Proyecto Final – Práctica en Empresa

Escalabilidad y Disponibilidad en Infraestructuras de Servicios Orientadas a la Web, basadas en Cloud Computing

Autor: Luis Nicolas Calmels

Tutor (Facultad): Abel Crespo

Tutor (empresa): Leonardo Torres

Carrera: Ingeniería en Sistemas

DNI: 32.578.252

Legajo: 3315

ÍNDICE

INTRODUCCIÓN

El presente trabajo se realiza como Trabajo Final para la Carrera Ingeniería en Sistemas (Plan 2004), de la Facultad de Ingeniería de la Universidad Nacional de La Pampa, bajo la modalidad "Práctica en Empresa".

El mismo persigue como objetivo, poder brindar una Solución de infraestructura TI Escalable y Disponible, utilizando tecnologías de Computación en la Nube o "Cloud Computing".

Se pretende dar a conocer el concepto de la computación en la nube, sus características principales, las diferencias que tiene con las demás alternativas TI en la actualidad, y sus ventajas y desventajas.

Se detallarán los diferentes pasos y las especificidades a tener en cuenta para lograr el objetivo propuesto, en un proveedor que reúna las características que se buscan para la infraestructura que se quiere conseguir.

Para poder llevar a cabo lo detallado en los párrafos anteriores, se detallará la solución que se le dio a un cliente en particular dentro de la empresa donde se llevó a cabo el presente trabajo, con el objetivo de poder dar a conocer cómo se realiza la configuración de una infraestructura de Alta Disponibilidad dentro de uno de los proveedores de Cloud Computing más grande del mundo.

Se evidenciarán las ventajas de utilizar Nubity como Administrador de plataformas en la Nube, para dar soluciones de administración, monitoreo y respaldo.

Por último, se describirán diferentes escenarios en los cuales el modelo creado pueda sufrir fallas de desempeño y/o caídas, y se buscará demostrar que dicho modelo alcanza los requerimientos necesarios para garantizar la Escalabilidad y Alta Disponibilidad mencionada anteriormente.

CAPITULO I

COMPUTACIÓN EN LA NUBE

1. Introducción

En este capítulo se introducirá conceptualmente el significado de computación en la nube o *cloud computing*, luego se hará lo propio en relación a los modelos de servicios que los proveedores en la nube ofrecen a organizaciones y particulares, y posteriormente se realizará una clasificación modular de cada una de las arquitecturas existentes. En la sección 1.2 y 1.3 se describirán las ventajas y desventajas de una infraestructura de software en la nube, y finalmente en la sección 1.4 se presentara y definirá sintéticamente el objetivo de este trabajo de tesis.

1.1 Modelos de Servicios en la nube

Cuando un desarrollador debe implementar una solución tecnológica para dar respuesta a las demandas de una organización en lo relativo a tecnologías de la información (TI), en primera instancia debe calcular cuales son los requerimientos de infraestructura necesarios para satisfacer la carga estimada a la que se la someterá y luego deberá seleccionar los componentes de redes, el hardware de computo (servidores), sistema operativo, aplicaciones, y todos los elementos que conformaran la solución de software definitiva. Concluida esta etapa, el sistema comenzara a operar y la infraestructura evolucionará en el tiempo en función de la demanda. La Fig. 1.1 ilustra un ejemplo descriptivo en la dinámica en una infraestructura en TI.

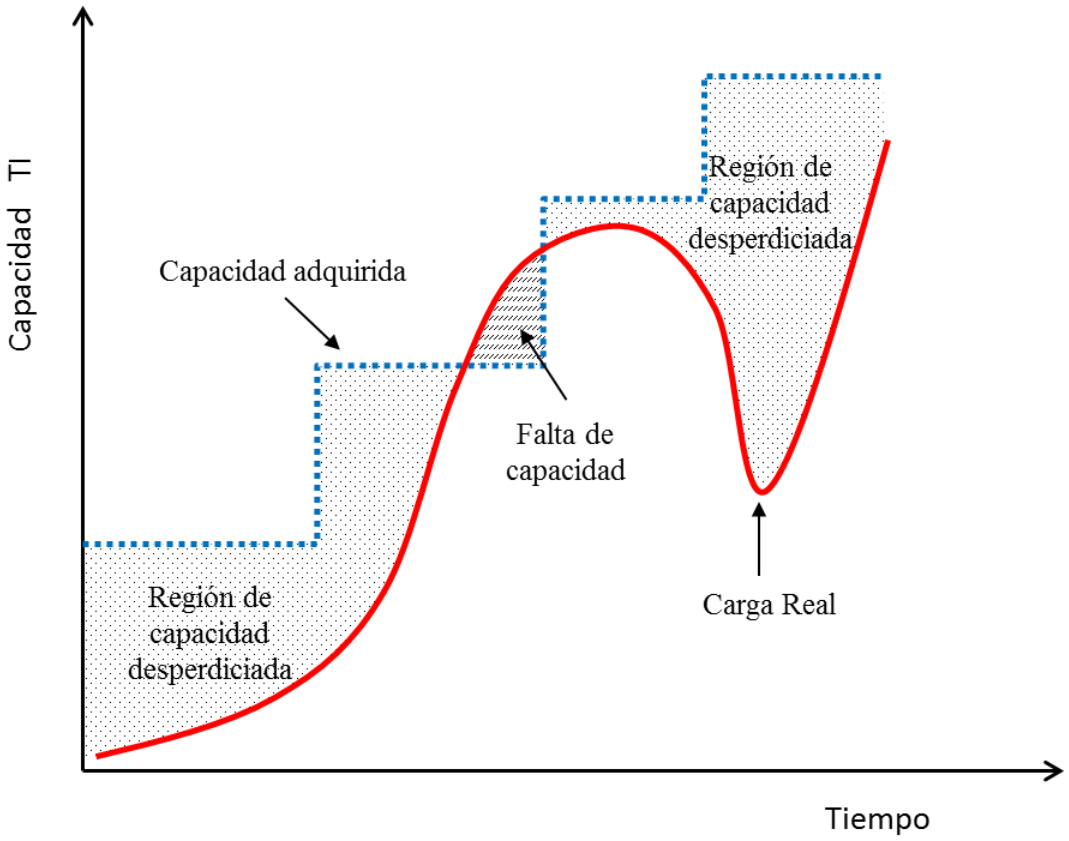

Fig. 1.1 - Dinámica en la Infraestructura TI

Tal como se observa en la Fig. 1.1 existen zonas en que la capacidad de la infraestructura TI es superior a la carga real y viceversa. Lo que sí es evidente, que para incrementar la capacidad de la infraestructura, es necesario realizar costosas inversiones que incluyen actualizaciones en tecnologías, incremento en el número de horas de ingeniería de recursos humanos especializados y mayores costos por servicios energéticos.

En una organización que posea su propia infraestructura TI, un esquema optimizado para responder en sintonía fina con la carga real de acuerdo a lo que expresa la Fig. 1.1, debería adaptarse a picos de alta demanda y desperdiciar capacidad cuando la carga disminuya considerablemente respecto de la adquirida. Las consecuencias de lo descrito en este párrafo derivan en un sistema de infraestructura en TI poco flexible que redundará en mayores costos para la organización. [1]

Fue así como las empresas tecnológicas y operadoras tradicionales vieron la oportunidad de ampliar sus negocios ofreciendo servicios de computación en la nube o *cloud computing*. Los servicios ofrecidos bajo este paradigma, reportan ventajas en forma de flexibilidad, eficiencia, y disminución del esfuerzo inversor a organizaciones y particulares.

Una definición formal de computación en la nube es la siguiente: “modelo que permite acceder de forma cómoda y ubicua, a petición del usuario, a una serie de recursos informáticos compartidos y configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden suministrar con rapidez y distribuir con un esfuerzo mínimo de gestión o interacción del proveedor de servicios” [2].

En la actualidad se observa una tendencia creciente en el número de pequeñas, medianas y grandes organizaciones que basan su infraestructura de servicios orientadas a aplicaciones en Internet sobre plataformas de computación en la nube o *cloud computing*. Los modelos de servicios que los proveedores ofrecen a través de la nube pueden clasificarse según tres categorías:

- Infraestructura como un Servicio o IaaS *(Infrastructure as a Service)*: modelo de servicios en el que al cliente se le ofrece tanto un medio de almacenamiento básico como una serie de capacidades de cómputo en la red. Todo ello haciendo uso de sistemas operativos virtualizados y servidores ubicados en la nube a los que el usuario accede a través de la Internet.
- Plataforma como un Servicio o PaaS *(Platform as a Service)*: para el desarrollo web y de software en general. Modelo que ofrece todo lo necesario para soportar el ciclo de vida completo de construcción y puesta en funcionamiento de aplicaciones y servicios a través de Internet con software en la nube.
- Software como un Servicio o SaaS *(Software as a Service)*: modelo de servicios en el que al cliente se le proporcionan aplicaciones a través de Internet. Tanto el software como los datos empleados por el usuario quedan alojados en los servidores del proveedor de servicios en la nube, accediendo el cliente a ellos mediante un navegador web. El cliente no realiza desarrollo de software sino que utiliza aplicaciones brindadas por el proveedor de la nube.

La Fig. 1.2 ilustra un modelo en capas de las categorías definidas en computación en la nube (IaaS, PaaS, y SaaS) en contraste con una infraestructuras de software clásico. A continuación se detallan, con independencia del modelo de servicio utilizado, los niveles o capas que integran los servicios TIC ofertados al cliente (Fig. 1.2):

- Networking: red de interconexión como medio de comunicación entre los diferentes dispositivos que integran la infraestructura TIC.
- Almacenamiento: capacidad de registro y de datos disponible en un determinado disco físico o virtual.
- Servidores: equipamiento con capacidad para ejecutar determinadas aplicaciones. De esta manera el usuario final únicamente tendría que instalar las aplicaciones en el servidor, que estaría listo para ser utilizado en todo momento por cualquier usuario.
- Virtualización: capa de abstracción entre el hardware y el sistema operativo de la máquina virtual, haciendo posible la compartición de recursos entre diversos entornos de ejecución.
- Sistema Operativo: programa o conjunto de programas que gestionan los recursos hardware (ya sean físicos o virtuales) disponibles, proveyendo acceso a las diferentes aplicaciones.
- Middleware: capa de abstracción software (complejidad y heterogeneidad de las redes de comunicaciones subyacentes) que posibilita la comunicación entre las aplicaciones y los sistemas operativos o lenguajes de programación.
- Ejecución: entorno para la prueba y ejecución de las aplicaciones lanzadas por el usuario a través del cual se gestionan las diferentes operaciones subyacentes.
- Datos: entorno en el que se sitúa la información de entrada/salida necesaria para la ejecución de una o varias aplicaciones.
- Aplicaciones: conjunto de programas de alto nivel que ofrecen al usuario final determinados servicios a modo de herramienta.

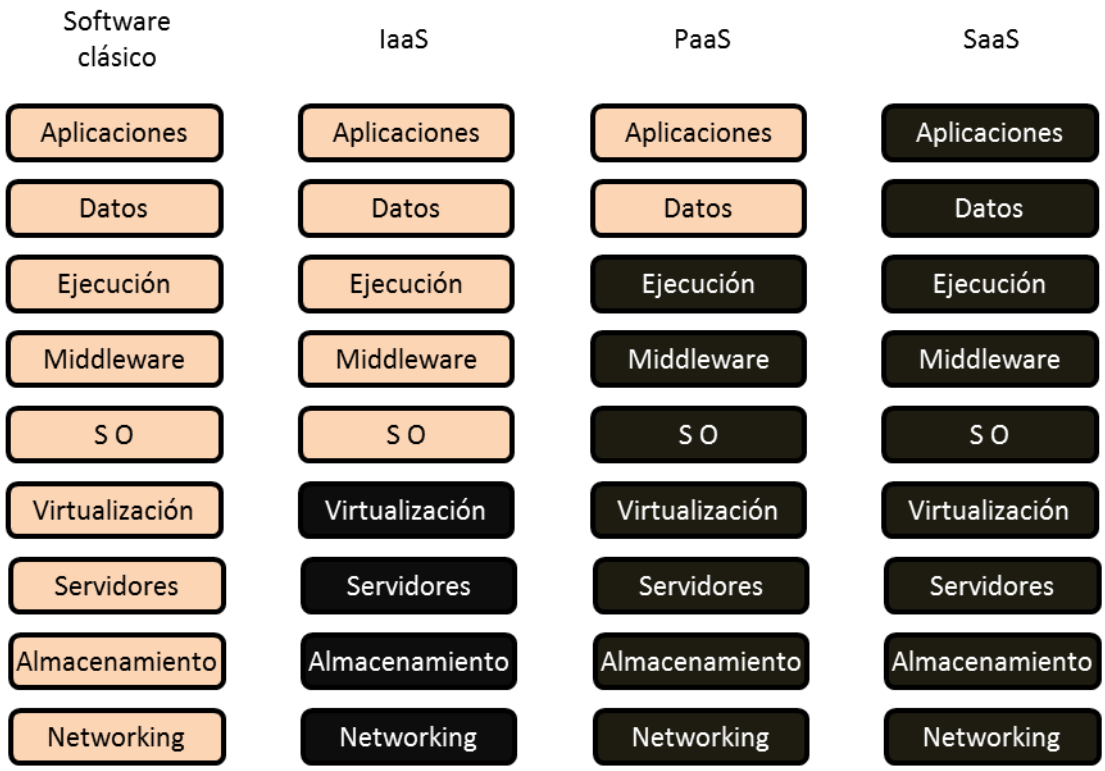

Fig. 1.2: Modelos de servicio tradicional y en la nube

El modelo tradicional definido en la Fig. 1.2 (software clásico), es aquel que no hace utilización de la nube y supone que el cliente u la organización es el encargado de la provisión del software y hardware necesario para garantizar el correcto funcionamiento de las diferentes aplicaciones, siendo responsable desde el momento de la planificación de la red hasta el de la generación de datos y/o software de aplicaciones. [1]

En el contexto IaaS, el usuario tiene a su alcance todo lo necesario para poder instalar los sistemas operativos en las máquinas virtuales ofrecidas por el proveedor a través de la nube. Para ello es necesario contratar los parámetros que sean necesarios (p. ej. capacidad de cómputo, almacenamiento, etc.) para poder hacer frente a la demanda estimada de los servicios. Por norma general, los parámetros son provistos mediante la virtualización y compartición de los recursos del proveedor

En el caso de PaaS, lo que se contrata es un entorno de desarrollo. Bajo este modelo el usuario final dispondrá de un entorno en el que podrá probar y ejecutar las aplicaciones necesarias. De este modo, el cliente se abstrae de toda la infraestructura, hardware y software necesarios para que el entorno funcione.

Por último, el escenario SaaS es aquel en el que el proveedor proporciona la totalidad de servicios al cliente. Así el usuario final dispondrá únicamente de las aplicaciones a las cuales accede a través de la nube, siendo ejecutadas directamente en las infraestructuras del proveedor.

Los servicios en nube se basan en los proveedores de infraestructura (proveedores de IaaS). Los proveedores de nube pública, como Amazon, Google y Microsoft, son grandes empresas, que poseen los recursos financieros para desarrollar los centros de datos y la conectividad global que se necesita para incursionar en este modelo de negocios. Estos proveedores poseen conocimientos especializados sobre hardware, software y seguridad, así como recursos para defenderse de los constantes ataques físicos y electrónicos. La infraestructura privada (incluida la que está dentro de nubes híbridas y comunitarias) puede estar gestionada por empresas y asociaciones más pequeñas, dependiendo de su escala.

Los proveedores de plataforma (o PaaS) añaden software intermedio a la infraestructura para que los desarrolladores de software puedan programar con mayor facilidad. En particular, facilitan que el software ascienda a un nivel más alto de uso de recursos a petición del usuario y gestione la facturación del cliente. Los propios proveedores de infraestructura normalmente ofrecen productos de plataforma. Otras organizaciones destacadas con especialización específica en plataformas son IBM, Force.com y Facebook. La complejidad de desarrollar y mantener estas plataformas tenderá a favorecer a las empresas más grandes en detrimento de las PYME´s como proveedores de plataforma. [3]

Los desarrolladores de software (proveedores de SaaS) pueden aprovechar la infraestructura y las plataformas en la nube y crear los servicios para que los clientes accedan directamente. Las plataformas en la nube deberían facilitar la entrada de las empresas pequeñas a este mercado. Por ejemplo, SlideShare, que permite que millones de usuarios compartan presentaciones de PowerPoint, fue creada por un matrimonio. Los mismos proveedores de infraestructura y plataforma también suelen ofrecer servicios desarrollados en sus propias nubes. Algunos ejemplos son Office 365 de Microsoft, Google Docs y la tienda de comercio electrónico de Amazon. Un ejemplo interesante del potencial para la integración de servicios entre distintos proveedores es el de Box.com, que permite a sus usuarios editar archivos en su nube de almacenamiento mediante Google Docs. El servicio Box Innovation Network de esta empresa invita a otros distribuidores de software a que proporcionen servicios adicionales en su plataforma.

1.2 Ventajas de la computación en la nube

El objetivo final de cualquier modelo de nube pasa por dar soluciones de TI a las organizaciones que así lo requieran. Teniendo presente lo anterior, parece obvio que tanto la instalación como el mantenimiento de la infraestructura corran a cargo del proveedor de servicios. Desde el punto de vista de los usuarios finales, estos dispondrán de los mismos servicios que utilizaban bajo el modelo tradicional de la infraestructura de Tecnologías de la Información. No obstante, el modelo *Cloud* en sí mismo reporta una serie de ventajas de diversa naturaleza para las organizaciones.

Para aquellos casos en que una organización decida no utilizar servicios en la nube a la hora de implementar su infraestructura de Telecomunicaciones, resultará imprescindible, como mínimo, realizar una inversión inicial en hardware y software para su construcción, sin mencionar el coste asociado a la contratación de personal cualificado para la implantación y el mantenimiento de dicha infraestructura. Conviene recordar que aproximadamente el 70% del presupuesto asignado a TI se dedica al mantenimiento de infraestructuras. Teniendo presente lo anterior se puede considerar adecuado contratar los servicios en la nube, no solo como una vía para evitar esta inversión inicial, sino también como un medio para no tener que gestionar el mantenimiento de la infraestructura de TI. De esta forma, a través de la estandarización del uso de la nube se estaría produciendo una transición entre un modelo que requiere acometer una inversión inicial importante hacia otro en el que únicamente se paga por el consumo que se hace de la infraestructura (en principio, sin requerimiento alguno de inversión inicial). Además, esta sustitución de inversión por consumo, permite que la estructura de costes de la organización sea más flexible y elástica a los cambios (tanto por potencial crecimiento como por disminución del negocio).

Otro de los grandes problemas que surgen en el momento en que una organización pretende implementar una infraestructura de TI adecuada es el tiempo que invierte en hacerlo. El proceso de implantación es largo y tedioso e incorpora fases como la planificación o el aprovisionamiento. De hecho, en término medio se podría decir que una implantación puede suponer meses para una organización. La ventaja de cualquier proveedor de servicios en la nube respecto a una organización cualquiera es que el primero ha tenido que realizar una planificación e implantación similar a la que deben hacer todas las organizaciones, por lo que su infraestructura de TI ya estará operativa. Es por ello que, cuando una organización cualquiera contrata los servicios en la nube de un proveedor, el tiempo que debe pasar hasta que el cliente tiene los servicios contratados operativos es mucho menor que el citado anteriormente, ya que bastará con asignar los recursos ya disponibles en el proveedor del servicio al cliente correspondiente. En este caso, se estaría hablando de un proceso con una duración del orden de los minutos.

Con la proliferación y mejora en las prestaciones de los dispositivos móviles los clientes hacen uso, cada vez más, de aplicaciones/servicios más potentes y accesibles desde cualquier lugar y en cualquier momento del día. Los servicios en la nube favorecen precisamente esta posibilidad, aumentando la movilidad con la que los empleados de las organizaciones que contratan los servicios de *Cloud* realizan su trabajo. Al menos en teoría, esa mayor movilidad de los empleados debería traducirse en un aumento de la productividad de la organización.

La capacidad que posea cualquier proveedor de servicios de Cloud será, presumiblemente, mayor que la que necesite cada uno de sus potenciales clientes. En este sentido, y teniendo en cuenta las economías de escala que se generan en el hardware y software que integran las redes que proporcionan el servicio, las ofertas que puedan hacer los proveedores serán más atractivas para los clientes, que a su vez comprobarán cómo contratar los servicios en la nube les resultará más barato que montar su propia red de TI. Si además se tiene en cuenta que los servicios de la nube son cada vez más populares y que existe la expectativa de que el número de clientes que

contraten dichos servicios crezca exponencialmente, es de suponer que a medida que avance el tiempo los productos se abaraten cada vez más.

Bajo el paradigma del modelo Cloud, los recursos de los que hacen uso los servicios ofertados al cliente están ahora centralizados en las instalaciones del proveedor. Esto posibilita la asignación y liberación de dichos recursos de manera dinámica. Cuando un cliente deje de usar un determinado recurso, éste se le podrá asignar a otro que requiera más. De esta manera se pueden realizar adquisiciones de servicios en los que la modalidad de pago sea por uso y en los que los recursos contratados se ajusten a medida que pase el tiempo con la demanda de la organización.

Por último, el uso de la nube supone una utilización de recursos de manera más eficiente respecto al gasto de energía comparado con el que harían las infraestructuras de TI de los clientes si estuvieran implantadas de manera individual.

1.3 Desventajas de la Computación en la nube

A pesar de las múltiples ventajas que para una organización puede suponer la utilización de la nube, no menos cierto es que existen ciertas dificultades ligadas al proceso de transición desde un modelo tradicional hacia cualquiera de los modelos *Cloud* descritos en apartados anteriores.

Sin lugar a dudas, el mayor inconveniente con el que se enfrentan las organizaciones a la hora de sustituir sus infraestructuras de red por servicios en la nube es la seguridad. Hasta ahora los datos se encontraban físicamente en las instalaciones de las propias organizaciones y no eran administrados por terceros. De ahora en adelante, bajo este nuevo paradigma de servicios en la nube, será el proveedor de servicios *Cloud* el encargado de gestionar y almacenar los datos de sus organizaciones cliente.

Otro de los grandes inconvenientes identificados es la de la privacidad. En este caso el origen de la problemática es idéntico al de la seguridad. La información y los datos no son administrados por los recursos de la propia organización, quedando relegada la función al ámbito de actuación del proveedor de servicios *Cloud*.

Al delegar en el proveedor de servicios *Cloud* toda la organización e implantación de la infraestructura TI se evita trabajo y costear grandes inversiones. Sin embargo, también puede llegar a generar situaciones de dependencia de un tercero en las que por cualquier falla en los sistemas del proveedor de servicios una organización completa puede llegar a quedarse sin servicio.

