniones y sus expectativas de una solución mejorada. (Ver modelo en el anexo)

Encuestas

Se pueden encontrar distintos tipos de encuesta de acuerdo a su alcance de sus objetivos, el modo de interacción con el encuestado y según el desarrollo temporal. De acuerdo a esto, se realizará una encuesta del tipo exploratoria por medio de la utilización de correo electrónico y otros medios de contacto por internet como mensajería instantánea mediante Google Forms. Este tipo de encuestas son realizadas a poblaciones pequeñas y muestras no probabilísticas que tienen como propósito identificar las dimensiones del problema de investigación, sugerir una hipótesis, perfilar categorías o precisar modelos teóricos. También se decide sistemáticamente a quien enviar esta encuesta para homogenizar la muestra con sus mayores variaciones posibles (Vieytes,2004, p. 328-329).

Análisis F.O.D.A

Se realizó un análisis de Fortalezas, Oportunidades, Debilidades y Amenazas para comparar las distintas implementaciones de las aplicaciones de alta demanda y elegir la más conveniente.

Tabla Metodológica

Siendo presentado el anterior marco teórico del diseño metodológico, se resumen dichos ítems con la tabla que sigue a continuación:

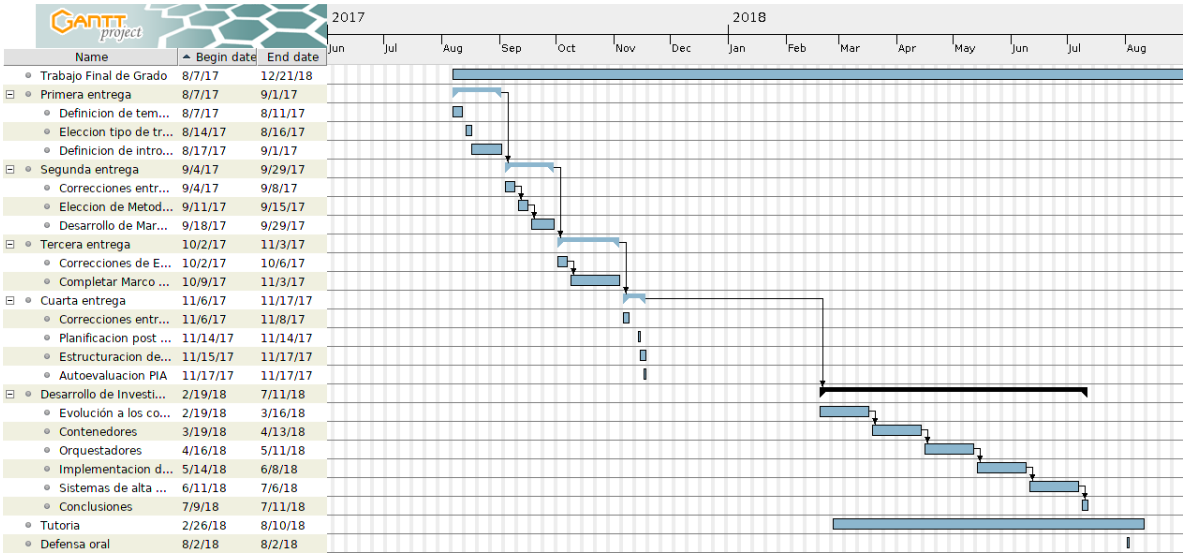
Tabla Metodológica	
Tipo de Investigación	Cualitativa/Exploratoria
Muestra	Orquestadores de Contenedores. Se analizarán 3 orquestadores principales del mercado (Mesos, Kubernetes y Swarm)

Población	Al menos 2 empresas con sistemas con alta demanda para entrevistas. 30 profesionales de IT relacionados a las tareas de operación de sistemas en producción para encuestas.
Modo de la muestra	No Probabilísticos
Herramientas a utilizar	- Encuestas a expertos del área de IT - Análisis de textos
Instrumentos	- Análisis F.O.D.A - Entrevistas a Expertos - Fuentes bibliográficas, documentales y datos secundarios.

Tabla 1: Tabla Metodológica

Diagrama de Gantt

El siguiente Gantt va a mantenerse actualizado en base a las entregas y el detalle de cada una.



Name	Begin date	End date
Trabajo Final de Grado	8/7/17	12/21/18
Primera entrega	8/7/17	9/1/17
Definicion de tem...	8/7/17	8/11/17
Eleccion tipo de tr...	8/14/17	8/16/17
Definicion de intro...	8/17/17	9/1/17
Segunda entrega	9/4/17	9/29/17
Correcciones entr...	9/4/17	9/8/17
Eleccion de Metod...	9/11/17	9/15/17
Desarrollo de Mar...	9/18/17	9/29/17
Tercera entrega	10/2/17	11/3/17
Correcciones de E...	10/2/17	10/6/17
Completar Marco ...	10/9/17	11/3/17
Cuarta entrega	11/6/17	11/17/17
Correcciones entr...	11/6/17	11/8/17
Planificacion post ...	11/14/17	11/14/17
Estructuracion de...	11/15/17	11/17/17
Autoevaluacion PIA	11/17/17	11/17/17
Desarrollo de Investi...	2/19/18	7/11/18
Evolución a los co...	2/19/18	3/16/18
Contenedores	3/19/18	4/13/18
Orquestadores	4/16/18	5/11/18
Implementacion d...	5/14/18	6/8/18
Sistemas de alta ...	6/11/18	7/6/18
Conclusiones	7/9/18	7/11/18
Tutoria	2/26/18	8/10/18
Defensa oral	8/2/18	8/2/18

Ilustración 2: Gantt de proyecto

Evolución a los contenedores

El camino hacia los contenedores empieza con el concepto de memoria virtual, inventado por Frizt-Rudolf Güntsch en 1956 que establece el almacenamiento administrado software y hardware integrado para ocultar esos detalles al usuario (Jessen, 1996, p. 216-219). Unos años más tarde, en 1959, John McCarthy que trabajaba en IBM da uno de los conceptos más revolucionarios de la computación. Teniendo en cuenta que los primeros ordenadores eran grandes equipos y muy costosos, se buscó la forma de optimizar su utilización ya que no se podían utilizar por múltiples usuarios. Solo podía hacerse trabajos por lote, es decir, secuenciales. A partir de esta limitación McCarthy idea el concepto de "Tiempo Compartido" en donde por medio de interrupciones del procesador y técnicas de multiprogramación permiten el uso concurrente de los recursos de un computador[6].

En 1964 IBM nuevamente reaparece con otro invento que agrupa los dos conceptos anteriores, el de la virtualización de sistemas operativos corriendo en diferentes contextos. Esto lo realiza en el centro Centro Científico de Cambridge sobre un equipo IBM CP-40, usando como hypervisor el sistema llamado SIMMON (Parmelee, 1976, p. 99-130). Aquí surgen dos tipos de virtualización (Goldberg, 1972):

- Tipo 1 – bare-metal hypervisor o nativo: estos corren directamente sobre los recursos de hardware. Por ejemplo, VMWare ESX.
- Tipo 2 – o hosted hypervisor: estos hypervisors corren directamente sobre un sistema operativo convencional. Por ejemplo, VirtualBox, Hyper-V, QEMU o VMWare Workstation.

Aquí se empieza a hablar robustez, ya que la separación de contexto de las VMs que ante un posible fallo de algún sistema operativo estaría aislado y los demás seguirían corriendo.

En el año 1979, en Unix se incorpora una nueva funcionalidad llamada chroot (change root) que permita aislar procesos en contextos aislados[7]. Esto quiere decir que el proceso y sus

[6] Fuente: http://www-formal.stanford.edu/jmc/history/timesharing/timesharing.html, recuperado el 02/07/18
[7] Fuente: https://docs.freebsd.org/44doc/papers/jail/jail-9.html, recuperado el 02/07/18

procesos hijos ejecutados no tiene acceso a las carpetas de orden superior[8]. Esto puede entenderse como una medida de seguridad, ya que permite ejecutar programas desconfiables, o en proceso de testing, sin afectar al sistema operativo padre.

En los años siguientes con los avances comentados, todo sigue orientado a mainframes con gran cantidad de procesamiento. A mediados de los años 60, se venía ideando una red de computadoras que surgió como proyecto militar (ARPANET) y luego fue hacia el contexto académico (NFSNET). Pero no fue hasta principios de los 90 que esta red de redes se explotó a nivel comercial y los usuarios de todo el mundo pudieron acceder a Internet, la gran red de redes. Muchos avances estuvieron asociados a la generación de esta idea, protocolos nuevos y muy importantes como TCP/IP, FTP y HTTP[9]. Este es el punto de inflexión donde la cantidad de usuarios concurrentes se dispara exponencialmente a través del tiempo.

Las primeras virtualizaciones llamadas livianas vienen de la mano de productos nacidos en 2000, como Virtuozzo[10]. En este año surge una plataforma que permite virtualizar sistemas operativos, pero a nivel de sistema operativo y no de hardware. Esto le permite ser más ágil a la hora de manejar los recursos del servidor, ya que se multiplexa el núcleo del sistema operativo el cual termina administrando los recursos, que al ponerle una capa adicional de hardware virtualizado que genera una sobrecarga adicional a diferencia de las tradicionales VMs[11].

También en año 2000 FreeBSD, implementa lo que le llama jails (jaulas en inglés)[12]. Este fue el primer contacto con crear contenedores en un sistema operativo. Esto permite una virtualización a nivel del sistema operativo, permitiendo ejecutar varias instancias de los mismos en forma independiente, es decir, que cada jail tiene sus procesos, sus usuarios y superusuarios (Intel, 2014, p. 3).

[8] Fuente: https://www.freebsd.org/cgi/man.cgi?query=chroot, recuperado el 02/07/18
[9] Fuente: https://www.internetsociety.org/internet/history-internet/brief-history-internet/, recuperado el 02/07/18
[10] Fuente: https://openvz.livejournal.com/49158.html, recuperado el 02/07/18
[11] Fuente: http://www.hpl.hp.com/techreports/2007/HPL-2007-59R1.html?jumpid=reg_R1002_USEN, recuperado el 02/07/18
[12] Fuente: https://www.freebsd.org/releases/4.0R/announce.html, recuperado el 10/07/18

Mientras tantos ingenieros de IBM en 2007 implementan la tecnología de Linux Containers (LXC), un proyecto que aprovecha las funcionalidades de Linux de cgroups y namespaces. Cgroups, provee la posibilidad de administrar medir los recursos utilizados por procesos en grupos de control. Namespaces permite aislar los recursos entre procesos a nivel de software (Intel, 2014, p. 5).

En 2013 se hace conocido el proyecto Docker, creado por el francés Solomon Hykes. Docker implementa una capa de abstracción para facilitar la implementación de contenedores LXC. Permite la virtualización a nivel de sistema operativo, con lo cual la administración de los recursos se delega al sistema operativo base. Docker permite también la creación de contenedores en Windows, utilizando Hyper-V para poder comunicarse con el sistema operativo host (Nickoloff, 2016, p. 4).

Docker es indiscutiblemente la herramienta de contenedores más reconocida y utilizada en el mundo (Forrester, 2016, p. 3). Debido a las características de ser livianos y eficientes en el manejo de recursos, se crea en 2015 el orquestador propietario de Docker Inc. llamado Docker Swarm, esto básicamente permite manejar a un grupo de dockers como un clúster. Es decir, garantizar una cantidad de instancias corriendo, administrarlo y monitorearlos en todo momento. Los beneficios claros de implementar un orquestador son que permiten una alta disponibilidad, y pueden mejorar el rendimiento, llamado comúnmente por su término en inglés throughput, por procesamiento distribuido (Soppelsa, 2016, p. 22).

A continuación, se sintetizan los hitos relacionados a contenedores:

Año	Hito	Creador
1956	Memoria Virtual	Frizt-Rudolf Güntsch
1959	Tiempo Compartido	John McCarthy
1964	Hypervisor (14 VM concurrentes)	IBM
1979	chroot	Bell Labs
circa 1993	Internet se transforma en una red global	Varios

2000	Virtualización OS	Virtuozzo
2000	jails	FreeBSD
2007	Primera tecnología de contenedores- LCX	IBM
2013	Primera tecnología de contenedores multiplataforma - Docker	Solomon Hykes
2015	Primer orquestador de contenedores - Swarm	Docker Inc.

Tabla 2: Evolución hacia los orquestadores. Fuente: propia

Patrones de investigación

Haciendo un repaso por la tendencia a través del tiempo se puede ver en el inicio de la computación, la centralización de los centros de cómputo, ya que en los 60's existió una orientación hacia la utilización de las potentes mainframes de forma centralizada, tanto a nivel de cómputo como físicamente.[13] Esto gracias a la innovación propuesta por John McCarthy. A principios de los 70´s se empiezan a crear las primeras implementaciones Clientes-Servidor, donde se caracteriza que los clientes son "pesados", es decir, muchas veces tienen las reglas de negocio, y hasta la conexión directa a base de datos. Nace con el lenguaje de Decode – Encode Language (DEL) que permite a una computadora recibir órdenes de otra computadora remota[14]. Este lenguaje fue desarrollado por el grupo de investigadores que crearon la ARPANET, que luego evolucionó a la hoy conocida Internet.

A medida que el hardware se fue abaratando se empezó a definir que una de las formas de escalar sistemas es paralelizar hardware. El primer clúster comercial que surgió fue en 1977, llamado ARCNet [15]. El mismo permitió compartir todos los recursos de los ordenadores conectados.

[13] Fuente: http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PR360.html, recuperado el 03/07/18
[14] Fuente: https://www.rfc-editor.org/rfc/rfc5.txt, recuperado el 03/07/18
[15] Fuente: http://arcnet.cc/history.htm, recuperado 04/07/18

Con la irrupción de Internet, las posibilidades aumentaron, y con ella SOA, que es la Arquitectura Orientada a Servicios [16]. Aquí se empieza a pensar en sistemas que podrían estar distribuidos que conforman distintos servicios que se comunican por protocolos de red. Una de las implementaciones más conocidas y más implementadas de los WebServices. Estos tienen contratos bien definidos que permiten una comunicación entre los sistemas consumidores de estos servicios con bajo acoplamiento [17].

Si bien el concepto era mucho más viejo, la popularidad de los sistemas distribuidos P2P empieza con Napster. Esta red permitía que no hubiera un control centralizado de la información, sino que cada cliente conectado era equivalente y tenía los mismos privilegios que los demás integrantes del sistema. Estos sistemas distribuidos se caracterizan por ser muy eficientes en alta disponibilidad y aumento del rendimiento en transferencia por dividir las transferencias entre distintas fuentes (Barkai, 2000, p. 3).

A mediados de 2006, aparece un jugador disruptivo en ofrecimiento de servicios Cloud. Amazon con su Amazon EC2 y luego hasta nuestros días con su servicio Amazon Web Services (AWS). Lo disruptivo aquí es ofrecer infraestructura como servicio (IaaS, Infraestructure as a Service), es decir que podemos pagar por utilizar los servidores de Amazon, por distintos criterios. Las características que tienen estos recursos es que los mismos pueden escalarse fácilmente (EC es por Elastic Computing) y podrían hasta elegirle donde se desean que estén los mismos. Esto último por ejemplo sirve para atender a Alemania, país que no permite que los datos puedan ser almacenados fuera del país. Todos estos beneficios son totalmente transparentes para el usuario. Luego aparecieron Azure, Google Cloud y muchos más. (Wittin, 2017, p. 8)

Y finalmente llegaron los Microservicios en 2012. Estos son una variación de la Arquitectura Orientada a Servicios donde los servicios son muy especializados y utilizan protocolos livianos. La gran diferencia con SOA es el tamaño del desarrollo realizado entre una plataforma SOA que generalmente cumple con un contrato, ante la no existencia de esto en la utilización de microservicios (Sharma, 2016, p. 39). Esto se suma a la ola de los DevOps, que impulsan el delivery continuo para que los mismos sean promovidos por las distintas

[16] Fuente: http://www.opengroup.org/standards/soa, recuperado el 04/07/18
[17] Fuente: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice, recuperado el 04/07/18

etapas de desarrollo de software. A continuación, se muestra en la tabla 3 que sintetiza la evolución de la arquitectura de las empresas en el tiempo de acuerdo a lo descripto.

Año	Hito
1964	Centralización de cómputo – Primer Mainframe de IBM
1970	Cliente Servidor
1977	Primer clúster de servidores comercial
1990	Arquitectura Orientada a Servicios – Web 2.0 - Descentralización
1999	Sistemas Distribuidos – P2P - Napster
2006	Primer servicio comercial de Cloud Computing – Amazon (vuelta a la centralización)
2012	Microservicios

Tabla 3: Evolución de la arquitectura de empresas.

Contenedores

Características

Según Docker (2016), es normal que se confundan las máquinas virtuales con los contenedores, pero en realidad no son lo mismo, cuestión que se discutirá más adelante en este mismo capítulo. Podemos sinterizar las características de la siguiente forma:

- Necesitan correr sobre un sistema operativo, ya que no administran directamente los recursos y lo delegan al mismo.
- Son una nueva forma de desplegar aplicaciones.
- El contenedor permite aislar a la aplicación, pretendiendo ocultar los detalles a la misma brindándole aquellos componentes que necesita para ser autosuficiente, en su defecto, ofrecer la portabilidad.
- Tiene la particularidad de ser flexible a su tamaño de acuerdo a los componentes que se necesiten.
- Livianos, el hecho de levantar los componentes que se requieren para una aplicación en vez de levantar todo el sistema operativo hace que ejecutar imágenes de contenedores sea más ágil.
- Permiten la conexión entre los mismos ya que pueden crearse redes virtuales para los mismos.

Implementación de Contenedores: Docker, Rkt

Los contenedores surgieron en Linux, y como la gran mayoría de estos proyectos son software libre. El hecho de que sea así es bueno, ya que la tecnología está disponible y generalmente es soportada por la comunidad y los desarrolladores, el problema es que pueden generarse múltiples opciones que ofrecen pequeñas diferencias. A continuación, se presentan las dos tecnologías más conocidas:

- **Docker:** es la tecnología de contenedores con mayor adopción en el mundo. Como explica Dua (2016), Docker toma varias funcionalidades del kernel de Linux para construir las imágenes de Docker. Entre estas tecnologías tenemos namespaces, cgroups, SELinux, perfiles de AppArmour junto a tecnologías de sistema de archivos como AUFS y BTRFS.

- **Rkt:** desarrollado por el proyecto Linux CoreOS, con el objetivo de ser un contenedor seguro. Esto lo logra utilizando parámetros aislados a nivel de contenedores o a nivel de aplicación. Disponible en múltiples distribuciones de Linux. Al ser posterior que Docker, tiene mucho menor segmento de mercado. Suma ser una tecnología compatible con App Container Specification, que es un estándar de formato de contenedores[18].

Existen otras tecnologías que empaquetan programas brindando una funcionalidad parcial a la que se logra con un contenedor, como SnapCraft, AppImage o Flatpak, estos básicamente logran un paquete agnóstico que se puede portar entre distintas distros Linux, sin necesitar por ejemplo, instalar dependencias o requerir privilegios root para correr. Pero carecen, por ejemplo, de posibilidades de generar redes virtuales por no tener su propia IP que es necesario para armar clúster para sistemas de alta disponibilidad. Por otro lado, este tipo de contenedores ven al sistema de archivos host, y pueden contener aplicaciones de usuario.

Contenedores vs máquinas virtuales

La confusión principal con la aparición de los contenedores fue el hecho de pensar que eran lo mismo a las máquinas virtuales. Tal como se explica en el paper de Docker (2017), aquellos profesionales que van tomando contacto con Docker piensan que es una máquina virtual liviana. Es normal que se asocien dichas tecnologías por el hecho de que tienen algunas similitudes como la de ofrecen un ambiente aislado en donde correr una aplicación. En ambos casos se representan en archivos binarios que pueden ser movidos entre diferentes computadoras. Pueden existir otras similitudes, pero estas son las más importantes.

Como se puede apreciar en la siguiente figura, ilustración 3, la aplicación se encuentra corriendo sobre una maquina virtual que tiene un sistema operativo, que a su vez corre sobre la base de un sistema operativo host. Esto genera lo que llamamos anteriormente overhead (sobrecarga).

[18] Fuente: https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html#rkt-vs-docker, recuperado el 25/06/2018

Ilustración 3: Arquitectura de VMs. Fuente: (Docker, 2016, p. 6)

Empezando hablar de las diferencias la arquitectura de los contenedores es diferente a la de una comúnmente llamada VM (por Virtual Machine). Las VMs son autocontenidas y ofrecen seguridad a visitantes no deseados. Por otro lado, tienen sus propios recursos, como memoria, procesador y disco. Toda VM en realidad es un completo sistema operativo que puede tener 0 o más programas incorporados.

