Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Despliegue y plataformado en entornos Cloud

Autor: Alberto Rodríguez de la Cruz
Tutor: Pablo Nebrera Herrera

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Despliegue y plataformado en entornos Cloud

Autor:
Alberto Rodríguez de la Cruz

Tutor:
Pablo Nebrera Herrera
Profesor adjunto

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Proyecto Fin de Carrera: Despliegue y plataformado en entornos Cloud

Autor: Alberto Rodríguez de la Cruz

Tutor: Pablo Nebrera Herrera

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Con este proyecto cierro mis estudios como Graduado en Ingeniería de las Tecnologías de Telecomunicación, que me han hecho aprender muchísimo y crecer como persona. Gran parte del mérito de haberlos realizado no es mío, sino que pertenece a muchas personas que han estado a mi alrededor apoyándome. Por ello, quisiera dedicar mi agradecimiento…

A mis padres, por su apoyo incondicional y sus ganas de ayudarme en todo lo necesario, facilitándome siempre cualquier cosa que pudiera hacerme falta.

A mi hermano Pablo, por su entrenamiento constante a base de puyas y piques, que sin duda me ha preparado para superar casi cualquier cosa.

A mis padrinos, por recordarme siempre lo que valgo y animarme en todo momento.

A Irene De Los Santos, por aguantar todas mis quejas en los momentos más agobiantes y tener ese don para hacer que me relaje a su lado.

A Jorge Rodríguez y Minerva Hurtado, por las grandes conversaciones compartidas y sus maravillosos consejos.

A mis amigos "los piltrafillas" y el resto de mis colegas de la escuela, por los grandes ratos que me han hecho pasar tanto dentro como fuera de ésta y por responder a mis dudas siempre que ha sido necesario.

Para terminar quiero agradecer la oportunidad que me ha ofrecido mi tutor Pablo Nebrera, que ha confiado en mí y me ha facilitado los recursos necesarios para la realización del proyecto, así como a todo el equipo de la empresa ENEO Tecnología, que siempre han estado dispuestos a resolverme cualquier duda que pudiera tener.

Resumen

En este proyecto se ha trabajado en la adaptación del sistema redBorder para su ejecución y despliegue en entornos cloud, así como en la creación y configuración de la plataforma para su correcto funcionamiento.

Comienza sentando las bases para poder desplegar el sistema en cualquier entorno de Intraestructura como servicio, para centrarse posteriormente en el despliegue en Amazon Web Services, así como en la integración con gran parte de los servicios ofrecidos por este proveedor.

Con esto se ha logrado posibilitar el uso de los entornos Cloud en un sistema que estaba preparado para el funcionamiento en entornos tradicionales, aprovechando las ventajas que este tipo de entornos nos ofrece.

Abstract

In this Project, we have worked in redBorder system adaptation for its execution over cloud environments, creating and configuring cloud platform to ensure that system works properly.

It starts making the basics for system deployment over any infrastructure as a service environments, focusing then in Amazon Web Services deployment and in service integration with this cloud provider.

We have reached the objetive of execute the system in cloud environemts when it only had been prepared to work in traditional compute environments, getting advantages from cloud environments.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Every problem has a solution. You just have to be creative enough to find it.

Travis Kalanick, co-founder of Uber

Cada vez más empresas de servicios de software utilizan los entornos de Cloud Computing para desplegar sus sistemas, aprovechando las ventajas que nos ofrecen este tipo de entornos: alta escalabilidad, fiabilidad y capacidad, acompañado de la posibilidad de rápido despliegue y crecimiento.

Además, ofrecen la posibilidad de automatizar prácticamente todos los procesos de creación y mantenimiento de la infraestructura, lo que está haciendo que pequeños equipos con pocos recursos puedan crecer y llegar a convertirse en grandes compañías.

Estas posibilidades son las que han motivado el desarrollo de este proyecto, que tiene la intención de adaptar el sistema de seguridad y monitorización redBorder para aprovechar las características del Cloud Computing, y así disponer de los recursos necesarios para su crecimiento e implantación a nivel mundial.