No resulta fácil para una organización migrar todo aquello que depende de las TI hacia la nube de manera rápida y eficaz. Por ello, se tenderá a una migración progresiva en la que primero se utilizarán los servicios *Cloud* para aplicaciones marginales que no sean de capital importancia. Esta situación provocará la aparición de un escenario de nube híbrida para luego ir, poco a poco, aumentando la presencia de los servicios en la nube. Este contexto intermedio de migración progresiva puede llegar a producir confusión e incertidumbre en los empleados de la organización.

Existencia de vacíos legales que aún no han sido regulados. La información puede almacenarse en países distintos al de la organización y/o el del proveedor de servicios, con las consiguientes divergencias en términos regulatorios de aplicación orientados a la protección y al acceso de datos.

A medida que aumenta la proporción de servicios de TI que se proporcionan a una organización a través de la nube, lo hace también la dependencia que se tiene de los proveedores. Ahora son ellos los que proporcionan todo lo necesario para que las TI en las organizaciones adheridas

funcione. En un caso extremo, podrían llegar a producirse situaciones de monopolio e incluso aumentos de precios por prestación de servicios. Como consecuencia de lo descrito en este párrafo, es imperante un proceso de estandarización de los distintos servicios de computación en la nube a nivel de proveedores de computación en la nube. Ello evitaría el sometimiento de los usuarios a situaciones abusivas producto de la dependencia, factor que facilitaría un proceso de migración inmediato ante condiciones desfavorables en lo relativo a costos y/o a calidad del servicio.

1.4 Objetivos Propuestos

La orientación de este trabajo de tesis, tiene como objetivo replicar el nivel de experticia alcanzado en la temática "computación en la nube", caracterizada por una dinámica exponencial respecto a la diversidad en lo que se refiere a los proveedores del servicio (opciones disponibles) y el alto grado de utilización de estas plataformas por parte de clientes en la forma de organizaciones y empresas de cualquier tipo de envergadura.

El presente trabajo mostrará cómo seleccionar un proveedor de servicios en la nube, respondiendo una serie de interrogantes iniciales:

- ¿Se pretende desarrollar software sobre la plataforma virtual en la nube?
- ¿Únicamente se almacenarán datos en la nube?
- ¿Se pretende configurar servicios de aplicación de red, escogiendo, y luego utilizando una opción de sistema operativo de la amplia gama que los proveedores en la nube ofrecen?
- ¿Cómo empresa desarrolladora de soluciones de software lo que se pretende es que los productos probados y aceptados por los clientes, estén disponibles sobre una plataforma de servicios en la nube?
- ¿Se ofrecerá a clientes productos de software que respondan a una infraestructura SaaS?

Las respuestas a los interrogantes planteados en el párrafo anterior definirá el tipo de infraestructura de servicios en la nube a utilizar: IaaS, PaaS o SaaS.

En el ámbito de este trabajo lo que se pretende es configurar una serie de servicios de aplicación de red que serán definidos en el capítulo 3 del presente informe, y que requieren de:

- Flexibilidad en lo que se refiere la capacidad demandada por los usuarios que acceden a los servicios ejecutados. Se sabe que los proveedores en la nube ofrecen la flexibilidad requerida y los costos asociados dependerá de la demanda. Una mayor demanda, implicará mayores costos en forma de pago al proveedor en la nube y viceversa.
- Alta disponibilidad: Cuando un servicio se ejecuta, necesita de la interrelación de numerosos componentes de software y cuando uno o más de esos componentes, por alguna razón dejan de funcionar, el servicio se interrumpirá. Por esta razón, se requiere de mecanismos que garanticen "alta disponibilidad", de manera que ante fallas de componentes en el sistema, nuevas instancias de los componentes de software afectados garanticen la disponibilidad del servicio considerado.
- Configuración diseño e implementación de servicios de aplicación de red en la infraestructura en la nube en el mínimo tiempo (pocas horas).
- Utilización de un servicio en la nube del tipo SaaS, provisto por IP Address, empresa en que desempeña la practica el autor de este trabajo. Al producto de software se lo conoce comercialmente con el nombre Nubity, y constituye una herramienta para monitorear y administrar aquellos clientes de una IaaS, que contrataron el servicio SaaS descrito.

Mediante Nubity, la empresa IP Address flexibiliza la capacidad demandada en la IaaS y administra los distintos servicios de sus clientes.

Todo lo expresado en esta sección del presente trabajo, deriva en una opción de computación en la nube del tipo IaaS, soportadas en lo referido al dimensionamiento o capacidad de la infraestructura, la administración y la configuración, utilizando una herramienta de servicio de software en la nube del tipo SaaS (Nubity).

1.5 Estructura del Proyecto Final

Esta sección trata sobre la estructura del trabajo de tesis en la modalidad "practica en empresa".

Habiendo sentado las bases conceptuales de computación en la nube (sección 1.1), los tipos de servicios que ofrecen a organizaciones y particulares (clasificación en IaaS, PaaS y SaaS), las ventajas y desventajas de este paradigma de infraestructura (sección 1.2 y 1.3) y una síntesis de los objetivos perseguidos (sección 1.4); el Capítulo 2 tratará sobre el tipo de servicios de aplicación de red a implementar, la topología resultante considerando lo escrito en la sección 1.4, la selección de una infraestructura IaaS en base a criterios de comparación entre proveedores y posteriormente la implementación de una infraestructura de servicios en la nube del tipo IaaS sobre un proveedor concreto.

En el capítulo 3, se procederá a detallar la configuración sobre el proveedor seleccionado. Se detalla la migración de los DNS a la nueva infraestructura, el alta de los nuevos servidores en la nube, las consideraciones generales referidas a la seguridad, la instalación de las aplicaciones específicas en los servidores, y las pruebas pertinentes para comprobar que todo lo instalado está funcionando correctamente.

El capítulo 4 estará dedicado a la migración de la aplicación. Se especificarán todos los pasos necesarios para que los datos de los sitios puedan ser migrados al nuevo proveedor, en los que se detallarán las consideraciones en cuanto a seguridad que es necesario tener, la forma en la cual se va a realizar la migración propiamente dicha, y las modificaciones necesarias que se tienen que realizar en los nuevos servidores para que la aplicación funcione correctamente.

El capítulo 5 se expone todo lo referido a garantizar la Alta Disponibilidad. Se trabaja sobre la redundancia que se tiene que garantizar en los distintos servicios críticos de la infraestructura, para garantizar que la aplicación esté operativa la mayor parte del tiempo posible.

En el capítulo 6 se da una introducción a los servicios de Nubity. Se detalla la forma en la cual se agrega una "nube" dentro de la plataforma, cómo se realizan los backups de toda la infraestructura, y de qué manera se realiza la configuración del monitoreo sobre la misma.

En el capítulo 7, se presentarán distintos escenarios en los cuales la infraestructura creada se va a ver comprometida, detallando el plan de acción a seguir para volver a tener operativa la misma. También se evidenciará a forma en la cual la configuración de la solución en la nube detallada en los capítulos anteriores, ayuda a garantizar la disponibilidad del sitio ante condiciones adversas.

Por último, se elaborarán las conclusiones que surjan del desarrollo de la presente tesis, teniendo en cuenta si los objetivos propuestos al inicio de la misma fueron cumplidos de forma exitosa.

CAPITULO II

SERVICIOS A IMPLEMENTAR EN IAAS

2.1 Introducción

Tal como se expresó en la sección 1.4 del Capítulo I, la infraestructura a configurar tiene que ser flexible a la demanda en lo que se refiere a su capacidad y proveer condiciones de alta disponibilidad en lo que se refiere los servicios de aplicación de red ofrecidos. Para lograrlo habrá que identificar los servicios críticos y configurarlos de modo tal que ante contingencias no deseadas, los servicios continúen funcionando "normalmente" a la percepción de los usuarios finales.

En el proyecto que fuera aprobado por la comisión de enseñanza del Consejo Directivo de la Facultad de Ingeniería de la Universidad Nacional de La Pampa, la infraestructura de servicios de aplicación de red a implementar considera "una organización con los siguientes componentes de software y de sistema operativo":

- Sistema operativo Centos 6.5 (GNU/Linux).
- Servicio web bajo Apache versión 2.2
- Varnish Cache: acelerador de aplicaciones web, también conocido como caché de proxy HTTP inversa.
- Un sistema de gestión de bases de datos relacional, multi-hilo y multiusuario, MySQL-Percona.
- Como lenguaje de programación de uso general de código web para el desarrollo web de contenido dinámico se utilizará PHP versión 5.3
- Se proveerá un servicio de FTP *(File Transfer Protocol)* basado en el software (lado servidor PROFTPD.

En lo relativo a la escalabilidad o al redimensionamiento de discos e instancias de cómputo, y lo que se refiere a recursos de red; la Infraestructura Como Servicio (IaaS) las provee automáticamente bajo la modalidad "pago por hora". Ello representa una de las ventajas inherentes de la computación en la nube. Los proveedores del tipo IaaS, brindan infraestructura en distintas escalas ofreciendo servicios en que se pueden seleccionar:

- Tipos y versiones de sistemas operativos, basadas en Windows, y en GNU/Linux
- Arquitecturas de microprocesador de 32 o 64 bits
- Máquinas virtuales en forma de contenedores para incrementar altas demandas por periodos de tiempo sostenidos
- Micro instancias, para proveer respuestas transitorias en cortos periodos a picos de alta demanda
- Optimización de memoria RAM
- Optimización de recursos de red
- Optimización de recursos para almacenamiento
- Opcionalmente se pueden contratar servidores físicos de distintas características para garantizar aislación y privacidad de los datos a una organización que así lo requiera

El lector debe notar que la organización que contrata un servicio IaaS, puede o no delegar la administración de los sistemas operativos, opciones de *networking* y la configuración de sus aplicaciones en la nube. En caso de "no delegación" la organización debe contar con

profesionales idóneos y calificados para la administración de la infraestructura de servicios en la nube. Empresas como Nubity vislumbraron la posibilidad de negocios en la brecha del monitoreo, configuración y mantenimiento de infraestructuras en la nube. Se trata de empresas intermediarias entre organizaciones y proveedores en la nube, con software a medida para abstraer a sus clientes de la complejidad y así garantizar que los servicios en la nube funcionen. Cuando este es el caso, las organizaciones disminuyen costos laborales, producto de la delegación (tercerización) de la administración de sus plataformas en las nubes. Esta temática será abordada en profundidad en el Capítulo VI de este trabajo de tesis.

A continuación se introducirá la topología a implementar en la nube, considerando el sistema operativo y las aplicaciones listadas en el inicio de esta sección.

2.2 Elección de la Topología en la Nube

La Fig. 2.1 ilustra la topología a implementar en la nube. Los proveedores de servicios en la nube deben poder facilitar el proceso de migración en lo que respecta a nombres de dominio. Para ello cuentan con herramientas administrativas amigables al usuario que asocian nombres a direcciones IP de las computadoras (reales o virtuales) en la nube, que son publicadas inmediatamente en los dominios de orden superior o *Top Level Domains* que correspondan. En la Fig 2.2, el componente de bloque R se encarga del proceso de migración descrito (este punto se describe con más profundidad en la sección 3.1 del presente trabajo). Se supone que el nombre de dominio del usuario que busca llevar su infraestructura a la nube es www.vaaestarbien.com.ar.

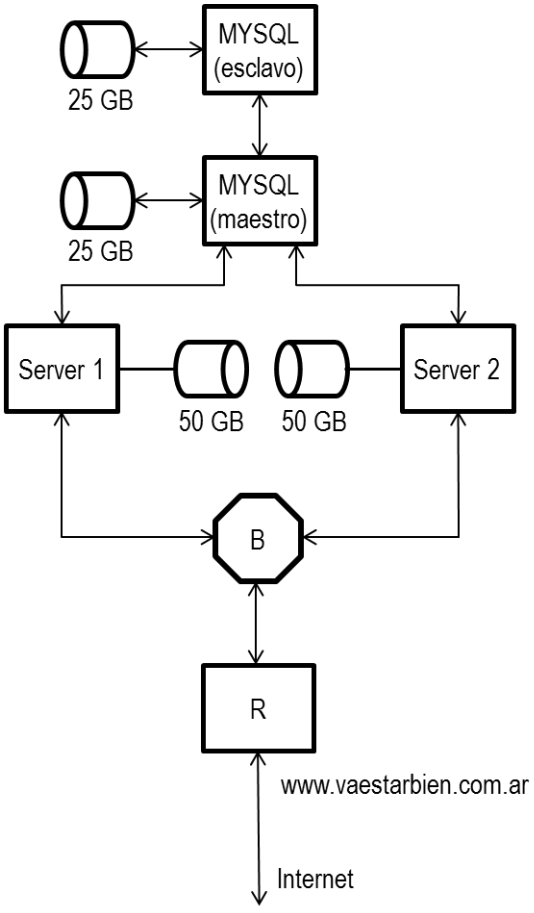

Fig. 2.1 - Topología en la nube para servicios sobre IaaS

Dado a que se necesita redundancia para condiciones de alta disponibilidad en los servicios, es deseable que el proveedor en la nube posea herramientas pre-construidas para una rápida y eficaz implementación. El bloque B en la Fig. 2.1, provee funciones de balanceo de carga entre las instancias de computo virtuales creadas en la nube Server 1 y Server 2. Es el componente en la topología que se encargará de reenviar el tráfico entrante, hacia los servidores (Server 1 y Server 2). Notar que el bloque B de balanceo de carga tiene que poder escalar fácilmente cuando el número de instancias de servidores aumente y ser tolerante a fallas.

Conociendo que una aplicación de red tiene los componentes, proceso cliente, proceso servidor y protocolo de capa de aplicación (modelo de referencia TCP/IP); en la jerga informática se acostumbra a denominar *backend* al lado servidor de la aplicación y *frontend* al lado cliente. La Fig. 2 ilustra dos instancias denominadas Server 1 y Server 2, encargadas de recibir el tráfico ingresante vía el balanceador de carga B, que entre otras funciones servirán a los clientes web que necesiten acceso al sitio implementado en ellos. Es importante configurar en Server 1 y Server 2 suficientes recursos de cache (en sus respectivas memorias) para consultar sobre recursos solicitados (por los clientes), y si existen, suministrarlos de manera inmediata y así aumentar el desempeño de la aplicación. Condiciones de alta disponibilidad se logran incorporando mayor redundancia (traducidas en réplicas de instancias servidoras) y variaciones en la demanda se corresponden con aumento o disminución de los recursos contratados al proveedor en la nube, tal como se lo expreso en la sección 2.1 de este capítulo.

Por último en la Fig. 2.1 se pueden observar dos instancias de aplicación referidas a las base de datos para la operación del servicio web. La replicación es una buena alternativa para tener disponibilidad de información cuando un servidor de base de datos por algún motivo deja de funcionar. La replicación no suplanta los *backups*, sino que, simplemente garantiza la operatividad de MySQL con un esquema de replicación asincrónica de un servidor maestro a uno o más servidores esclavos. El servidor MYSQL “maestro” escribe las transacciones en un archivo de tipo binario, que sirve como registro de actualizaciones para el o los servidores “esclavos”.

Habiéndose descrito los componentes de la topología en la Fig. 2.1, solo resta escoger un proveedor de servicios en la nube (sección 2.3) y proponer la topología en la nube considerando los módulos que el proveedor ofrece para implementar los servicios enunciados en la sección 2.1 de este capítulo.

2.3 Elección del proveedor de la Infraestructura en la nube.

Para cumplir con lo descripto anteriormente en este capítulo, es necesario poder contar con un proveedor de infraestructura en la nube que permita una serie de factores: flexibilidad para una segura escalabilidad, robustez en los componentes claves de una infraestructura, facilidad para la administración y creación de plataformas en la nube en tiempos medidos en unas pocas horas. Con el propósito señalado en este párrafo, se evaluaran simplificadamente tres de los más importantes proveedores de servicios de infraestructura en la nube:

- DigitalOcean [https://digitalocean.com]**:** una característica atractiva de este proveedor es el bajo precio de sus servicios. Posee planes económicos para disponer de servidores virtuales privados *(Virtual Private Servers)* en la nube, con variados recursos disponibles. Agregar uno o más servidores es sencillo y rápido. Ofrece todo lo referido a servidores de nombre de dominio para una rápida y eficaz migración de sitios web.

  Como desventajas se puede citar la escasa flexibilidad al no ofrecer la posibilidad de almacenamiento adicional en los servidores creados. Cuando se requiera escalar en capacidad de almacenamiento, será necesario contratar una infraestructura de otra empresa en la nube para adecuar la infraestructura a las necesidades actuales, o en su

defecto crear un nuevo servidor virtual con el único propósito de utilizar su disco de almacenamiento.

- Softlayer [https://softlayer.com]: es un proveedor muy robusto, que ofrece gran variedad de servicios. Tiene un soporte de configuración eficaz, para soluciones escalables. A través de un panel web que incluye una gran variedad de opciones se construye una topología a medida de las necesidades del cliente, aunque exige un conocimiento acabado para poder realizar la infraestructura en un tiempo prudencial. Un administrador que no esté familiarizado con la interfaz web de este producto consumirá mucho de su tiempo en comprender aspectos que hacen a la construcción de la infraestructura.

  Es un producto claramente orientado a soluciones corporativas de gran magnitud. Pequeñas o medianas organizaciones que requieran de Softlayer como IaaS, tendrán que mitigar con numerosos problemas burocráticos derivados de la evaluación comercial que realiza la empresa (viabilidad). Ofrece servicios en la nube privados o compartidos. En el modo privado el cliente corporativo dispone de servidores físicos exclusivos para su infraestructura, mientras que en el modo compartido, se comparte la infraestructura de servidores con otros clientes de la empresa.

- Amazon Web Services (AWS) [https://aws.amazon.com]**:** el líder mundial en lo relativo a soluciones en la nube. Provee un servicio robusto, muy amigable para el usuario en su administración. El aprovisionamiento de la infraestructura se logra de manera inmediata y sin trámites burocráticos. Ofrece varias opciones en cuanto a la capacidad de los servidores a crear, con una fácil escalabilidad por servidor (recursos de CPU, memoria RAM, almacenamiento en disco, etc.). Se caracteriza por ofrecer numerosos módulos pre-construidos, entre los cuales se destacan el balanceador de carga automático denominado *Elastic Load Balancer (ELB)*, y el módulo *ElastiCache (EC)* para crear capas de caché en forma dinámica.

  Permite la creación de volúmenes de almacenamiento adicionales, que se asocian fácilmente a los servidores creados, permitiendo flexibilidad y seguridad en cuanto a la integridad de los datos. Un aspecto a considerar es su elevado precio, y como consecuencia exige precisión a la hora del diseño de la infraestructura en lo que se refiere a la capacidad de los servicios que se contratan.

En función de todo lo escrito en esta sección, la opción que se va a utilizar en el ámbito de este trabajo de tesis es AWS. Un aspecto a destacar es que AWS cobra el servicio de las instancias en la nube por hora. Ello implica que ante un eventual aumento de tráfico en distintos intervalos de tiempo, la infraestructura crecerá en unidades “hora” y el costo estará asociado a esa unidad. El mismo modo cuando el tráfico disminuya se producirá la situación inversa.

2.4 Propuesta de la Infraestructura

Según lo descrito en este capítulo, en lo que a servicios de aplicación se refiere y la elección del proveedor de IaaS en la nube, la propuesta de la infraestructura es la que se observa en la Fig. 2.2. A continuación se describirán los componentes creados y aquellos provistos por AWS

- Amazon Route 53: es un servicio web DNS escalable y de alta disponibilidad en la nube AWS. Mediante esta aplicación, se puede delegar de forma efectiva los dominios y re direccionarlos a los que ofrece AWS o en su defecto a instancias externas a AWS (nubes hibridas). [4]

- Elastic Load Balancer – ELB: modulo que provee balanceo de tráfico entre los servidores creados (Front 1 y Front 2 en la Fig. 2.2). Es un servicio altamente escalable y de fácil configuración. ELB se encarga de ofrecer la alta disponibilidad requerida de manera transparente al usuario

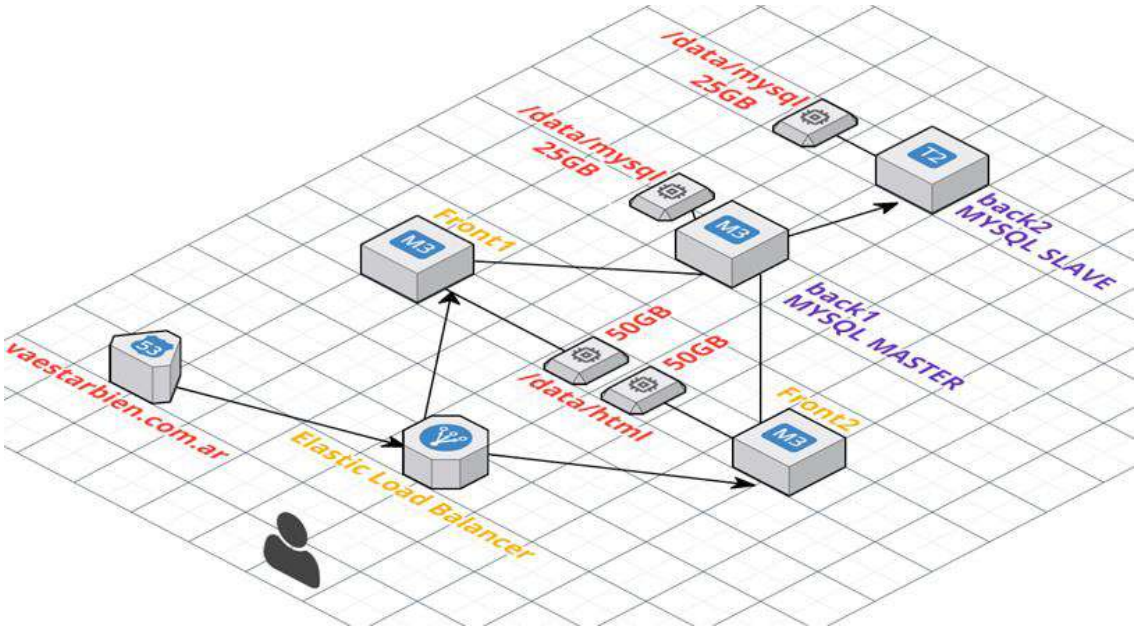

Fig. 2.2: IaaS provista por AWS

- Instancias EC2: son los servidores propiamente dichos. Se crearán y configuraran cuatro (4) instancias EC2. Dos de ellas serán utilizadas como *frondend* de las aplicaciones Varnish, Apache y PHP, y otras dos instancias para la implementación del servidor de base de datos MYSQL en modo maestro - esclavo. Las instancias escogidas son del tipo M3.medium que disponen 1 núcleo virtual de CPU y 3.75 GB de memoria RAM.