Sin embargo, los contenedores también ofrecen protección sobre visitantes no deseados, pero sobre una infraestructura común, es decir, los contenedores no manejan sus propios recursos, sino que los comparten. Por otro lado, los contenedores pueden variar mucho en su tamaño ya que no necesitan incluir un sistema operativo completo. Las aplicaciones solo necesitan tener aquellas dependencias que necesiten, lo demás lo obtendrán del sistema operativo donde van a estar ejecutándose. Esto demuestra la primera limitación de los contenedores, las aplicaciones tienen que estar contenidas en los sistemas operativos en los que fueron diseñados y el contenedor también tiene que correr sobre el mismo tipo de sistema operativo. En simples palabras un contenedor de Linux, no puede correr en Windows. Con lo cual podríamos definir a los contenedores como una tecnología para despliegue, pero no una tecnología para virtualización. Y otra característica muy importante es que la imagen de un contenedor es sin estado (stateless), esto quiere decir que cualquier cosa que se haya guardado en un contenedor no va a estar disponible si el contenedor se cae y vuelve a levantarse. Para

evitar este problema, el contenedor tiene un mecanismo para mapear directorios del sistema host en una ubicación en su propio sistema de archivos, de esta forma los datos quedan por fuera del contenedor (Docker, 2016, p. 3-4). Se sintetizan las diferencias de tecnologías en la siguiente tabla:

	Contenedor	Máquina virtual
Velocidad de booteo	Menor por no levantar un sistema operativo completo.	Mayor por levantar sistema operativo completo.
Tamaño	Solo contiene librerías especificar y la aplicación para que sea autocontenida	Incluye aplicaciones, librerías y el sistema operativo completo.
Madurez	Tecnología nueva, aparición en 2013	Tiene mayor madurez de desarrollo, primera aparición en 2000.
Seguridad	Los contenedores comparten infraestructura, con lo cual es posible acceder a todos desde un mismo punto.	Tiene su propia seguridad ya que funcionan como sistemas independientes
Agilidad para el desarrollador	No requiere levantar un entorno completo por lo cual otorga mayor rapidez en debugging, testing	Es costoso en tiempo y esfuerzo configurar y levantar una VM para realizar tareas de debugging y testing

Tabla 4: Beneficios de Contenedores y VMs. Fuente: (Docker, 2016, p. 3-4)

Ahora bien, como vimos son tecnologías diferentes que si lo miramos desde el punto de vista de la actualidad se complementan. El hecho de que los contenedores son dependiente del tipo de sistema operativo, se puede resolver desplegando sobre distintas máquinas virtuales de distintos sistemas operativos. Por otro lado, la una máquina virtual puede tener muchos recursos asignados y los contenedores de distintas aplicaciones los que se ejecuten sobre el

mismo para aprovechar mejor los recursos. Este es el escenario más común ya que en nuestros días es difícil encontrar infraestructura que no implemente virtualización.

De acuerdo a lo presentado podemos sintetizar los siguientes beneficios entre los contenedores y las máquinas virtuales, que combinado en contenedores dentro de máquinas virtuales pueden aprovecharse.

Arquitectura de los contenedores

Como se explicó en la sección anterior, los contenedores se virtualizan a nivel de sistema operativo, es decir, que necesitamos que corra sobre un sistema operativo base. A continuación, se muestra en la ilustración 4 la arquitectura de los contenedores:

Ilustración 4: Arquitectura de Contenedores. Fuente: (Docker, 2016, p. 6)

En un primer lugar están los recursos como la base de la arquitectura, que será la memoria RAM, procesadores, unidades de disco, interfaces de red que existen en el sistema donde correrán los contenedores. Para administrar estos recursos tenemos un sistema operativo, sobre el cual correrá un runtime de contenedores. Tal como explica Smith (2017), el runtime de contenedores es el proceso que corre y controla los contenedores en cada máquina que se pretenda correr contenedores. También es el motor que permite funcionar a clústeres de contenedores. Provee una API (Application Protocol Interface) con varias herramientas para manipular los contenedores desde una consola (Smith R, 2017, p. 32).

Luego sobre esta capa intermediaria tendremos los contenedores propiamente dichos, los cuales estarán compuestos de los ejecutables (bins) y librerías (libs) necesarios para que los ejecutables puedan funcionar.

Ciclo de vida de contenedores

Los contenedores son definidos mediante una especificación declarativa que indica cuales cada una de las capas (layers) que se ejecutaran en un contenedor.

El ciclo de vida de los contenedores lo podemos sintetizar de la siguiente forma:

- Debemos construir la imagen

- Ejecutarla con lo cual se convierte en un contenedor

- Podemos detener un contenedor.

Los contenedores no guardan los cambios realizados dentro de ellos, por lo cual, si se requiere guardar el estado de ese contenedor en ejecución, se puede hacer mientras el mismo se mantenga en ejecución, ya que una vez detenido los cambios se pierden y vuelven al estado definido por la imagen (Kolhi, 2017, p 14-15).

A continuación, se resume el proceso en la ilustración 5:

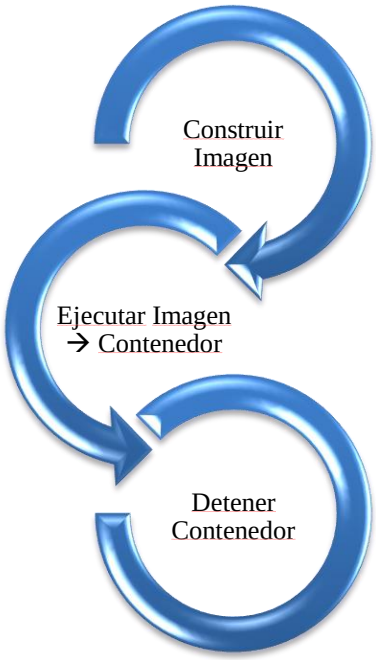


Ilustración 5. Ciclo de vida de contenedor. Fuente: (Kohli, 2017, p. 14-15)

Aplicaciones con múltiples contenedores

Al momento, se explicó de contenedores y su composición, sus beneficios y ventajas sobre máquinas virtuales convencionales. Lo que no se debe perder de vista es que un contenedor es un elemento básico de despliegue, es decir, difícilmente tengamos un solo contenedor con una base de datos, servidor de archivos y servidor de aplicaciones para desplegar una aplicación web. Una de las primeras restricciones que tenemos es que los contenedores no deberían almacenar información dentro de su sistema de archivos virtual ya que al reiniciarlo la información se pierde (stateless). A partir de lo visto tenemos que pensar en descomponer las aplicaciones en contenedores con la funcionalidad acotada y de esta forma tener bajo acoplamiento. A modo de ejemplo, en la figura 6, descomponemos en 3 contenedores, uno que tiene la base de datos, otro el servidor de aplicaciones y por ultimo un contenedor que provee acceso al sistema de archivos.

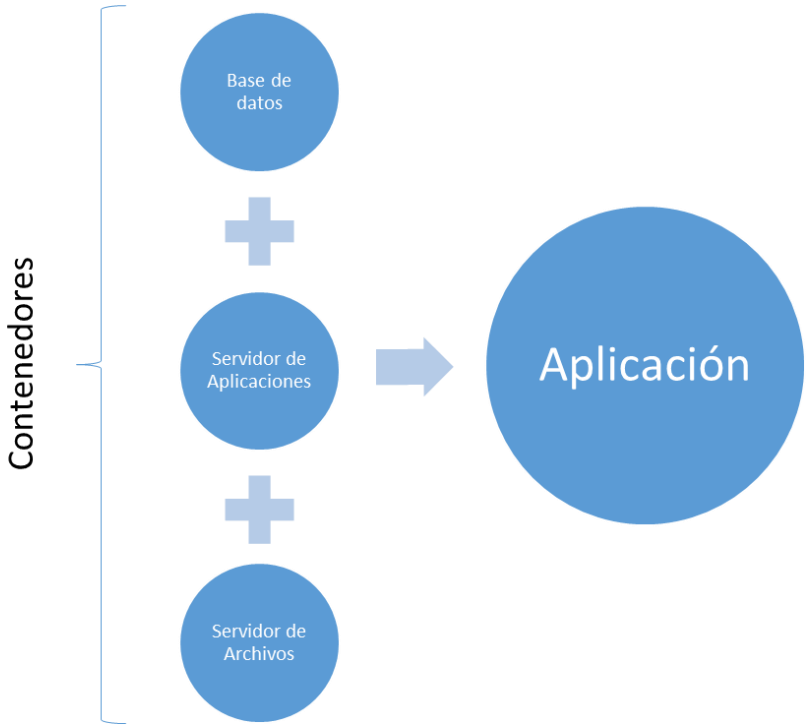


Ilustración 6: Descomposición de aplicaciones. Fuente: Propia

La posibilidad con esta tecnología es que distintos contenedores puedan correr en la misma máquina. También facilita el mantenimiento, ya que se pueden actualizar los componentes que lo requieran sin afectar a todo el contenedor. Por otro lado, los contenedores habilitan interfaces de red virtuales que permiten visualizarse entre ellos y para que sean vistos deben exponerse explícitamente los puertos. En el caso de Docker, tenemos una herramienta específica para hacer aplicaciones multicontenedor, llamada Docker Compose.

Casos de uso de contenedores

Se recomienda utilizar contenedores ante los siguientes casos:

- Aislar aplicaciones para mejorar portabilidad.
- Simplificar despliegues
- Escalabilidad de sistemas

Por otro lado, la utilización de contenedores agrega costos y complejidad. El hecho de que los contenedores sean simples en su punto de vista de despliegue, no quita que se tenga que agregar una capa más de complejidad ya que requieren configuración y un trabajo en definir

la comunicación entre ellos en la etapa de desarrollo. Ese tiempo adicional significa dinero, con lo cual es una inversión que se tiene que recuperar luego a nivel de operaciones del sistema, lo cual nos dará mayor robustez y fiabilidad.

Los contenedores permiten la aplicación de automatización para entregar más valor y agilizar los procesos de despliegue.

Las utilizaciones de los contenedores permiten utilizar mejor los recursos de una computadora, ya que funcionan como procesos en un sistema operativo y generan menor sobrecarga (overhead).

Tendencia de adopción de esta tecnología a nivel de empresas

Dada la imposibilidad de hacer un estudio de tal magnitud que escapa al alcance del presente trabajo, el mismo se basó en una investigación realizada en junio de 2016 por la reconocida consultora Forrester por pedido de la reconocida empresa Redhat llamada "The State of Contaneirization". La misma se basó consultando a 150 empresas en Estados Unidos, Reino Unido, India, China y Alemania. Las mismas pertenecen a distintos sectores, como empresas de software/IT, manufactura, químicos y metales, servicios financieros y productos de consumo.

Según Forrester (2016), la rapidez del desarrollo continuo y entregar software de alta calidad ya no son diferenciadores competitivos como en el pasado. Es un requerimiento ser una empresa viable, sin importar la industria. Como consumidores, se demandan mayores capacidades de innovación y las firmas requieren infraestructuras cada vez más robustas y aplicaciones que sirven incrementalmente consumidores cada vez más disimiles. Los profesionales de IT involucrados en el ciclo de vida del desarrollo de software demandan por herramientas y procesos que reduzcan la fricción y los tiempos (llamado generalmente Time to market).

La tecnología de contenedores existe desde hace muchos años, pero el interés sobre esta tecnología coincide con la aparición del proyecto open source Docker en 2014.

Ilustración 7: Interés por Docker a partir de su aparición. Fuente: Google Trends, extraído el 19 de mayo del 2018.

Dado que Docker ayuda a las nuevas demandas del mercado, al 2016 ya el 48% estaba usando contenedores en alguna parte de su etapa de desarrollo u operaciones y los restantes planeaban incorporarlas al siguiente año.

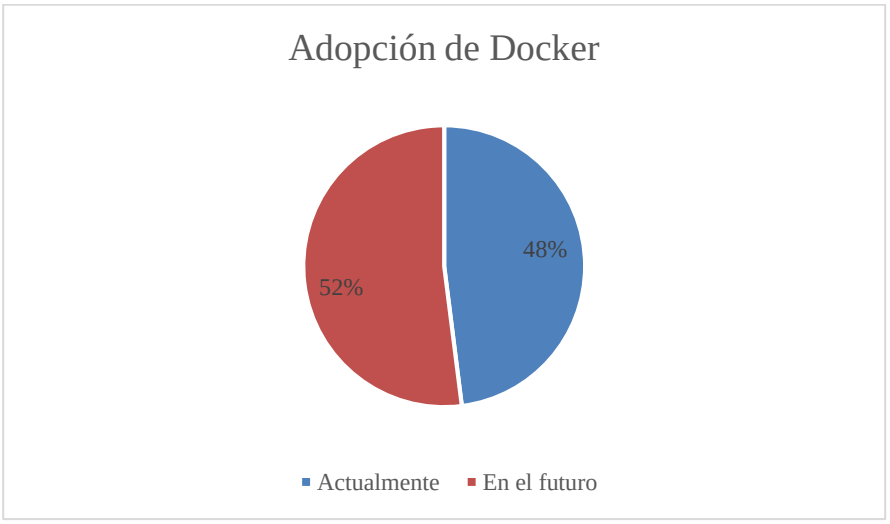


Ilustración 8: Adopción de Docker. Fuente: (Forrester, 2016, p. 3)

Respecto a los factores que fueron importantes para decidirse a implementar contenedores se capturaron las siguientes razones por su orden de mayor a menor importancia:

- Eficiencia mejorada de los contenedores respecto a máquinas virtuales

- Oportunidad de lanzar funcionalidades rápidamente
- Manejo simplificado de administración y despliegue de aplicaciones
- Interacción rápida.
- Oportunidad de mejorar la interacción entre DevOps
- Rentabilidad de contenedores a través de plataformas/maquinas.
- Mantenimiento simplificado respecto a máquinas virtuales.
- Reducción de tiempos de configuración.

Ahora bien, son bien conocidos los beneficios brindados por los contenedores, pero cuales son las características a los cuales las empresas más les interesa hoy. Forrester (2016), tuvo en cuenta esto y detecto como puntos importantes del stack de contenedores las siguientes características (de mayor a menor):

- Seguridad
- Mejor flexibilidad de despliegue
- Aprovisionamiento más veloz.
- Facilidad en creación y despliegue de aplicaciones y servicios modularizados.
- Habilidad de entregar/desplegar aplicaciones rápidamente.
- Posibilidad de escalar verticalmente hacia arriba o abajo fácilmente.
- Mejor movilidad/portabilidad de aplicaciones.
- Simplicidad de creación/configuración de ambientes.

Seguridad en contenedores

En la sección anterior se enfatiza que es de sumo interés para las empresas el factor de seguridad de los contenedores. La seguridad de contenedores juega un papel principal en ambientes productivos. Hay ciertos puntos que nos remarca en su paper Scott Gallager (2016), en donde indica tener especial atención en las siguiente buenas practicas:

1. Tener en cuenta que cualquiera que tenga acceso a la plataforma de dockers tiene acceso a cada uno de las instancias que están corriendo, con las posibilidades de detenerlos o borrarlos e iniciar nuevas instancias.

2. Recordar que se pueden montar volúmenes de Docker desde unidades del sistema host en forma de solo lectura. Con lo cual restringir de esta forma si no es necesario un privilegio mayor.
3. Existen tools como "Docker Bench For Security" que nos van a ayudar a configurar eficientemente nuestros contenedores, validando docenas de buenas prácticas.
4. Se pueden auditar los cambios en imágenes Docker utilizando las herramientas de línea de comandos.

$ docker diff

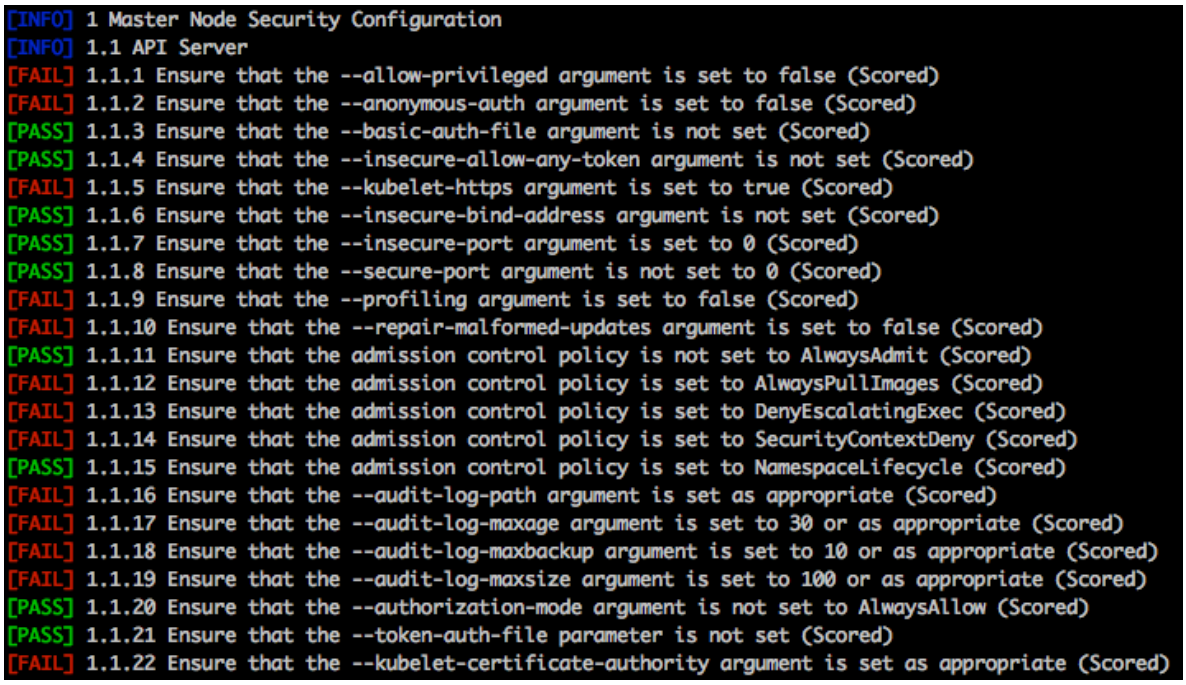


Ilustración 9 – Salida de script "Docker Bench for Security"[19]

[19] fuente: https://github.com/docker/docker-bench-security, extraído 19/05/2018

Orquestadores

Introducción

Inicialmente, los servicios de internet corrían directamente sobre computadoras físicas y la vida iba bien. Para escalar los servicios y manejar los picos de demanda solo se tenía que comprar suficiente hardware para manejar la carga. Cuando ya no existía la carga, el hardware quedaba en desuso o desperdiciado. Entonces, luego siguieron las máquinas virtuales y la vida era buena. Las comúnmente llamadas VMs (por las siglas en inglés de máquinas virtuales) pueden ser escaladas al tamaño necesitado. Muchas VMs pueden correr en el mismo hardware. Si hay un incremento en las demandas, nuevas VMs pueden ser iniciadas siempre que haya recursos disponibles en el servidor físico. Mas puede ser realizado en menos hardware. Aún mejor, nuevas VMs pueden ser iniciadas en minutos cuando sean necesarias, y destruidas luego cuando la demanda decaiga. Luego, con la aparición de cloud computing, fue posible hacer outsourcing a servicios de Amazon, Google y Microsoft. Ahí entonces nació el elastic computing.

Volviendo a las VMs, ellas también tienen sus problemas. Cada una de ellas requiere que memoria y almacenamiento sea reservado para soportar al sistema operativo. Además, cada plataforma de virtualización tiene su propia forma de hacer las cosas. La automatización que funciona para un sistema necesita que ser totalmente reescrito para funcionar con otro. Depender del proveedor de virtualización se convierte en un problema.

Entonces aquí vino Docker. Lo que hacen las VMs por el hardware, contenedores lo hacen por la VM. Los servicios pueden ser iniciados en múltiples servidores e incluso múltiples proveedores (entornos híbridos). Una vez desplegados, contenedores puede ser iniciados en segundos sin la sobrecarga de una máquina virtual completa. Y aún más, las aplicaciones desarrolladas en Docker pueden ser desplegadas exactamente como fueron construidas, minimizando problemas de configuración y mantenimiento de paquetes.

Entonces surge la pregunta, ¿Qué hace todo esto? La respuesta el proceso es llamado orquestación, y como una orquesta, hay un número de elementos necesarios para construir un clúster funcional. En dicho capitulo se enumeran los componentes necesarios para tener una aplicación escalable, servicios confiables con rápidos y despliegues consistentes (Smith, 2017, p. 31) traducción propia.