1.1 Situación Actual

El proyecto parte con el sistema redBorder implementado y probado en entornos de computación tradicional, estando preparado para ser escalable y de alta disponibilidad. Sin embargo, no está aún preparado para ser ejecutado en la nube, ya que debe implementar mecanismos que permitan su correcto funcionamiento en este tipo de entornos.

1.2 Objetivos y alcance

El objetivo del proyecto es preparar entornos cloud y adaptar redBorder para su despliegue en dichos entornos, y así aprovechar sus ventajas y los servicios ofrecidos. Se pretende poder implementar el sistema en la cloud de cualquier proveedor, pero prestando especial atención a Amazon Web Services, ya que es donde se realizará la primera puesta en producción del sistema en la nube.

Cabe tener en cuenta que en este proyecto se ha trabajado en preparar el entorno cloud y otorgar al sistema de las herramientas y el acceso necesarios para aprovechar los servicios prestados por los proveedores de la infraestructura. Los detalles del uso e implementación de los sistemas de redBorder para la utilización de estos servicios no es objetivo del proyecto, aunque en algunos casos se ha tenido que colaborar con los desarrolladores para lograr la integración.

2 CLOUD COMPUTING

If someone asks me what cloud computing is, I try not to get bogged down with definitions. I tell them that, simply put, cloud computing is a better way to run your business.

Marc Benioff, CEO of Salesforce.com

La computación en la nube es un área que está cobrando gran importancia en el mundo de la informática y las telecomunicaciones. Continuamente vemos como las aplicaciones tanto domésticas como empresariales van teniendo una versión en la nube, convirtiéndose en un servicio que se ofrece desde ésta.

Dado que la finalidad del proyecto es desplegar el sistema redBorder en la nube, comenzaremos haciendo una introducción al Cloud Computing para así aclarar el concepto y entender qué es exactamente.

2.1 ¿Qué es Cloud Computing?

Para explicar qué es la Cloud, nos vamos a basar en la definición que da el NIST (National Institute of Standards and Technology) de EEUU [1].

Cloud Computing es un paradigma de computación que permite el acceso a través de la red a un conjunto de recursos computacionales (redes, servidores, aplicaciones, almacenamiento y servicios) bajo demanda, sin importar desde donde se soliciten esos recursos, siendo éstos aprovisionados y desplegados con un esfuerzo en gestión e interacción con el proveedor mínimos.

La nube permite acceder a recursos computacionales sin necesidad de trabajar con equipos físicos, haciendo que se puedan manejar infraestructuras y servicios de forma remota y mediante el uso exclusivo de software.

2.2 Tipos de Cloud

Los entornos cloud se suelen clasificar de dos formas: atendiendo al tipo de servicio ofrecido y en función de qué clientes pueden usarlos.

2.2.1 Tipos de Cloud en función del servicio ofrecido

Dependiendo del servicio ofrecido, se pueden clasificar los entornos cloud de la siguiente manera.

- **IaaS**: Infraestructure as a Service (Infraestructura como servicio).
- **PaaS**: Platform as a Service (Plataforma como servicio).
- **SaaS**: Software as a Service (Software como servicio).

2.2.1.1 IaaS (Infraestructura como servicio)

Se permite al cliente provisionar capacidad de procesamiento, almacenamiento, de red y otros recursos computacionales de forma que pueda desplegar y ejecutar software arbitrario, incluyendo sistemas operativos y aplicaciones. El cliente no tiene control físico de la infraestructura cloud que hay por debajo, pero puede trabajar sobre los recursos aprovisionados y ejecutar software sobre ellos.

Un ejemplo muy utilizado de cloud IaaS es Amazon EC2 (Elastic Compute Cloud), que permite desplegar instancias[1] a partir de una imagen (AMI[2]) que contiene el sistema operativo y otro software, en una máquina virtual con una serie de recursos asociados en la nube, permitiendo además configurar aspectos de red y realizar integración con otros servicios de la cloud. También se ofrece este servicio en otras nubes como Google Cloud Platform, Microsoft Azure, y las basadas en OpenStack, entre otras.