- Elastic Block Storage – EBS: son unidades de almacenamiento que se acoplan a instancias EC2 de forma sencilla. Sirven para alojar la información de las aplicaciones a instalar. Tal como lo ilustra la Fig. 2.2 se suministrara un volumen EBS a cada una de las instancias creadas.

A continuación, en el Capítulo 3 se tratará sobre alta y configuración de la IasS de Amazon (AWS)

CAPITULO III

ALTA Y CONFIGURACIÓN DE LA INFRAESTRUCTURA

3.1 Introducción

En este capítulo, se va a realizar el alta de la primera parte de la infraestructura en AWS (front1 y Mysql Master de la figura 2.2). Tanto la creación de las instancias y los servicios a utilizar, como la configuración propiamente dicha de las aplicaciones dentro de los servidores.

3.2 Migración de DNS a Route53

Se van a migrar los DNS a Route53 (R53), aunque todavía los sitios apuntarán a la infraestructura anterior. Al momento de realizar la migración final, solo resta cambiar los registros A de los dominios, y apuntarlos al balanceador de la nueva infraestructura.

Esto va a acelerar los tiempos de migración, dado que el TTL (tiempo de propagación) por defecto de AWS es muy bajo (300 segundos), lo que quiere decir que en 5 minutos se tendrá totalmente propagados los DNS a la nueva infraestructura.

Lo primero que se realizará, es la creación de la zonas DNS. Se selecciona la opción “Create Hosted Zone” dentro del Panel de Administración de Route53, y se introduce el nombre del dominio (ver figura 3.2.1). Como es un dominio de tipo público, se va a crear una zona de tipo pública.

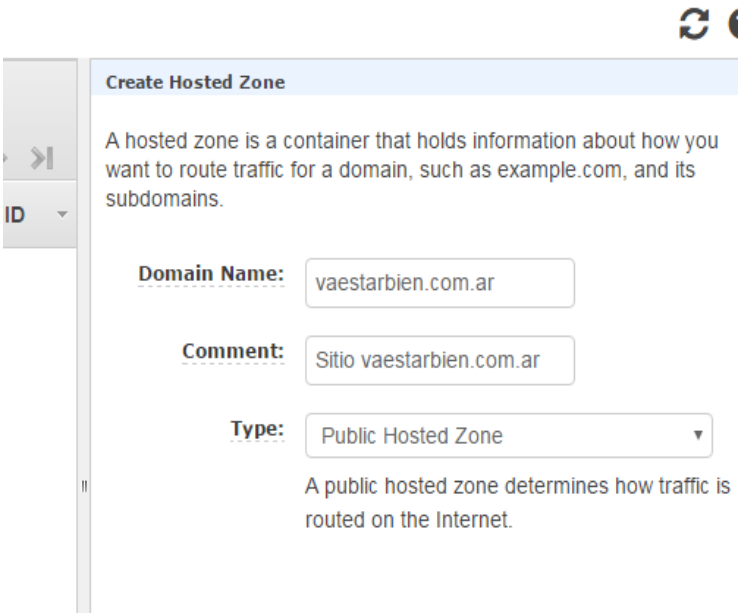

Figura 3.2.1 – Creación de Zona de DNS.

Una vez creada la zona, AWS creá los registros NS a los cuales se debe delegar el dominio. En este caso, primero se copiarán los registros tal cual están en el servidor DNS anterior, y luego se procederá a notificarle al cliente que debe realizar la delegación del dominio en la entidad correspondiente (ver figura 3.2.2). Dichos registros se pueden crear uno a uno, o también importar todos los registros desde un archivo de texto, haciendo más sencilla esta tarea.

Figura 3.2.2 – Registros NS donde delegar el dominio.

Una de las razones por las cuales se eligió el uso de este servicio, es porque como se decía anteriormente, los DNS se pueden asociar fácilmente a las demás soluciones que ofrece Amazon. Un ejemplo de esto es la manera que Route53 tiene de apuntar el registro A principal de cada sitio al balanceador. Todo registro "A" principal en cualquier servidor DNS, tiene necesariamente que apuntar a una IP, no puede ser un alias o "CNAME" de otro dominio. La dirección del Balanceador de Amazon es un dominio específico o "DNS Name", por ejemplo en este caso es "ELB-Productivo-1613830531.us-east-1.elb.amazonaws.com". R53 ofrece la posibilidad de colocar dicho dominio como registro "A" del dominio principal, tal como se puede ver en la figura 3.2.3. AWS es el encargado luego, de traducir dicho alias en IPs para el correcto funcionamiento de los DNS.

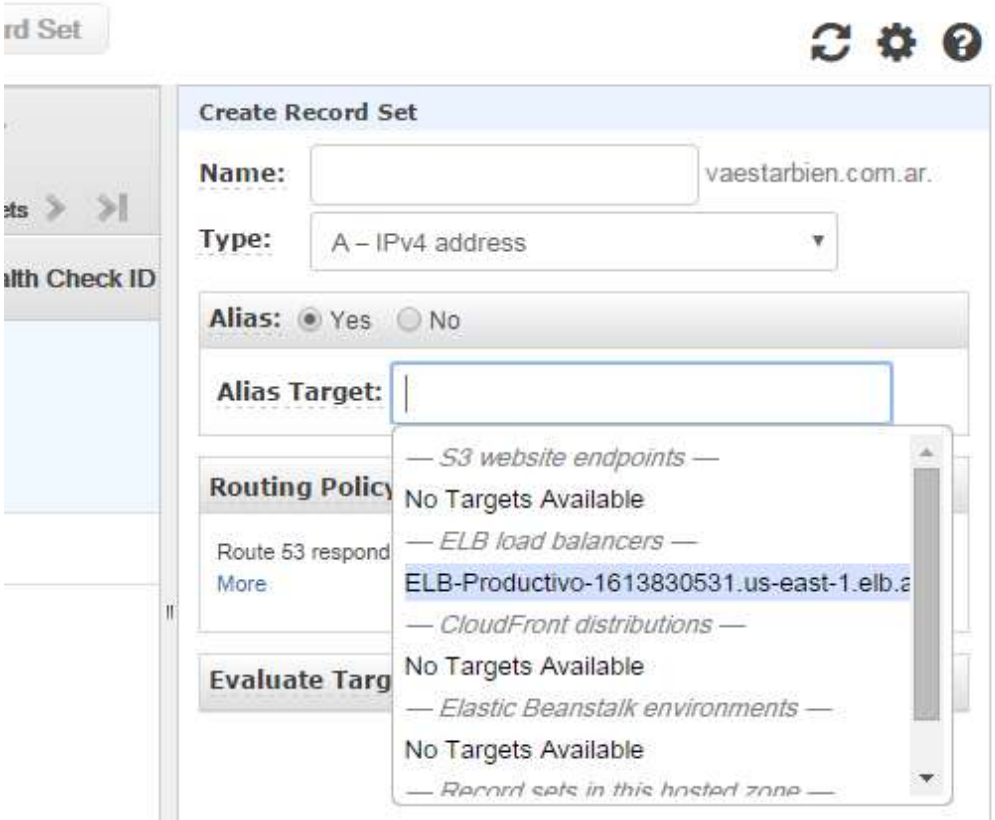

Figura 3.2.3 – Registro A asociado a ELB.

3.3 Alta de Instancias en AWS

La primer tarea que se va a realizar en EC2, es la creación de las 2 instancias principales (una instancia para el servidor web, y otra para el mysql maestro).

Las 4 instancias de la nueva infraestructura van a utilizar la AMI (acrónimo que se les da a las imágenes sobre las cuales se virtualizan los servidores): "CentOS Linux 6 x86_64 HVM EBS 20150928_0-74e73035-3435-48d6-88e0-89cc02ad83ee-ami-2b35794e.2 (ami-57cd8732)", elegida entre todas las que ofrece "AWS Marketplace" (lugar donde se encuentran las AMIs para crear las instancias). El motivo de dicha elección, se debe a que la misma viene con las configuraciones básicas del Sistema Operativo Centos 6, y ninguna otra aplicación agregada, lo que permite instalar y configurar las aplicaciones que se necesiten, y no tener otra consumiendo recursos de la instancia.

Primero se creará una instancia llamada "front1", donde se configurará todo lo referido al servidor web, la capa de cache Varnish, PHP y el servidor FTP. Como segundo paso, se procederá a crear la instancia que va a contener el servidor de base de datos maestro (llamada "back1").

Cuando esto esté configurado, testeado y funcionando, se procederá a crear el segundo Servidor Web, desde una imagen del primero, para que los dos queden exactamente iguales. Luego, creamos el segundo servidor de base de datos, y lo configuramos como esclavo del primero.

La creación de las instancias lo hacemos desde el panel de AWS, en la sección de Elastic Compute Cloud (ver figura 3.3.1).

"Amazon Elastic Compute Cloud" (EC2) es un servicio web que proporciona capacidad de cómputo con tamaño modificable en la nube. Reduce el tiempo necesario para obtener y arrancar nuevas instancias de servidor en cuestión de minutos, lo que permite escalar rápidamente la capacidad, ya sea aumentándola o reduciéndola, según vayan cambiando las necesidades. Dicho servicio se cobra por la cantidad de recursos de la instancia, en una hora. Esto permite ser mucho más precisos en cuanto a la capacidad de la infraestructura, y así dejar mucho menos tiempo de capacidad ociosa en los servidores. [5]

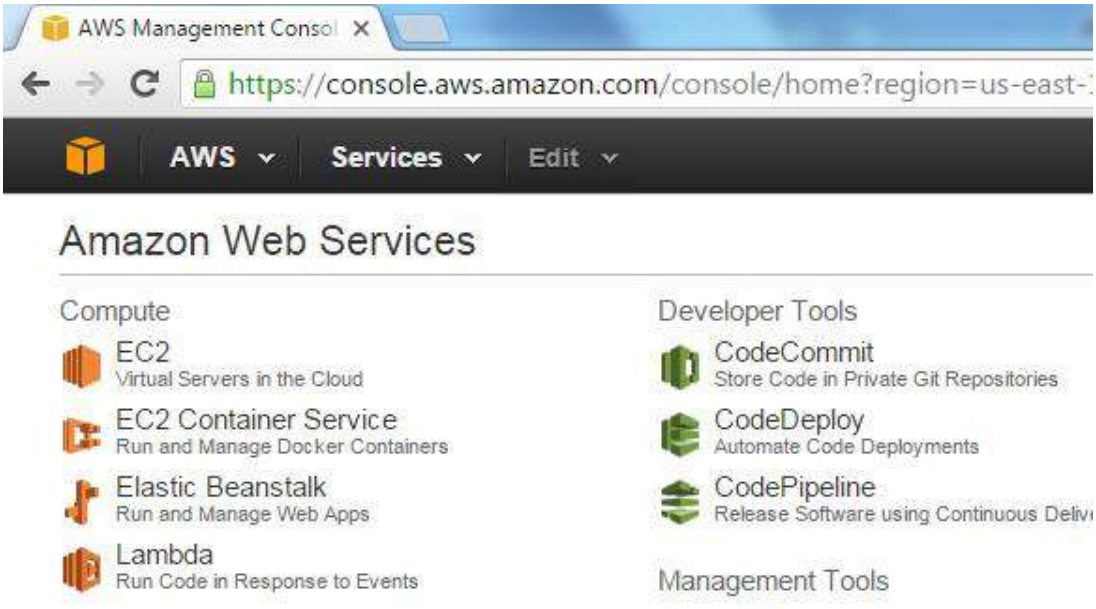

Figura 3.3.1 – Consola de Administración Principal de AWS.

Dentro del panel de EC2, en el apartado “Instances”, se procede a lanzar la nueva instancia, desde la opción “Launch Instance” (ver figura 3.3.2). Eso lleva al primer paso en la creación de una instancia, el cual consiste en la elección de la AMI que vamos a utilizar. Aquí se tiene que tener en cuenta el Sistema Operativo, el tipo de virtualización de la instancia y las características generales que ofrece la misma (ver figura 3.3.3).

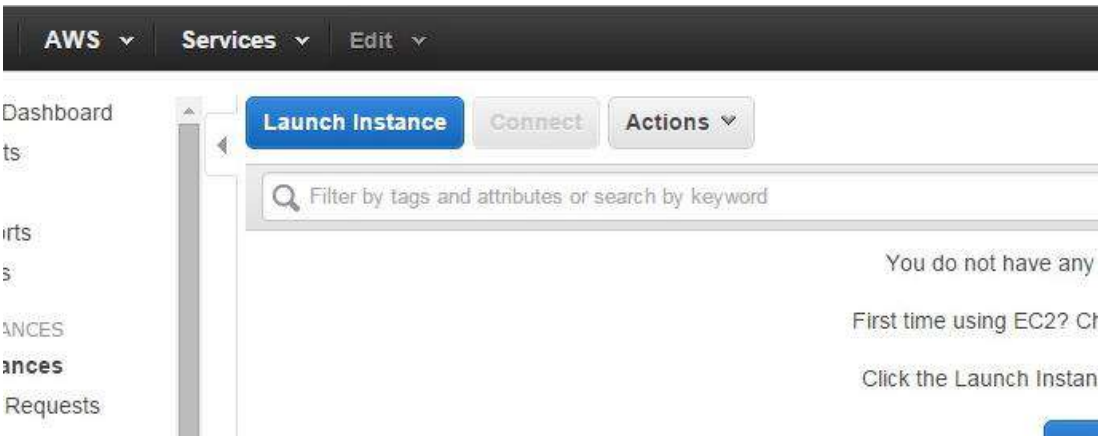

Figura 3.3.2 – Crear Nueva Instancia.

Figura 3.3.3 – Selección de AMI.

Una vez elegida la AMI, se indica qué tipo de instancia vamos a crear. Para este caso, elegimos la opción T2.MICRO. Por cuestiones de costos, se dejará las instancias a su mínima capacidad durante el tiempo que dure el alta inicial. Luego, se procederá a agrandar las instancias al tamaño real (ver figura 3.3.4).

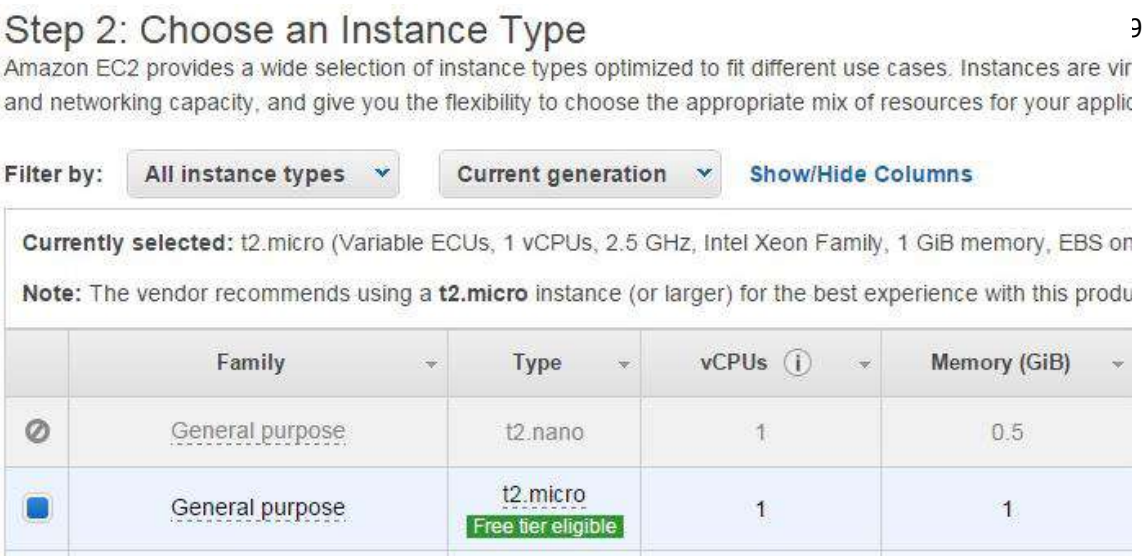

Figura 3.3.4 – Elección del Tipo de Instancia.

La próxima pantalla, pide detalles de la instancia que se está creando. Para este caso, se indica la Zona de Disponibilidad ("Availability zone" o AZ) donde AWS va a crear el servidor (ver figura 3.3.5). Dichas zonas, consisten de infraestructura totalmente separada en lugares físicos diferentes, independiente una de la otra. Para este caso, donde vamos a tener 2 instancias destinadas al servidor web y 2 destinadas a las bases de datos, es importante crear las instancias en diferentes AZ, lo cual permitirá una mayor disponibilidad. Para la primera instancia, se elije la opción "us-east-1a". Al momento de crear las demás, se tendrá en cuenta la selección de otro AZ diferente.

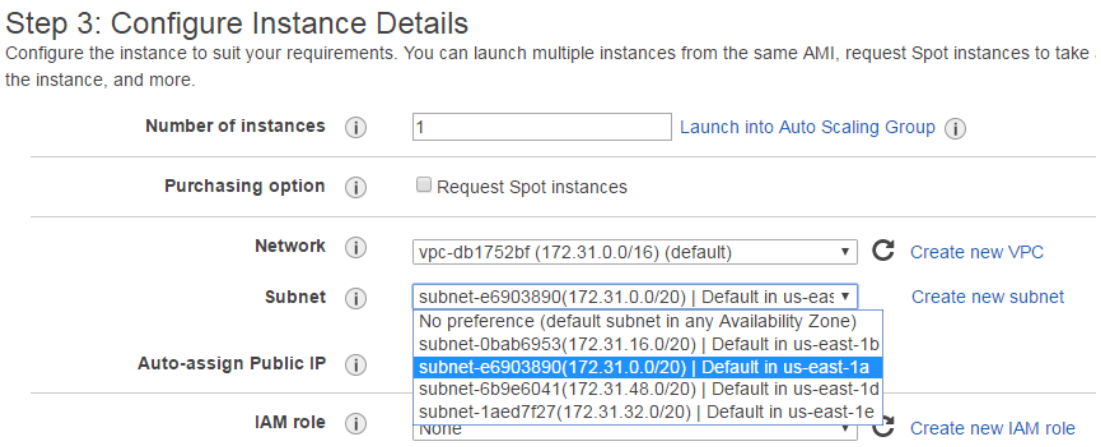

Figura 3.3.5 – Detalles de la instancia.

En el punto 5 de la creación de una instancia, se procede a agregar las etiquetas del servidor. Dichas etiquetas sirven para poder clasificar las diferentes instancias que creamos dentro de EC2. La más importante, es la etiqueta “name”, en la cual se tiene que indicar el nombre que le vamos a dar a la instancia (ver Figura 3.3.6).

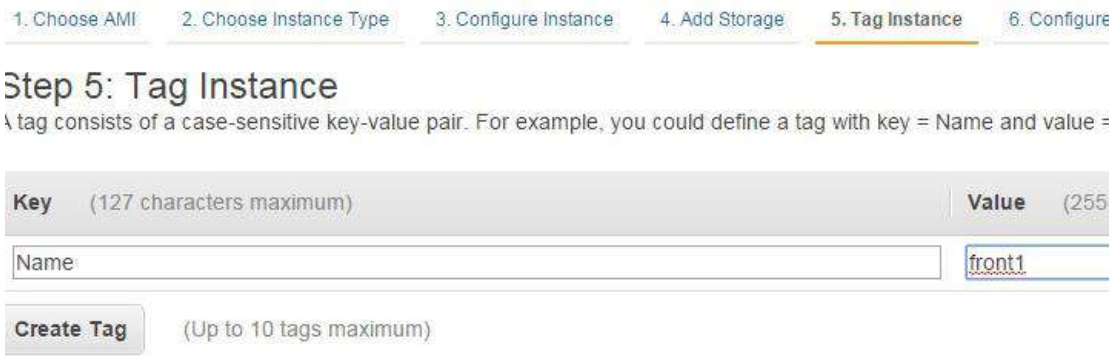

Figura 3.3.6 – Etiquetas de la instancia

El próximo paso, es la selección del “Security Group”. El “Security Group” es un firewall de AWS, el cual se asocia a cada instancia. El mismo puede estar asociado a un grupo de instancias. En esta infraestructura, se utilizarán dos, uno para los servidores web, y otro para los referidos a las bases de datos. El uso del mismo tiene como ventaja, la posibilidad de tener un nivel más de seguridad sobre la infraestructura, además del propio firewall de cada instancia.

El último paso de la creación de la instancia, es un repaso de las configuraciones previas que se realizaron en los pasos anteriores. Se chequea que todo esté bien, y luego se selecciona “Launch”, para crear la instancia bajo la configuración configurada anteriormente.

Al momento de lanzar la instancia, el asistente de instalación pide un par de llaves que AWS va a utilizar para crear el acceso al nuevo servidor. Se puede elegir un par de llaves existentes (que hayamos usado para otra instancia), o crear un nuevo par de llaves, opción que se utiliza para este caso. Una vez de indicar el nombre de la llave, se descarga la llave privada y se guarda en un lugar seguro, dado que con esa llave es la única manera de ingresar a la instancia.

Una vez terminado con la creación de la instancia, se puede ver a la misma en estado de inicialización en el panel (ver figura 3.3.8). Cuando la instancia este en estado “Running” (encendida), ya se puede ingresar a la misma para realizar la configuración.

De esta manera, vamos a realizar la creación de todas las instancias que necesitamos para la infraestructura planteada anteriormente.

Figura 3.3.8 – Inicializando instancia.

3.4 Seguridad de las Instancias

La primera medida en cuanto a la seguridad de las instancias es cambiar el modo en el que SELINUX viene configurado. Se cambiará el modo a “Disabled” (desactivado) y procederemos a reiniciar la instancia para que tome los cambios. Esto se hace para poder tener control sobre los servicios críticos del sistema, como por ejemplo SSH, FTP, firewall, etc., para poder configurarlos sin problemas.

Luego se va a poner el servicio de SSH a escuchar en un puerto diferente al que tiene por defecto, con el objetivo de poder garantizar un nivel más de seguridad en el acceso a la infraestructura. Este puerto estará restringido sólo para IPs específicas desde los lugares donde se necesite acceder.

En los servidores donde se configura el servidor web, sólo se dejará abierto al mundo el puerto 21 (para el acceso FTP del cliente), junto con los puertos utilizados como pasivos por FTP (puertos 49152 a 65534), dado que el cliente accede a este servicio desde un ISP (Proveedor de Servicios de internet) con IP dinámica.

Para el caso del puerto del servidor web (80), se dejará abierto solo para las IPs privadas de la red interna (red donde se encuentran los servidores), dado que los mismos estarán bajo el ELB de Amazon, y los sitios estarán apuntando a dicho Balanceador. En caso de necesitar un acceso público a un servidor específico, se concederá en el momento que lo necesiten, sólo por el tiempo que se requiera acceder a dicha instancia.