Características

Según Vohra (2016), los orquestadores tienen cuatros grandes objetivos:

1. **Programar – Coordinar:** decidir que contenedores correrán en que nodo.
2. **Distribuir la carga:** enviar la carga, generalmente llamadas tareas, de acuerdo a la disponibilidad de recursos de cada nodo.
3. **Escalar:** aumentar o disminuir la cantidad de contenedores de acuerdo a la carga de la aplicación.
4. **Comunicar:** Comunicación dentro de los contenedores

Arquitectura de solución con orquestadores

Si se tiene un único nodo con Docker, no hay necesidad de herramientas adicionales. Los orquestadores ayudan a manejar clústeres de contenedores. Los contenedores necesitan conocer el estado del clúster para poder comunicarse, con lo cual existen diferentes roles, nodos que almacenarán el estado del clúster y redirigirán las tareas (managers), y nodos que simplemente procesarán las tareas (workers). Esto permite que el sistema distribuido pueda coordinar las tareas a dichos nodos que se encuentren trabajando (activos). Este es el principio de funcionamiento del Elastic Computing. Los managers eligen un líder del cluster el cual dirigirá el estado del clúster y lo replicará a los demás managers.

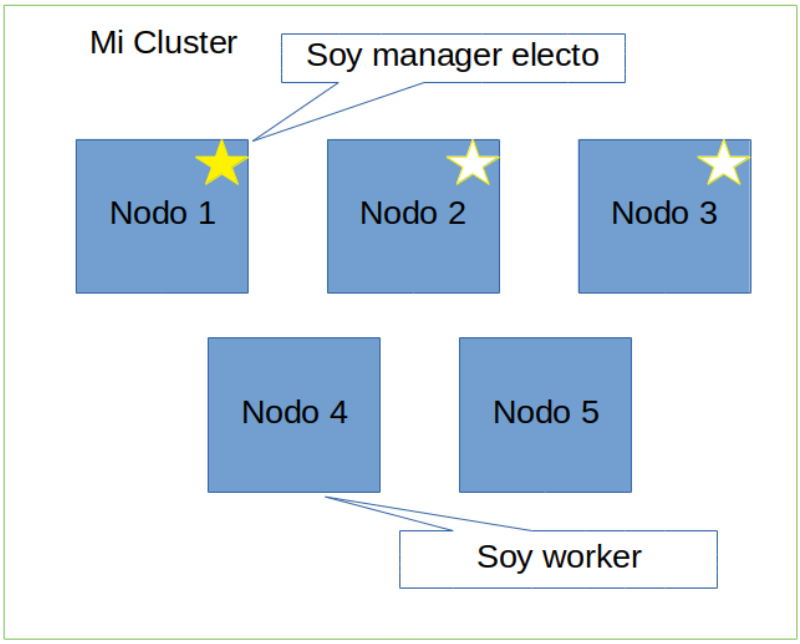


Ilustración 10: Arquitectura Clúster. Fuente: propia

Los componentes del clúster, los podemos sintetizar en:

Nombre del clúster: esto va a definirnos un conjunto de nodos que pertenecen a un clúster

Nodo: es cada uno de los contenedores que se ejecuta en un clúster. Estos pueden ser:

- Manager: es un tipo de nodo que almacena el estado del clúster, es decir, que nodos forman parte y cuál es el estado de cada uno de ellos. Generalmente entre el grupo de nodos managers se aplican algoritmos para que se elija dinámicamente un líder que maneje el estado del clúster general.
- Workers: son nodos que resuelven tareas y dan un resultado.

Nombre de nodo: es el identificador del nodo.

Tareas: es una unidad de trabajo que llega al clúster y que puede ser atendida por el mismo.

Servicio: es una o más tareas que pueden ser atendidas por un nodo.

Réplicas: número de copias que se tendrán de un contenedor en el clúster para asegurar la alta disponibilidad.

Despliegues Manuales vs Despliegues con orquestadores

Respecto a lo desarrollado en el presente trabajo de investigación podemos enumerar las diferencias entre despliegues manuales en contraste con despliegues con orquestadores.

Despliegues Manuales	**Despliegues con Orquestadores**
Se requiere un monitoreo y acciones reactivas manuales, a veces no logrando evitar las caídas.	El monitoreo puede o no existir y las acciones son reactivas automáticas, evitando las caídas.
Proceso lento.	Proceso rápido.
Ante caídas del sistema, puede demorarse en levantar un servidor ocasionando daño a la imagen de la empresa.	Se pueden configurar los orquestadores para levantar automáticamente la caídas de nodos y desviando las tareas a nodos activos del clúster.
En caso de aumentar la demanda, se deben crear VM o disponer equipos físicos con las dependencias listas (aprovisionamiento) para desplegar las aplicaciones.	El aprovisionamiento y despliegue se unifican haciéndose automáticamente de acuerdo a las configuraciones realizadas en el clúster.

Tabla 5: Comparativa Despliegue Manual vs Despliegues con Orquestadores. Fuente: propia

Orquestadores Populares

Existen muchas opciones en el mercado de orquestadores que realizan en mayor o menor medida las mismas operaciones. En este trabajo se tomaron las tres tecnologías que son referentes en el mercado: Swarm (propio de Docker), Kubernetes y Apache MesOS.

Swarm

Como explica Smith (2017), Docker Swarm es el orquestador nativo de Docker, la tecnología de contenedores más popular. Y como es de suponerse usar Docker Swarm es tan simple como utilizar Docker. Para poder comunicar distintos hosts que corran Docker runtime se requieren los siguientes puertos en cada uno de ellos:

- Puerto TCP 2377 para administración del clúster
- Puertos TCP/UDP 7946 para comunicación entre nodos
- Puertos TCP/UDP 4789 para red superpuesta (overlay network)

El algoritmo de consenso utilizado es RALF, donde existe un nodo líder donde pasan todos los cambios de clúster, nodos candidatos que pueden reclamar ser líderes si el mismo no se encuentra disponible y finalmente los seguidores que atienden las peticiones de nodo líder.

Por otro lado, describe a Swarm como una herramienta que da la ilusión de administrar un simple pero gran Docker host hecho por muchos Docker host, como si fueran uno y tienen un punto de entrada de comandos. Esto permite orquestar y operar un cierto número contenedores en dichos hosts, usando las rutinas de Docker tools, las primitivas de Docker, el cliente Python de Docker o incluso la herramienta curl usando Docker Remote API. (Soppelsa, 2015, p. 9).

Apache Mesos

Apache MesOS es un núcleo distribuido. Este permite que se puedan trabajar los recursos de un clúster como si fuera único. Mesosphere y DC/OS (Datacenter Operating System) combinados proveen un sistema robusto antes cualquier carga de trabajo, incluyendo contenedores Docker para el cual tiene soporte nativo. Este orquestador tiene un soporte más extendido a nuevas tecnologías, lo cual lo hace muy flexible. Como algoritmo de consenso, es utilizado PAXOS. (Smith, 2017, p. 178).

Originalmente iniciado en la Universidad de Berkeley en 2009, es un proyecto maduro y ha sido utilizado en producción con éxito, como el caso de Twitter. Es incluso una plataforma multipropósito más grande que Kubernetes y Swarm, siendo multiplataforma (puede correr en Windows, Linux, MacOS) y capaz de correr trabajos heterogéneos. Es decir, se pueden estar administrando un clúster de contenedores al mismo tiempo que puros trabajos de BigData (Hadoop o Stark) a otros tipos de trabajos, incluyendo integración continua, procesamiento en tiempo real, aplicaciones web, almacenamiento de datos y aún más. (Soppelsa, 2015, p. 16).

Kubernetes

En su libro Smith (2017) explica que Kubernetes (conocido también como k8s) fue iniciado en Google como una forma propia de administrar los contenedores y fue lanzado en junio de 2014. Otras compañías rápidamente se unieron al proyecto y hoy es el backend para Red Hat OpenShip, Deis Workflow y la plataforma Fabric8 como PaaS (Platform as a Service).

Todos los días, más y más organizaciones están desplegando Kubernetes para orquestar sus servicios[20].

Kubernetes es una complicada y grande pieza de software. Tiene algunos conceptos propios como el de POD, que se refiere a una organización de uno o más contenedores que componen a una aplicación y es la unidad mínima de despliegue. Los mismos pueden ser replicados, desplegados y actualizados independientemente de otros pods corriendo en el clúster. Cada pod tiene su dirección IP que es usada para exponer puertos utilizados por los contenedores y permitir la comunicación con otros pods.

En el caso de Kubernetes, al igual que Swarm, utiliza RALF como algoritmo de consenso.

Análisis F.O.DA y comparativa entre orquestadores

A este punto de la teoría desarrollada es importante identificaron cuáles son las Fortalezas, Oportunidades, Debilidades y Amenazas para una organización con un análisis F.O.D.A:

Fortalezas:

- Elasticidad (escalabilidad horizontal) de capacidad de procesamiento de nuestra demanda.
- Open source, la mayoría de la tecnología es tecnología abierta sin gastos de licenciamiento y con amplio soporte en la comunidad.
- Reduce las configuraciones de despliegue y el tiempo del mismo.

Debilidades:

- Tecnología nueva, con lo cual muchos profesionales de sistemas no tienen conocimiento de cómo trabajar con contenedores.
- Mayor complejidad al principio de la implementación de estas herramientas.
- No ofrece al momento un soporte a Windows como para Linux. Al ser una tecnología proveniente de Linux es mucho más madura para esa plataforma.

Amenazas:

[20] Fuente: https://www.cncf.io/blog/2017/06/28/survey-shows-kubernetes-leading-orchestration-platform/ , recuperado 02/07/2018

- El hecho de que sea una tecnología nueva y por el desconocimiento puede generar desconfianza en el usuario en el usuario.

Oportunidades:

- El soporte a tecnologías Windows, tiene mucho por recorrer.
- Apache MesOS ofrece mayor flexibilidad para el armado de ambientes híbridos, es decir, partes pueden estar alojado en servidores propios y parte en servicios cloud.

Teniendo en cuenta el análisis previo, se comparan las tecnologías más fuertes en el mercado relacionadas orquestación de contenedores en tabla 4.

	Swarm	**Kubernetes**	**MesOS**
Madurez	3 años	3 años (en realidad está basado en Google Borg que tiene más de 16 años)	9 años
Comunidad	Las contribuciones son de Docker Inc.	Es uno de los proyectos opensource más importantes en GitHub (número 1 en ranking 2017)	El contribuidor más importante es Mesosphere.
Soporte de comunidad/proveedores (PaaS)	Docker/Moby	Más de 70	Mesosphese
Proveedores públicos Cloud	Azure	Google, AWS, Azure, Open Telekom Cloud,	AWS, Azure, Mesosphere

Fortalezas	Al ser dirigido por Docker, tiene mayor visibilidad de cuál es el futuro del orquestador. Aprovechamiento de nuevas características de Docker, ya que usa la misma API. Swarm Mode: no es necesario un software adicional para que un contenedor sea un nodo en un cluster.	Líder de Mercado por mayor adopción.	Utilizado por algunas empresas grandes como Twitter. Este orquestador puede utilizarse complementariamente con kubernetes o swarm.
Debilidades	Mayor orientación a desarrolladores que a operaciones.	No existen certificaciones. Varias versiones distintas que pueden requerir soporte de terceras partes.	El fuerte de Mesos no es contenedores, sino BigData y monitoreo. Es la solución más compleja, en la mayoría de los casos de requiere implementar frameworks específicos para extender la funcionalidad.

Tabla 6: Comparación de orquestadores. Fuente: (Smith, 2017, p. 20-60)

Implementaciones de Orquestadores y Contenedores

Cloud Computing

Este término es muy popular hoy en el mundo IT y beneficia la utilización de orquestadores con contenedores. Esto podría justificarse por ciertos factores que contribuyen a que sea una solución importante como lo explicado a continuación.

Según lo que explica Wittin (2017), hay un creciente consenso entre analistas, proveedores cloud y usuarios que cloud computing es el más alto nivel que terceras partes ofrecen un servicio de cómputo, disponible cuando es necesitado y que puede ser escalado dinámicamente ante los cambios de necesidades. Este concepto representa la salida de las normas de desarrollo, operaciones, y el manejo de sistemas/IT. Desde la perspectiva económica, no solo la adopción de esta tecnología tiene vastos beneficios económicos, sino que también posiciona a las empresas a ser más agiles y flexibles (Wittin, 2017, p. 3-4) traducción propia.

Los cinco principios de Cloud Computing

Podemos sintetizar los principios de cloud computing en cinco según Wittin (2017):

1- Pool de recursos disponibles para cualquier usuario suscripto.
2- Recursos de computación virtualizados para maximizar la utilización de hardware
3- Elasticidad hacia arriba o abajo de acuerdo a la necesidad.
4- Creación de nuevas máquinas virtuales y borrado de las existentes de forma automatizada.
5- Facturación de solo los recursos utilizados.

Se desarrollan a continuación los detalles de cada principio:

Pool de recursos disponibles para cualquier usuario suscripto

La primera característica de los servicios de Cloud Computing es que es un pool de recursos que pueden ser externamente comprados y controlados en cambio de ser internos recursos no agrupados y dedicados. Cuando se califica que cualquier usuario suscripto tiene el acceso a estos recursos, quiere decir que cualquier persona física o jurídica que tenga una tarjeta de crédito puede acceder al mismo.

Considerando el ejemplo de un sitio web corporativo, existen 3 tipos de opciones de despliegue comúnmente usadas en el día de hoy. La primera opción, es la de tener el hosting propio, la segunda es la de no tener un datacenter centralizado y no exclusivo (cloud computing) y finalmente es la de tener el outsourcing del servidor, pero dedicado.

Primero diferenciando entre el hosting propio de las opciones de outsourcing, tiene sus implicaciones económicas. Las primeras implicaciones son que los gastos de capital (CAPEX) se reducen con la tercerización como así también los gastos de operación (OPEX) por la dinámica de reducirlos recursos que no son utilizados. Esto baja las barreras de iniciar un nuevo proyecto por cuestiones financieras.

En los servicios en hosting propio, las empresas deben presupuestar por adelantado la compra de hardware y licencias de software. En el caso de un servidor tercerizado, los costos son similares a un mes de operación y las contrataciones tienen que ser por un año al menos por adelantado. Aquí están alineados en casi el mismo valor de CAPEX, pero con la diferencia a favor de OPEX ya que la infraestructura no es manejada por la empresa. En un marcado contraste, contratando cloud computing no hay gastos iniciales, ya que con una tarjeta de crédito literalmente se empieza a utilizar la infraestructura en mucho menos tiempo del requiere leer este capítulo.

Estos fundamentos son los más fuertes en el momento de la evaluación en una empresa para decidirse a un modelo de cloud computing contra uno de hosting propio.

Recursos de computación virtualizados para maximizar la utilización de hardware

El segundo principio de cloud computing, está relacionado con el ya antiguo concepto de virtualización. La gran mayoría de las empresas han cambiado de servidores físicos a servidores virtuales entre los últimos 5 a 10 años. En cloud computing es un factor de mucha importancia porque el escalamiento tiene que ser enorme basado en miles de servidores donde cada uno ocupa espacio físico, consume electricidad y requiere enfriamiento. Conseguir el máximo aprovechamiento tiene implicancia directa con los costos eficientes en la infraestructura.

El ultimo quiebre tecnológico importante, es que un servidor físico es dividido mediante virtualización en muchas máquinas virtuales. Cada una es un servidor con un sistema operativo y un conjunto complementario de aplicaciones. Los servidores virtualizados son la unidad primaria los cuales pueden ser consumidos en una nube, que constituyen un gran conjunto disponible de recursos.

Elasticidad hacia arriba o abajo de acuerdo a la necesidad:

El hecho que exista un gran conjunto de recursos habilita el concepto conocido como elasticidad, el tercero de los cinco principios. La elasticidad es tan importante como concepto de cloud computing, al punto que Amazon decidió llamar su servicio *Amazon Elastic Compute Cloud*.

Elasticidad es sinónimo de escalabilidad dinámica, y se refiere a la capacidad de adaptar el consumo de recursos disponibles en respuesta a que cantidad es la necesitada. Las aplicaciones típicas requieren un nivel base de recursos en un funcionamiento normal, menos cantidad mientras no realizan procesamiento y mayores cantidades ante picos de demanda.

En un mundo no cloud, las empresas necesitan no solo poder soportar un funcionamiento normal de las operaciones sobre las aplicaciones, sino que también tienen que ofrecer un funcionamiento correcto en situaciones de picos de demanda para no afectar la percepción de los usuarios. Esto se traduce directamente en que se necesita más hardware de forma previsional. Normalmente una aplicación nace con un limitado conjunto de recursos, que va expandiéndose a medida que la aplicación va creciendo en funcionalidad y requerimiento, y siempre sumándole el adicional para manejo de altas demandas. Esto en las empresas es muy difícil de mantener, porque para que se compre nuevo hardware y se instale puede tomar en el mejor de los casos semanas y hasta meses.

Creación de nuevas máquinas virtuales y borrado de las existentes de forma automatizada:

La habilidad de automáticamente mediante API de realizar aprovisionamiento y despliegue de una nueva instancia de una máquina, es el cuarto principio. Una aplicación cloud puede aprovisionar nuevas instancias bajo demanda y esas instancias son disponibilizadas en minutos. Una vez que el pico de demanda ocurrió y los recursos no son necesarios, las

instancias son quitadas y puestas fuera de línea, con lo cual estas no van a ser facturadas. Es decir que los costos adicionales se pagaran solamente por el momento donde las instancias fueron utilizadas y por las que se mantienen en línea.

Facturación de solo los recursos utilizados:

La quinta característica distintiva de cloud computing un sistema de facturación medido. En el caso de hosting propio usualmente hay un costo de contratación inicial y luego un contrato fijo anual. El modelo de cloud rompe esta barrera económica ya que se abona por lo utilizado. No hay un contrato fijo ni condiciones mínimas de consumo especificadas.

Típicamente se reservan recursos como van siendo necesitados y estos son pagados en un esquema por hora. Esta ventaja económica no solo beneficia a los proyectos llevados a cabo por empresa IT, sino también a los emprendedores iniciando nuevos negocios. En vez de necesitar conseguir un capital como el que necesitaban en el pasado, pueden utilizar vastos recursos de cómputo por algunos centavos de dólar por hora. Este nuevo esquema niveló las posibilidades de un programador independiente con respecto a una gran corporación.

Cloud privadas como precursoras de Cloud públicas

Dentro de las clasificaciones de cloud/nube, una cloud privada es una variante de la genérica cloud computing donde datacenters internos de una empresa u organización no son puesto en disponibilidad para el público en general.

Si las organizaciones tienen suficientes usuarios y una capacidad económica para tener un buen datacenter que soporte todas las operaciones una cloud privada se va a nivelar en comportamiento a como lo hace una cloud pública, pero a escala menor.

Hace años hubo inversiones muy grandes en equipamiento para armar costosos datacenters, en el día de hoy todo esto se está moviendo a un modelo de cloud pública.

Un punto intermedio son las cloud hibridas, que generalmente se da cuando en organizaciones donde sus datacenter están muy al límite de capacidad y se lo combina con recursos en cloud pública para aliviar la carga (Wittin, 2017, p. 17-18).

Como productos que ayudan a la creación de cloud privadas se encuentran OpenStack o VMWare.

Granularidad Cloud

En el contexto cloud es común escuchar "X as a Service" (X como un Servicio, traducido al español), donde X puede ser P (Plataforma), S (Software), I (Infraestructura), A (Aplicación), F (Framework) y hasta D (Datacenter). Los proveedores no están de acuerdo con están convenciones de nombres porque muchas veces no se sienten encasillados en ninguna de ellas. Se interpreta que "X as a Service" significa bajo demanda, requiriendo mínima o nula inversión. Significa consumible remotamente a través de Internet, y tiene una facturación medida de acuerdo al uso.

Según explica Wittin (2017), podemos clasificar según su granularidad a los servicios cloud:

- **IaaS (Infraestructure as a Service):**

  Es el nivel más bajo de servicio cloud, a veces llamado también HaaS (Hardware as a Service). Un usuario de IaaS tiene acceso al nivel más detallado y con el conjunto más pequeño de funcionalidad preempaquetada. Un proveedor IaaS provee distintos OS (Sistema Operativo por sus siglas en inglés), con sus distintas versiones o sabores en caso de las distribuciones Linux en forma de imágenes de VMs. Estas imágenes pueden ser ajustadas por el desarrollador para correr aplicaciones conocidas o software desarrollado por ellos mismos. Estas aplicaciones pueden correr nativamente en el OS y pueden ser almacenadas para un cierto propósito. El usuario puede traer esas estas instancias online de estas VMs de acuerdo a su necesidad. El uso de estas instancias es medido y facturado por hora.