2.2.1.2 PaaS (Plataforma como servicio)

Se permite al cliente desplegar contenido software creado por él mismo, sin necesidad de gestionar servicios básicos como los del sistema operativo, ni el hardware (tanto físico como virtual) que está debajo, lo que provee de un mayor nivel de abstracción que en IaaS, permitiendo al cliente centrarse sólo y exclusivamente en su aplicación.

Muchos proveedores de IaaS también proveen servicios de PaaS como por ejemplo Amazon Web Services, que ofrece el servicio BeanStalk[3], que permite el despliegue de aplicaciones web sin necesidad de preocuparse de los recursos hardware necesarios para escalarlos, ni de mantener ni configurar los servidores web. Otros ejemplos de PaaS son Google App Engine[4] u OpenShift Origin[5].

2.2.1.3 SaaS (Software como servicio)

El cliente utiliza una aplicación desplegada en la nube por el proveedor, siendo accedida mediante una interfaz de usuario, sin que se realice el procesamiento principal de aplicación en los dispositivos del cliente. Además, el cliente no controla ni gestiona la infraestructura cloud, ni siquiera el software de la propia aplicación salvo algunos parámetros de configuración. Normalmente, los proveedores de estos servicios utilizan clouds de tipo IaaS o PaaS para ofrecerlos.

Existen muchísimos ejemplos de SaaS en internet, ya que está orientada a los usuarios finales.Ejemplos de SaaS son Google Docs, Dropbox, iCloud, YouTube, etc.

2.2.2 Tipos de Cloud en función de los clientes que pueden utilizarla

Dependiendo de qué clientes pueden utilizar el servicio, se puede clasificar de esta otra forma:

- Públicas
- Privadas
- Híbridas

2.2.2.1 Públicas

Los servicios son ofrecidos por proveedores que gestionan sus infraestructuras y permiten a los clientes utilizarlas, ya sea forma gratuita o mediante pago. Los clientes no conocen los detalles de su implementación, y no se hacen cargo del mantenimiento de los sistemas, por lo que se centran sólo en el software desplegado.

El uso de este tipo de nubes tiene la ventaja de que no es necesaria una inversión inicial en infraestructura y que tiene gran capacidad de escalado. Como inconvenientes destacamos la falta de control sobre los sistemas subyacentes y la dependencia con el proveedor.

---

[1] Instancia: máquina virtual con una serie de recursos asociados que es deplegada en un servicio de IaaS.
[2] AMI (Amazon Machine Image): plantilla que contiene el software a desplegar en instancias de Amazon EC2. Serán explicadas con más profunidad en el capítulo 0.
[3] Más información en la documentación de Amazon Beanstalk [35].
[4] Google App Engine: permite correr aplicaciones sobre la infraestructura de Google sin necesidad de mantener instancias ni servidores. Más información en su documentación [36].
[5] OpensShift Origin: servicio PaaS open source de Red Hat. Más información en su documentación [37].

2.2.2.2 Privadas

La plataforma Cloud es mantenida por los propios usuarios de la misma, sin ofrecer servicios a terceros. Ofrecen más control y seguridad que las nubes públicas, por lo que son utilizadas por organismos donde estas características son cruciales.

Tiene el inconveniente de que es necesaria una gran inversión en infraestructura y mantenimiento, y el escalado es limitado, a cambio de ese plus de control que es tener en tu propio centro de datos toda tu infraestructura.

2.2.2.3 Híbridas

En este tipo de nubes se mezclan elementos de los dos tipos anteriores. Una parte de la infraestructura es mantenida por el cliente, mientras que la otra, por un proveedor, estableciéndose conexiones (normalmente mediante VPN[6]) entre ambas partes. De esta forma se pueden tener los sistemas críticos controlados en la nube privada mientras se aprovecha la escalabilidad de la nube pública.

Como inconveniente hay que señalar que puede ser difícil integrar los dos tipos de infraestructura, y que se debe controlar muy bien el acceso a los recursos de la nube privada, para evitar problemas de seguridad.