Para el caso de los servidores de base de datos, dejaremos el puerto mysql (3306) abierto solo para la red privada, dado que no se necesita acceso desde el exterior. El único acceso público que los mismos tendrán, es el referido al puerto SSH, solo desde las IPs necesarias.

3.5 Configuración del servidor de web

3.5.1 Creación del Volumen extra

Como primera medida, se creará el volumen de 25GB que va a contener los datos de la aplicación (ver figura 3.5.1.1). Para crear el disco hay que tener en cuenta que se debe crear en la misma AZ que el servidor al cual vamos a asociar dicha unidad, dado que después se va a montar el disco en ese servidor.

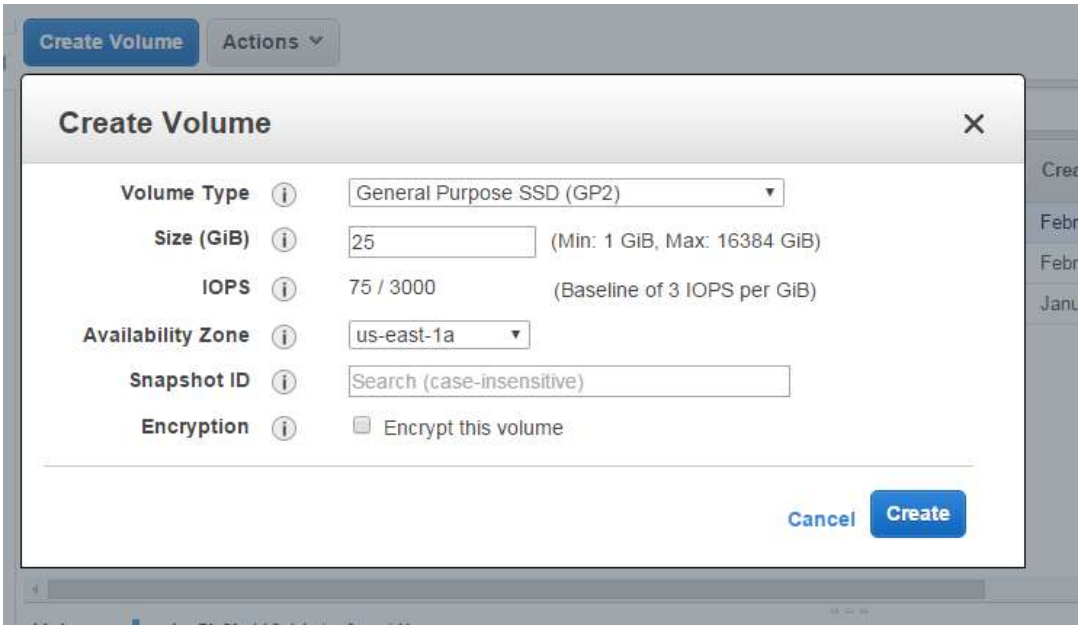

Figura 3.5.1.1 – Creación de Volumen.

AWS posee tres tipos de Volúmenes (o EBS, por su nombre en inglés) que se pueden crear para montar en los servidores:

- **General Purpose SSD (GP2):** Es el tipo de volumen de costo intermedio que se utiliza para propósito general, como datos de la aplicación, base de datos relacionales de poco tráfico, o también volúmenes de arranque de las instancias EC2.
- **Provisiones IOPS (IO1):** Este tipo es el más caro, dado que se puede garantizar una velocidad de Entrada/Salida (E/S) a los discos mucho mayor que el anterior, provisionando una cantidad de IOPS (Input/Output Operations Per Second – Operaciones de Entrada/Salida por segundo) mayor por cada GB del volumen. El IOP es una unidad de medida que se utiliza para garantizar mayor desempeño sobre el disco que necesitamos crear.
- **Magnetic:** Es el volumen más barato, pero de más lentitud. Se puede utilizar para accesos poco frecuentes, como puede ser un volumen para alojar backups o en ambientes no productivos.

Para este caso, se eligió GP2 como tipo de disco, dado que la aplicación no necesita que el mismo contenga posea tanta cantidad de IOPS garantizados.

Luego de crear el volumen. El mismo se habilita rápidamente en el panel para que se pueda asociar a la instancia donde se va a montar el mismo. El estado *"available"* indica que el disco está listo para poder ser asociado a la instancia (ver figura 3.5.1.2). Haciendo click derecho sobre el disco, y apretando la opción "Attach Volume" lo asociamos a la instancia indicada (ver figura 3.5.1.3).

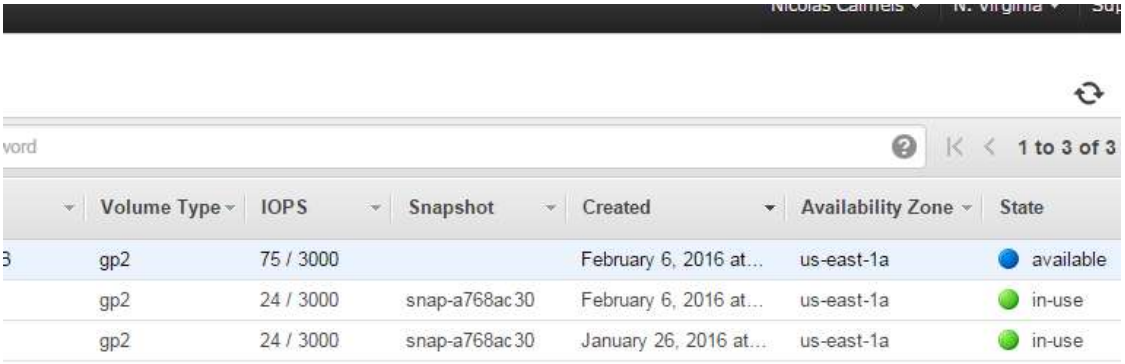

Figura 3.5.1.2 – Estado de los Volúmenes.

Dentro de la instancia, se procede a formatear el disco y montarlo en la carpeta "/datos". Para esto, con el comando *"fdisk –l"*, se va a ver cuál es el disco nuevo que tenemos que formatear (ver figura 3.5.1.4). Luego, se crea un link simbólico, para que la carpeta "/var/www/html" apunte a "/datos/html", y así tener los datos de la aplicación alojados en la nueva partición. De esa manera, ya queda montada la unidad de 25GB en la instancia (ver figura 3.5.1.5).

[root@front1 html]# mkfs.ext4 /dev/xvdf
[root@front1 html]#mkdir /datos
[root@front1 html]#mount /dev/xvdf /datos
[root@front1 html]#ln -s /datos/html/ html

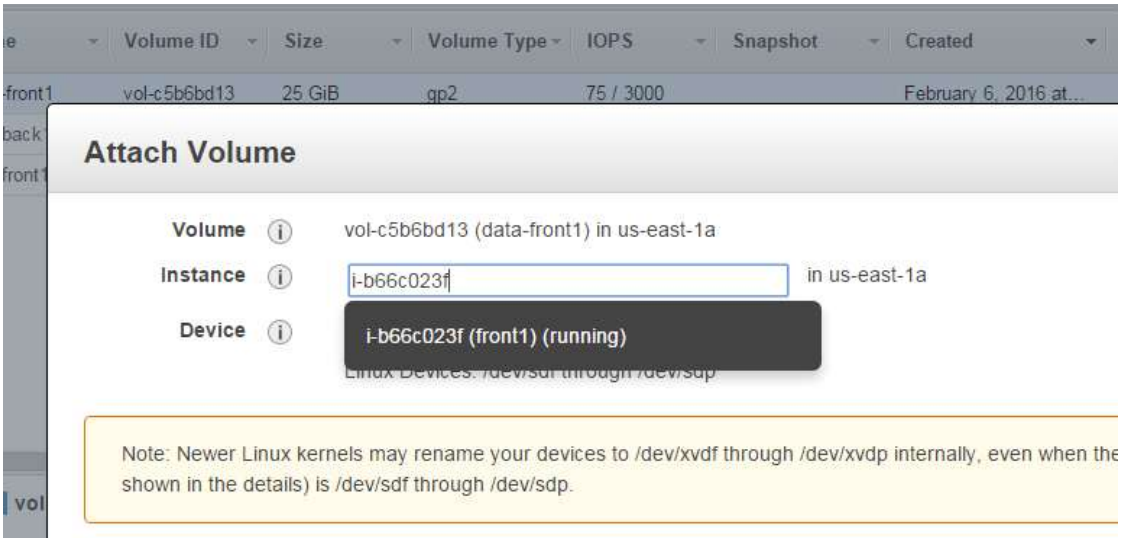

Figura 3.5.1.3 – Montar Volumen a la instancia.

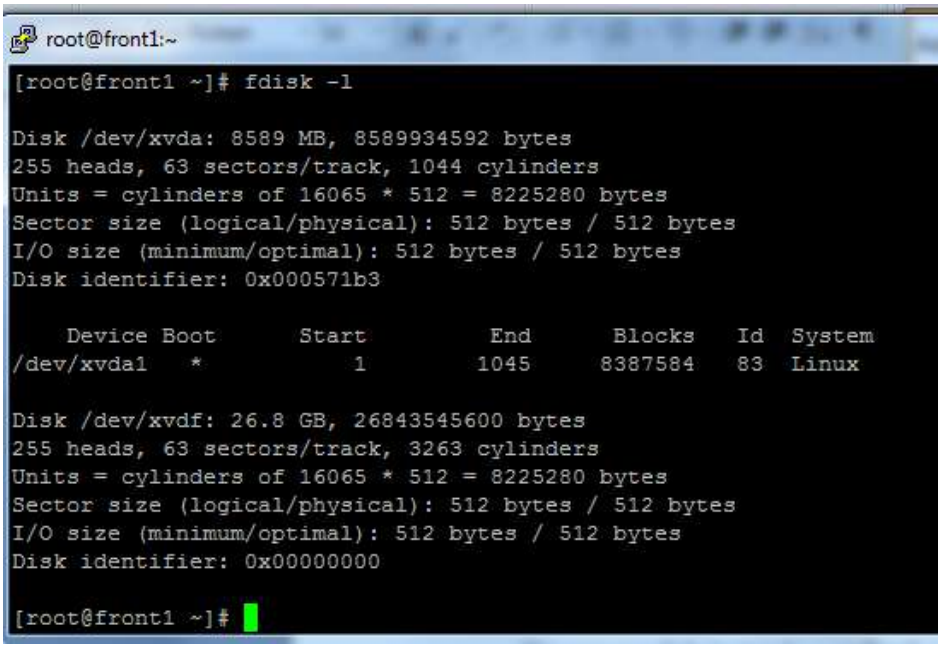

Figura 3.5.1.4 – Identificación de volumen nuevo en el

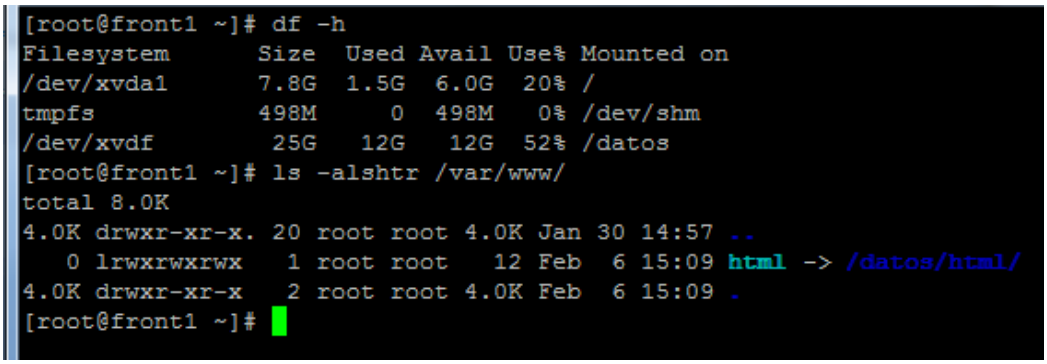

Figura 3.5.1.5 – nuevo disco montado y link simbólico

El último paso, es agregar al archivo */etc/fstab*, la información del disco que se agregó al servidor. Esto garantiza que cuando se reinicie el equipo, el volumen se va a montar de manera correcta, y los servicios van a poder iniciarse sin ningún inconveniente.

Para esto, ejecutamos el comando *blkid* sobre el disco extra que se agregó (ver figura 3.5.1.6). Este comando permite poder obtener el ID del disco en cuestión. Luego, esa información la utilizamos para poder agregar en el archivo mencionado, una línea que indique que dicho volumen sea montado en la carpeta */datos* siempre que se inicie el servidor (ver figura 3.5.1.7).

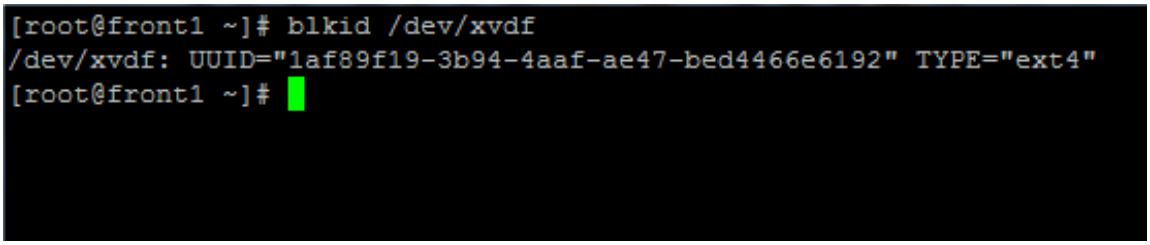

Figura 3.5.1.6 – ID del disco.

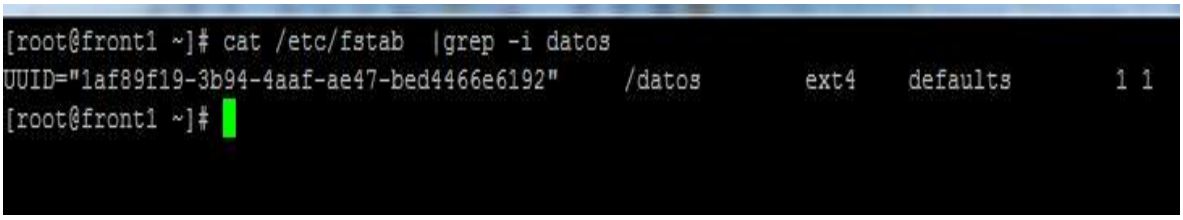

Figura 3.5.1.7 – Linea agregada en /etc/fstab.

Una vez creada la instancia, el volumen EBS extra y realizadas las acciones para garantizar la seguridad de la misma, se realizará la instalación de las diferentes aplicaciones.

3.5.2 Instalación del Servidor FTP

El servidor FTP que utilizaremos, será el mismo que se venía utilizando en la infraestructura anterior, ProFTPD. El mismo no utiliza usuarios del sistema, sino que se crean usuarios virtuales, sobre los cuales se setean los ACL (Lista de Control de Accesos) correspondientes para que puedan acceder a los archivos propios de ese usuario. De esta manera, garantizamos más seguridad en el uso de esta aplicación.

El mismo se instala desde el repositorio propio del Sistema Operativo, y se utilizarán los mismos usuarios y configuración que se utiliza en el servidor productivo actual.

Para instalarlo por repositorio, se ejecuta el comando *"yum install proftpd"*; aceptamos la instalación, y el Sistema Operativo se encarga de conectarse al repositorio y bajar los paquetes necesarios (junto con la resolución de dependencias que se necesiten para que este programa funcione correctamente).

La configuración de proftpd se encuentra en la ruta */etc/proftdp.conf*. Dicha configuración se va a copiar exactamente como está en el servidor anterior, para poder conservar los mismos ACLs que se habían configurado. Para esto, primero hacemos un backup de la configuración original del ProFTPD en el servidor, tal como se ve en la figura 3.5.2.1.

```
[root@front1 etc]# cp proftpd.conf proftpd.conf-original
```

Figura 3.5.2.1 – Copia de la configuración original

```
Port                            21
PassivePorts                    49152 65534

Umask                           002
MasqueradeAddress 54.85.187.146

# Default to show dot files in directory listings
ListOptions                     "-a"

# See Configuration.html for these (here are the default values)
RootLogin                       off
```

Figura 3.5.2.2 – Directiva MasqueradeAddress

Cuando se tiene dicho backup, se reemplaza el archivo de configuración con el contenido de ese archivo en el servidor anterior. Lo único que cambia es la directiva **"MasqueradeAddress"**, dónde se va a cambiar la IP del servidor anterior, por la del front actual (ver figura 3.5.2.2). Dicha directiva es la encargada de indicar cuál es la IP pública por la cual se van a conectar al servidor FTP.

Los ACL de la configuración original se preservan tal cual estaban, dado que también se van a preservar las rutas a donde están apuntados los sitios actualmente. En la figura 3.5.2.3 vemos un ejemplo de un ACL para un usuario FTP en particular, el cual puede acceder a dos sitios diferentes.

```
<Directory /var/www/html/vaestarbien.com.ar/>
        <Limit ALL>
                Order allow,deny
                AllowUser vaestarbien_ftp
                DenyAll
        </Limit>
</Directory>

<Directory /var/www/html/ajeciudad.org/>
        <Limit ALL>
                Order allow,deny
                AllowUser vaestarbien_ftp
                DenyAll
        </Limit>
</Directory>
```

Figura 3.5.2.3 – Ejemplo de ACL

Con respecto a los usuarios FTP, se hará lo mismo que con la configuración del servidor. En este caso, la configuración de los usuarios virtuales creados se alojan en la ruta */etc/ftp.passwd*. Se copiará dicho archivo desde el servidor anterior. Una vez finalizada con la configuración, solo queda abrir los puertos necesarios en el "Security Group" de la instancia en AWS para el puerto 21 y el rango de puertos pasivos. También los puertos deben abrirse en el firewall del servidor (iptables).

3.5.3 Instalación del Servidor Web

Se va a utilizar Apache en su versión 2.4.18. Dicha versión es el último lanzamiento estable que la "Apache Software Fundation" había realizado durante la confección del presente trabajo. Dicha versión no se encuentra disponible por repositorio para el Sistema Operativo que se eligió, por lo cual se procederá a compilar la misma.

Para esto, desde el sitio oficial de apache se busca algunos de los sitios de descarga oficiales donde estén disponibles los paquetes para la instalación de esta versión. En este caso, también descargaremos los archivos necesarios para instalar APR junto con apache.

```
[root@front1 src]# cd /usr/local/src
[root@front1 src]# wget http://mirrors.nxnethosting.com/apache//httpd/httpd-2.4.18.tar.gz
[root@front1 src]# wget http://mirrors.nxnethosting.com/apache//apr/apr-1.5.2.tar.gz
[root@front1 src]# wget http://apache.dattatec.com//apr/apr-util-1.5.4.tar.gz
```

Apache Portable Runtime (APR) es una biblioteca de soporte para el servidor web, que provee un conjunto de APIs que permiten recurrir al sistema operativo (SO) subyacente. En los casos en que el SO soporta una función particular, APR provee una emulación de la misma. Por tanto, los programadores pueden emplear APR para lograr que un programa sea verdaderamente portátil a través de diversas plataformas.

Entre las funciones independientes de plataformas ofrecidas por APR se incluyen: funcionalidad de gestión de memoria y de pool de memoria, Instrucciones atómicas, Gestión dinámica de bibliotecas, Entrada/salida de archivos, Análisis de argumentos de la la línea de comandos, Locking, Tablas hash y arreglos, etc. [6]

Como la compilación va a ser de forma manual, el repositorio no va a resolver las dependencias de paquetes necesarios que se tengan que instalar para poder realizar la correcta instalación del mismo, por lo que de antemano tenemos que prever esto. En este caso, se instalarán un conjunto de paquetes necesarios para la compilación:

```
[root@front1 src]# yum install make gcc openssl-devel pcre-devel
```

Dichos paquetes instalados, son librerías necesarias para que apache se ejecute correctamente. También se instalan los comandos necesarios para la compilación, asi como también el compilador propiamente dicho.

Luego, se descomprimirán los paquetes descargados y se moverán los archivos de APR dentro de la carpeta de compilación de apache:

```
[root@front1 src]# tar xzvf apr-1.5.2.tar.gz
[root@front1 src]# tar xzvf apr-util-1.5.4.tar.gz
[root@front1 src]# mv apr-1.5.2 httpd-2.4.18/srclib/apr
[root@front1 src]# mv apr-util-1.5.4 httpd-2.4.18/srclib/apr-util
```

Una vez realizado esto, se procede a la compilación del servidor web:

```
[root@front1 src]#cd httpd-2.4.18/

[root@front1 httpd-2.4.18]#./configure --prefix=/usr/local/httpd-2.4.18 --enable-cgi --enable-ssl --enable-headers --enable-module=expires --enable-module=headers --enable-module=so --enable-rewrite --enable-vhost_alias --enable-deflate --with-pcre --with-mpm=prefork --with-included-apr --with-included-apr-util

[root@front1 httpd-2.4.18]#make
[root@front1 httpd-2.4.18]#make install
```

Al terminar la compilación del servidor, se procede a agregar algunas rutas que ayuden a mantener los estándares que el Sistema Operativo (CentOS) utiliza con respecto a la configuración de apache, como ser la ruta donde se encuentran los logs y la ruta de los archivos de configuración:

```
[root@front1 httpd-2.4.18]#cd /usr/local/
[root@front1 local]#ln -s httpd-2.4.18/ httpd
[root@front1 local]#echo 'pathmunge /usr/local/httpd/bin' > /etc/profile.d/httpd.sh
[root@front1 local]#chmod 755 /etc/profile.d/httpd.sh
[root@front1 local]#. /etc/profile
[root@front1 local]#ln -s /usr/local/httpd/conf/ /etc/httpd
[root@front1 local]#mkdir /var/log/httpd/
[root@front1 local]#ln -s /usr/local/httpd/logs/error_log /var/log/httpd/error_log
[root@front1 local]#ln -s /usr/local/httpd/logs/access_log /var/log/httpd/access_log
[root@front1 local]#mkdir /etc/httpd/vhosts
```

La carpeta */etc/httpd/vhosts* (que es creada en el último comando), es donde se alojarán los archivos ".conf" de cada uno de los sitios que tenemos que servir en los Servidores web. Este tipo de configuración ayuda a ordenar la información referida a los sitios que tenemos alojados, dado que de esa manera no se tiene toda la información referida al servidor web, en un solo archivo de configuración (httpd.conf) de mucha cantidad de líneas.