  El almacenamiento y el ancho de banda son también commodities consumibles en un entorno IaaS, con el almacenamiento típicamente cobrado por Gb por Mes y el ancho de banda cobrado por tránsito hacia dentro y fuera del sistema.

  IaaS provee gran flexibilidad y control sobre los recursos cloud siendo consumidos, pero típicamente más trabajo es requerido por el desarrollador para operar eficientemente en el ambiente.

- **PaaS (Platform as a Service):**

  Los PaaSs se facturan de forma similar a los IaaS: consumo de CPU, ancho de banda y almacenamiento opera bajo modelos similares. La principal diferencia es que PaaS requiere menos interacción con respecto a la infraestructura del sistema. No se necesita interactuar con una administración de sistemas operativos virtuales. En

cambio, se puede dejar a la plataforma abstraer esa interacción y concentrarse directamente en escribir la aplicación. Esta simplificación generalmente viene con el costo de menor flexibilidad y el requerimiento de codificar en el lenguaje de programación soportado por el proveedor particular del PaaS.

- **SaaS (Software as a Service) y FaaS (Framework as a Service):**

  SaaS, como se describe anteriormente, refiere a servicios y aplicaciones que son disponibilizadas en una forma bajo demanda, por ejemplo, Salesforce.com. FaaS es un ambiente adjunto a un SaaS ofreciendo y permitiendo a los desarrolladores extender la funcionalidad preconstruida de la aplicación SaaS. Por ejemplo, Force.com extiende las funcionalidades de Salesforce.com.

  Las funcionalidades de FaaS son útiles específicamente para aumentar y mejorar las capacidades de un sistema SaaS base. Esto se traduce en crear aplicaciones personalizadas y especializadas para una organización especifica o una aplicación de propósito genérico que puede ser utilizada por cualquier cliente del servicio SaaS. Al igual que PaaS, en FaaS el desarrollador solamente puede usar el lenguaje especifico y las APIs provistar por el FaaS.

(Wittin, 2017, p. 14 - 17).

Servicios Cloud Públicos: Google Cloud, Azure y AWS (Amazon Web Services)

Con la aparición de Amazon ofreciendo IaaS (Infraestructura como servicio, por sus siglas en inglés) en 2006, todo cambió para siempre. En muchos casos, para las empresas, es factible mover su infraestructura a las de un proveedor como Google Cloud, Azure (Microsoft) y AWS. Se puede apreciar el crecimiento de adopción de acuerdo a distintos estudios

realizados, como el que se muestra en la gráfica (ilustración 11) donde se muestra el crecimiento en el gasto en cloud públicas según la consultora Nemertes[21].

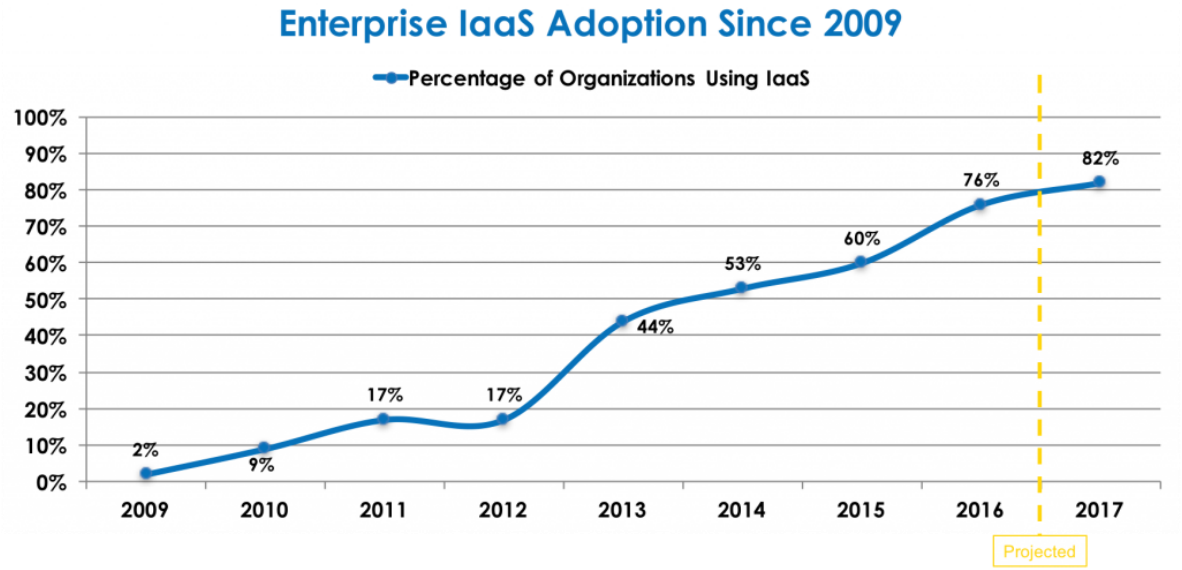


Ilustración 11: Adopción de IaaS en empresas desde 2009. Fuente: Nemertes (2017)

A continuación se hace extracto de una gráfica (ilustración 12) tomada del estudio "The State of Cloud" realizado por RightScale, donde se encuestaron a 997 profesionales de distintas empresas del mundo para conocer la adopción de cloud computing en 2018 respecto al año

[21] Fuente: https://nemertes.com/overwhelmed-floundering-welcome-cloud/, recuperado el 12/07/18

anterior.

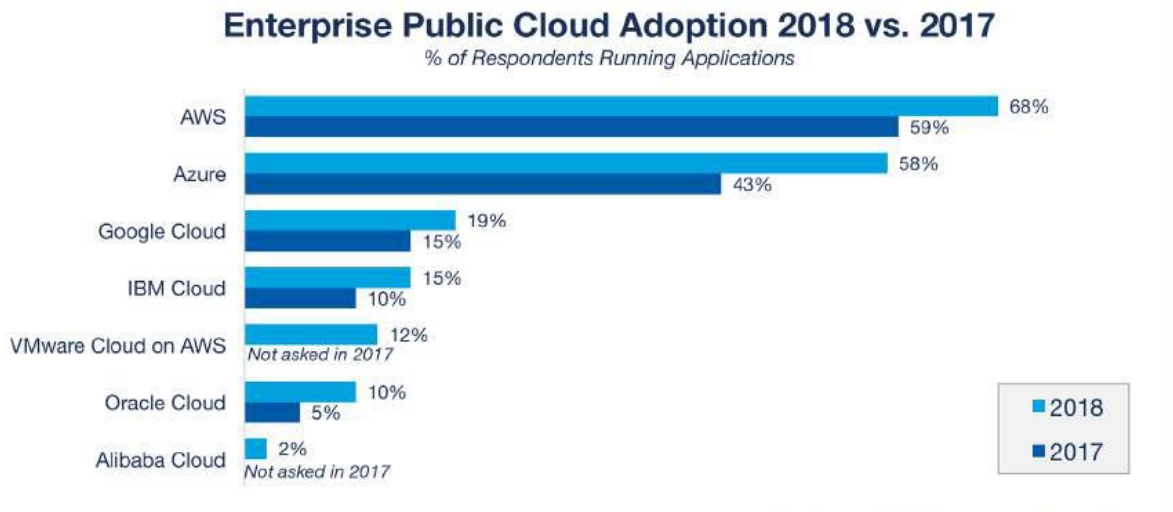


Ilustración 12: Adopción de proveedores públicos cloud. Fuente: (RightScale, 2018, p. 31)

Este alquiler de almacenamiento y poder de cómputo tiene costo por transferencias, espacio en disco almacenado, unidades de CPU utilizadas, RAM utilizada y más. La verdadera ventaja de estos proveedores es su gran capacidad de procesamiento ante una necesidad puntual de procesamiento adicional, lo que se conoce como elasticidad de cómputo. Esto tiene sentido si se realiza automáticamente, como los orquestadores lo pueden hacer. Esto es clave si se quiere ofrecer un servicio de ventas de entradas online o ante otros eventos masivos de venta online como CyberMonday y HotSale, donde muchas veces es difícil tener una estimación de cuantos recursos van a ser necesario.

Comparativa de Servicios Cloud Públicos

A continuación, se comparan los proveedores públicos de cloud computing más importantes del mercado. El problema de comparar proveedores cloud públicos es que no existen estándar de cómo deberían estar constituidas y que servicio/funcionalidades deben ofrecer. En la tabla 5, se ofrecen detalles de las diferencias entre algunos aspectos relevantes que pueden influir en la selección final de uno respecto a los otros.

	AWS	Microsoft Azure	Google Cloud
Numero de servicios	Muchos	Bastantes	Algunos
Numero de Datacenters	9	13	3

Estandares cumplimentados	Estándares comunes (ISO 27001, HIPAA, FedRAMP, SOC), IT Grundschutz (Germany), G-Cloud (UK)	Estándares comunes (ISO 27001, HIPAA, FedRAMP, SOC), ISO 27018 (cloud privacy), G-Cloud (UK)	Estándares comunes (ISO 27001, HIPAA, FedRAMP, SOC)
Lenguajes SDK	Android, Navegadores Web (JavaScript), iOS, Java, .NET, Node.js (JavaScript), PHP, Python, Ruby, Go	Android, iOS, Java, .NET, Node.js (JavaScript), PHP, Python, Ruby	Java, Navegadores (JavaScript), .NET, PHP, Python
Integración en el proceso de desarrollo	Media, no enlazado a ecosistemas específicos	Alto, conectado al ecosistema Microsoft (por ejemplo, desarrollo con .NET)	Alto, conectado al ecosistema Google (por ejemplo, desarrollo en Android)
Almacenamiento Block level	Sí	Si (puede ser utilizado con varios servidores virtuales simultáneamente)	No
Bases de Datos Relacional	Si (MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server)	Si (Azure SQL Database, Microsoft SQL Server)	Sí (MySQL)
Base de Datos NoSQL	Sí (propietaria)	Sí (propietaria)	Sí (propietaria)
DNS	Sí	No	Sí

Red Virtual	Sí	Sí	No
Mensajería de publicación/suscripción	Sí (propietaria, JMS disponible)	Sí (propietaria)	Sí (propietaria)
Herramientas de Machine Learning	Sí	Sí	Sí
Herramientas de despliegue	Sí	Sí	Sí
Integración con servidores on premises (cloud hibrida)	Sí	Sí	Sí

Tabla 7: Diferencias entre proveedores cloud públicos. Fuente: (Wittin, 2017, p 35-36)

Hosting propio vs Cloud

Como se repasa en los principios de Cloud computing, el hecho de tener el hosting propio requiere costos de capital (CAPEX) y costos de operación (OPEX) mayores que si decidimos optar por un entorno en Cloud. En el caso de un ambiente cloud requiere CAPEX a nivel 0 y un OPEX que mejor se adapta a lo que utilizamos. Como se encontró, es un problema tanto para una empresa como Ilumno como para M2M Monitoreo, donde los entrevistados tienen una preocupación especial con el sobredimensionamiento de los servidores. Con lo cual dependiendo de nuestro ambiente podemos optar por un entorno Cloud puro o un Cloud hibrido (supongamos que tenemos mainframes aun dentro de nuestra arquitectura). En el capítulo 5 vamos a revisar como conocer el punto de stress de nuestra aplicación, que es sumamente útil cuando queremos conocer que cantidad de conexiones soporta nuestra aplicación con una cierta configuración de servidor.

Seguridad en la nube: CASB

Un CASB es un término creado por la consultora Gartner que se refiere a las iniciales de *Cloud Access Security Broker*. Se define como una plataforma que provee visibilidad en aplicaciones cloud, monitorea el acceso a la aplicación y utilización entre varios servicios de cloud, asegura políticas de control de acceso, protección de datos y conformidad.

El CASB recolecta información de los siguientes aspectos:

- Aplicaciones basadas en Cloud (Dropox, Google Apps, Office 365)
- Usuarios que acceden la aplicación
- Dispositivos utilizados para acceder a las aplicaciones
- Archivos y datos creados y almacenados entre aplicaciones
- Actividades relacionadas al acceder aplicaciones y archivos, relacionadas con trabajar con aplicaciones y archivos, y finalmente relacionados con manejar permisos y permisos en compartir los archivos.

Pero CASB es más que solamente una herramienta de monitoreo, ya que puede correlacionar y analizar la información para:

- Identificar vulnerabilidades y riesgos.
- Detectar comportamientos anómalos e indicar ataques.
- Demonstrar conformidad con políticas y regulaciones.

(Friedman, 2015, p. 17-18) traducción propia.

Microservicios

Desde muchos años se ha intentado construir cada vez software mejor. Se ha aprendido mucho de cómo se construían las cosas anteriormente, adoptando nuevas tecnologías, y observando como nuevas empresas de IT emplean nuevas tendencias para tener clientes y también a sus propios desarrolladores felices.

En principio los profesionales de IT escribían aplicaciones monolíticas, es decir, que consisten en grandes componentes. Esto hace evidente que en la evolución de los sistemas los mismo se vuelven más grandes y complejos. Esta tendencia afecta principalmente en la dificultad de mantenimiento y el aislamiento de los problemas. Una mejor solución es la de granularizar las funcionalidades de sistemas en componentes más pequeños desacoplados. Esto reduce la complejidad, y va más alineado a la aplicación de metodologías agiles. Por otro lado, permite mayor flexibilidad de despliegues ya que se reemplazan porciones más pequeñas del sistema permitiendo mantener mejor control sobre los cambios realizados, por el bajo acoplamiento.

Se cita al autor Newman que propone a la siguiente definición: los microservicios son pequeños servicios autónomos que trabajan juntos. (Newman, 2015, p. 2) traducción propia.

Es evidente que, alineando a los microservicios con la utilización de contenedores, van a servir en forma conjunta para aprovechar:

- Controlar la complejidad
- Facilitar el despliegue de estos bloques de construcción
- Tener una mejor capacidad de escalabilidad de cada uno de ellos a medida que sea necesario.

Procesos de Integración Continua y Delivery Continuo

En el presente trabajo se investigó como tener sistemas robustos, y esto no es solamente haciendo aprovisionamiento de cada vez más y más hardware para cubrir las necesidades de la aplicación, sino que esta tiene que estar bien construida para que no se cubran ineficiencias del sistema con asignaciones de recursos adicionales.

Los contenedores ayudan a construir un sistema de CI, ya que se evita tener equipos dedicados para compilar el software (agentes de construcción) y equipos dedicados para hacer testing, una vez que estos contenedores terminan de hacer su trabajo se detienen y dejan las salidas como reporte de testing o el software construido en ubicaciones predefinidas. Sin ir más lejos, Jenkins y otros productos de solución de CI traen ya soporte a Docker para usarlos como agentes.

En la ilustración 13, se ejemplifica un pipeline (tubería en inglés), que establecen los pasos obligatorios del sistema de integración continua que deberán cumplirse previo a continuar con el siguiente. Todos esos pasos pueden implementarse utilizando contenedores.

Ilustración 13: Ejemplo de pipeline de CI. Fuente: propia

Una vez que el proceso de Integración Continua termina y el sistema construido pasó las pruebas exitosamente, tenemos la posibilidad de promover esa construcción por las diferentes instancias o fases que la organización tenga definidas respecto de ambientes, como Desarrollo, Pruebas, Pre producción y Producción. Aquí lo importante es que el sistema

construido no sea modificado en todo el proceso. La oportunidad de generar imágenes de Docker ayuda a este requerimiento, ya que tanto el desarrollador, el tester y el DevOps tendrían exactamente la misma imagen inmutable. Por otro lado, dicha imagen de Docker se podrá implementar corriéndola en cada ambiente sin cambios, con lo cual se reduce el tiempo de despliegue significativamente.

El rol del DevOps

Existe un rol nuevo que surge de las nuevas estrategias de delivery continuo, integración continua y cloud computing. Este es el de DevOps, como dice el nombre es una mezcla entre un desarrollador y una persona orientada a operaciones. Tiene capacidades de automatizar procesos, utilizar herramientas para agilizar los procesos de desarrollo y también capacidades de una persona de operaciones. Este es el rol que generalmente trabaja en implementaciones de contenedores y orquestadores, pero no es el único. No sería recomendado que el DevOps construya la imagen del sistema que se está codificando y definir/construir el proceso de integración continua y delivery continuo. Lo mejor es que desde desarrollo se adopte embeber el código en la imagen de Docker desde un principio, generando que esa misma imagen vaya del desarrollador al tester, del tester al personal de infraestructura o DevOps, y desde allí al usuario.

Licenciamiento de las herramientas necesarias

En este trabajo se han expuesto muchas herramientas que conforman el ecosistema de contenedores. Pero en el caso más simple, si se desea implementar contenedores y orquestadores sin tener en cuenta la tecnología del sistema desarrollado, no se tienen gastos de licenciamiento en la tecnología lo cual es muy importante, por ser la totalidad de ellas opensource.

Producto	Licenciamiento
Docker	OpenSource – Licencia Apache 2.0
Rkt	OpenSource – Licencia Apache 2.0
Kubernetes	OpenSource – Licencia Apache 2.0
Apache MesOS	OpenSource – Licencia Apache 2.0
Docker Swarm	OpenSource- Licencia MIT

Jenkins	OpenSource – Licencia MIT

Tabla 8: Tabla de licenciamiento de herramientas. Fuente: Propia

En el caso de los sistemas operativos virtualizados en caso de usar servicios en la nube, las licencia en los casos de Windows está incluida en el costo del servicio.

¿Es factible implementar orquestadores en empresas tipo Pyme?

Según lo investigado, la respuesta es sí. Teniendo en cuenta que se pueden implementar las herramientas necesarias ya que son en su mayoría open source y que implementar en Cloud no requiere una inversión inicial son las razones más fuertes. Hay empresas que se especializan en nuestro ámbito local en dar consultoría en cuestiones de infraestructura con lo cual con este aspecto se podría aspirar a tener un sistema de alta disponibilidad soportada por orquestadores y contenedores en cloud computing.

Sistemas de alta disponibilidad

Introducción

Cuando se diseña un sistema que va a ser mantenido en línea siempre deben estimarse cuantos usuarios podrían llegar a conectarse en el caso más extremo. Esto permite investigar si el hardware que se tenía pensado en un principio podría requerir múltiples instancias adicionales para soportar picos. En casos reales muchas veces se sobredimensiona el hardware de lo necesario en condiciones normales, pero no demasiado por cuestiones de costos. Esta no es una solución adecuada ni eficiente. Recordando que este trabajo busca analizar diferentes alternativas para tener un sistema robusto y fiable, que no tenga caídas, que funcione de con buenos niveles de respuesta bajo carga y que de esta manera se eviten consecuencias de imagen empresarial y costos por operaciones pérdidas o interrumpidas.

Zero Downtime

La expresión anglosajona "Zero Downtime" está relacionada al hecho de que el sistema no se tenga tiempo de caídas, es decir, de indisponibilidad. Las caídas pueden estar asociadas a varias circunstancias que podemos clasificar de dos maneras:

1- Planificadas:
   a. Despliegues de actualizaciones de aplicación: cuando existen aplicaciones en línea, es común que se necesite actualizar el software para ofrecer nuevas funcionalidades o para resolver problemas en las mismas. Cuando esto no se realiza con ciertas técnicas el sistema puede estar no disponible por minutos o hasta horas.
   b. Actualización de hardware: Esta situación es común cuando se tienen servidores físicos, los cuales no pueden ser mantenidos en funcionamiento para agregar hardware. Si bien esto es menos común y posible de realizar desde la implementación de virtualización.

2- No planificadas
   a. Picos de Demanda: muchos usuarios se conectan y operan en el mismo momento sobre el sistema causando en un primer momento impacto en el nivel de respuesta del sistema, pudiendo en el peor de los casos no lograr a

responder. Esto es lo que debería ser controlado con orquestadores, cloud computing y contenedores.

b. Ataques DoS (Denial of Service): El sistema recibe múltiples peticiones, en contraste con el caso anterior, no son auténticas. Las mismas llevan el sistema a un colapso de las capacidades de procesamiento y el servidor deja de responder a las peticiones en un cierto punto. En este caso deberían utilizarse servicios de protección como CloudFlare que ofrecen protección ante ataques DoS. El no tener una protección de este tipo colapsará los servidores, o en el caso de tener un buen sistema de alta disponibilidad en cloud, elevar los gastos en servidores para atender las peticiones.
c. Errores en la aplicación: En caso de que la aplicación tenga errores puede provocarse que el sistema tenga falta de disponibilidad por excepciones que no fueron bien manejadas.