2.3 Ventajas y desventajas de los entornos Cloud (Públicos)

En la siguiente tabla se exponen las principales ventajas y desventajas del uso de los entornos basados en Cloud Computing, a fin de realizar una valoración de la decisión de utilizar este tipo de infraestructura.

Tabla 1 Ventajas y desventajas de los entornos Cloud

Ventajas	**Desventajas**
Inmediatez de despliegue	Posibles problemas de seguridad y confidencialidad, ya que la infraestructura es compartida
Se paga por lo que se usa, sin gastos fijos en infraestructura y con el beneficio de la economía de escala masiva (gran cantidad de clientes usando la infraestructura)	Pérdida de control (dependencia de proveedor, debe ser de confianza, ya que tiene acceso a los recursos físicos)
Gran escalabilidad	Los hosts son virtuales, cuyo rendimiento siempre es algo menor que el de los hosts físicos.
Evita el mantenimiento y la operación de la infraestructura	Disponibilidad (dependemos de la disponibilidad de la nube)
No hay necesidad de hacer predicciones precisas de capacidad (elasticidad)	No recomendado para servicios gubernamentales o críticos (como de salud o bancarios)
Posibilidad de despliegue en múltiples regiones	Acceso exclusivamente a través de Internet

Cómo observamos en la tabla, el principal inconveniente es la pérdida de control sobre recursos y la necesidad de confiar en el proveedor. Sin embargo, en una gran cantidad de aplicaciones (entre ellas la que se trata en este proyecto), si se emplean proveedores que ofrezcan garantías en su servicio, la Cloud ofrece una infraestructura de gran calidad y escalabilidad, sobre todo para las empresas que no tengan medios para

[6] VPN: Virtual Private Network. Tecnología que permite establecer conexiones entre redes privadas de forma segura a través de una red pública, como por ejemplo Internet.

mantener su propio centro de datos.

2.4 Buenas prácticas en entornos Cloud

Los entornos basados en Cloud Computing tienen como característica la posibilidad de la automatización de prácticamente todos los procedimientos de despliegue y mantenimiento de un servicio, y la capacidad de escalar y repartir la infraestructura por diversas zonas de disponibilidad. Es importante por lo tanto, aprovechar estas ventajas mediante el seguimiento de una serie de buenas prácticas:

- **Diseñar el sistema para que sea tolerante a fallos**: si se diseña asumiendo que se van a producir fallos y desastres tanto de hardware como de software, el sistema será mucho más resistente a éstos. La gran capacidad y la dispersión geográfica de la nube nos ayuda a aumentar esta tolerancia.
- **Desacoplar componentes**: la división del sistema en componentes independientes hace que resulte mucho más fácil escalarlos y aumentar la capacidad en caso de cuellos de botella en un componente concreto.
- **Automatizar la infraestructura**: es fundamental tratar de automatizar el despliegue y el escalado de la infraestructura, ya que nos evita errores humanos de despliegue y configuración y permite crear y preparar entornos de forma rápida y sencilla.
- **Paralelización de procesos**: para conseguir escalabilidad es fundamental tratar de conseguir concurrencia y paralelización a la hora de realizar operaciones, de forma que se aproveche la gran capacidad de escalado de la nube.
- **Mantener la información dinámica cerca de las instancias y la estática cerca de los usuarios**: dado que el acceso a la nube se realiza a través de internet, es importante que la información que necesita ser procesada continuamente se mantenga lo más cerca posible de las instancias (es decir, dentro de la nube), para reducir costes y latencias. Sin embargo, la información estática que no necesita ser procesada pero sí debe ser servida a los clientes es conveniente tenerla lo más cerca posible de éstos, lo que reduce la latencia y aumenta el rendimiento.
- **Reducir el acceso a recursos de la nube desde internet lo máximo posible**: por seguridad, desde internet sólo se debe permitir el acceso a los servicios de front-end, estando el back-end accesible sólo desde las propias instancias de la nube.
- **Integración continua**: es importante tratar de aprovechar las facilidades de despliegue en la nube para mantener el sistema actualizado y poder reparar fallos y problemas rápidamente.