Luego se editan los archivos de configuración necesarios (haciendo un backup previo del original por cualquier problema). Dicha configuración se realiza bajo los estándares Nubity tiene determinado para este tipo de aplicaciones:

```
[root@front1 local]#cd /etc/httpd/
[root@front1 httpd]#mv httpd.conf httpd.conf-original
[root@front1 httpd]#vim httpd.conf
[root@front1 httpd]#mv /etc/httpd/extra/httpd-vhosts.conf /etc/httpd/extra/httpd-vhosts.conf-original
[root@front1 httpd]#vim /etc/httpd/extra/httpd-vhosts.conf
```

También se tiene que crear el script de inicio para poder iniciar, parar o reiniciar el servicio (entre otras cosas). Este es un paso necesario en el tipo de instalación que realizamos, dado que si se realiza la instalación por repositorio, dicho script de inicio ya queda instalado:

```
[root@front1 httpd]#vim /etc/init.d/httpd
[root@front1 httpd]#chown root.root /etc/init.d/httpd
[root@front1 httpd]#chmod 755 /etc/init.d/httpd
```

Lo mismo pasa con el usuario y grupo apache. Para este tipo de instalación, se debe realizar la creación de forma manual:

```
[root@front1 httpd]#groupadd --gid 48 apache
[root@front1 httpd]#useradd --home /var/www --shell /sbin/nologin --no-create-home --uid 48 --gid 48 apache
```

Para la comprobación de la correcta instalación de dicho servicio, se procede a la creación de un "Virtual Host" de prueba, para comprobar que el servicio esté funcionando correctamente:

[root@front1 httpd]#mkdir -p /var/www/html/default
[root@front1 httpd]#chown apache:apache /var/www/html/default
[root@front1 httpd]#nano -wc /var/www/html/default/index.html
[root@front1 httpd]#nano -wc /etc/httpd/vhosts/0_default.conf
[root@front1 httpd]#/etc/init.d/httpd start

El "Virtual Host" es la configuración de un dominio que queremos servir en dicho servidor web. Cuando el servidor va a tener varios dominios, se necesita de dichos "Virtual Host" para indicar la ruta de acceso de cada sitio, redirecciones específicas que pueda tener el sitio, o nombre de cada log en particular, entre otras configuraciones. El Virtual Host que se creó tiene como dominio "example.com" (ver figura 3.5.3.1).

```
[root@front1 vhosts]# cat /etc/httpd/vhosts/0_default.conf
<VirtualHost *:80>
    ServerAdmin webmaster@sitio.com
    DocumentRoot /var/www/html/default/
    ServerName www.example.com
    ServerAlias example.com

        <Directory /var/www/html/default/>
                Options +MultiViews +Includes +FollowSymLinks -Indexes
                AllowOverride all
                allow from all
        </Directory>

    ErrorLog /var/log/httpd/www.example.com-error_log
    CustomLog /var/log/httpd/www.example.com-access_log common
</VirtualHost>
[root@front1 vhosts]#
```

Figura 3.5.3.1 – Virtual Host creado para pruebas.

Para la comprobación, debemos apuntar dicho dominio en nuestro archivo "hosts" de la computadora, para forzar a que "example.com" resuelva a la ip del servidor. Dicho archivo se encuentra en */etc/hosts* en los Sistemas Linux, en */etc/prívate/hosts* en sistemas MAC OS, y *C:\Windows\System32\drivers\etc\hosts* en los Sistemas Windows. Al final del archivo, se agrega la línea:

54.85.187.146 example.com www.example.com

De esta manera se comprueba la correcta instalación del servidor web, tal como muestra la figura 3.5.4.2.

Figura 3.5.3.2 – Comprobación de funcionamiento de

3.5.4 Instalación de PHP

Se va a instalar PHP en su versión 5.3.29, la última versión estable de la rama 5.3. Es una versión antigua, pero por el momento se va a dejar la misma que había en el servidor a migrar, dado que cambiar la versión puede traer problemas de compatibilidad con los sitios que están alojados en la infraestructura.

Al igual que el servidor web, PHP también va a ser compilado, ya que de esa manera podemos tener más control sobre los módulos de PHP que instalamos, junto con la configuración del mismo. Por lo tanto, se va a descargar el paquete de la mencionada versión desde el sitio oficial de php, se lo va a descomprimir, y luego se instalará el mismo:

[root@front1 ~]#cd /usr/local/src/
[root@front1 src]#wget http://php.net/distributions/php-5.3.29.tar.gz
[root@front1 src]#tar xzvf php-5.3.29.tar.gz
[root@front1 src]#cd php-5.3.29

Se necesita instalar librerías extras, antes de la compilación de PHP. Las mismas son necesarias para el correcto funcionamiento de dicha aplicación, por lo cual se procederá a instalarlas, y luego se compilará e instalará PHP:

[root@front1 src]#yum install libxml2-devel bzip2-devel libcurl-devel libjpeg-turbo-devel libpng-devel freetype-devel t1lib-devel readline-devel libxslt-devel mysql-devel

[root@front1 php-5.3.29]# ./configure --enable-bcmath --enable-calendar --enable-cli --enable-soap --enable-wddx --exec-prefix=/usr/local/php5.3.29 --with-mhash --with-xsl --with-mysqli --enable-pdo --enable-mbstring --with-openssl --with-mysql --with-mysql-sock --with-gd --with-jpeg-dir=/usr --enable-gd-native-ttf --with-pdo-mysql --with-libxml-dir=/usr --with-curl --enable-zip --enable-sockets --with-zlib --enable-exif --enable-ftp --with-iconv --with-gettext --with-t1lib=/usr --with-freetype-dir=/usr --prefix=/usr/local/php5.3.29 --with-fpm-user=apache --with-apxs2=/usr/local/httpd/bin/apxs --with-bz2 --enable-pcntl --with-readline --enable-shmop --enable-sysvsem --enable-sysvshm --enable-sysvmsg --enable-opcache --sysconfdir=/etc/php5 --with-config-file-path=/etc/php5 --with-config-file-scan-dir=/etc/php5/conf.d --libdir=/usr/lib64 --with-libdir=lib64

[root@front1 php-5.3.29]# make

[root@front1 php-5.3.29]# make install

Una vez instalado PHP, se procede a realizar las configuraciones pertinentes para poder dejarlo configurado bajo los estándares que el sistema operativo tiene sobre la ubicación de los archivos de configuración. Por último, se reinicia apache para que tome las nuevas configuraciones:

[root@front1 php-5.3.29]# cd /usr/local/

[root@front1 local]# ln -s php5.6.17/ php5
[root@front1 local]#ln -s /usr/local/php5/bin/php /usr/bin/php
[root@front1 local]#mkdir /etc/php5/conf.d
[root@front1 local]#vim /etc/httpd/extra/php.conf
[root@front1 local]#vim /etc/php5/php.ini
[root@front1 local]#/etc/init.d/httpd restart

Dentro de la configuración de apache que se realizó en el punto 3.5.3 del presente trabajo, está incorporado el módulo de PHP que apache necesita agregar para que dichas aplicaciones funcionen en forma conjunta.

Para poder probar el correcto funcionamiento, junto con la comprobación de que todos los módulos necesarios fueron instalados, podemos utilizar el vhost configurado para la prueba de apache. Lo que se cambiará es el archivo index, modificando su extensión a "index.php" y escribiendo dentro del mismo la función "phpinfo();" de php (ver figura 3.5.4.1).

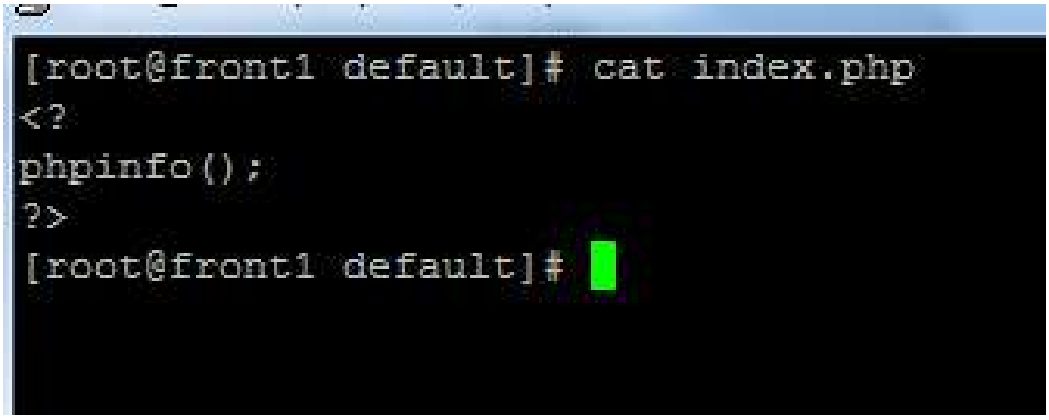

Figura 3.5.4.1 – archivo index.php

De esta manera, visitando el sitio "example.com" desde el browser, se puede visualizar si apache está interpretando de manera correcta PHP, y también la configuración que se acaba de realizar sobre esta aplicación (ver figura 3.5.4.2).

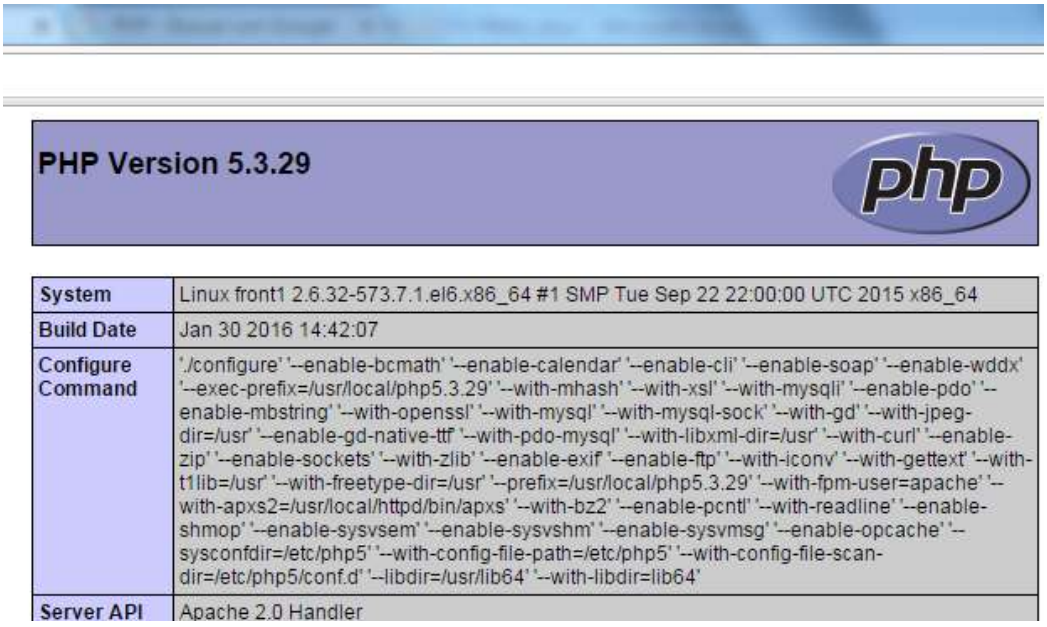

Figura 3.5.4.2 – archivo index.php.

3.5.5 Instalación y Configuración de Varnish

Varnish Cache es un acelerador de aplicaciones web, también conocido como caché de proxy HTTP inversa. Se instala delante de cualquier servidor HTTP y se configura para almacenar en el caché del servidor una copia del recurso solicitado. Está ideado para aumentar el rendimiento de aplicaciones web con contenidos pesados y APIs altamente consumidas.

Varnish se configura en el puerto HTTP (puerto 80) y se encarga de recibir todas las peticiones que llegan al servidor web. Luego decide, en función a un conjunto de reglas que tiene definidas, si el contenido que tiene que servir lo hace desde el caché, o si la petición se la pasa al servidor Web propiamente dicho para que el mismo sirva dicho contenido. [7]

De esta manera, ayuda al servidor web a atender las peticiones del sitio, y también ofrece una mayor rapidez del servicio, dado que su contenido es servido directamente de la memoria RAM del servidor, haciendo mucho más ágil al sitio web.

Esta aplicación, se instalará desde su repositorio oficial. Para esto, se agrega el repositorio de varnish en el servidor, y luego se procede a la instalación utilizando *"yum"* tal como se hizo con el servidor FTP:

```
[root@front1 ~]# cd /usr/local/src/
[root@front1 src]# wget https://repo.varnish-cache.org/redhat/varnish-4.0.el6.rpm
[root@front1 src]#rpm --nosignature -i https://repo.varnish-cache.org/redhat/varnish-4.0.el6.rpm
[root@front1 src]#yum install epel-release
[root@front1 src]#yum install varnish
```

Luego de la instalación se procede a la configuración de dicha aplicación. Se configurará el puerto 80, como puerto por defecto por el cuál varnish va a escuchar. Dicha configuración, se realiza en el archivo */etc/sysconfig/varnish*, configurando la línea:

```
VARNISH_LISTEN_PORT=80
```

Cabe aclarar, que luego se deberá configurar apache para que escuche en el puerto que se definirá como "backend", mas adelante en esta sección.

Otra de las directivas importantes a configurar dentro del archivo */etc/sysconfig/varnish* es la que indica la cantidad de memoria RAM que va a ser reservada para Varnish. Como el servidor cuenta con 3,75 GB de memoria RAM, se le va a indicar a varnish que utilice 1GB de dicha memoria para poder alojar el contenido estático del sitio. Por defecto este valor es de *128M*, pero teniendo en cuenta la capacidad de los servidores que se van a utilizar, es posible aumentar este valor 8 veces más, dado que se tienen 2,75GB más de RAM para las demás aplicaciones que se ejecutan en el servidor:

```
VARNISH_STORAGE_SIZE=1024M
```

Luego se procederá a la configuración del archivo */etc/varnish/default.vcl*. Dicho archivo, contiene todas las reglas de cacheo que se definen para la aplicación, las cuales se configuran bajo el lenguaje VCL (*Varnish Configuration Language*).

VCL es un lenguaje propio, que permite determinar las políticas a tomar sobre las peticiones de entrada. En esta política se puede decidir qué contenido desea servir, desde donde se desea obtener el contenido y la forma en que la solicitud o la respuesta debe ser alterada.

El primer parámetro a configurar dentro de dicho archivo, es el *backend* que varnish va a utilizar para servir el contenido que no tiene cacheado en memoria. Para esto se indica el puerto

y el host, que en este caso es el propio servidor en el puerto 8080 (en el cual va a estar escuchando apache).

```
# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8080";
}
```

De esta manera se define que el *backend* por defecto que utiliza varnish va a ser apache. Esto quiere decir que para las peticiones en las que varnish no tenga que servir el contenido, apache es el encargado de hacerlo.

Para este caso, vamos a utilizar una configuración básica, donde se indicará a varnish que cachee todo el contenido estático del sitio (archivos .gif, .jpg, .swf, .ttf, etc). Para esto, se agrega una regla, la cual indica que si una petición contiene uno de los tipos de archivos especificados, el mismo tiene que ser servido por varnish. Para los otros archivos, la petición pasa para el servidor web.

```
# Clean GET vars from the URL of static content and force get from cache
    if (req.url ~ "\.(gif|jpg|jpeg|swf|ttf|css|js|flv|mp3|mp4|pdf|ico|png)(\?.*|)$") {
            set req.url = regsub(req.url, "\?.*$", "");
            return (hash);
    }
```

También definiremos el Tiempo de Vida (TTL) que dichos contenidos tienen que ser cacheados. Cuanto el TTL caduca, varnish consulta a apache nuevamente para cachear ese contenido. En este caso, se configurará en 72h:

```
if (bereq.url ~ "\.(gif|jpg|jpeg|swf|ttf|flv|mp3|mp4|pdf|ico|png)(\?.*|)$") {
            set beresp.ttl = 72h;
    }
```

Cuando se termina de optimizar la configuración, se reinician varnish y apache y se comprueba el correcto funcionamiento de ambos. Para esto, se utiliza el comando “cURL”, el cual es es una herramienta software para transferencia de archivos con sintaxis URL mediante intérprete de comandos. El objetivo del uso de dicha herramienta, es poder simular las acciones de un usuario en un navegador web, y así poder saber si el contenido del sitio lo está sirviendo varnish o apache.

3.6 Configuración del servidor de Bases de Datos.

Dicho servidor va a contar sólo con Percona Mysql, ya que está dedicado exclusivamente a la base de datos. Percona Server es una variante del servidor tradicional de base de datos relacionales, MySQL. Es una compañía de desarrollo de software Open Source, especializada en el soporte, consultoría, administración y enseñanza de MySQL. [7]

3.6.1 Seguridad de la Instancia.

Como primera medida, se realizará la configuración del Security Group creado para la instancia donde, tal como se dijo anteriormente, sólo quedarán habilitados puertos de SSH y MySQL.

La conexión a la base de datos solo se habilita desde la red privada de la infraestructura, y la conexión SSH solo para las IPs de las oficinas de Nubity, más las que el cliente solicite en caso de que necesite acceder vía SSH al servidor.

También se creará una unidad EBS, de tipo GP2 de 25GB, tal como se hizo para la instancia dedicada al servidor web. La diferencia es que en este caso, el link simbólico va a ser en la carpeta "/var/lib/mysql", la cual va a apuntar a "/datos/mysql/".

3.6.2 Instalación de Percona Mysql

La instalación del servidor de base de datos, se realizará con la descarga de los paquetes RPM (manejador de Paquetes de Red Hat), descargando el archivo ".tar" desde el sitio oficial de Percona, el cual contiene dentro todos los paquetes necesarios para realizar la instalación:

[root@back1 ~]#cd /usr/local/src

[root@back1 src]#wget https://www.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.28-76.1/binary/redhat/6/x86_64/Percona-Server-5.6.28-76.1-r5759e76-el6-x86_64-bundle.tar

[root@back1 src]#tar xvf Percona-Server-5.6.28-76.1-r5759e76-el6-x86_64-bundle.tar

Luego de desagrupar los archivos, se procede a instalar los paquetes previos necesarios, utilizando el repositorio de Centos. Una vez instalados dichos paquetes, se realiza la instalación de mysql. Aquí es importante el orden de los archivos ".rpm", dado que sino arrojarán errores de dependencia al momento de instalarlos:

[root@back1 src]#yum install libaio perl-Data-Dumper numactl
[root@back1 src]#rpm -ivh Percona-Server-shared-56-5.6.28-rel76.1.el6.x86_64.rpm
[root@back1 src]#rpm -ivh Percona-Server-client-56-5.6.28-rel76.1.el6.x86_64.rpm
[root@back1 src]#rpm -ivh Percona-Server-devel-56-5.6.28-rel76.1.el6.x86_64.rpm
[root@back1 src]#rpm -ivh Percona-Server-server-56-5.6.28-rel76.1.el6.x86_64.rpm

Una vez terminado de instalar el servidor de base de datos, ejecutamos un comando propio de Percona MySQL, que me permite realizar las configuraciones iniciales (password de usuario root, borrado de base de datos "test", configuración del usuario root para que solo pueda acceder de manera local):

[root@back1 src]#/etc/init.d/mysqld start
[root@back1 src]#/usr/bin/mysql_secure_installation

De esta manera, queda creada la instancia de base de datos con su configuración básica. Dicho servidor va a ser utilizado como maestro, configuración que se va a realizar en el capítulo 4 del presente trabajo.

3.7 Creación y configuración del ELB

Elastic Load Balancing distribuye automáticamente el tráfico entrante de las aplicaciones entre varias instancias de Amazon EC2. Permite conseguir niveles más altos de tolerancia a errores en las aplicaciones, ya que ofrece de manera integral la capacidad de equilibrio de carga necesaria para distribuir el tráfico.

ELB garantiza que solo las instancias de Amazon EC2 en buen estado reciban tráfico mediante la detección de instancias en mal estado. Para esto, ELB tiene un "Health Check" que sirve para poder saber si una instancia se encuentra en condiciones de poder recibir tráfico. Si el ELB ve la instancia como "saludable", el mismo rutea tráfico hacia ese servidor. [8]

Para la creación de dicho balanceador, se ingresa al panel de administración de EC2, y en el apartado *"Load Balancers"* se realiza el alta y configuración del mismo. Al momento de seleccionar la opción "crear load Balancer", se despliega un asistente de instalación muy similar al de la creación de una instancia EC2.

En primer lugar, se define la configuración básica del ELB: nombre y puertos a balancear. Como paso 2, se define el Security Group asociado a dicho balancer. En este caso, se dejará abierto el puerto 80 a toda conexión entrante solamente, dado que es el único puerto a balancear.

El paso 3 es para realizar la configuración necesaria de seguridad, en caso de que se necesite balancear el trafico bajo HTTPS (puerto 443). ELB ofrece la posibilidad de instalar y configurar el certificado dentro del balanceador ("SSL offloading"), encargándose de esta manera del cifrado y entregando a las instancias tráfico sobre el puerto 80. Esto tiene como ventaja, no tener que configurar en todas las instancias detrás del balanceador, con el certificado correspondiente.

En el paso 4, se configura el "Health check". Este chequeo es el que permite al ELB poder saber si una instancia se encuentra apta para recibir tráfico (*InService*), o si no está apta para hacerlo (*OutOfService*). En este caso, se configurará el" Health check" para que valide sobre un archivo en particular ("alive.html") si el servidor está funcionando correctamente (ver figura 3.7.1).

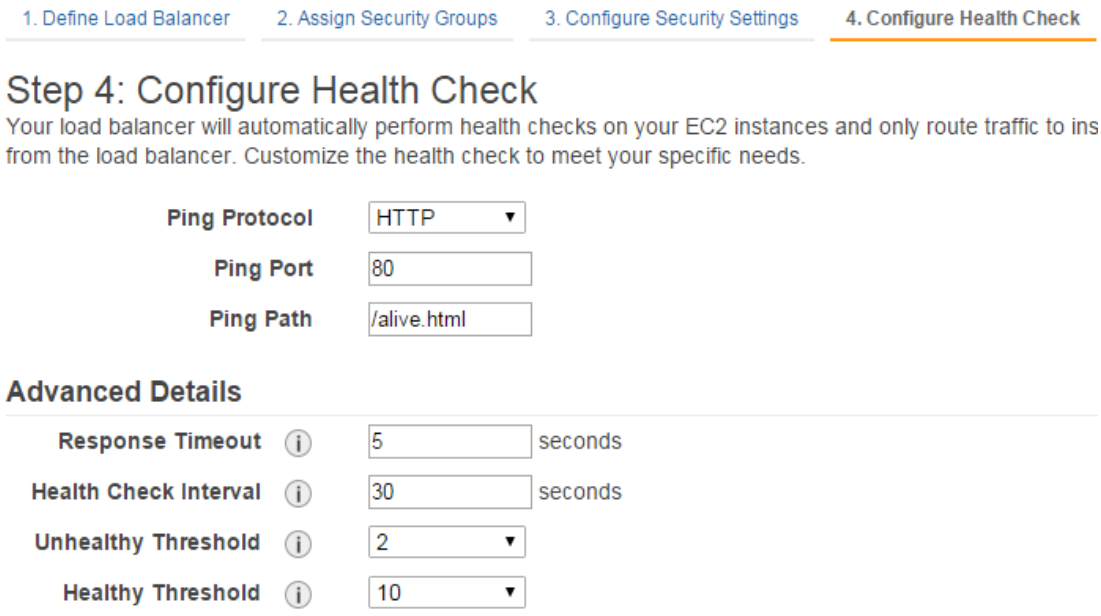

Figura 3.7.1 – Configuración de Health Check.