Estrategias de despliegue

Como se explicó en el párrafo anterior es importante manejar los despliegues de nuevas versiones para no afectar la disponibilidad de los sistemas. Por lo cual es esencial establecer estrategias de despliegues y poder volver atrás un despliegue en caso de que algo haya salido mal. El hecho de intentar hacer depuración de problemas en entornos en producción terminará con grandes probabilidades en largas noches, más errores con consecuencias impredecibles y lo peor: usuarios enojados. La mejor forma es poseer una alternativa de recuperar el entorno ante una situación donde el despliegue fue fallido, y tener la comodidad de revisar el problema durante la jornada de trabajo normal. Ante estas situaciones existen técnicas para volver atrás rápidamente, e incluso técnicas más avanzadas como Blue – Green, y Canary como para hacer despliegues con zero downtime.

Bien desarrollado por Humble (2011), antes de adentrarnos a las técnicas debemos tener dos factores en cuenta respecto a los rollbacks (vuelta atrás en inglés):

1- Los datos: si el nuevo sistema realiza cambios en los datos es difícil volver atrás sin generar inconsistencias o problemas con ellos a los usuarios.

2- Los sistemas con los cuales se integra la plataforma. Es lógico pensar que en escenarios donde existen mayor cantidad de despliegues de sistemas orquestados mayor es la dificultad.

Como precondiciones de despliegue, primero se debe asegurar el estado de nuestro sistema, incluyendo base de datos y sistemas de archivos. Por otro lado, en el plan de vuelta atrás (rollback), que funcione y que haya sido validado, y que permita recuperar desde backups y migrar las bases de datos. (Humble, 2011, p. 259).

Volver atrás usando la última versión estable

En muchos casos, es la forma más simple de hacer rollback. Si se tiene un proceso automatizado para desplegar la aplicación, la forma más simple de volver a un buen estado es redesplegar la versión previa estable desde el principio. Esto también incluye reconfigurar el ambiente donde corre, para que quede en el mismo estado antes de desplegar la nueva versión.

Según Humble, las buenas razones de realizar este simple procedimiento son (Humble, 2011, p. 260):

- Si no se tiene un procedimiento automatizado de rollback pero si se tiene un procedimiento de despliegue, entonces redesplegar la última versión se realiza en un tiempo fijo que tiene menor riesgo (menor probabilidad que salga mal).
- Generalmente es el mismo proceso que se prueba muchas veces antes de ir a producción, mientras que los planes de rollback son probados menos frecuentemente y son por esto más propensos a fallar.

También está claro que esto tiene ciertas desventajas (Humble, 2011, p. 260):

- Si bien el tiempo para redesplegar viejas versiones es fijo, no es cero. Esto conduce a que tengamos una caída del sistema.
- Es más difícil de depurar cuando los problemas se presentan. Redesplegar la vieja versión en ocasiones sobreescriben la nueva versión. Con esto ya se pierde la posibilidad de trabajar en los problemas encontrados.
- Si se recupera desde una base de datos de respaldo previo al despliegue de la última versión, se pueden perder datos luego del despliegue fallido. Esto puede ser un

problema grave si no se hace un rollback lo suficientemente rápido, pero en ciertas ocasiones es inaceptable.

Depliegues Zero-Downtime

Un despliegue Zero-Downtime, también llamado despliegue en caliente, es aquella técnica de cambiar usuarios entre distintas versiones de un sistema de forma instantánea. Y más importante aún, permite volver usuarios a la versión anterior de forma también automática si algo sale mal.

La clave para despliegues de zero-downtime es desacoplar las partes del proceso de despliegue para que puedan realizarse lo más independientemente posible. Esto debería poder lograr poner en lugar nuevas versiones de recursos compartidos por las aplicaciones de los cuales dependen, como bases de datos, servicios, recursos estáticos, previamente antes de actualizar la aplicación. (Humble, 2011, p 260-261).

Despliegues Blue - Green

Es una de las técnicas más poderosas y conocidas para manejar despliegues. La idea es tener dos entornos idénticos con idénticas versiones del entorno productivo, donde al primero lo llamamos Green (clon) y al segundo Blue (entorno productivo). El funcionamiento se establece poniendo por delante de los entornos un router, balanceador, proxy reverso donde el mismo estará apuntando al entorno de producción (blue). Entonces se desea lanzar una nueva versión, con lo cual se realiza en el ambiente clon (green). Sobre este último ambiente podemos validar que el despliegue fue correcto, sin afectar al entorno productivo. Cuando el ambiente esta validado, cambiar de versión es tan simple como cambiar que se enrute el tráfico al ambiente nuevo en segundos, y ahí el ambiente Green se transforma en producción. En la ilustración 14, puede visualizarle lo explicado:

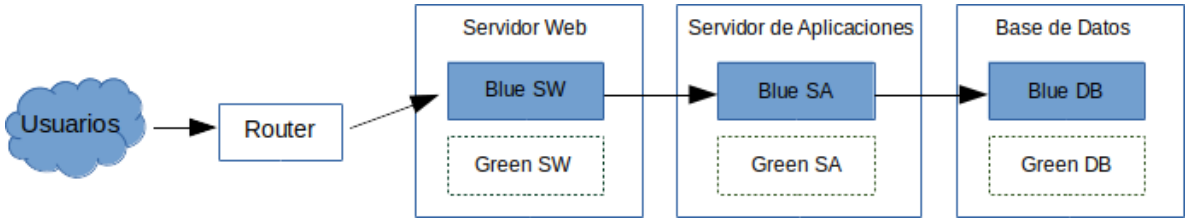


Ilustración 14: Despliegue Blue-Green. Fuente: (Humble, 2011, p. 261)

Si algo malo ocurre, simplemente se puede hacer un switch nuevamente en segundos al ambiente blue. La ventaja de esta técnica es que permite depurar que ocurrió. Aquí los cuidados tienen que estar tomados en el manejo de las bases de datos. Generalmente no es posible cambiar de base de datos de un entorno a otro ya que la migración de datos tomaría un tiempo adicional. Una solución a esto es mantener la base de datos en modo solo lectura mientras se hace el cambio, y cuando se verifica que todo está bien, se migran los datos y allí se vuelve a habilitar escritura.

Este enfoque puede ser muy costoso por el doble costo de entorno idéntico. Algunas alternativas que pueden plantearse, es aplicar blue green con entornos de diferentes fases, como por ejemplo producción y preproducción o correr dos instancias sobre el mismo ambiente. (Humble, 2011, p. 260-262).

Lanzamiento Canario (Canary Release):

Siempre el usuario supone que solo hay una versión del software en el entorno productivo en un momento dado. Esto hace más fácil manejar parches (hotfixes), y de hecho la infraestructura general. Sin embargo, esto presenta un impedimento para probar el software en este estadio. Aún con una sólida y completa estrategia de testing, los defectos aparecen en producción. Aunque se tenga un bajo tiempo de ciclos, los equipos de desarrollo pueden obtener beneficios del rápido feedback de nuevas funcionalidades y cualquier otra posibilidad que pueda hacer que el software provea mayor valor.

Lanzamiento Canario trata de tomar ventaja de estos factores, y consiste en realizar un despliegue de una versión de software a un subconjunto de servidores para obtener rápido feedback, como se muestra en ilustración 15, donde un pequeño grupo accede a la versión nueva y la mayoría de usuarios siguen siendo enrutados a la versión estable. Como un canario en una mina de carbón, esto rápidamente revela cualquier problema con la nueva versión de software sin causar impacto en la mayoría de los usuarios. Esto reduce el riesgo de lanzar nuevas versiones de software masivamente.

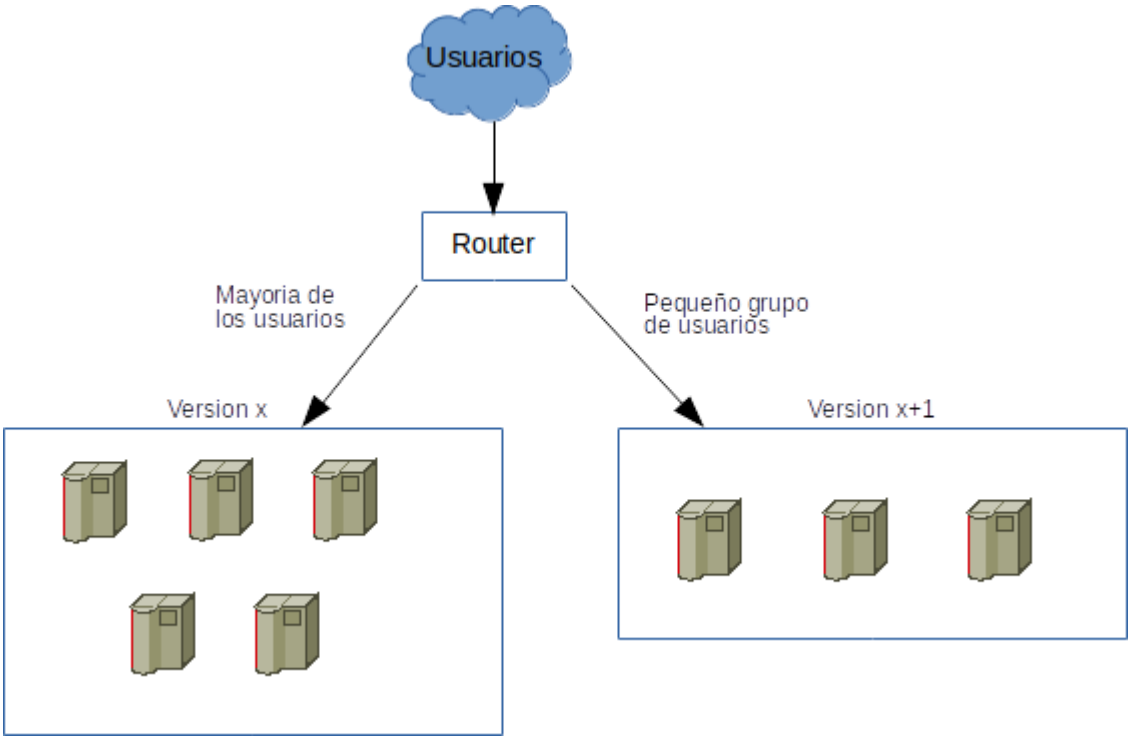


Ilustración 15: Canary Release. Fuente: (Humble, 2011, p. 263)

Los beneficios de realizar releases canary son (Humble, 2011, p. 263-264):

- Volver atrás es rápido, si algo ocurre con los usuarios que apuntan a la nueva versión, se los enruta a la versión estable. En el ambiente donde se apuntaron usuarios persiste información del error en logs que permitirá su posterior análisis.
- Permite realizar pruebas para validar si una funcionalidad causa interés y si la misma provee valor, antes de lanzarlo masivamente a todos los usuarios.
- Puede implementarse para calcular el punto de stress de un sistema. Esto se logra aumentando gradualmente los usuarios enrutados al subset de servidores, y analizando cuando los tiempos de respuesta no son aceptables.

Pruebas de performance o de carga

El testing de performance o de carga constituye uno de los tipos de pruebas no funcionales, es decir, que no están atadas a los requisitos del sistema. Como explica Sommerville (2011), "*Los requerimientos no funcionales, como el rendimiento, la seguridad o la disponibilidad,*

especifican o restringen por lo general características del sistema como un todo. Los requerimientos no funcionales a menudo son más significativos que los requerimientos funcionales individuales." "*Los requerimientos no funcionales afectan más la arquitectura global de un sistema que los componentes individuales*" (Sommerville, 2011, p. 87).

Las pruebas de performance son aquellas donde se verifica que el sistema puede procesar la carga pretendida. La misma consiste en ir aumentando la carga hasta que en un punto la capacidad de respuesta está afectada. (Sommerville, 2011, p. 227).

Esto generalmente se logra capturando algunas operaciones comunes de usuario, los cuales se automatizan y luego se repiten escaladamente hasta llegar a un punto de stress, y el sistema deja de responder de una manera aceptable por el usuario. Hay herramientas muy conocidas para realizar este tipo de trabajos como JMeter.

Punto de stress

Cuando se especificó la situación de las pruebas de performance, se describió al punto de quiebre en donde ya el sistema no es capaz de responder de forma aceptable para el usuario. Este punto varía dependiendo de varias condiciones que van a tener que ser contextualizadas como, por ejemplo, la configuración de hardware, las versiones de los componentes de software de la arquitectura (base de datos, servidor de aplicaciones), las configuraciones a nivel de software, como caches, que nos afectaran mucho nuestras mediciones. Las mismas deben ser documentadas para poder establecer una línea base (baseline) contra el cual estimar hardware requerido por demanda.

Métricas de performance

Cuando se proponen objetivos, en todo aspecto, se necesitan que sean medibles para que no sean subjetivos. Con las pruebas de performance se obtiene una referencia de lo que soporta el sistema bajo análisis, pero: ¿bajo en que unidades se podrían especificar? Estas métricas deben monitorearse mediante herramientas de monitoreo y alarmas. Según Bourke (2001), tenemos 3 métricas validas con las cuales medir la capacidad del sistema. Cada métrica tiene su nivel de importancia dependiendo de las necesidades del sitio:

- Conexiones por segundo

Es quizás la métrica más importante relativo al rendimiento puro, especialmente en HTTP. Conexiones por segundo se refiere al número de peticiones entrantes que el sistema es capaz de aceptar, dependiendo del proveedor. Es usualmente el factor limitante de cualquier sistema, la primera de todas las métricas de performance para alcanzar el límite de rendimiento. La razón por la cual esta métrica es tan crítica es porque es costoso el abrir y cerrar conexiones HTTP en la pila de red o el procesamiento de red. Ejemplificando esta carga de trabajo, se muestra a continuación los pasos necesarios para transferir un archivo por HTTP:

1. El cliente inicia una conexión HTTP enviando un mensaje HTTP SYNC destinado al puerto 80 de un servidor web.
2. El servidor web envía un paquete ACK en respuesta al paquete del mensaje HTTP sync con detalles adicionales al cliente
3. El cliente envía un mensaje ACK al mensaje HTTP SYNC en respuesta al server.

El inicio de una conexión HTTP se conoce como negociación en tres vías. Luego de que la negociación es realizada, los datos pueden ir y volver. En el caso de HTTP, es una página web.

Este proceso tiene algunos pasos para enviar solamente 30kb datos netos, y esto impacta los recursos de red. Los mismos se usan intensivamente creando y bajando las conexiones. Esto explica en mejor detalle porque es tan importante la velocidad que un sistema puede cumplir con este trabajo.

- Conexiones concurrentes totales

  Las conexiones concurrentes totales es una métrica que determina cuantas sesiones de usuario TCP un sistema puede soportar. Usualmente este número está dado por la memoria y el procesador del sistema o dispositivo de red encargado de aceptarlas. El rango de la cantidad va desde la infinidad a algunos miles de conexiones. La mayoría de las veces este límite es teórico, con lo cual es recomendable encontrar otra métrica antes que encontrar el límite de conexiones.

Por tráfico UDP, las conexiones concurrentes no son un factor, ya que UDP no es un protocolo orientado a conexión. El trafico UDP está relacionado generalmente al tráfico de streaming y de DNS, pero son varios protocolos que corren sobre UDP.

- Throughput (Rendimiento en bits por segundo):

  El throughput es otra métrica importante. Generalmente medido en bit por segundo, el throughput es la razón en la cual el sistema/dispositivo es capaz de pasar tráfico a través de su infraestructura interna. Todos los dispositivos tienen factores limitantes basados en su diseño arquitectural, entonces es un punto importante a revisar en la información brindada por el proveedor.

  Mientras que el throughput es medido en bit por segundo, es de hecho una combinación de dos variables diferentes: tamaño de paquete y paquetes por segundo. Los paquetes ethernet varían en tamaño, con un MTU (Maximum Transmitable Unit) de alrededor de 1.5 KB. Si una pieza particular de datos es superior a 1.5KB, entonces es particionado en partes de 1.5 KB para ser transportado. El número de paquetes por segundos es realmente el factor más importante en un balanceador de carga o cualquier otro dispositivo de red. La combinación de esto y de los tamaños de paquetes determinan los bits por segundo (Bourke, 2001, p. 32-34).

Escalabilidad vertical y Escalabilidad Horizontal

Teniendo en cuenta que aumentar la capacidad del sistema es aumentar los recursos asignados, existen dos estrategias posibles:

- Escalabilidad vertical: esto se constituye en agregar mayor hardware a los servidores para aumentar su rendimiento y capacidad, como más CPU, más memoria RAM, discos más veloces.
- Escalabilidad horizontal: Consiste en agregar instancias para aumentar la capacidad total del sistema. Esta es la tendencia actual, como ocurre en cloud computing o a nivel de la fabricación de nuevo hardware.

En cloud computing y orquestadores se tiende a esta última forma, ya que permite que esa ampliación o disminución de instancias de acuerdo a la demanda en el sistema, lo que permite la elasticidad del sistema.

En la ilustración número 16, se muestra gráficamente la diferencia entre estos dos tipos de escalabilidad.

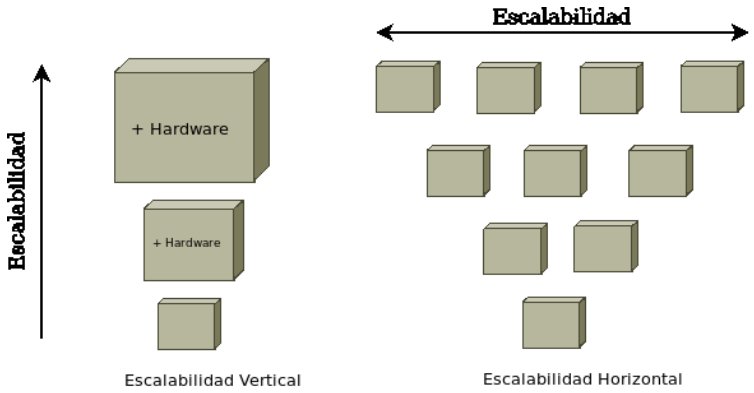


Ilustración 16: Escalabilidad Vertical vs Escalabilidad Horizontal. Fuente: Propia

Mecanismos de mitigación actuales

El hecho que la temática de caída de sistemas siga presente en la actualidad, no es porque no existieran formas de mitigar o controlar la carga de los servidores. A continuación, se enumeran ciertas técnicas utilizadas y todavía validas en la actualidad para mejorar los tiempos de respuesta al usuario y controlar la carga en los servidores que se pueden combinar con orquestadores de contenedores.

Cache server

El caching es utilizado para mejorar la performance de servidores web por medio de la percepción de latencia del usuario y el uso de ancho de banda, y además reduciendo la carga en servidores y facilitando la escalabilidad de sistemas. Los servidores de cache pueden estar desplegados en varios puntos de una red. Los mismos tienen que tener algoritmos para administrar que se va mantener en cache y que va a ser descartado en un buffer limitado. (Iyengar, 2004, p. 1).

CDNs

Los CDNs (Content Distribution Networks) son una forma de cacheo distribuido, que suplementa el cacheo en el cliente a otro punto en la red controlado por el proveedor de servicio de CDN. Un CDN es una red compartida de servidores o caches que entregan contenido a los usuarios en representación de proveedores de contenido usando diferentes técnicas de ruteo de peticiones. La intención de un CDN es servir contenido a un cliente desde un servidor CDN con lo cual la respuesta es obtenida en un mejor tiempo que contactar el servidor de origen directamente. El tener demasiados CDNs también reducen la carga en los servidores de origen. (Iyengar, 2004, p. 2).

Balanceadores de carga

Es quizás el tema más ampliamente discutido a la hora de ofrecer sistemas con alta disponibilidad. Según Bourke (2001), los balanceadores constituyen un proceso y tecnología que distribuye el tráfico entre diferentes servidores usando un dispositivo en la red. Este dispositivo intercepta tráfico destinado a un server y lo redirige a otros. Este proceso es totalmente transparente al usuario final. Pueden existir docenas e incluso cientos de servidores trabajando detrás de una única URL. Las funciones de los balanceadores de carga se pueden resumir en los siguientes puntos (Bourke, 2001, p 14-15):

- Interceptar tráfico en una red, como el trafico web destinado para un sitio.
- Dividir el tráfico en peticiones individuales y decidir que servidor las revisa.
- Monitorea los servidores disponibles, asegurando que todos responden al tráfico. En caso contrario son quitados de la rotación.
- Provee redundancia por emplear más de una unidad en un escenario de tolerancia a fallos (failover)
- Ofrece distribución de acuerdo con contenido, realizando lectura de URLs, intercepción de cookies, y parseo XML.

.