2.5 Principales proveedores

A continuación hacemos referencia a los 5 mayores proveedores de clouds públicas, y a su cuota de mercado en el año 2014:

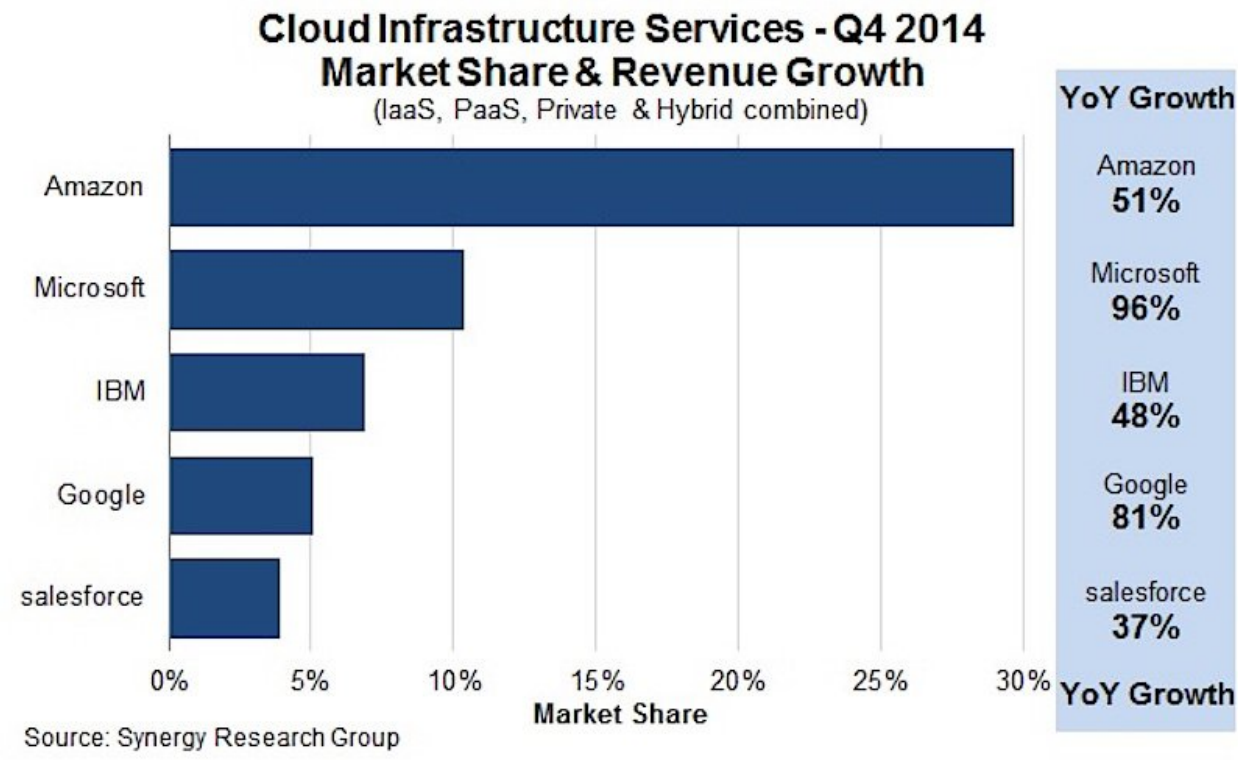

Figura 1 Cuota de mercado de los principales proveedores de servicios de Cloud Computing [2].

Como se observa en la figura, el principal proveedor y el que más servicios ofrece es Amazon Web Services, seguido de Microsoft Azure.

Además, aunque no aparecen en la figura, existen muchos proveedores que utilizan clouds basadas en OpenStack [3], que es un sistema de código abierto que permite implementar un servicio de IaaS en cualquier centro de datos.

En redBorder, se ha trabajado sobre todo con Amazon Web Services, debido a la gran variedad de servicios que ofrece, su gran fiabilidad y su fuerte presencia en el mercado. También se ha trabajado con nubes basadas en OpenStack, las cuales no serán tratadas en este proyecto.

3 INTRODUCCIÓN A REDBORDER

The people who are crazy enough to think they can change the world are the ones who do.