En el paso 5, se configuran las instancias que vamos a incorporar al balanceador (ver figura 3.7.2). En este punto del presente trabajo, solo se incorporará el servidor que tenemos configurado para servir contenido web.

La opción de "Enable Cross-Zone Load Balancing" permite que el ELB pueda enrutar tráfico entre diferentes Zonas de Disponibilidad. Esta opción, reduce la necesidad de mantener la misma cantidad de instancias dentro de los Diferentes "AZs", aunque siempre es recomendable crear las instancias en diferentes Zonas de Disponibilidad para agregar un nivel mas de tolerancia a fallas a la infraestructura. [8]

Por otro lado, en la opción de "Enable Connection Draining", configuramos la cantidad de tiempo que el ELB va a esperar antes de que cierre las conexiones hacia una instancia reportada

como “OutOfService”. Cuando ese tiempo se supera, el balanceador cierra forzosamente todas las conexiones que quedaron abiertas hacia ese servidor.

En el paso 6, se configuran las diferentes etiquetas que queramos configurar para este recurso de AWS, de la misma manera que se hace en la creación de las instancias EC2. El punto 7 es una revisión de todo lo configurado anteriormente, permitiendo cambiar algo que eventualmente haya quedado mal configurado.

Al finalizar el asistente de instalación se crea el Balanceador. Una vez creado, podemos ver el estado de las instancias que tenemos detrás del mismo, y también editar las configuraciones que consideremos necesarias (ver figura 3.7.3).

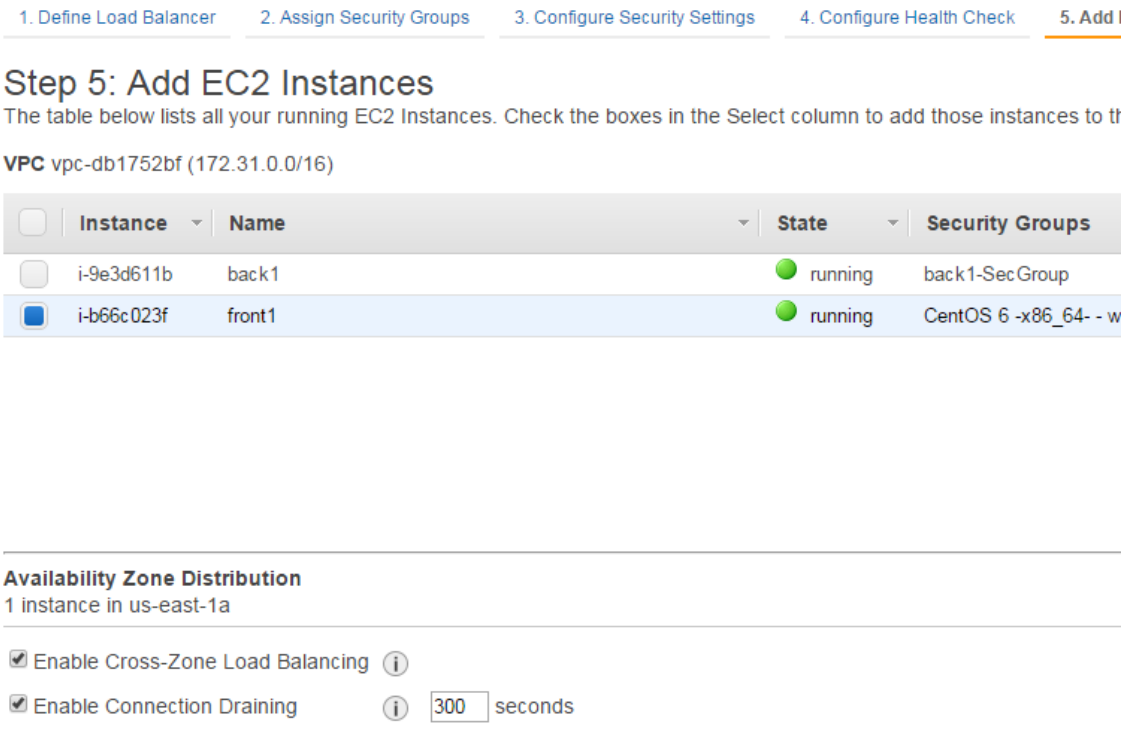

Figura 3.7.2 – Agregar instancias al ELB.

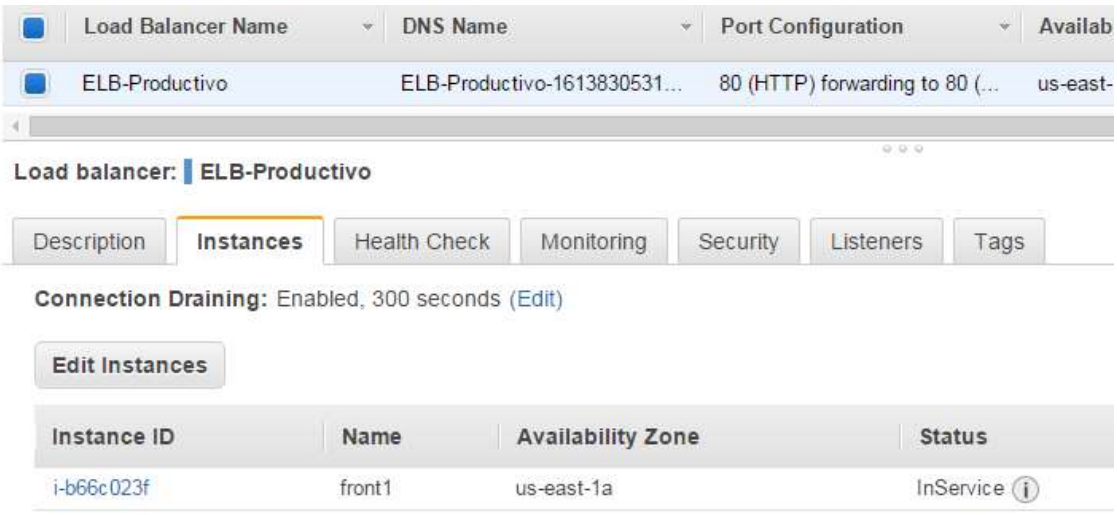

Figura 3.7.3 – Vista del Balanceador creado.

En los logs de apache dentro del servidor, vamos a poder ver el chequeo que realiza el balanceador (ver figura 3.7.4).

```
[root@front1 httpd]# tail -f www.example.com-access_log |grep alive.html
172.31.19.213 - - [04/Mar/2016:13:46:33 +0000] "GET /alive.html HTTP/1.1" 200
172.31.7.62 - - [04/Mar/2016:13:46:33 +0000] "GET /alive.html HTTP/1.1" 200 -
172.31.19.213 - - [04/Mar/2016:13:46:53 +0000] "GET /alive.html HTTP/1.1" 200
172.31.7.62 - - [04/Mar/2016:13:46:53 +0000] "GET /alive.html HTTP/1.1" 200 -
172.31.19.213 - - [04/Mar/2016:13:47:13 +0000] "GET /alive.html HTTP/1.1" 200
172.31.7.62 - - [04/Mar/2016:13:47:13 +0000] "GET /alive.html HTTP/1.1" 200 -
172.31.19.213 - - [04/Mar/2016:13:47:33 +0000] "GET /alive.html HTTP/1.1" 200
172.31.7.62 - - [04/Mar/2016:13:47:33 +0000] "GET /alive.html HTTP/1.1" 200 -
^C
```

Figura 3.7.4 – Chequeo del Balanceador sobre el servidor.

De esta manera, quedan configurados los dos servidores principales, junto con el Balanceador. Para realizar una primera prueba sobre esta infraestructura reducida, se procederá a realizar la primera migración de los datos de la aplicación, con el objetivo de poder corroborar el correcto funcionamiento de los sitios, antes de la configuración de las instancias redundantes.

CAPITULO IV

MIGRACION DE LA APLICACIÓN

4.1 Introducción

En el presente capítulo, se abordará todo lo referido a la migración de la aplicación hacia los nuevos servidores. En el punto 4.2, se detallarán los aspectos de seguridad a tener en cuenta para realizar la migración del servidor anterior a la nueva infraestructura. En el punto 4.3, se realizará la migración propiamente dicha, de los archivos del sitio y las bases de datos. En el punto 4.4 y 4.5 se especificarán cuáles son las tareas a realizar en el servidor web, y el servidor de base de datos respectivamente, luego de la migración de los datos. Los puntos 4.6 y 4.7 están destinados a desarrollar todo lo referente a las pruebas sobre la nueva infraestructura.

4.2 Configuraciones de seguridad

Para poder llevar a cabo la migración, hay que realizar unas configuraciones temporales a nivel servidor SSH, con el objetivo de que el servidor anterior pueda comunicarse vía ssh y realizar así la migración de todos los datos de manera correcta. Para esto, se editarán tres líneas de la configuración del servidor SSH en la instancia "front1" (en el archivo de configuración ubicado en "/etc/ssh/sshd_config"):

PermitRootLogin yes
PasswordAuthentication yes
AllowUsers centos nubity root

Dichas directivas permiten ingresar al usuario root por ssh (usando solo su contraseña) como cualquier otro usuario sin privilegios que configuremos sobre la instancia. Esto será solo temporal hasta que se concluya con la migración, dado que por cuestiones de seguridad no es recomendable ingresar a la instancia bajo el usuario con privilegios de administrador (root), y tampoco que los usuarios comunes puedan ingresar sólo con su contraseña.

La decisión de realizar la migración bajo el usuario root, y no un usuario creado para tal efecto, es que si se realiza bajo el superusuario, se conservan los permisos de los archivos tal como están en el servidor anterior, garantizando el correcto funcionamiento de la aplicación, y garantizando que los usuarios FTP de cada sitio van a poder tener los archivos con los mismos privilegios (siendo la migración totalmente transparente en este caso).

El próximo paso, es habilitar la ip del servidor a migrar en el "Security Group" de AWS creado para las instancias que se encargan de servir el contenido web. También la habilitaremos en el firewall del propio servidor.

4.3 Migración de archivos y Bases de Datos

La aplicación utilizada para realizar la migración, es RSYNC. La misma, es una aplicación libre para sistemas de tipo Unix y Microsoft Windows que ofrece transmisión eficiente de datos incrementales, que opera también con datos comprimidos y cifrados. Permite sincronizar archivos y directorios entre dos máquinas de una red o entre dos ubicaciones en una misma máquina, minimizando el volumen de datos transferidos. Una característica importante de rsync no encontrada en la mayoría de programas o protocolos es que la copia toma lugar con sólo una transmisión en cada dirección. rsync puede copiar o mostrar directorios contenidos y copia de archivos, opcionalmente usando compresión y recursión.

Para este caso, se le indicará a rsync que realice la transferencia de los archivos mediante la conexión SSH que configuramos previamente, por lo que el puerto a utilizar va a ser el del ssh del nuevo servidor front:

rsync -avP -e 'ssh -p 6000 -c blowfish' /var/www/html/ root@54.85.187.146:/datos/html/

Con la opción “a”, se le está indicando a rsync que conserve en el destino, todos los atributos de archivos y carpetas a transferir. Con la opción “v”, se puede ver en la consola donde se ejecuta el comando, el estado de la transferencia, indicando el archivo y la carpeta que está transfiriendo en cada momento. Con la opción “P”, se le indica que el progreso lo haga de manera parcial, con el objetivo de poder soportar pequeñas interrupciones al momento de la transferencia. La opción “e” permite utilizar otra aplicación para poder realizar la conexión (en este caso, utilizamos SSH).

Para evitar cualquier problema con el link simbólico al momento de la transferencia, es recomendable realizar el rsync directamente hacia la unidad creada para alojar los archivos de los sitios, es por eso que en el destino se configuró “/datos/html/”.

4.4 Configuración de datos migrados en servidor front

Una vez que los datos de la aplicación se encuentran en el nuevo servidor, es necesario poder configurar los nuevos parámetros de conexión a la base de datos de cada sitio. Para eso, se editarán los archivos donde se detalla el acceso a la base de datos, cambiando la IP de la misma. Por ejemplo, para los sitios manejados bajo el CMS “Wordpress”, el archivo a editar es el “wp-config.php”, donde la configuración pasará de ser “localhost” a tener la ip del servidor back:

define('DB_HOST', '172.31.1.143');

Para sitios manejados bajo “Joomla”, el archivo a modificar es el “configuration.php”, en el cual se edita bajo el mismo concepto que el ejemplo anterior.

Para sitios que no son sobre CMSs conocidos, se deberá hablar con el cliente para que de su lado puedan realizar el cambio de la conexión de la base de datos, o en su defecto indique dónde realizarlo.

Pensando en la alta disponibilidad que se tiene que garantizar sobre la infraestructura, es una desventaja poder configurar la conexión a la base de datos a través de la ip privada del servidor de base de datos (en este caso back1). En un escenario donde dicho servidor se encuentre comprometido, y haya que promover el servidor esclavo a maestro, se va a tener que cambiar la IP de conexión a base de datos en todos los archivos de configuración nuevamente, perdiendo mucho tiempo para volver a tener operativa la infraestructura.

Para poder sortear dicho problema, la configuración a la conexión de base de datos se realizará a través de un dominio y no de una ip:

define('DB_HOST', 'back1-aws-master.com');

Luego, para que dicho dominio resuelva a la ip que tiene que resolver, se editará el archivo *“hosts”* de cada servidor web. En *“/etc/hosts”* se va a agregar la siguiente línea:

172.31.1.143 back1-aws-master.com

De esta manera, si se tiene que cambiar la ip de conexión a la base de datos, solo se hará en ese archivo, y no en todos los archivos de configuración que posean las aplicaciones.

4.5 Configuración de datos migrados en servidor Back

Como paso previo de la migración de los datos, se realizaron los dumps de todas las bases de datos de los sitios, en la carpeta en "/var/www/html/dumps" del servidor a migrar, con el objetivo de poder pasarlos con el rsync. En este paso, es importante hacer el dump de la base "mysql" también, dado que de esa manera, copiamos los usuarios y contraseñas de mysql, haciendo aún más transparente la migración.

El próximo paso, es llevar los dumps (backups) de la base de datos al servidor definido para tal fin. Para esto se empleará nuevamente el comando rsync, pero en este caso nos comunicaremos con back1 a través de la red privada:

rsync -avP -e 'ssh -p 6000 -c blowfish' /vr/www/html/dumps/ root@172.31.1.143:/root/dumps/

Una vez que se tienen los dumps en el servidor back, se procede a realizar los restores de las bases de datos. Primero se tienen que crear las bases de datos, con el comando *"create database nombre-base-de-datos;"* desde la consola de mysql, y luego realizar el restore de dicha base de datos (desde la consola del sistema operativo):

[root@back1 dumps]# mysql –p –u root nombre-base-de-datos < dump_base_de _datos

Como a la base de datos anterior se accedía de forma local, dado que era un solo servidor, se tienen que editar los usuarios mysql para que se los pueda acceder de forma remota. Para eso se procede a cambiar el valor de los usuarios en la Columna "Host" la tabla "user" de la base de datos mysql. El valor por el cual se lo va a cambiar es "%", indicando que se lo puede acceder desde cualquier lugar. Esto se realiza desde el Shell de mysql:

```
mysql> use mysql;
mysql> UPDATE mysql.user SET Host='%' WHERE User='usuario-mysql';
mysql> FLUSH PRIVILEGES;
```

Uno de los problemas que se presentan ante el cambio de versión del servidor mysql, es que algunas tablas de la base de datos "mysql" tienen el orden de sus columnas cambiadas, por lo que es necesario realizar ese cambio para esta nueva versión. Percona MySQL permite realizar dicha actualización de manera automática a través de uno de sus comandos que se ejecutan desde el Shell del servidor:

[root@back1 dumps]# mysql_upgrade -p --upgrade-system-tables

De esta manera, la bases de datos del sistema se ordenan de acuerdo a la versión actual, lo que permite seguir con el siguiente cambio, el cual consiste en editar los privilegios de cada usuario sobre la base de datos correspondiente, dado que dichos privilegios estaban configurados para el acceso de manera local:

mysql> GRANT ALL PRIVILEGES ON base-de-datos. to 'usuario-mysql'@'%';*

Para poder corroborar todos los cambios realizados, se hace desde el servidor front1, ingresando con las credenciales de cada usuario MySQL y probando que todos funcionen correctamente.

[root@front1 html]# mysql -p -h 172.31.1.143 –u usuario-mysql

4.6 Pruebas finales

Las pruebas finales se realizan visitando los sitios desde un navegador, para comprobar el correcto funcionamiento del mismo. Como los DNS todavía no están apuntando a la nueva infraestructura, se editará el archivo “hosts” de la computadora para poder forzar a que el sitio apunte al nuevo balanceador, de la misma manera que se realizó al momento de configurar apache y php.

En este punto, se le notifica al cliente mediante un ticket de soporte, indicando la forma en la que tiene que realizar las pruebas, y solicitándole que nos comunique en caso de que vea alguna falla o mal funcionamiento del sitio.

4.7 Migración final de la infraestructura

Al momento de finalizar las pruebas, tanto del lado de Nubity como del lado del cliente, se puede comenzar a planificar de manera conjunta el plan de migración final. En cuanto a los pasos para poder realizarlo, son los mismos que se explicaron en los 5 primeros puntos del presente capítulo, solo que en esta oportunidad, tenemos que estar coordinados con el cliente, dado que será necesario tener el sitio fuera de línea.

La única diferencia a los pasos de la migración inicial, es que aquí se deben modificar los DNS en Route53 para que apunten a la nueva infraestructura.

Para llevar a cabo la migración, primero se van a cambiar los DNS al nuevo balanceador, y en el servidor web, el cliente configurará una placa de mantenimiento para que los usuarios que visiten el sitio vean dicha placa y estén informados acerca de las tareas. Como Route53 nos permite tener bajos TTL en los registros de DNS, dicho cambio se propagará totalmente en 5 minutos (300 segundos), que es el número que se tiene configurado en los registros DNS. Una vez replicados totalmente, se procede con los pasos siguientes de la migración

Como medida de seguridad, se para el servicio de apache en el server de origen, con el objetivo de asegurarnos que no entre tráfico de ningún tipo al mismo. Una vez detenido apache, se realizarán los backups (dumps) de las bases de datos, y se parará el servicio de mysql también.

Luego se ejecuta el rsync para hacer el paso de datos. El rsync va a pasar solo los datos nuevos, o que hayan tenido una modificación reciente en el servidor actual, por lo que no será prolongado el tiempo de pasaje de los datos al nuevo frontend.

Por último, Se realizarán los restore correspondientes en el servidor de base de datos (tal como se especificó en el punto 4.5).

A partir de este momento, se está en condiciones de notificarle al cliente que las tareas del lado de Nubity han finalizado, y que puede sacar la placa de mantenimiento del sitio para comenzar a servir tráfico en el nuevo servidor.

Es importante tener en cuenta que en este momento se tiene la infraestructura productiva, pero reducida, de manera tal que inmediatamente después a dejar productivos los servidores, se procederá a crear las instancias redundantes que permitan darle la alta disponibilidad requerida a la infraestructura.

CAPITULO V

CREACIÓN DE LOS SERVIDORES REDUNDANTES

5.1 Introducción

Para poder garantizar alta disponibilidad, se procederá a crear dos nuevas intancias; una para servir el contenido web, y otra para que funcione como mysql esclavo del servidor back1. Para poder empezar a realizar dichas tareas, es importante que se haya confirmado con el cliente que el sitio está funcionando correctamente en la nueva infraestructura. De esa manera, se pueden crear estas nuevas instancias a partir de las recientemente productivas, haciendo mucho más fácil el proceso.

AWS permite crear de forma muy sencilla una imagen de cualquiera de los servidores, tomando "snapshots" de los discos. Un "snapshot" es una imagen que se toma de la totalidad del disco, el cual puede servir como un método de backup de la información contenida en el mismo, o (como en este caso) para la creación de una instancia que necesite tener la misma información que otra ya creada.

Los "snapshots" nos permiten crear AMI's (Amazon Machine Images) en AWS. Dichas AMI's son imágenes del sistema operativo que permiten poder crear instancias nuevas en la nube, bajo las especificaciones en cuanto a permisos, información y configuración que se requiera en ese momento.

Dichas AMI's sirven también como punto de partida ante la recuperación de un eventual escenario en el cual las instancias sufran una caída de la que no se puedan recuperar. Se pueden comenzar a crear nuevas instancias a partir de las AMI's y tener de forma más rápida los servidores operativos nuevamente.

5.2 Creación de AMI

Se va a desarrollar la explicación de la creación de la AMI, utilizando el servidor de base de datos ya configurado como ejemplo. Para la creación del nuevo servidor web, el procedimiento es el mismo que el detallado a continuación.

Para la creación del servidor redundante, que va a servir como esclavo del servidor principal antes configurado, se procederá a realizar snapshots de los dos volúmenes EBS que tiene la instancia principal. Para eso, desde el la interfaz de administración de EC2, se ingresará al apartado donde se listan todos los volúmenes, y desde la opción "actions" se creará el snapshot de ese disco en particular (ver figura 5.2.1). Luego, se indica el nombre y una breve descripción del snapshot que vamos a crear (ver figura 5.2.2).

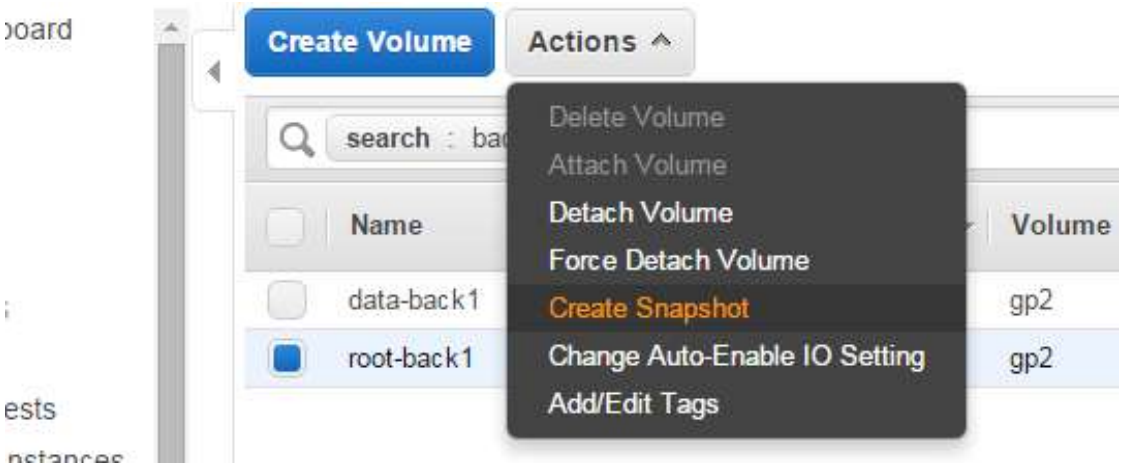

Figura 5.2.1 – Seleccionar volumen para snapshot.