Los balanceadores pueden ser agrupados en dos categorías, de acuerdo a su operación en modelo OSI (Open Systems Interoperatibility): de capa 4 o capa 7. Los balanceadores de carga de capa 4 actúan de acuerdo a los datos en protocolos de transporte como UDP, TCP, IP, FTP. En cambio, los balanceadores de capa 7 distribuyen las peticiones de acuerdo a los

datos presentes en protocolos de aplicación como HTTP. La distribución de tráfico se puede realizar utilizando distintos tipos de algoritmos[22] como:

- Round Robin
- Weight Round Robin
- Menos conexiones
- Menor tiempo de respuesta.

Proxy reverso

Es un proxy que actúa como un servidor web convencional, que puede redirigir las peticiones a otros servidores. Ninguna configuración especial es necesaria en los clientes. El cliente hace una solicitud ordinaria de contenido al nombre de proxy inverso. El proxy reverso decide donde derivar esa solicitud, y retorna la respuesta como si fuera propia. Usos típicos de proxy reversos son, proveer acceso a internet a usuarios que están detrás de un firewall. Otro caso es la de realizar balanceo de carga entre varios servidores, o proveer cache a un servidor de backend lento[23].

Traffic shapers/Priorizadores de carga

Según la norma RFC2475 de la IETF, los traffic shapers atrasan algunos o todos los paquetes en un flujo de tráfico en orden de traer al flujo aquel tráfico priorizado de acuerdo con el perfil del tráfico[24]. Un priorizador usualmente tiene un buffer finito, y los paquetes pueden ser descartados si no queda suficiente espacio en buffer para mantener los paquetes retrasados.

Este mecanismo de encolado puede ayudar a contener y priorizar las request que nos interesan, pero hay un riesgo en perdida de peticiones.

[22] fuente: https://f5.com/glossary/load-balancer, recuperado el 13/06/2018
[23] fuente: http://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse, recuperado el 06/06/2018
[24] fuente: https://tools.ietf.org/html/rfc2475, recuperado el 26/06/2018

Realidad en mercado local

Para conocer el ámbito local se realizaron entrevistas y encuestas para identificar las técnicas implementadas, las apreciaciones hacia contenedores, orquestadores y cloud computing. Las mismas fueron realizadas a 30 profesionales de sistemas de empresas de IT radicadas en Córdoba (Argentina), pero en mayor parte trabajando para proyectos internacionales. Entre las empresas de donde provienen los encuestados son McAfee, BitLogic, Ilumno, MercadoLibre, Olapic, entre otras. Los perfiles de las personas encuestadas son:

- Administradores de sistemas

- Personal de operaciones/infraestructura

- DevOps.

Esta preselección fue necesaria para conseguir profesionales con sistemas en producción, es decir, con disponibilidad general en la Internet.

Recorremos primero cada una de las respuestas de la encuesta realizada de acuerdo a modelo en ANEXO D:

1- Se consultó sobre las consecuencias de no evitar caídas de sistema:
   Todos estuvieron de acuerdo que las caídas del sistema causan consecuencias, ya sea a la imagen de una marca (56,7%) o directamente perdidas económicas (43,3%) por no poder concretar operaciones económicas.

Ilustración 17: Encuesta pregunta Nº1. Fuente: propia.

2- Se preguntó si sus sistemas tenían implementados mecanismos de mitigación de carga:

   En su gran mayoría expresaron tener algún mecanismo (73,3%) y solo el resto completaron el total sin mecanismo alguno de control de alta demanda.

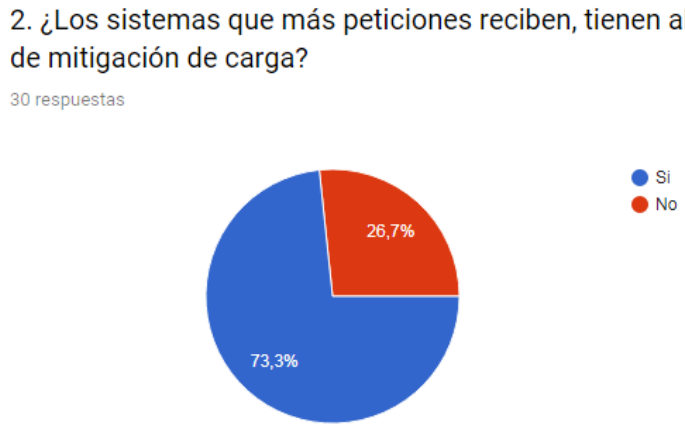


Ilustración 18: Encuesta pregunta Nº2. Fuente: propia.

3- Relacionado a conocer el límite que soporta su sistema actual:

   Este es un aspecto que no está muy claro en el ámbito, ya que los encuestados estuvieron repartidos en un 50% entre los que conocían el límite y los que no. Como

bien se explicó en este trabajo no es un cálculo trivial y requiere realizar pruebas no funcionales para conocerlo.

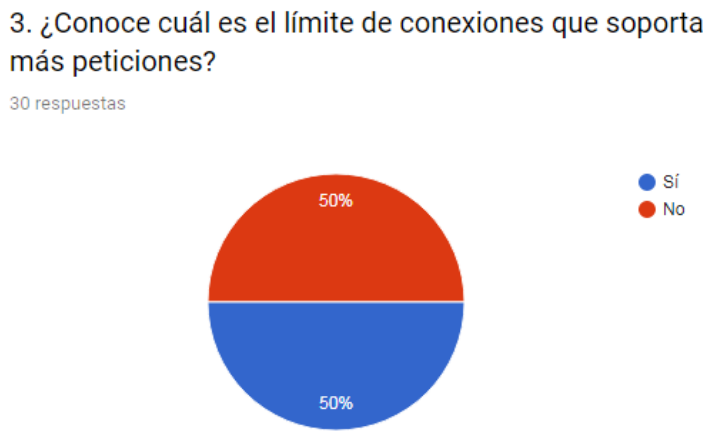


Ilustración 19: Encuesta pregunta Nº3. Fuente: propia.

4- Respecto a evitar ataques de denegación de servicio:

Aquí se obtuvo un porcentaje importante del 80% de los encuestados que tienen implementados mecanismos para soportar ataques DoS (Denial of Service).

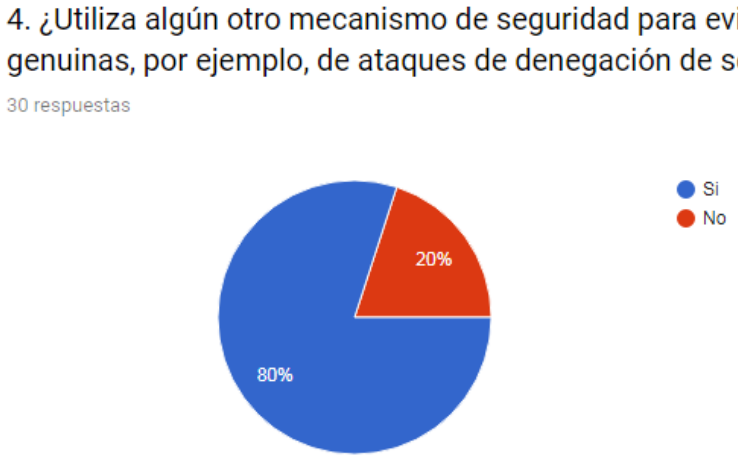


Ilustración 20: Encuesta pregunta Nº4. Fuente: propia.

5- Cuando se consulta acerca de seguridad y robustez en sistemas propios:

Tenemos alta tasa de respuestas positivas (73,3%) que piensan que sus sistemas están en buenas condiciones respecto a los aspectos planteados.

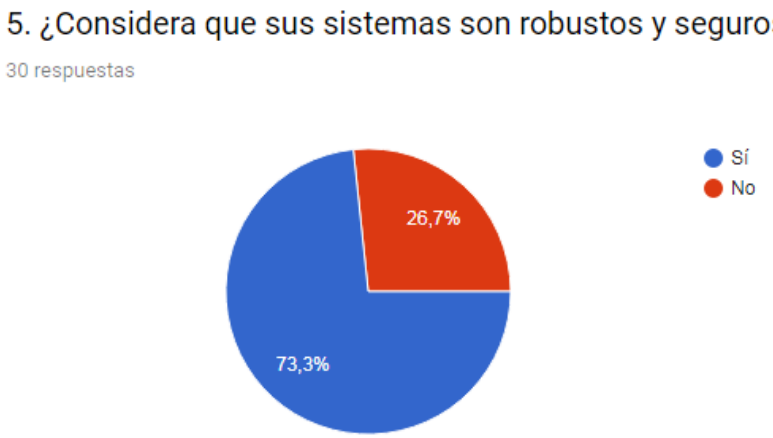


Ilustración 21: : Encuesta pregunta Nº5. Fuente: propia.

6- Cuando se consulta sobre las técnicas consideradas para mitigar caídas:

La opción preferida fue la utilización de balanceadores en VMs o maquinas físicas con un 63,3%, seguido por contenedores y orquestadores (20%). Tengamos en cuenta que aquí los orquestadores permiten implementar balanceadores de capa 7. En tercer lugar, se ubicó la creación de ambientes temporales para soportar la carga (13%), esto sería correcto si fuera automático, pero manualmente es una práctica que funciona, pero no es eficiente.

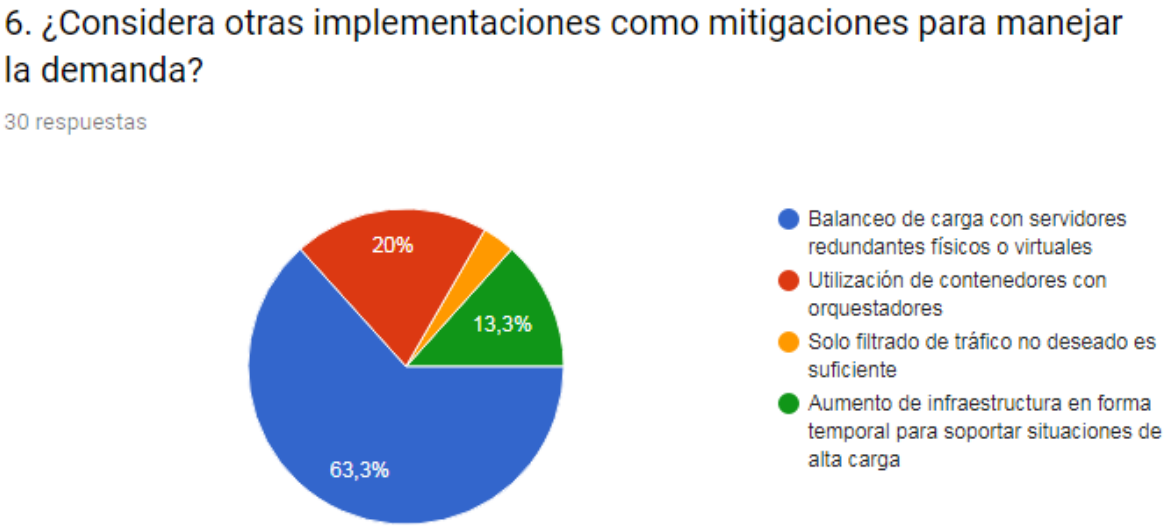


Ilustración 22: Encuesta pregunta Nº6. Fuente: propia.

7- Se consulta sobre el conocimiento sobre orquestadores y contenedores:

Solo el 50% tiene conocimiento de la tecnología, y un 10% no conoce de la tecnología. El resto conocen los beneficios, o escucharon hablar de la temática, pero no conocen como implementarlo.

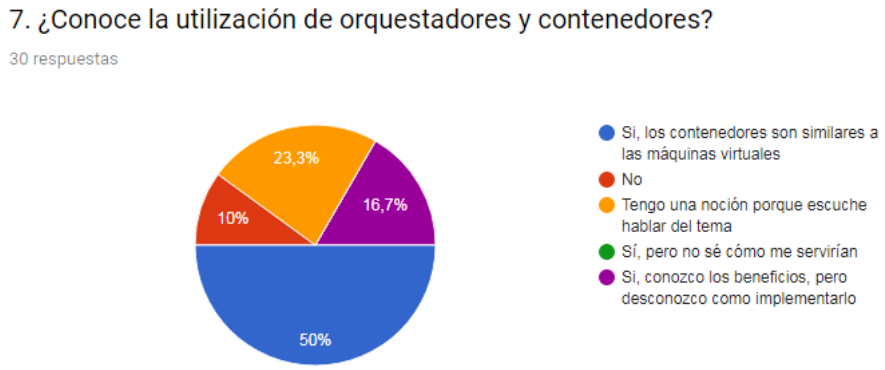


Ilustración 23: Encuesta pregunta Nº7. Fuente: propia.

8- Se buscó entender si realmente se conocían los orquestadores y balanceadores, y se consultó a si los imaginaban como futuros reemplazantes de los balanceadores y máquinas virtuales:

La mayoría respondió que no las iban a reemplazar (63,3%), y eso es correcto. Las VMs se complementan con los contenedores, ya que permiten dar una base a los mismos para distintos sistemas operativos. Por otro lado, los balanceadores se siguen utilizando por más que existan orquestadores, ya que tienen funcionalidades distintas. Además, hay casos de uso donde los contenedores respecto a las VMs son más recomendables, como con operaciones temporales como la ejecución de un proceso de forma programada o en la utilización de sistemas de integración continua.

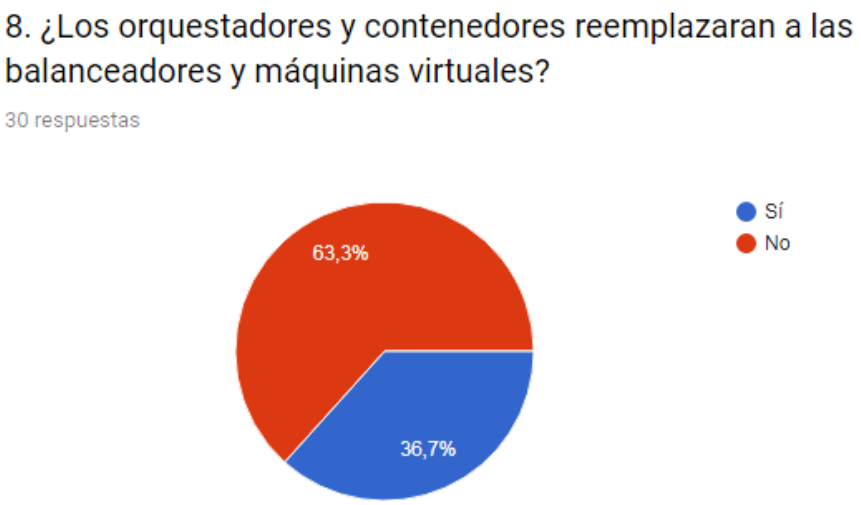


Ilustración 24: Encuesta pregunta Nº8. Fuente: propia.

9- Se consulta por alternativas a los balanceadores y orquestadores:

Gran porcentaje (70%) no tuvo otra opción por nombrar. Los restantes propusieron CDNs, Servidores de cache, Priorizadores de Tráfico como ejemplos. Estas respuestas fueron claves para poder brindar técnicas complementarias en esta investigación en capítulo 5.

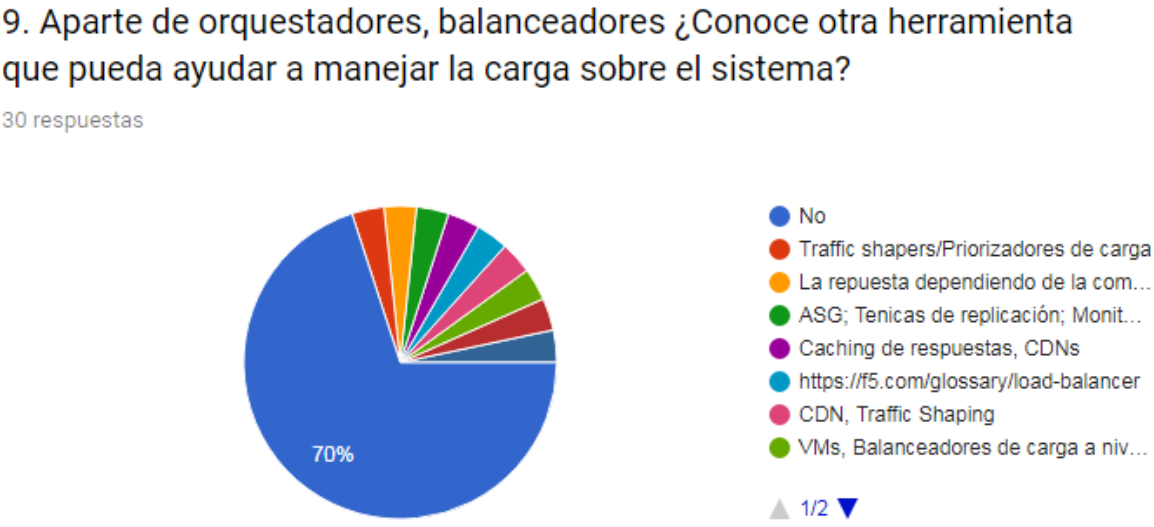


Ilustración 25: Encuesta pregunta Nº9. Fuente: propia.

10- Respecto a la factibilidad de implementación en los tipos de empresa:

Los profesionales ven factibles implementarlos prácticamente en todo ámbito, pero mayoritariamente en grandes empresas y en menor medida en empresas PyMes. De acuerdo a esto, se plantea la posibilidad de tercerizarlo en caso de que no se dispongan profesionales para llevarlo adelante.

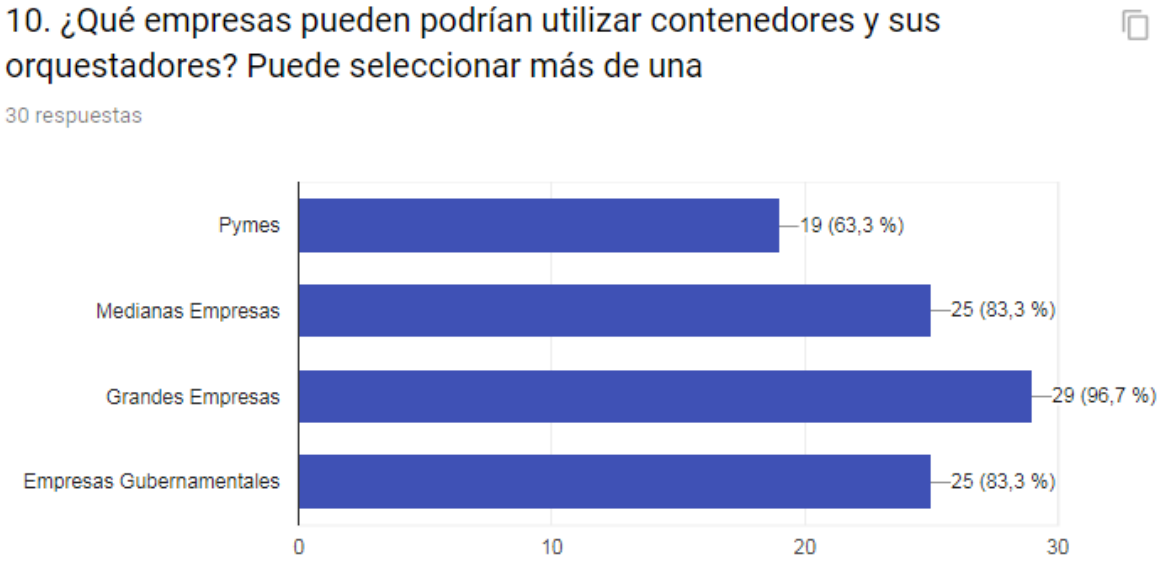


Ilustración 26: Encuesta pregunta Nº10. Fuente: propia.

11- Respecto al costo de la implantación de contenedores y orquestadores:

El 60% responde que considera que la implementación es económica y el resto desconoce. La mayoría conoce de la cuestión de licenciamiento opensource que facilita la adopción. Las dudas vienen por el costo de profesionales con conocimiento para implementar esta tecnología.

Ilustración 27: Encuesta pregunta Nº11. Fuente: propia.

12- Se preguntó sobre la implementación de orquestadores y contenedores en cloud computing:

Predominaron las respuestas afirmativas acerca de que es más rápido, seguro y más económico (en este orden) que la implementación en infraestructura propia.