Steve Jobs, CEO of Apple.

En este capítulo se hará una introducción al sistema que será portado, adaptado y desplegado en entornos Cloud, a fin de entender qué características de estos entornos serán necesarias para su funcionamiento.

3.1 ¿Qué es redBorder?

redBorder es un sistema de seguridad y monitorización de redes en tiempo real y altamente escalable que permite disponer de una gran visibilidad de todo lo que ocurre en las redes monitorizadas, procurando un gran control de la infraestructura y de los servicios ofrecidos en ésta.

Este sistema ofrece varios módulos de servicios. Los clientes pueden contratar los módulos que deseen, dando cada uno información diferente sobre la red:

- **redBorder IPS**: implementa un sistema de prevención de intrusos (IPS) que se instala en la red del cliente, pudiéndose obtener información organizada y filtrada de los eventos de seguridad generados desde la web del sistema.
- **redBorder Flow**: monitoriza sistemas de red compatibles con Netflow[7] y SNMP[8], permitiéndonos ver en la web información sobre el tráfico de la red y sobre los clientes que la utilizan.
- **redBorder Malware**: analiza los datos que pasan por la red para detectar y bloquear malware, a fin de evitar que actúe y se extienda por la red.
- **redBorder Social**: obtiene información social sobre los usuarios que utilizan la red para realizar análisis de sentimiento[9].

Además, redBorder nos ofrece la posibilidad de crear **Dashboards**, es decir, pantallas personalizadas donde se muestra la información que el usuario decida y donde tenemos una paronámica general de lo que ocurre en la red.

3.2 Arquitectura general

Dado que el objetivo de redBorder es monitorizar y controlar redes, es necesario que parte del software corra en equipos dentro de la red a monitorizar, y que tenga acceso directo al tráfico y a equipos de la red. Por ello

[7] Netflow: protocolo de red desarrollado por Cisco para recolectar información sobre tráfico IP.
[8] SNMP: Simple Network Management Protocol: protocolo para el intercambio de información de gestión y administración de equipos de red.
[9] Análisis de sentimiento: proceso de determinación de si la interacción en redes sociales contiene una opinión general positiva o negativa sobre una entidad o concepto [38].

tenemos dos tipos fundamentales de equipos en el sistema:

- **Sensores**: son los equipos que se encuentran en la red del cliente. Se encargan de obtener información de la red, y en algunos casos, de actuar sobre ella de distintas formas. Envían la información a los Managers, que son explicados a continuación.
- **Managers**: se encargan de procesar la información obtenida desde los sensores, analizándola y presentándola mediante una web, de forma que los gestores de la red puedan tener visibilidad de lo que está ocurriendo en la red. Estos equipos no tienen por qué estar en la red del cliente y de hecho son los que se van a desplegar en entornos cloud.

En función de los módulos que se utilicen se deben implementar distintos tipos de sensores, ya que, por ejemplo, no es lo mismo implementar un IPS que necesita una gran capacidad de procesamiento, que un recolector de NetFlow.

En el caso de la nube además existe un nuevo tipo de equipo, llamado **Cloud Proxy**. Es un intermediario que recolecta la información generada por los sensores y la envía a la nube a través de una conexión segura.

En el siguiente esquema podemos ver la arquitectura de una forma más visual:

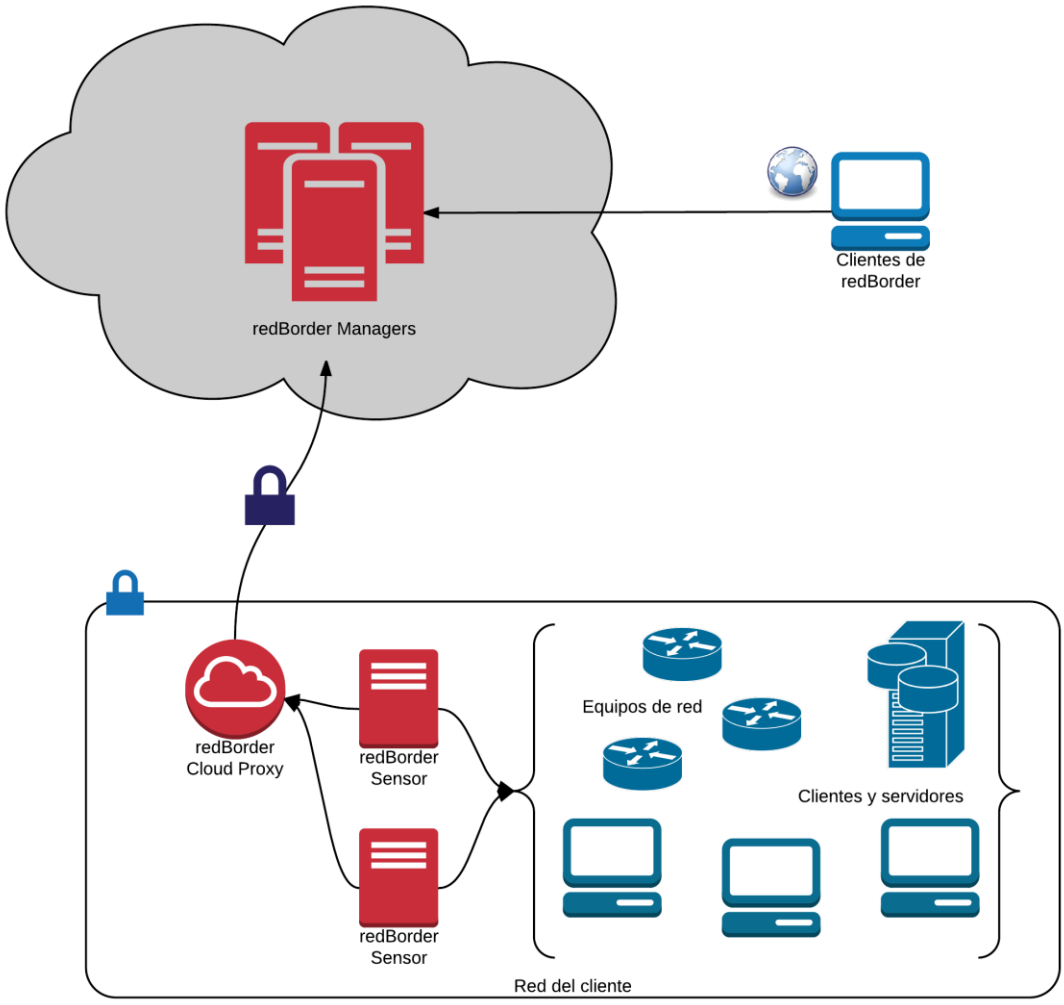

Figura 2 Arquitectura general de redBorder

3.3 Servicios de redBorder Manager

Los managers son los equipos que se desplegarán en la cloud, por lo que vamos a entrar más en profundidad en los servicios que implementan y ofrecen, y en cómo son capaces de procesar los eventos que llegan desde

los sensores.