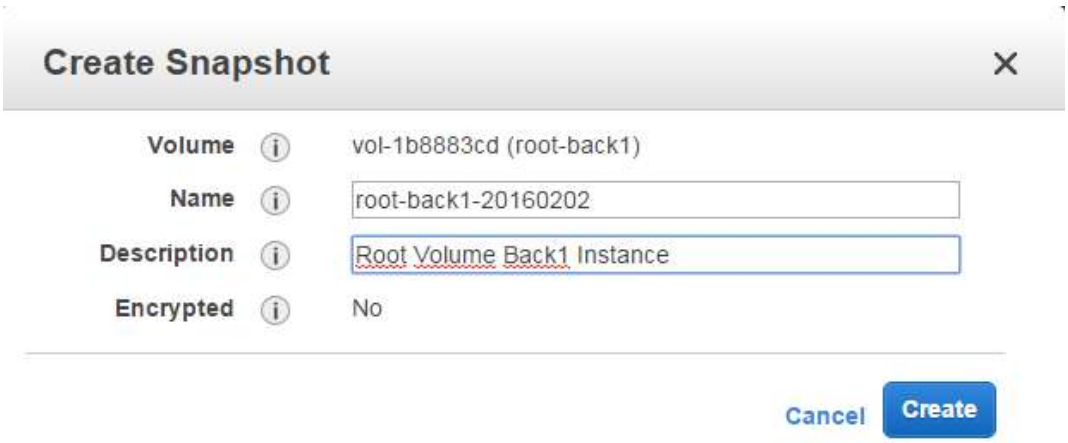

Figura 5.2.2 –Nombre y Descripción del Snapshot.

Para este caso en particular, se crearán dos snapshots, uno del volumen principal de la instancia, y otro del volumen montado en "/datos". Dichos snapshots se pueden hacer sin la necesidad de apagar la instancia, lo cual es una gran ventaja en este momento que ya tenemos los servidores productivos.

Una vez que están creados (se puede ver el estado de la creación desde el apartado *snapshots* en la interfaz web – Figura 5.2.3), se procederá a crear la imagen (AMI). Para esto se configuran una serie de opciones, referidas al tipo de virtualización de la instancia, el ID de núcleo de linux, y otras configuraciones que se obtienen de la instancia original (ver figura 5.2.4.).

De esta manera, se crea la AMI necesaria para luego poder crear la instancia back2 (slave). Para la creación de dicha instancia, se procede de la misma manera que en la creación de la primera instancia de esta infraestructura (sección 3.3), solo que al momento de elegir la Imagen que se utilizará,, seleccionamos la AMI creada en este caso para tal fin.

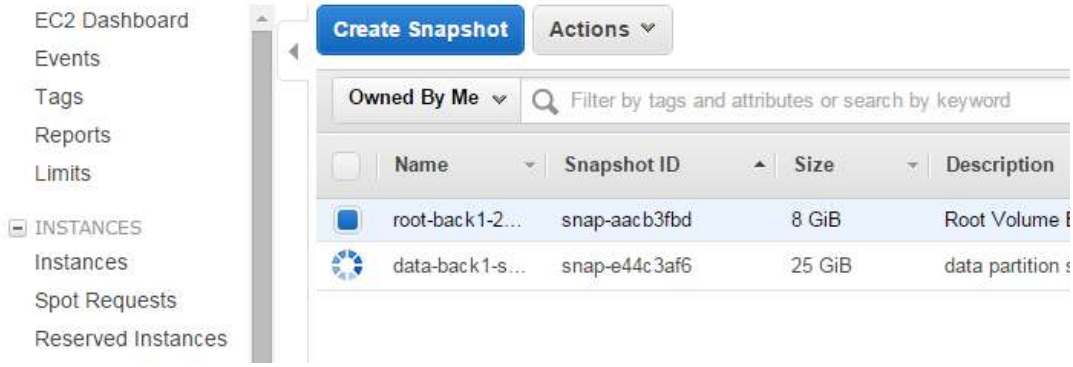

Figura 5.2.3 –Estado de la creación de los snapshots

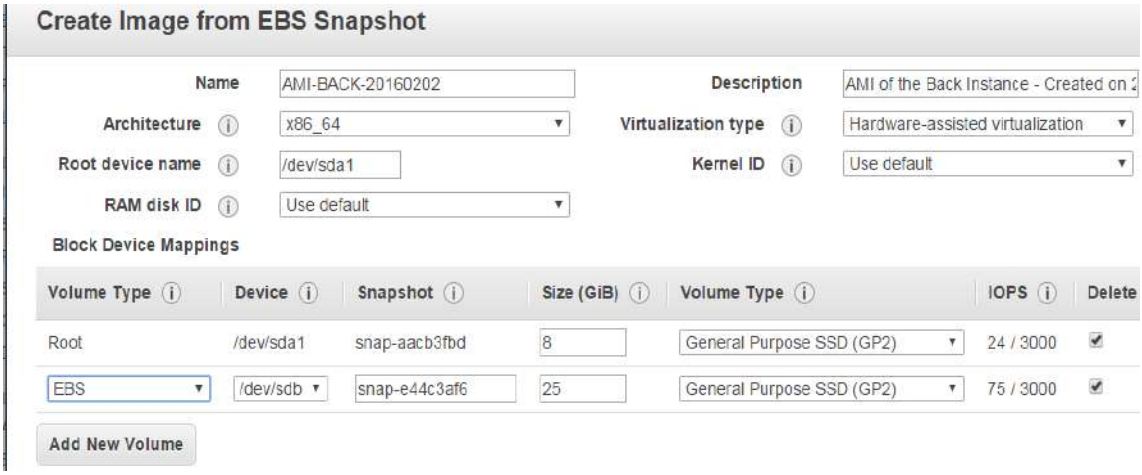

Figura 5.2.4 –Creación de Imagen a partir de los Snapshots

5.3 Configuración de Servidores de Bases de Datos Maestro/Esclavo

Una vez creados los servidores redundantes, se procede a crear la replicación de las bases de datos en los mismos.

La configuración que se eligió en el presente trabajo, es de tipo Maestro/Esclavo (o Master/Slave). En este tipo de configuración, el servidor maestro es el que tiene permisos de escritura y lectura sobre todas las bases de datos. El servidor esclavo, sólo tiene permisos de lectura sobre las mismas.

Para garantizar la replicación, el servidor maestro escribe en un log binario, todas las actualizaciones que se realizan sobre las bases de datos que se están replicando. El servidor esclavo, lee de dicho log y en base al mismo va actualizando sus bases de datos. Para esto, se establece una conexión entre los dos servidores, la cual siempre permanece abierta esperando que el maestro actualice el log binario, para que el esclavo pueda realizar las actualizaciones pertinentes.

Como primer paso, se configura el servidor maestro, editando el archivo de configuración de mysql (/etc/my.cnf), en el cual se agregan las siguientes opciones:

bind-address = 172.31.1.143
server-id = 1
log_bin = /var/lib/mysql/mysql-bin.log
binlog_do_db = <base-de-datos>

La primera opción, indica la IP privada del servidor donde se encuentra el maestro de la replicación (en este caso, back1). Luego, el *server-id* es una variable que identifica a cada uno de los nodos de la replicación. Cada servidor de base de datos tiene que tener necesariamente Números de identificación diferentes. El *log_bin* es la ruta en la cual mysql escribirá los logs que luego serán leídos por el servidor esclavo para poder replicar la información. Por último, la opción *binlog_do_db* indica la base de datos a replicar. Dicha línea se repite por la cantidad de base de datos que necesitemos replicar.

Una vez configurado el maestro, se procede a reiniciar el servidor mysql para que tome los cambios que se realizaron. Luego del reinicio, se crea el usuario de mysql con los privilegios necesarios para realizar la replicación:

*GRANT REPLICATION SLAVE ON *.* TO 'usuario_slave'@'%' IDENTIFIED BY 'contraseña';*

Para poder seguir con la configuración de la replicación, es necesario poner a todas las tablas de las bases de datos en modo lectura, dado que se requiere que por este período de tiempo las bases no se actualicen:

USE base-de-datos;
FLUSH TABLES WITH READ LOCK;
SHOW MASTER STATUS;

El último comando mostrado, permite identificar la posición que el servidor maestro lleva escribiendo en el log de replicación en este momento. Este es un dato muy importante a tener en cuenta, ya que se tiene que configurar en el servidor esclavo (ver figura 5.3.1):

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      107 | newdatabase  |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Figura 5.3.1 – Ejemplo Salida comando “SHOW MASTER STATUS;”

Luego, se procede a realizar los “dumps” de las bases de datos a replicar. Al terminar los dumps, se desbloquean las tablas para que puedan escribir otra vez, bajo el comando “UNLOCK TABLES”, desde la consola de mysql.

Una vez realizados los pasos del lado del maestro, se procede a configurar la base de datos esclava. Como primer paso, se realiza el restore de las bases de datos a partir de los dumps creados anteriormente en el back maestro.

El próximo paso, es la edición del archivo de configuración del servidor esclavo. En el mismo, se procede a crear las siguientes líneas:

server-id = 2
log_bin = /var/lib/mysql/mysql-bin.log
binlog_do_db = <base-de-datos>
relay-log = /var/lib/mysql/mysql-relay-bin.log

Como se había especificado anteriormente, el *server-id* es diferente para este servidor. La variable *log_bin* y *bin_log_do_db* tiene el mismo propósito que en el servidor maestro. La variable que cambia es *relay-log*, en la cual se especifica el log que el esclavo va a utilizar para registrar todo lo referido a la replicación. Una vez configurado el archivo my.cnf del slave, se reinicia el servidor mysql para que tome los cambios.

Luego del reinicio, se procede a indicarle al servidor esclavo, mediante el comando "CHANGE MASTER", el usuario que se utilizará en la replicación, la posición del log en la cual tiene que comenzar a replicar (figura 5.3.1), el archivo de replicación a utilizar, y la dirección IP del servidor maestro:

CHANGE MASTER TO MASTER_HOST='172.31.1.143',MASTER_USER='usuario_slave',
MASTER_PASSWORD='contraseña', MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS= 107;

START SLAVE;

Por último, se activa el slave en dicho server, y la replicación comienza a funcionar. Con el comando *SHOW SLAVE STATUS\G;* podemos ver el estado de la replicación, en donde se visualiza si el esclavo se encuentra replicando correctamente.

5.4 Configuración de nuevo front.

Como se indico en el presente capítulo, el alta del nuevo servidor web se realiza de la misma manera que para el esclavo de las bases de datos. El principal cambio que debemos realizar sobre la nueva instancia, es la edición del archivo */etc/fstab*, en el cual se debe editar la línea agregada en la sección 3.5.1 del presente trabajo. En dicha línea, se cambia el ID del disco por el correspondiente al front2.

Para finalizar, lo que resta es agregar el nuevo front al balanceador. Esto se realiza de la misma manera que se agregó el front1 al momento de la creación del ELB. Es importante que cuando se agregue, los dos servidores queden en estado "InService" para que el balanceador redirija el tráfico hacia los dos servidores (ver figura 5.4.1).

Figura 5.4.1 – Servidores de front en estado InService.

CAPITULO VI

ALTA Y CONFIGURACIÓN DE LOS SERVICIOS DE NUBITY

6.1 Introducción

Nubity es un Administrador de Servicios en la nube (o "Cloud Manager") que le permite al usuario centralizar toda su infraestructura en un solo panel de Administración. Permite agregar servidores de diferentes proveedores (tanto físicos como en la nube).

Es un SaaS que permite poder monitorear todos los servidores en tiempo real, tener gráficos del comportamiento de la infraestructura y sus aplicaciones, como así también tiene la opción de delegar la administración de la infraestructura a un equipo de soporte que trabaja 24x7.

También les brinda a sus clientes la posibilidad de poder llevar a cabo una solución integral para los problemas de infraestructura que puede tener una empresa. Tal como se detalló en el presente trabajo, Nubity se puede encargar de armar una infraestuctura completamente nueva, o también proponer mejoras sobre infraestructuras ya creadas anteriormente. De esta manera, Nubity trabajará en el diseño, alta y configuración de los servidores, como así también en la posterior administración de la infraestructura una vez que la misma ya se encuentre productiva.

En el presente capítulo, se va a detallar cómo se integra una nube del cliente a Nubity, como funcionan el sistema de backups, y qué es lo que se puede monitorear con este SaaS.

6.2 Integración de la Nube de AWS en el panel de Nubity

Para la integración de la nube de AWS en el panel de Nubity, se tiene que crear un usuario en la cuenta de amazon, que sólo tenga acceso a los servicios de EC2. Para esto, dentro de la sección de "Security Credentials" (ver figura 6.2.1), se crea el usuario "nubity.account", el cual se va a utilizar para realizar la integración.

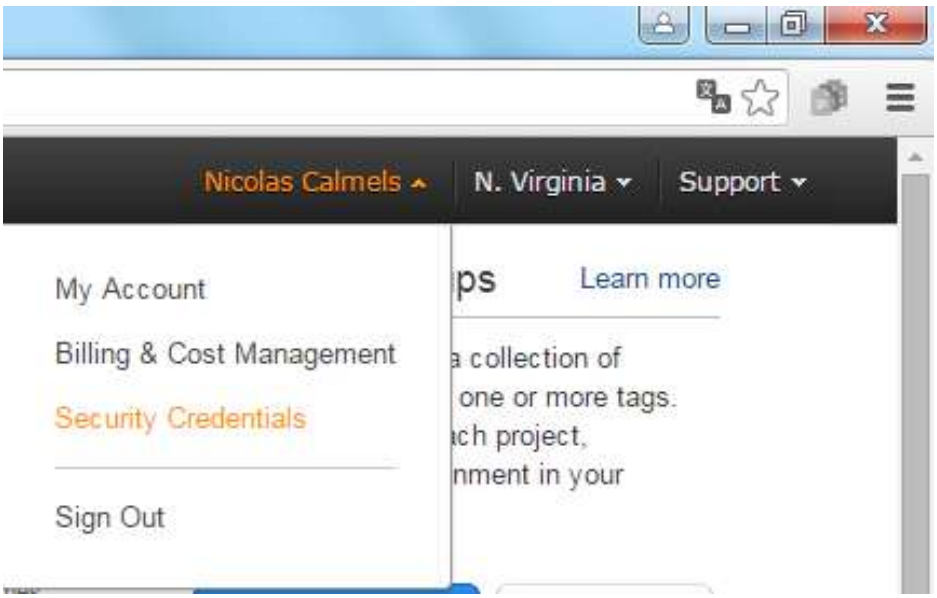

Figura 6.2.1 – Seccion "Security Credentials".

En la sección "Users" se selecciona la opción "create new user" y luego el botón "create". En la próxima pantalla, se visualizarán las credenciales de acceso de Dicho Usuario (Access Key ID y

Secret Access Key), las cuales son necesarias para realizar la integración en el panel de Nubity (ver figura 6.2.2).

Figura 6.2.2 – Credenciales de Acceso de nuevo usuario.

Luego de la descarga de las credenciales necesarias para la integración, se deben configurar las políticas que va a tener el nuevo usuario. Dichas políticas indican los privilegios del usuario sobre los servicios de AWS. Para este caso, la política a asociar al usuario es la "AmazonEC2FUllAccess", la cual le da permisos sobre todo lo que referido a EC2 (ver figura 6.2.3).

Dicha política se crea para que el usuario no tenga acceso a otro de los servicios que AWS ofrece, garantizando así la seguridad de la cuenta.

Figura 6.2.3 – Agregar Política a usuario.

Una vez configurado dicho usuario, se procede a agregar la nube en el panel de Nubity. Se selecciona el tipo de "cloud" que se quiere agregar (ver figura 6.2.4). Para este caso, se elige

Amazon Web Services. Luego, se agregan los Accesos para el usuario “nubity.account” y se le da un nombre a la Cloud.

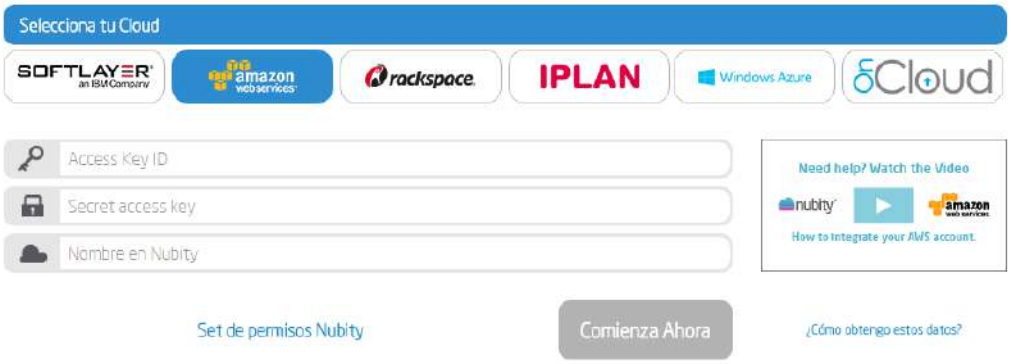

Figura 6.2.4 – Integrar nube al panel de nubity.

Luego de agregar los accesos, Nubity se sincroniza con AWS (ver figura 6.2.5) y a partir de ese momento, se puede visualizar todas las instancias en el panel de administración, pudiendo realizar acciones sobre las mismas, como instalar el monitoreo, encenderlas o apagarlas, crear un ticket de soporte, chatear con el soporte técnico, o ver los gráficos y alertas.

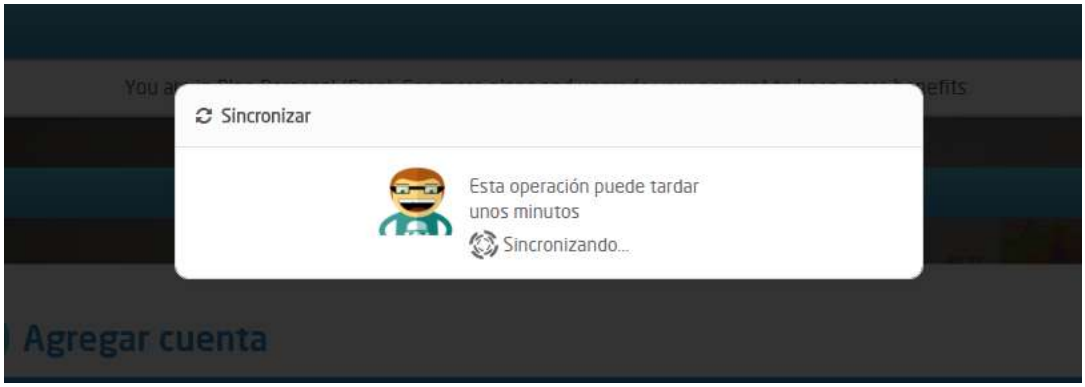

Figura 6.2.5 – Sincronización en proceso.

6.3 Alta y Configuración de backups

La configuración y administración de los backups, está totalmente delegada a Nubity. La empresa se encarga de realizarlos y monitorear que los mismos se estén llevando a cabo de manera correcta. En el caso que el cliente solicite algún archivo de los backups, tendrá que hacerlo mediante ticket de soporte y un técnico de la empresa se encargará de llevar a cabo la solicitud.

Los backups corren en el momento que la infraestructura tiene menos tráfico, para no afectar a la performance de la misma. Los backups que se ejecutan durante la semana (de lunes a sábado) son incrementales, y los domingos son backups *Full*. La diferencia entre ambos es que los primeros solo guardan la información que se modificó desde la fecha del último backup

incremental y la actual, mientras que los *Full* guardan toda la información contenida dentro de las carpetas que se van a respaldar.

El primer paso para el alta de los backups, será definir con el cliente cuáles son los archivos que se respaldarán, y con qué frecuencia. Los archivos más comunes a respaldar, siempre son los archivos del sitio y las bases de datos. A esto Nubity le suma todos los archivos de configuración de las diferentes aplicaciones instaladas en los servidores. También suelen respaldarse los archivos que el cliente posee en la carpeta home de su usuario ssh (en caso de tenerlo).

Una vez que se acordaron cuáles son las rutas que se van a resguardar de cada servidor, se define la rotación de los mismos. Esto es, por cuánto tiempo se va a almacenar un backup. Por defecto, la política que se toma es respaldar los backups incrementales con una rotación de 7 días y los backups *Full* con una rotación de 15. Es decir, que los incrementales sólo se guardan de manera semanal, y se almacenan los últimos 3 backup *Full* que se ejecutaron.

Luego, se hace una estimación lo más precisa posible de la cantidad de espacio necesario para poder almacenar los backups, teniendo en cuentra los 3 *Full* y los 6 incrementales. Al total de espacio requerido, se le suma un 30% más, teniendo en cuenta un futuro crecimiento del sitio y las bases de datos. Cuando se tiene el número del espacio a crear, se asocia un disco EBS (de tipo magnético) a una de las instancias, y se lo monta en la carpeta */backup*. En esa ubicación, se van a almacenar los backups de toda la infraestructura.

La ventaja de poder implementar los backups de esta manera, y no por ejemplo a través de *snapshots* de los discos desde la consola de AWS, hace que en el escenario en el que se requiera un solo archivo de un sitio en particular, podamos obtener sólo eso, sin necesidad de restaurar todo el directorio. Obtener *snapshots* con una periodicidad de tiempo estipulada con el cliente es muy conveniente, pero con el objetivo de brindar alta disponibilidad a la infraestructura, y no como respaldo de los archivos del sitio.

6.4 Alta y configuración del Monitoreo

Una vez creada la nube en Nubity, se puede comenzar a configurar el monitoreo sobre la instancia. Para eso, en la sección del panel donde se visualizan todos los dispositivos, seleccionamos la opción “Instalar Ahora!”, opción que nos lleva a un paso a paso bien detallado de cómo se tiene que instalar el monitoreo (ver figura 6.4.1).