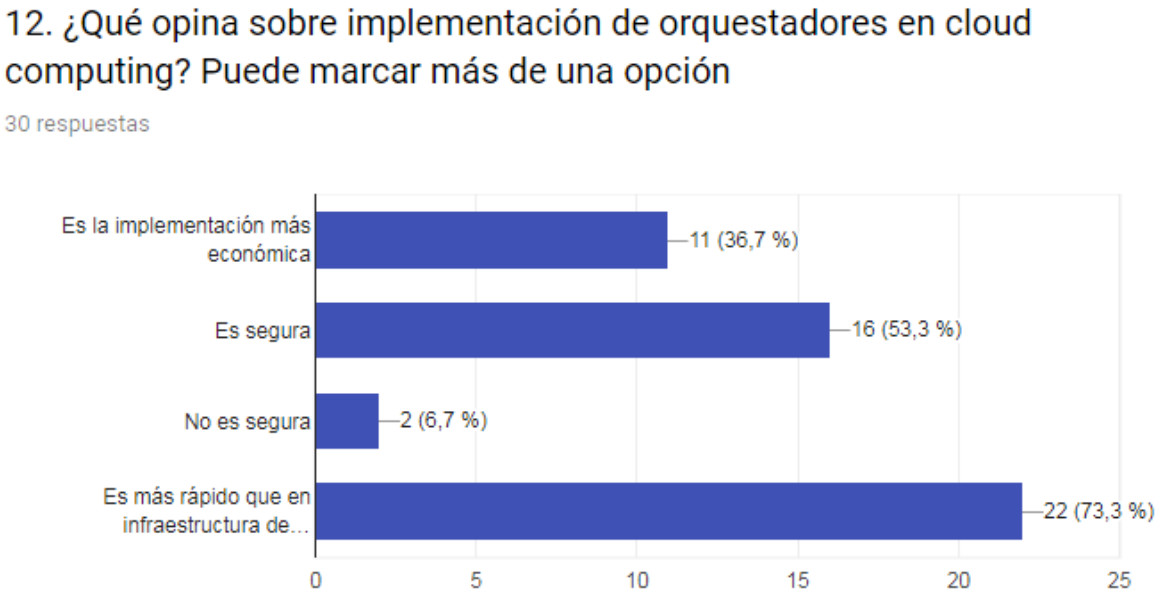


Ilustración 28: Encuesta pregunta Nº12. Fuente: propia.

13- Respecto a la opinión sobre qué aspectos son importantes a controlar en un sistema online:

El 100% de los encuestados mostraron preocupación por mantener los tiempos de respuesta adecuados a sus clientes, luego las caídas y en tercer la seguridad. Esto nos valida que los sistemas de alta disponibilidad deben monitorear en forma proactiva las métricas para atender adecuadamente a los usuarios.

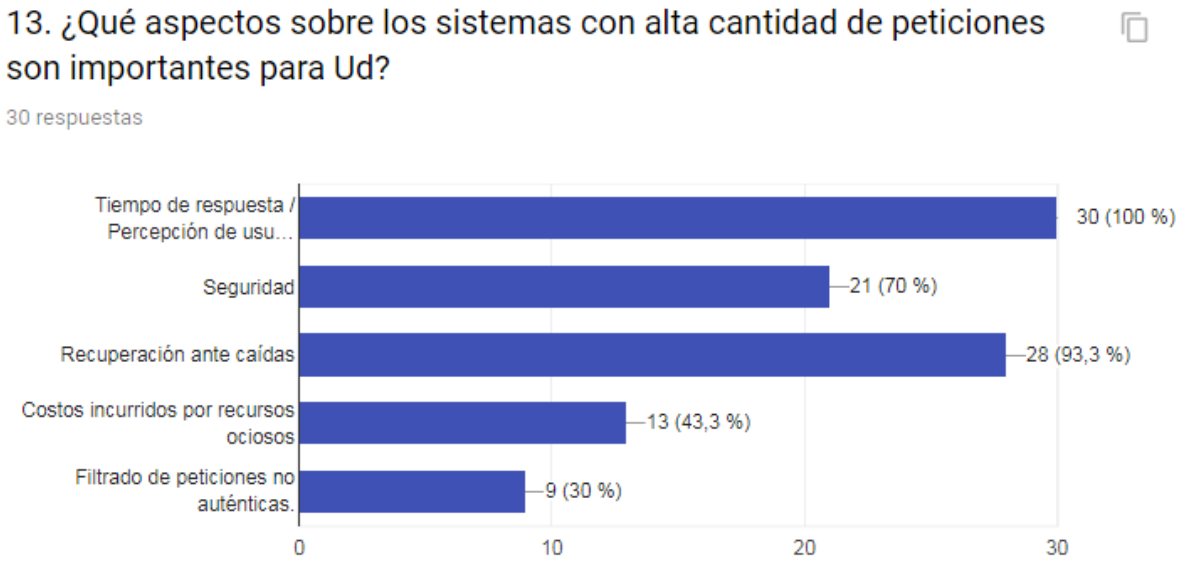


Ilustración 29: Encuesta pregunta Nº13. Fuente: propia.

Por otro lado, se llevaron a cabo entrevistas según modelo de encuesta en ANEXO A. En primer lugar, se entrevistó a Fernando Hevia, Director de Infraestructura de una red internacional de universidades llamada Red Ilumno, de la cual Universidad Siglo 21 forma parte. En segundo lugar, se entrevistó a la Ingeniera Dafne Defant, al frente de la parte técnica de la empresa M2M Monitoreo, una PyMe del interior de la provincia de Córdoba. Las entrevistas completas se encuentran en ANEXO B y C respectivamente. Algunos puntos por destacar:

- En ambos casos poseen esquemas de asignación de hardware excedente para soportar picos. En el caso de Ilumno, ya implementando balanceadores con servidores redundantes, y también AlwaysOn, una solución propietaria de alta disponibilidad en Microsoft SQL Server. Por el lado de M2M Monitoreo tienen sobrado hardware luego de cambiar la arquitectura de la aplicación para que la parte de mapas sea

consumida desde internet, aliviando al servidor que antes tenía su propio servicio de mapas. Los dos profesionales vieron como una oportunidad utilizar contenedores como una forma de usar sus recursos de manera más eficiente.

- El director de Ilumno, ve como potencial aplicación de contenedores y orquestadores para sus implementaciones de ESB interno de la empresa, ya que observa que cada vez más aplicaciones están transitando sus comunicaciones por ellos. Además, como manejo más eficiente de su actual sistema con más peticiones llamado Epic.
- En M2M monitoreo ven de manera optimista utilizar el servidor que tienen contratado para instalar orquestadores y contenedores y así poder utilizar versiones distintas de PHP, ya que esto le genera una dependencia de tecnología de las aplicaciones que tienen desplegadas en el mismo, por otro lado, permitiría tener más contenedores de forma dinámica de acuerdo al uso para cada aplicación.

Conclusiones del proyecto

Como conclusiones del análisis realizado en este proyecto de investigación se identificaron tendencias importantes. Lo primero fue el cambio disruptivo de la aparición de Internet en los 90s facilitando el acceso a la información, por el cual se fue sumando la presencia de organizaciones en la Internet, ya sea con páginas web institucionales y como también ofreciendo servicios web con fines comerciales. Este crecimiento exponencial de usuarios ocasionó en muchas oportunidades que los sistemas no sean capaces de responder grandes cantidades de peticiones simultaneas. Esto afecta factores asociados con la calidad de software como la fiabilidad y la robustez. Con la aparición de la virtualización, puede fácilmente ofrecerse redundancia con el uso de balanceadores de carga, o proxies reversos para balancear las peticiones. Aquí surge lo que se definió como alta disponibilidad. Esto funcionó bien un tiempo, pero los usuarios online de muchos sitios fueron creciendo fuertemente por diferentes nuevas prácticas de orden masivo, como la venta de entradas online, los eventos de marketing digital como Black Friday o HotSale, donde resultó difícil tener un datacenter lo suficientemente grande para asegurar que el sistema no va a degradar la velocidad de respuesta en peticiones o en el peor de los casos llegar directamente a la no disponibilidad del sistema por la sobrecarga. Las empresas que logran tener buena cantidad de recursos en sus datacenters, desembolsan grandes cantidades de capital (CAPEX) y a su vez requieren de servicios especiales de personal de operaciones para mantener en funcionamiento (OPEX). En este punto se refirió a Cloud Computing, que permite iniciar con CAPEX en 0, reducir los costos por recursos ociosos e indirectamente costos relacionados con OPEX, como refrigeración, espacio físico, permitiendo ajustar la infraestructura necesaria a la demanda actual, y lo más importante solamente pagando por lo utilizado. En este punto, cuestiones como el aprovisionamiento de ambientes y la elasticidad de los servidores son resueltas. Pero esto soluciona el problema en parte, porque si los sistemas son complejos y grandes requieren de difíciles procedimientos de despliegue, que pueden ocasionar caídas del sistema esta vez por errores en el despliegue o tiempo de ventana necesario para realizarlo donde el usuario puede ser afectado. La complejidad de despliegues se soluciona con la incorporación de contenedores, que aíslan los detalles de dependencias y tienen mucho menor de sobrecarga para iniciarse ya que solamente contienen lo necesario para correr en las VMs disponibles en Cloud. Combinándose efectivamente con estrategias

como las de Canary Release y despliegue Blue-Green constituyen una solución óptima. La inclusión de contenedores en el ciclo de desarrollo y en operaciones ayudan a adoptar prácticas de Integración Continua y Delivery Continuo por sus características de poder levantarse y bajarse muy simplemente, que las diferencian con las VMs convencionales. Esto último administrado y monitoreado por la generación de clústeres de contenedores gestionados por orquestadores que van a estar velando por la salud de los mismos. Estas últimas tecnologías no son todavía muy adoptadas en Argentina, pero como se investigó le sería posible y conveniente a un emprendedor empezar a utilizarlas ya que no se requiere inversión inicial. Entonces desde el punto de vista de esta investigación la combinación de las tecnologías de cloud computing, contenedores y orquestadores, sumadas a técnicas que desde hace más tiempo vienen ayudando a mitigar la carga de los servidores como el servidor de cache, priorizadores de carga, balanceadores de carga, proxies reversos y CDNs que son excelentes complementos a la solución propuesta para sistemas críticos en la suma de fiabilidad y robustez. Se puede vislumbrar que, aunque la complejidad de la implementación de estas tecnologías se ve incrementada al principio, el resultado a largo plazo es muy satisfactorio respecto a costos y tiempo. Por otro lado, se encontraron algunas restricciones para poder implementar esta solución, como la necesidad de estructurar el sistema para que se descomponga efectivamente en contenedores, por ejemplo, con una arquitectura de microservicios, y no se intente llevar un sistema tradicional monolítico directamente a un contenedor. Por otro lado, existen situaciones donde no toda la información o tecnologías pueden llevarse a cloud y/o contenedores como, por ejemplo, tecnologías de servidores IBM que pueden ser subsanadas mediante implementaciones de cloud híbridas con conectores ofrecidos para tal fin.

En el estudio realizado sobre el mercado local (Córdoba, Argentina) mediante instrumentación de encuesta y entrevistas a profesionales de IT se observó que se conoce de las tecnologías, pero no muy bien cuáles son los beneficios que se obtienen del uso de contenedores y orquestadores. Fue evidente que la preocupación principal de los encuestados y entrevistados se alinea a los tiempos de respuesta de la aplicación y la utilización de recursos, donde el primer factor afecta a la imagen de empresa y el segundo a los costos asociados a mantener dicho rendimiento. Esto ayuda a sustentar la idea de que los contenedores y orquestadores permitirían atender ambos aspectos ya que representan las

ventajas de los mismos, asociados a rendimiento y elasticidad. Un punto importante es que la mayoría tienen implementados mecanismos de mitigación de carga (como balanceadores, servidores cache y priorizadores de carga), que complementadas con orquestadores y contenedores contribuyen a un excelente rendimiento desde el punto de vista de usuario. La mayoría de las empresas utilizan máquinas virtuales, y las mismas se podrían utilizar para correr contenedores sobre ellas, ya que como se explicó se complementan perfectamente. Finalmente, se comprobó que la adopción de esta tecnología en el ámbito local está recién comenzando, lo cual representa una oportunidad de mejora a las aplicaciones actuales.

De acuerdo a lo expuesto anteriormente se recomienda el uso de **orquestadores, contenedores y cloud computing**. Esta combinación es muy potente, ya que como se explicó durante el desarrollo del presente trabajo, los orquestadores ayudan a la elasticidad de la demanda recibida y son capaces de ajustar el uso de recursos en Cloud. Adicionalmente, Cloud permite que se ahorre en cuestiones mantenimiento e inversión en hardware, siendo más económicos en uso de recursos. Y por otro lado los contenedores facilitan la implementación de microservicios, continuous integration y delivery, reduciendo la complejidad de mantenimiento de software ya que se trabajan como unidades atómicas de despliegues.

Bibliografía

- **Barkai, D** (2000), An Introduction to Peer-to-Peer Computing, Estados Unidos, Intel Press
- **Bourke, T** (2001), Server Load balancing, Estados Unidos, O'Really & Associates Inc.
- **Docker** (2016), Docker for Virtualization Admin eBook, Estados Unidos, Docker
- **Dua, R** (2016), Introduction to Docker, Estados Unidos, Packt Publishing Ltd.
- **Friedman, J** (2015), Definitive Guide to Cloud Access Security Brokers, Estados Unidos, CyberEdge Press
- **Forrester Research** (2016), The state of Containerization, Estados Unidos, Forrester
- **Gallagher, S.** (2016), Docker in Production, Estados Unidos, Packt Publishing Ltd.
- **Gindi, J.** (2016), What is Docker and Why is it So Popular?, Estados Unidos, Packt Publishing Ltd.
- **Goldberg, R.P** (1972), Architectural Principles for Virtual Computer Systems, Estados Unidos, Division of Engineering and Applied Physics, Harvard Univ., Cambridge Mass.
- **Hernandez-Campos, F** (2003), Tracking the Evolution of Web Traffic: 1995-2003, Estados Unidos, University of North Carolina at Chapel Hill
- **Humble, J** (2011), Continous Delivery, Estados Unidos, Pearson Education, Inc.
- **Intel** (2014), Linux Containers Streamline Virtualization and Complement Hypervisor-Based Virtual Machines, Estados Unidos, Intel Corporation
- **Iyengar, A** (2004), "Web Caching, Consistency, and Content Distribution", Estados Unidos, IBM Resarch - CRC Press LLC
- **JCGs (Java Code Geeks)** (2016), Docker Containerization Cookbook, Estados Unidos, Exelixis Media P.C.
- **Jessen, E** (1996), Die Entwicklung des virtuellen Speichers, Alemania, Informatik Spektrum
- **Kohli, V** (2017), Troubleshooting Docker, Estados Unidos, Packt Publishing Ltd.
- **Miells, I** (2016), Docker in practice, Estados Unidos, Manning Publications Co.
- **Newman, S** (2015), Building Microservices Designing Fine Grained Systems, Estados Unidos, O'Really Media Inc.
- **Nickoloff, J** (2016), Docker in Action, Estados Unidos, Manning Publications Co.
- **Parmelee, R.P., Peterson, T.I., Tillman, C.C. and Hatfield, D.J** (1976), Virtual storage and virtual machine concepts, Estados Unidos, IBM Systems
- **Sharma, S** (2016), Microservices: Building Scalable Software, Estados Unidos, Packt Publishing Ltd.

- **Sourabh, S** (2016), Mastering Microservices with Java, Estados Unidos, Packt Publishing Ltd.
- **Smith, R** (2017), Docker Orchestration, Estados Unidos, Packt Publishing Ltd.
- **Sommerville, I** (2011), Ingeniería de Software, Estados Unidos, Pearson Education
- **Soppelsa, F** (2016), Native Docker Clustering with Docker Swarm, Estados Unidos, Packt Publishing
- **RighScale** (2018), The State of Cloud Report, Estados Unidos, RighScale Inc.
- **Tanenbaum, A** (2009), Sistemas Operativos Modernos, 3ra edición. México, Pearson Education
- **Tanenbaum, A** (2003), Redes de Computadoras, 4ta edición, México, Pearson Education
- **Verona, J** (2016), Practical DevOps, Estados Unidos, Packt Publishing Ltd.
- **Vieytes, R** (2004), Metodología de la Investigación en Organizaciones, Mercado y Sociedad, Argentina, Editorial de las Ciencias.
- **Vohra D** (2016)**,** Kubernetes Microservices with Docker, Estados Unidos, Apress
- **Wittin, M** (2017), Exploring Cloud Computing, Estados Unidos, Manning Publications Co.

ANEXOS

ANEXO A - Modelo de Entrevista

Entrevista

Fecha: --/--/--

Entrevistado: ---------

Posición: -------

- ¿Cómo resuelven hoy la demanda de su servicio de más alta demanda?
- ¿Hasta cuantas conexiones concurrentes puede estar atendiendo como pico?
- ¿Cuáles son las tecnologías utilizadas?
- ¿Cuáles son las desventajas o ineficiencia al método actual?
- ¿Podrías explicar cuáles serían tus expectativas del manejo automático de altas demandas?
- ¿Conoce otras aplicaciones en la empresa que necesiten de este tipo de manejos?

ANEXO B - Entrevista a Fernando Hevia

Fecha: 18/10/17 19:00 hs

Entrevistado: Fernando Hevia

Posición: Director de Infraestructura de Ilumno / Dueño de Consultora de IT SplendIT

Web: https://www.ilumno.com

- ¿Cómo resuelven hoy la demanda de un servicio mayor demanda?
En el caso de Epic se poseen entre 4 y 6 nodos que tienen los servidores de aplicaciones y una capa de APIs. Luego tenemos un cluster de SQL Server 2012 con un servidor master (escritura/lectura) y un esclavo (solo lectura) donde la aplicación hace balanceo de carga de operaciones de bases de datos. Para distribuir el tráfico se utilizan balanceadores que balancean según la cantidad de conexiones activas en cada nodo. Todas estas máquinas son máquinas virtuales que se encuentran en un datacenter en Miami.

- ¿Hasta cuantas conexiones concurrentes puede estar atendiendo como pico?
En un día como hoy estamos viendo que aproximadamente 1500 personas por día. En el caso de Epic, no es univoca con la cantidad de conexiones. Por ejemplo, en este momento (19:15hs del 18/10) veo 700 conexiones en cada uno de los servidores web.

- ¿Cuáles son las tecnologías utilizadas en el sistema?
La aplicación está básicamente construida en .NET, por lo cual está utilizando Internet Information Services (IIS) con Microsoft SQL Server

- ¿Cuáles son las desventajas o ineficiencia al método actual?

Nosotros tenemos recursos fijos asignados, que la mayoría del tiempo no están siendo aprovechados. Me refiero a CPU, memoria que tiene cada servidor de aplicaciones. Tranquilamente si se cayera hoy uno de los cuatro servidores, no afectaría la percepción del servicio. Con lo cual cuando no existen los picos la cantidad de hardware esta sobreestimada. Pero, por otro lado, si existen problemas de performance por picos de demanda, tenemos que agregar al balancer manualmente nuevos servidores de aplicación para atender las peticiones.

- ¿Podrías explicar cuáles serían tus expectativas del manejo automático de altas demandas?

Hay dos cuestiones que me gustarían de un sistema de manejo de demanda: por un lado, es que cuando una aplicación no tenga mucha demanda, los recursos estén disponibles, y en el caso de que los necesite los tome. El hecho de tener servidores ociosos tiene gastos aparejados desde la electricidad que consume, al hecho de no poder asignarlo a otra aplicación si no hubiera más recursos sin asignar.
Por otro lado, también me parece interesante que ante la caída de servidores los mismos se recuperen automáticamente. Ya que hoy, hay que hacerlo de acuerdo al monitoreo que tenemos implementado en Zabbix y posteriormente realizarlo a mano.

- ¿Conoce otras aplicaciones en la empresa que necesiten de este tipo de manejos?

Epic es el sistema que más demanda externa tiene por ser utilizado por la red ilumno (13 Universidades y más de 200000 alumnos). Pero me imagino que los servicios de integración de la empresa necesitan de un mecanismo similar. Los mismos van a interconectar, microservicios, Learning Management System, Customer Relationship Management, Enterprise Resourse Planning, Content Management Systems y otros sistemas legados de las universidades. Las cantidades de conexiones van a ser muy altas por la gran cantidad de sistemas interconectados.

ANEXO C -Entrevista a Dafne Defant

Fecha: 26/05/18 17:00 hs

Entrevistada: Dafne Defant

Posición: Responsable de IT / co-fundadora de M2M Monitoreo

Web: http://www.m2mmonitoreo.com.ar/

- ¿Cómo resuelven hoy la demanda de un servicio mayor demanda?
El sistema de seguimiento de nuestra empresa tiene un backend donde los equipos de vehículos envían peticiones informando la ubicación, donde tenemos al momento 410 equipos reportando y generando peticiones cada 20 segundos. Allí registran la posición GPS y reciben un ACK. Luego se hace un post procesamiento de las coordenadas GPS a una geocodificación.