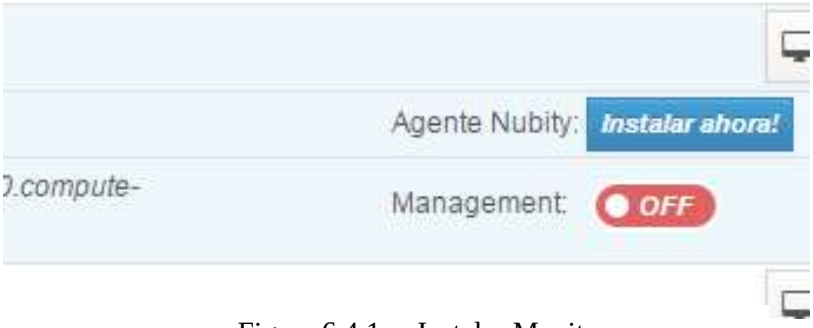

Figura 6.4.1 – Instalar Monitoreo.

Para la instalación del monitoreo, se ejecuta un script con privilegios de super usuario dentro del servidor, el cual se descarga desde la página de *Nubity* (ver figura 6.4.2). Al ejecutar el script, el mismo analiza cuál es la distribución del sistema operativo del servidor e instala el agente de monitoreo acorde a dicha información.

```
[root@front1 src]# wget http://packages.nubity.com/installer/nubity-installer-last.sh
--2016-03-12 12:43:10--  http://packages.nubity.com/installer/nubity-installer-last.sh
Resolving packages.nubity.com... 54.158.134.133
Connecting to packages.nubity.com|54.158.134.133|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4477 (4.4K) [application/x-sh]
Saving to: "nubity-installer-last.sh"

100%[======================================================================================

2016-03-12 12:43:10 (14.4 MB/s) - "nubity-installer-last.sh" saved [4477/4477]

[root@front1 src]# chmod +x nubity-installer-last.sh
[root@front1 src]# ./nubity-installer-last.sh
```

Figura 6.4.2 – Instalación del agente en el servidor.

Una vez instalado el agente en el servidor, se visualiza en el panel de Nubity los diferentes monitoreos que se pueden configurar sobre el mismo (ver figura 6.4.3).

Figura 6.4.3 – Configuración del Monitoreo en el panel de Nubity.

Por defecto, el template que ya tiene configurado es el de “Linux Basic Monitoring”, el cual monitorea los aspectos básicos de un Sistema Operativo Linux: uso de la partición swap, cantidad de procesos totales, conectividad del puerto SSH, I/O de disco, carga del servidor, uso de CPU, uso de memoria RAM, espacio disponible en los discos montados en el servidor y el ancho de banda de entrada y salida en sus interfaces de red.

También se agregará en los servidores web, el monitoreo sobre métricas de apache, monitoreo de varnish, FTP y monitoreo de “Web Content” sobre el sitio. Este ultimo monitoreo analiza una palabra o frase que se encuentre en el contenido del sitio web. En caso de no encontrar dicho contenido, o si supera el tiempo de espera, Nubity generará una alerta de dicho error.

Las métricas de apache, hacen referencia a monitoreos que permiten saber si apache está ejecutándose correctamente, si llego a la cantidad máxima de procesos hijos que tiene permitido abrir y si está recibiendo conexiones de manera correcta.

Con respecto a FTP, monitorea si el servicio se encuentra disponible, alertando en caso de que se haya detenido.

En lo que respecta a varnish, el monitoreo nos alerta en caso de que el servicio se haya detenido, si perdió conectividad con el “backend” configurado, y si el porcentaje de cacheo en función a la cantidad de peticiones que tiene el servidor es muy bajo.

En los servidores back1 y back2, se configurará todo lo referido al monitoreo sobre bases de datos Mysql. Se monitorea si el servicio está levantado, si llego a la cantidad límite de procesos simultáneos, tiempo de ejecución de las consultas SQL, y si dichas consultas generaron “locks” sobre las bases de datos.

En el servidor esclavo, también se configuran los chequeos pertinentes para saber si el esclavo está corriendo y si está totalmente sincronizado con el maestro.

Una vez configurados los monitoreos, ya se comenzarán a visualizar los gráficos sobre lo que se está monitoreando, teniendo la posibilidad de ver el estado del servidor en tiempo real, o también un histórico de los diferentes recursos y aplicaciones del mismo.

Otra opción que tiene el cliente, es configurar las notificaciones que desee recibir a su cuenta de correo. Dependiendo de la gravedad de la alerta, el cliente puede configurar la opción para que le llegue un mail cada vez que surge una alerta sobre su infraestructura.

De esta manera, queda configurada la infraestructura en el panel de Administración de Nubity. El usuario ahora tiene la posibilidad de monitorear todos sus servidores, como así también solicitar cualquier tipo de soporte técnico que requiera sobre la misma (ya sea a través de un ticket de soporte, o abriendo una cadena de chat con el técnico disponible en ese momento).

CAPITULO VII

Pruebas de Alta Disponibilidad

7.1 Introducción

Antes de la migración final del sitio, se realizan diferentes tareas que permiten corroborar si la infraestructura que se creó puede efectivamente servir el tráfico para la cual la misma fue preparada.

En la primera sección del presente capítulo, se va a detallar cuáles son las tareas que se llevan a cabo para comprobar que la nueva infraestructura no posea ningún cuello de botella o falta de desempeño que se haya podido pasar por alto al momento del diseño de la misma.

Luego de realizar los cambios que sean necesarios, producto de las pruebas detalladas anteriormente, se procederá a la migración final del sitio. Luego de que el sitio esté migrado, se procederá a indicar cuáles son los escenarios en los cuales la infraestructura puede sufrir una degradación en su desempeño, o hasta una caída.

En la segunda sección de este capítulo, se planteará el plan de acciones a ejecutar sobre cada escenario propuesto, para poder tener nuevamente disponible la infraestructura en el menor tiempo posible. El objetivo de esto es tener presente cuáles son los diferentes escenarios en los cuales el sitio puede verse comprometido, y cuáles son las acciones a llevar a cabo por parte de los administradores.

7.2 Pruebas de Performance

Las pruebas de rendimiento se llevarán a cabo con una herramienta que permita poder simular una cantidad determinada de visitas sobre la infraestructura, con el objetivo de poder determinar dónde puede haber un cuello de botella, tanto en la aplicación como en la infraestructura, para poder tomar las acciones pertinentes.

Como primer paso, debemos realizar las pruebas sobre diferentes URLs del sitio, tal que la simulación sea lo más real posible. No sirve solo realizar la prueba sobre la página principal del sitio, o sobre pocas secciones del mismo, dado que no sería del todo real.

Para esto, se tomará el log de accesos de apache del servidor a migrar, que es el lugar donde queda detallados los accesos que se hacen al sitio, y se procederá a obtener las URLs, ordenadas de manera tal que las primeras sean las que son más visitadas:

[root@front1 httpd]# cat /var/log/httpd/acces_log.log | awk '{print $7}' | sort | uniq -c | sort -nr | awk '{print $2}' >> url_para_prueba.txt

De esta manera, en el archivo de texto “url_para_pureba.txt” quedará una lista de URLs que utilizaremos para realizar la prueba de performance sobre la nueva infraestructura.

El software que se utilizará para llevar a cabo la prueba es “siege”. El mismo es un simulador HTTP que permite realizar pruebas de carga sobre la infraestructura. Soporta autenticación básica, protocolos HTTP, HTTPS y también FTP. Permite apuntar tráfico a un server específico con una cantidad determinada de usuarios concurrentes. [10]

Para las pruebas que se van a realizar sobre la infraestructura, se iniciarán las mismas simulando una cantidad de 100 usuarios durante 1 minuto. A partir de ahí, se irá incrementando el número de usuarios concurrentes en 50 para las posteriores pruebas en caso de que no se vea ningún tipo de degradación en el sistema.

El comando que utilizaremos es el siguiente:

siege -t 1m -c 100 -f url_para_prueba.txt

Con "-f" indicamos el archivo donde están las urls que utilizará para realizar las pruebas, con "-t" el tiempo de duración de la prueba, y con "-c" la cantidad de usuarios concurrentes que se simularán.

Las pruebas que se realizaron mostraron que el sistema comenzó a degradarse cuando se simularon 500 usuarios concurrentes. Del 100% de disponibilidad que se habían obtenido en las pruebas anteriores, se pudo observar que, al momento de simular las visitas con 500 usuario, dicho número cayó a 90%, dando un tiempo máximo de respuesta de 40 segundos en la transacción más larga.

Una de las primeras conclusiones que se obtuvieron, fue que la decisión de poner una capa de caché (varnish) por encima del servidor web sea acertada, dado que se logró que muchas de las consultas fueran servidas por dicha aplicación, lo cual generó mucho menos tráfico sobre apache en los servidores de front. Esto hace que los servidores estén dentro de los parámetros normales de carga y consumo de recursos durante la prueba.

Al compartir dichos resultados con el usuario, se decidió que la infraestructura estaba correctamente dimensionada, dado que ese número de usuarios concurrentes es mucho mayor al que tienen pensado llegar.

No obstante, se analizó por el motivo por el cual se llegó a ese cuello de botella en los 500 usuarios concurrentes, y se pudo diagnosticar que se debía a consultas de mysql que tardaban mucho tiempo en poder resolverse.

Se planteó que, a futuro, dicho problema sea resuelto poniendo una capa que cachee dichas consultas para poder darle más performance a la infraestructura. Para esto, la aplicación que se pensó a futuro implementar es redis. Se va a evaluar también si se instala de forma local en los servidores, o si se utiliza el servicio de "ElastiCache" que ofrece AWS.

7.3 Escenarios donde la infraestructura puede estar comprometida

7.3.1 Escenario 1 – Crecimiento del tráfico

Se va a partir de la base que el crecimiento del tráfico se debe a un aumento real de la cantidad de visitas en el sitio, y no debido a un ataque, caso que será analizado más adelante.

Este escenario es uno de los que la creación de la infraestructura ya prevé, dado que con el diseño modular implementado se puede alcanzar una solución de manera rápida y sencilla. El crecimiento necesario para poder sortear este tipo de problemas, se puede lograr de manera horizontal o vertical. Esto es, crecer en cantidad de instancias que se encargan de servir el contenido, ó crecer en capacidad de cómputo de las instancias ya creadas.

Para tomar esa decisión, es importante poder tener en cuenta si dicho aumento de tráfico es previsto con tiempo, o si se tiene que salir a solucionar el problema al instante. Por ejemplo, si se realiza una fuerte campaña de marketing y estimamos que el crecimiento del sitio va a subir, se puede planificar junto con el cliente un aumento de la infraestructura. En este caso, se puede lograr un crecimiento horizontal, dado que tenemos el tiempo para poder crear instancias nuevas, configurarlas y disponer de las mismas para sumarlas al balanceador al momento que la campaña comience.

En el caso en el que el problema se presente y no se haya previsto (puede pasar por ejemplo, por un cambio de código que mejora el rating del sitio sobre google), no se requiere de mucho tiempo para poder realizar la creación de una nueva instancia y poder crecer horizontalmente por lo que se deberá crecer de forma vertical, dándole más capacidad de cómputo a las instancias que ya tenemos creadas. Este crecimiento se puede realizar sin tener una caída en el sitio, dado que lo que se hace es sacar un servidor del balanceador, apagarlo, cambiar la capacidad de cómputo, prenderlo, ponerlo nuevamente en el balanceador y repetir lo mismo para el segundo. En estos simples pasos, se puede hacer crecer la infraestructura de manera rápida, sin tiempo de caída y permitiendo tener en muy poco tiempo más capacidad.

En el momento en que el tráfico vuelva a los niveles normales, se planifica la reducción de la infraestructura, para no tener mucha capacidad ociosa generando costo extra.

7.3.2 Escenario 2 – Caída de un servidor web

Puede suceder que un servidor web se vea afectado por una caída del servicio de nuestro proveedor de infraestructura. Para esto es que se pensó previamente en tener los fronts en diferentes Zonas de Disponibilidad, tal que permitieran que si un servidor está comprometido, se tendrá el otro disponible, virtualizado en otro lugar físico.

El sitio no va a sufrir caídas, dado que el balanceador va dirigir el tráfico al servidor que vea saludable, pero la infraestructura no será tolerante a fallas luego del incidente. Para estos casos, se puede prever tener una instancia en una tercera zona de disponibilidad, pero apagada. Esto no genera mayores costos extras en el cálculo al final del mes, dado que solo se pagaría por los discos asociados a dicha instancia. Al momento de tener este problema, se prende dicha instancia, se actualizan los datos de la aplicación y se la agrega al balanceador para que comience a recibir tráfico de manera normal.

Al momento de recuperar la instancia comprometida, se procederá a apagarla, y está quedará disponible en caso de que alguna vez pueda suceder lo mismo. Si la instancia no se puede recuperar más, se creará otra instancia desde una imagen que se tenga guardada previamente, se configurará y se la dejará apagada.

7.3.3 Escenario 3 – Caida del master MySQL

En este escenario, necesariamente vamos a tener un downtime en la aplicación, dado que la misma va a fallar al intentar conectarse con la base de datos. Para poder volver a tener operativo el sitio nuevamente, solo se tienen que realizar una serie de pasos sobre el servidor esclavo para promoverlo como maestro.

En primer lugar, se refrescarán los logs, cerrando y volviéndolos a abrir nuevamente mediante el comando *"FLUSH LOGS;"*; de esa manera, el log binario comienza a escribir en otro archivo diferente y queda preparado para ser promovido como master.

Luego, se para el esclavo y se resetea el maestro, con dos comandos diferentes (ver figura 7.3.3.1). Esto se realiza para poder indicarle al servidor de mysql el nuevo host del master, que en este caso va a ser la ip privada de su servidor (ver figura 7.3.3.2).

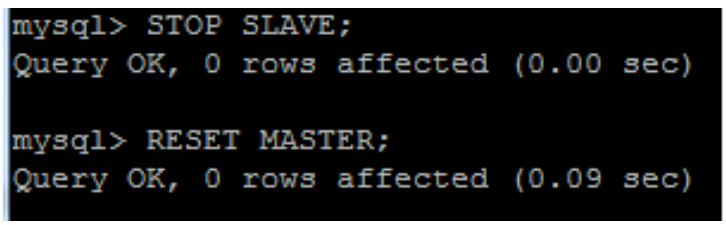

Figura 7.3.3.1 – Parar slave y resetear master.

```
mysql> CHANGE MASTER TO MASTER_HOST='172.31.21.15';
Query OK, 0 rows affected (0.02 sec)
```

Figura 7.3.3.2 – cambiar la ip del master.

Luego de estos pasos, el servidor esclavo ya se encuentra promovido como maestro. Para poder configurar las aplicaciones a la nueva base de datos, se cambiará la ip del dominio "back1-aws-master.com" que se configuró en el archivo */etc/hosts* de cada servidor web, especificado en la sección 4.4 del presente trabajo.

En pocos minutos, podemos tener operativo nuevamente el sitio gracias a la replicación de mysql creada. Queda pendiente reconfigurar nuevamente la base de datos maestro/esclavo, dado que debido a esta caída no se tiene más redundancia en este servicio crítico de la infraestructura.

7.3.4 Escenario 4 – Ataque DOS sobre la infraestructura

Los ataques de denegación de servicio (DOS, por sus siglas en ingles), son uno de los problemas con los que se enfrenta día a día el mundo TI. La naturaleza de este ataque es simular un incremento muy grande de tráfico sobre un sitio en particular, generando la inaccesibilidad del sitio para los usuarios legítimos del mismo.

Dicho problema no se afronta desde la infraestructura, sino que se debe ingresar a los servidores y en tiempo real poder detectar quién está causando dicho ataque y frenar el mismo.

La manera de hacerlo, es visualizando los logs de apache. Allí se va a poder ver cuál es la/s IP/s que están realizando dicho ataque. Es sencillo visualizarlo, dado que, al ser tráfico no real, se va a ver siempre la misma URL siendo accedida por la misma ip, tal como se muestra en la figura 7.3.4.1.

Al identificar la ip que está causando el ataque, se bloquea la misma desde el firewall del servidor (ver figura 7.3.4.2).

```
208.79.239.238 - - [11/Mar/2016:16:44:12 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:14 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:16 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:18 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:20 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:22 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:24 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:26 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:28 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:30 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:32 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:34 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:36 -0300] "POST /xmlrpc.php HTTP/1.0" 200 403 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:36 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:38 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:40 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:42 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
208.79.239.238 - - [11/Mar/2016:16:44:44 -0300] "POST /xmlrpc.php HTTP/1.0" 200 55483 "-" "-"
```

Figura 7.3.4.1 – Ejemplo de Ataque DOS sobre un Wordpress.

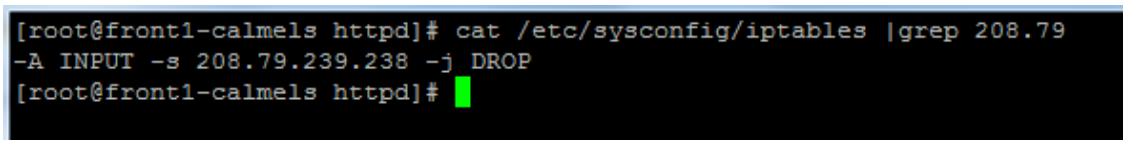

Figura 7.3.4.2 – IP Bloqueada en el Firewall del serviddor.

De esta manera, se puede bloquear rápidamente un ataque de Denegación de Servicio. Para los ataques de denegación desde múltiples IPs (DDOS por sus siglas en ingles), se puede tardar un poco más en identificar todas las IP responsables del ataque, pero se procede de la misma manera.

7.3.5 Escenario 5 – Caída del Balanceador

Como al inicio del presente trabajo se nombró al balanceador como uno de los servicios críticos que se debían tener en cuenta para garantizar la Alta Disponibilidad del sitio, se debe considerar en este capítulo como uno de los escenarios posibles. Sin embargo, dicha tarea fue delegada a AWS, dado que al utilizarse el servicio ELB, también se delega la responsabilidad de que el mismo esté garantizado todo el tiempo.

AWS asegura la disponibilidad de este servicio, teniendo dos balanceadores detrás del ELB. Esto es fácilmente comprobable, dado que si ejecutamos el comando "host" o "dig" (comandos de Linux) para saber la ip a la que responde el "DNS Name" del balanceador, se verán dos ips diferentes respondiendo a dicho balancer. Otra forma de comprobarlo es viendo en los logs de apache de las instancias el chequeo que realiza sobre las mismas para saber si están en buen estado. Como se puede ver en la figura 3.7.4, son dos IP privadas diferentes la que realizan el chequeo sobre las instancias dedicadas a servir el contenido web.

De esta manera, AWS hace totalmente transparente a su cliente la disponibilidad de este servicio, dado que puede cambiar uno de los servidores que está detrás el ELB, y no va a afectar al comportamiento del sitio de ninguna manera.

7.3.6 Escenario 6 – Hackeo sobre el sitio Web

Más allá de haber configurado la infraestructura con todos los recaudos necesarios en cuanto a seguridad para no tener intrusiones de ningún tipo en la misma, el riesgo de que pueda ser comprometida siempre existe.

Hay muchas técnicas hoy en día de hackeo de páginas web, como Ataque por fuerza bruta, Mysql Injection, Spoofing, o la propia Ingeniería social, encargada de "hackear" a las propias personas para robar datos sensibles que permitan ingresar de manera ilegal a una infraestructura de una empresa.

Para poder tener el sitio operativo nuevamente después de una intrusión, se puede utilizar el servicio de backups que Nubity ofrece, restaurando una versión anterior que se tenga guardada del sitio, y así tenerlo operativo nuevamente.

El próximo paso, consiste en trabajar con el cliente para poder encontrar el origen de dicha falla de seguridad para poder solucionarla.

CONCLUSIONES

Como conclusión del presente Trabajo Final, podemos afirmar que en el desarrollo del mismo, se pudo llevar a cabo todos los puntos propuestos al inicio del presente informe.

Se ejecutó una solución de infraestructura que pudiera contar con un servicio disponible en todo momento, el cual pueda responder de manera rápida y eficaz a las demandas que el sitio pueda llegar a tener en cualquier momento.

La modularidad que se logró, hace que se pueda escalar tanto horizontal como verticalmente de manera sencilla y rápida, solucionando así cualquier cuello de botella que se pueda tener en un futuro en cualquiera de las partes que hacen a toda la infraestructura.

Por otro lado, la elección del proveedor garantizó que el alta y configuración se haya realizado rápida y fácilmente. En un futuro en el cual se tenga que planificar un crecimiento debido al incremento constante del tráfico sobre el sitio, se puede tener en cuenta los demás servicios de AWS que ahora se descartaron, pero que en algún momento pueden servir para cubrir alguna necesidad que se pueda llegar a tener.

Se pudo dar a conocer cómo una empresa como Nubity, puede ayudar a realizar el diseño, alta y configuración de una infraestructura con las características a las planteadas en el presente trabajo. También, se pudo detallar de qué manera dicha empresa realiza el monitoreo y respaldo de los servidores, y cómo actúa al momento de cualquier incidente que pueda llegar a haber para solucionarlo de la manera más rápida posible.

El diseño, alta y configuración que se realizó, se pudo lograr sin mayores complicaciones o contratiempos, dado que dichas tareas son las que se realizan de manera cotidiana dentro de la empresa donde se llevó a cabo la práctica. Esto se debe, a que dicha empresa cuenta con años de experiencia en el armado de infraestructuras de similares características.

Por último, se puede destacar que, en el desarrollo del armado de la infraestructura, se notaron todas las ventajas que la nube nos brinda al momento del diseño de una solución TI, y la rapidez de adaptación que se puede tener conforme la demanda vaya aumentando o disminuyendo, garantizando de esta manera ser mucho más preciso en la relación capacidad adquirida y carga real del sitio.

BIBLIOGRAFÍA

[1] “La nube: oportunidades y retos para los integrantes de la cadena de valor”, Managment Solutions, http://www.managementsolutions.com/PDF/ESP/La-nube.pdf - Año 2012.

[2] “COMPUTACIÓN EN LA NUBE”, Parlamento Europeo, Anna Fielder y Ian Brown, http://www.europarl.europa.eu/RegData/etudes/etudes/join/2012/475104/IPOL-IMCO_ET(2012)475104_ES.pdf – Mayo de 2012.

[3] “Plataforma Como Servicio”, Documento Técnico de Intel IT Center, http://dialogoti.intel.com/sites/default/files/documents/10110442_overcomingbarriers_whitepaper_v2f_dwc.pdf - Agosto de 2013.

[4] Documentación oficial de AWS sobre Route 53, http://docs.aws.amazon.com/es_es/Route53/latest/DeveloperGuide/Welcome.html

[5] Documentación oficial de AWS sobre EC2, http://aws.amazon.com/es/documentation/ec2/

[6] Documentación oficial sobre APR, https://apr.apache.org/

[7] Sitio Oficial de Percona, https://www.percona.com/

[8] Documentación oficial sobre Varnish, https://www.varnish-cache.org/docs/4.0/

[9] Documentación Oficial sobre ELB, https://aws.amazon.com/es/elasticloadbalancing/

[10] Sitio Oficial de siege, https://www.joedog.org/siege-home/