- ¿Hasta cuantas conexiones concurrentes puede estar atendiendo como pico?
Hoy se tiene contratado un servicio de hosting con recursos de más que se están desaprovechando. Estimo que fácilmente podría atender al doble de los equipos actuales. En caso de que se termine la capacidad de procesamiento del actual servidor de producción, contrataría un servidor adicional paralelo. El problema actual es el post procesamiento de las coordenadas GPS por el servidor Postgresql que convierte más mismas a direcciones (geocodificacion).

- ¿Cuáles son las tecnologías utilizadas en el sistema?
El sistema es de geolocalización: Posgresql es el motor de BD, después el backend es Java y el frontend es PHP con framework Zen.

- ¿Cuáles son las desventajas o ineficiencia al método actual?

Desaprovechamiento de recursos. En un principio se tenía un servidor de mapas propios, luego al cambiar la arquitectura del sistema y hacer peticiones directamente a OpenStreet, ese procesamiento no es más realizado por nosotros, por esa razón disminuimos el procesamiento en servidores propios, y allí los servidores quedaron sobredimensionados.

- ¿Podrías explicar cuáles serían tus expectativas del manejo automático de altas demandas?

Que el cliente no note la diferencia en horas pico. Por otro lado, me imagino que ahorraría dinero en infraestructura si se adaptara a la demanda. No me podría cambiar a un servicio cloud porque necesito la ip publica que ya tengo en los equipos instalados más antiguos, ya que es un requerimiento de los equipos de geolocalizacion.

- ¿Conoce otras aplicaciones en la empresa que necesiten de este tipo de manejos?

Tenemos una aplicación web del tipo administrativa, si bien en este momento tenemos poca demanda de recursos para esta aplicación, identifico que te me está generando una limitación a la hora de elegir tecnologías porque, por ejemplo, no puedo usar una versión más nueva de PHP ya que el frontend de nuestra aplicación tiene una versión antigua ya instalada en el servidor compartido.

ANEXO D - Modelo Encuesta

1. ¿Considera que las caídas de sus sistemas perjudican sobre qué aspectos mayoritariamente?

- Imagen de la empresa
- Perdidas económicas
- No considero que afecte considerablemente.
- Otras:

---

2. ¿Los sistemas que más peticiones reciben, tienen algún mecanismo de mitigación de carga?

- Si
- No

---

3. ¿Conoce cuál es el límite de conexiones que soporta su sistema con más peticiones?

- Si
- No

4. ¿Utiliza algún otro mecanismo de seguridad para evitar peticiones no genuinas, por ejemplo, de ataques de denegación de servicio?

- Si
- No

En caso de ser afirmativa la respuesta por favor mencione cuales:

---

5. ¿Considera que sus sistemas son robustos y seguros?

- Si
- No

En caso de responder de manera negativa por favor mencione que otros aspectos usted considera necesarios cubrir:

---

6. ¿Considera otras implementaciones como mitigaciones para manejar la demanda? Puede responder más de una opción si así lo considera.

- Balanceo de carga con servidores redundantes físicos o virtuales
- Utilización de contenedores con orquestadores
- Solo filtrado de tráfico no deseado es suficiente.
- Aumento de infraestructura en forma temporal para soportar situaciones de alta carga

---

7. ¿Conoce la utilización de orquestadores y contenedores?

- Si, los contenedores son similares a las máquinas virtuales
- No
- Tengo una noción porque escuche hablar del tema.
- Sí, pero no sé cómo me servirían.
- Si, conozco los beneficios, pero desconozco como implementarlo.

---

8. ¿Los orquestadores y contenedores reemplazaran a las máquinas virtuales?
    - Si
    - No

---

9. Aparte de orquestadores, balanceadores ¿Existe otra herramienta que pueda ayudar a manejar la carga sobre el sistema?

- Si
- No

En caso de responder afirmativamente por favor mencione cual:

---

10. ¿Qué empresas pueden podrían utilizar contenedores y sus orquestadores? Puede seleccionar más de una

- Pymes
- Medianas Empresas
- Grandes Empresas
- Empresas Gubernamentales

---

11. ¿Considera que orquestadores y contenedores son implementación económica?

- Si
- No

---

12. ¿Qué opina sobre implementación de orquestadores en la cloud computing? Puede marcar más de una opción

- Es la implementación más económica
- Es segura
- No es segura
- Es más rápido que en infraestructura de la empresa.

---

13. ¿Qué aspectos sobre los sistemas con alta cantidad de peticiones son importantes para Ud?
    - Tiempo de respuesta
    - Seguridad
    - Recuperación ante caídas
    - Costos incurridos por recursos ociosos
    - Filtrado de peticiones no auténticas.

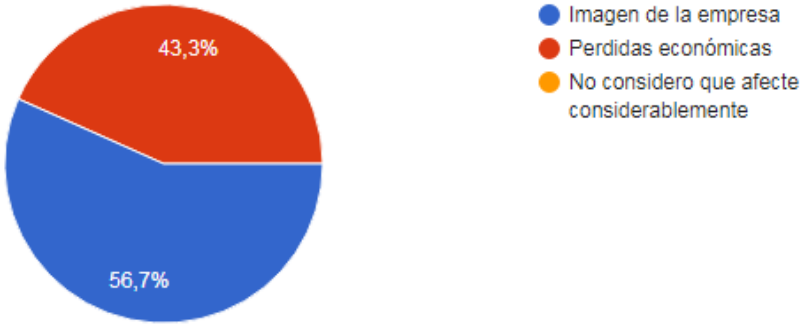
En qué orden los ubicaría de mayor a menor de acuerdo a su importancia:

ANEXO E - Resultado de Encuesta

La misma fue realizada entre 06/06/2018 al 13/06/2018. Los entrevistados fueron preseleccionados de acuerdo a su perfil (DevOps, Infraestructura, Administradores de Sistema, Operaciones). Los mismos se encuentran en la provincia de Córdoba (Argentina), y pertenecen a empresas como: MercadoLibre, McAfee, Olapic, Universidad Siglo 21, BitLogic, Ilumno, entre otras.

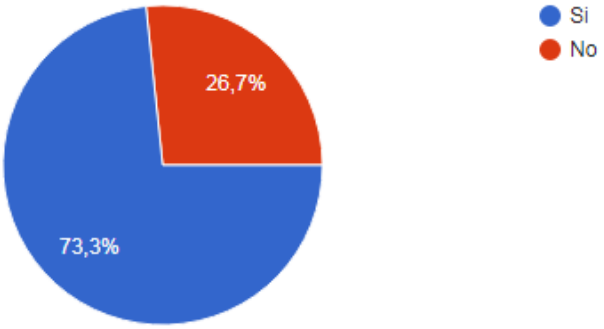
1. ¿Considera que las caídas de sus sistemas perjudican sobre qué aspectos mayoritariamente?

30 respuestas



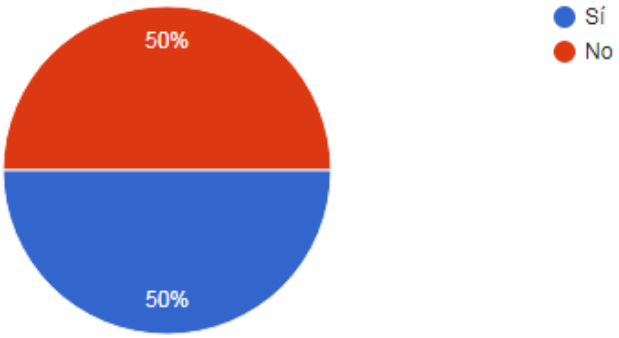
2. ¿Los sistemas que más peticiones reciben, tienen algún mecanismo de mitigación de carga?

30 respuestas



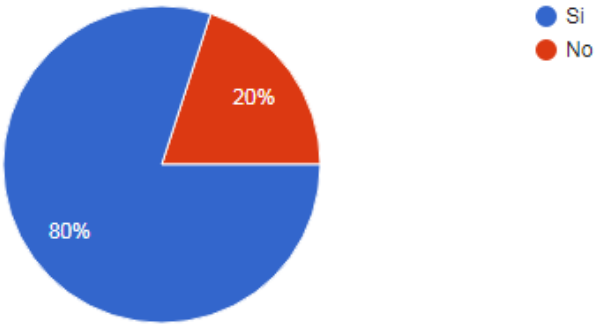
3. ¿Conoce cuál es el límite de conexiones que soporta su sistema con más peticiones?

30 respuestas



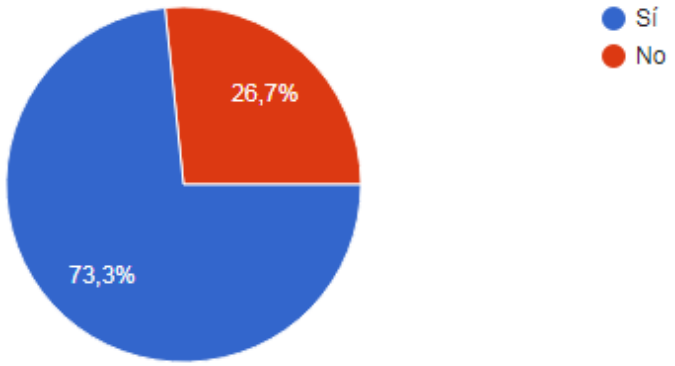
4. ¿Utiliza algún otro mecanismo de seguridad para evitar peticiones no genuinas, por ejemplo, de ataques de denegación de servicio?

30 respuestas



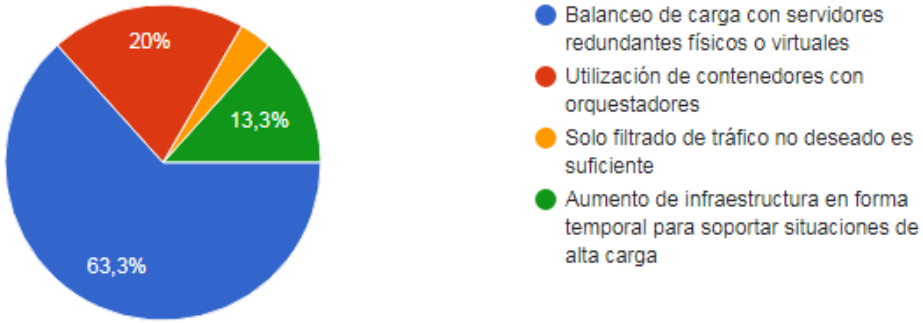
5. ¿Considera que sus sistemas son robustos y seguros?

30 respuestas



6. ¿Considera otras implementaciones como mitigaciones para manejar la demanda?

30 respuestas

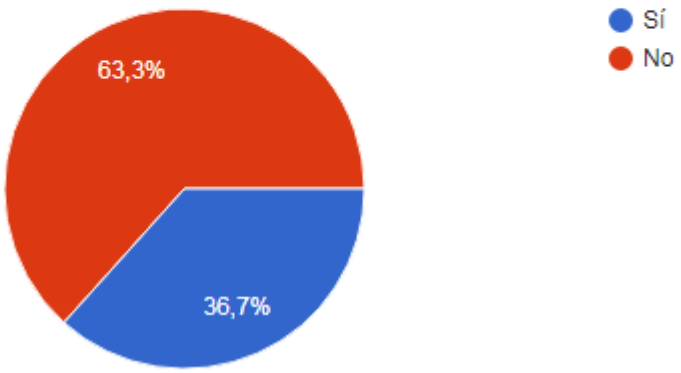


7. ¿Conoce la utilización de orquestadores y contenedores?

30 respuestas

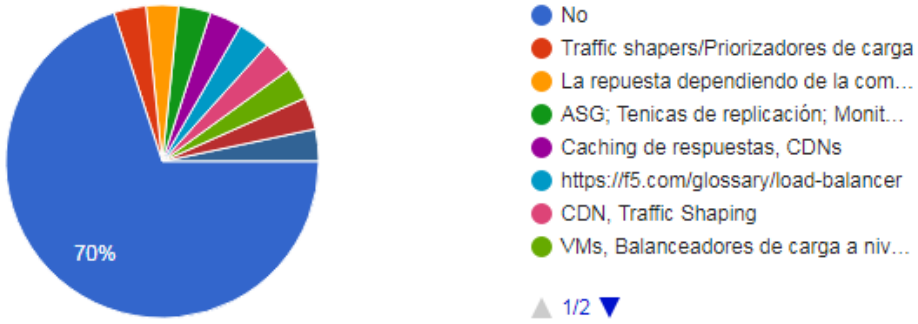
8. ¿Los orquestadores y contenedores reemplazaran a las balanceadores y máquinas virtuales?

30 respuestas



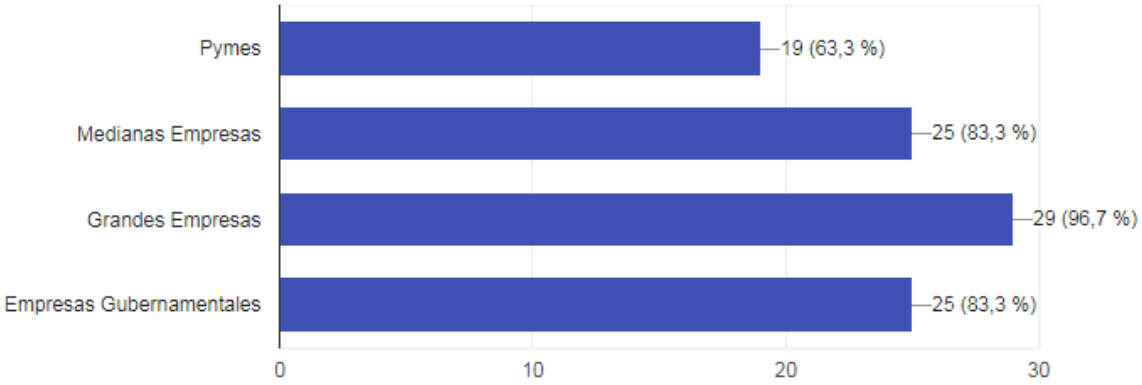
9. Aparte de orquestadores, balanceadores ¿Conoce otra herramienta que pueda ayudar a manejar la carga sobre el sistema?

30 respuestas



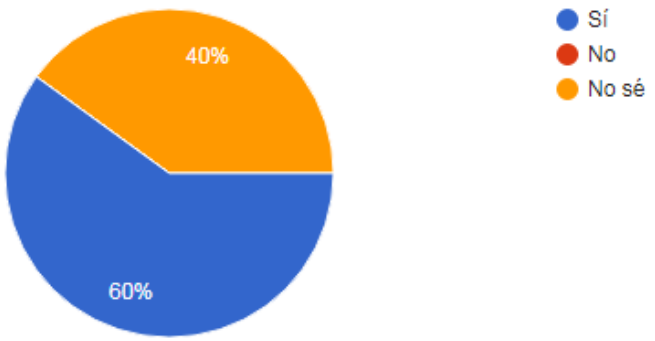
10. ¿Qué empresas pueden podrían utilizar contenedores y sus orquestadores? Puede seleccionar más de una

30 respuestas



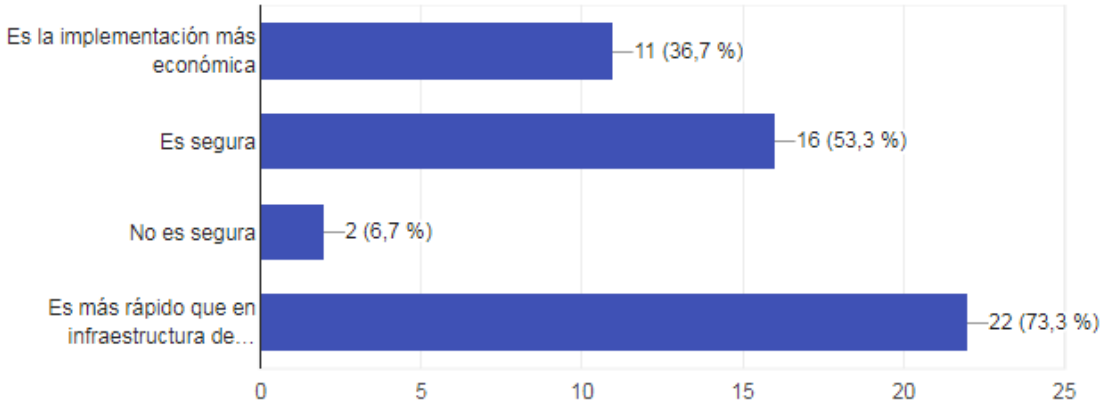
11. ¿Considera que orquestadores y contenedores son implementación económica?

30 respuestas



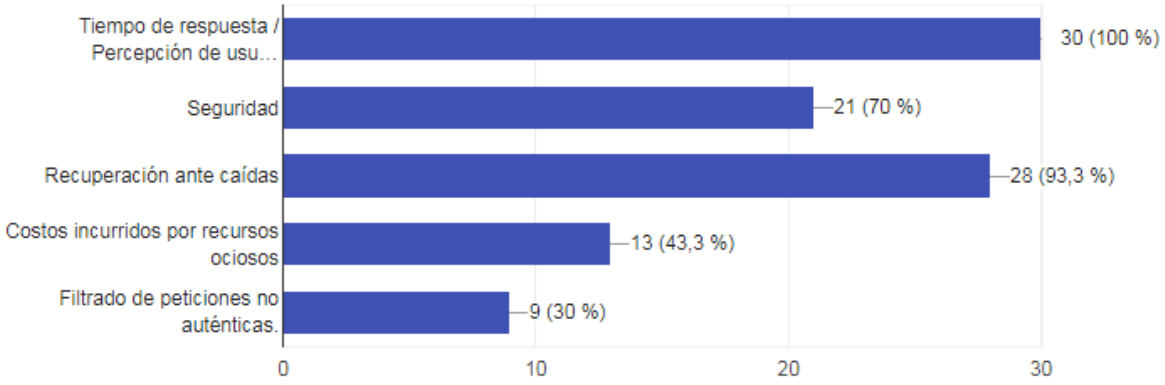
12. ¿Qué opina sobre implementación de orquestadores en cloud computing? Puede marcar más de una opción

30 respuestas



13. ¿Qué aspectos sobre los sistemas con alta cantidad de peticiones son importantes para Ud?

30 respuestas



ANEXO F – FORMULARIO DESCRIPTIVO DEL TRABAJO FINAL DE GRADUACIÓN

AUTORIZACIÓN PARA PUBLICAR Y DIFUNDIR TESIS DE POSGRADO O GRADO A LA UNIVERIDAD SIGLO 21

Por la presente, autorizo a la Universidad Siglo21 a difundir en su página web o bien a través de su campus virtual mi trabajo de Tesis según los datos que detallo a continuación, a los fines que la misma pueda ser leída por los visitantes de dicha página web y/o el cuerpo docente y/o alumnos de la Institución:

Autor-tesista *(apellido/s y nombre/s completos)*	Funes, Franco Nelson
DNI *(del autor-tesista)*	30782169
Título y subtítulo *(completos de la Tesis)*	Análisis de solución eficiente para escalabilidad de sistemas online mediante orquestación de contenedores
Correo electrónico *(del autor-tesista)*	Funesfranco@gmail.com
Unidad Académica *(donde se presentó la obra)*	Universidad Siglo 21

Otorgo expreso consentimiento para que la copia electrónica de mi Tesis sea publicada en la página web y/o el campus virtual de la Universidad Siglo 21 según el siguiente detalle:

Texto completo de la Tesis *(Marcar SI/NO)*[1]	SI
Publicación parcial *(Informar que capítulos se publicarán)*	

Otorgo expreso consentimiento para que la versión electrónica de este libro sea publicada en la página web y/o el campus virtual de la Universidad Siglo 21.

Lugar y fecha: _____

Firma autor-tesista **Aclaración autor-tesista**

Esta Secretaría/Departamento de Grado/Posgrado de la Unidad Académica: _____certifica que la tesis adjunta es la aprobada y registrada en esta dependencia.

Firma Autoridad Aclaración Autoridad

Sello de la Secretaría/Departamento de Posgrado

[1] Advertencia: Se informa al autor/tesista que es conveniente publicar en la Biblioteca Digital las obras intelectuales editadas e inscriptas en el INPI para asegurar la plena protección de sus derechos intelectuales (Ley 11.723) y propiedad industrial (Ley 22.362 y Dec. 6673/63. Se recomienda la NO publicación de aquellas tesis que desarrollan un invento patentable, modelo de utilidad y diseño industrial que no ha sido registrado en el INPI, a los fines de preservar la novedad de la creación.