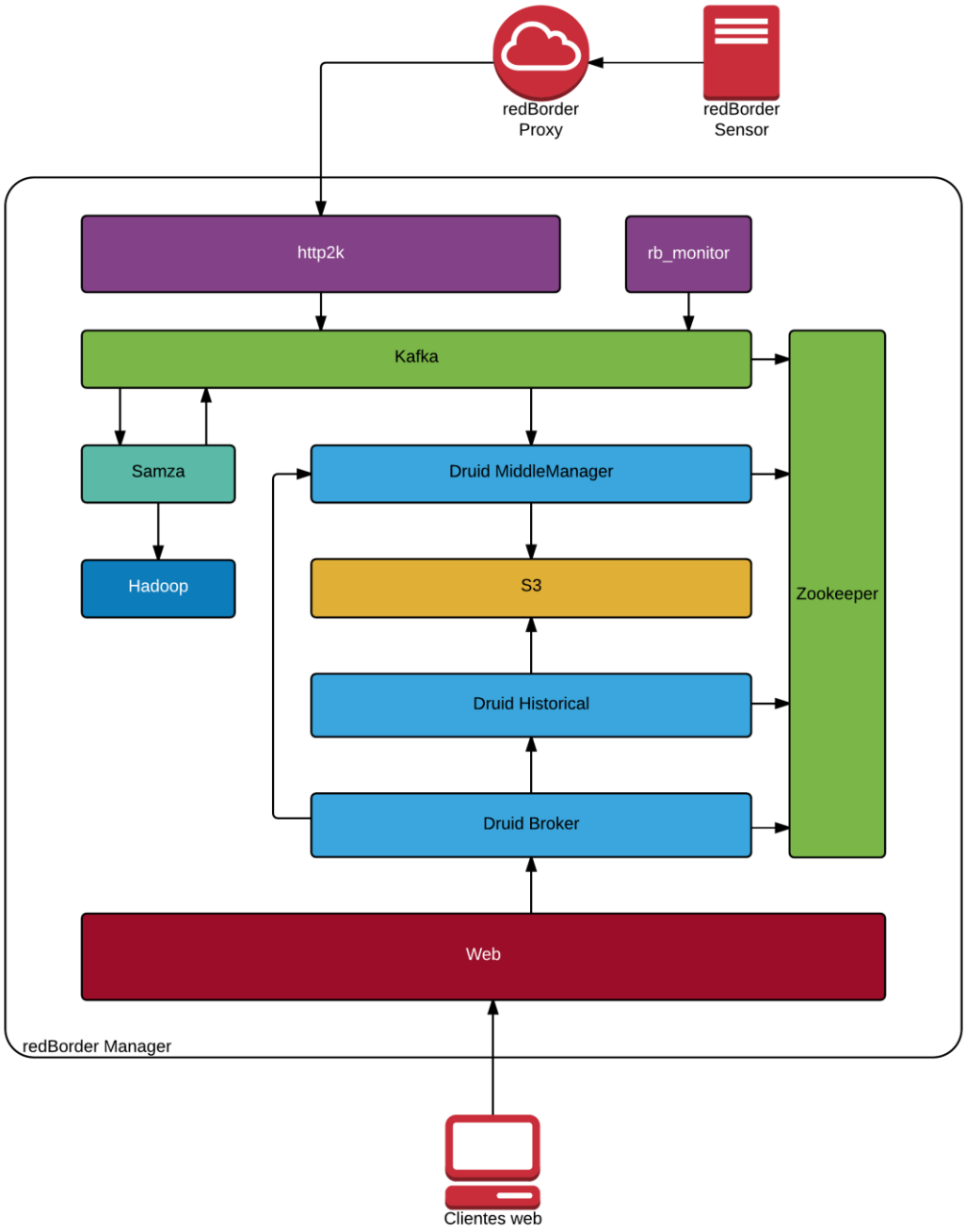

Figura 3 Servicios principales de redBorder

En la figura se observan los servicios principales de redBorder, que son brevemente introducidos a continuación junto con otros que no aparecen en la figura:

- **Http2k**: servicio que recibe información de sensores de redBorder a través de HTTP y la envía a un Topic de Kafka. Se utiliza para crear un túnel HTTPS entre la red del cliente y los managers (que pueden estar por ejemplo en la nube).
- **Zookeper** [4]: sistema que proporciona un servicio de configuración centralizada, registros de nombres y sincronización para grandes sistemas distribuidos, ofreciendo además alta disponibilidad y escalabilidad.
- **Kafka** [5]: servicio distribuido, particionado y con posibilidad de replicación que ofrece la

funcionalidad de un sistema de colas de mensajes con el paradigma publicador / subscritor. Los eventos generados por las distintas aplicaciones de redBorder pasan por este servicio, a fin de desacoplar la obtención de datos de su procesamiento. Utiliza zookeeper para la sincronización de las colas y los mensajes.

- **Rb_monitor**: obtiene métricas del estado de los managers y los sensores asociados al sistema para su monitorización, y las envía a kafka.
- **Druid (general)** [6]: es un sistema de almacenamiento y análisis de infomación que permite la consulta y exploración de grandes cantidades de datos en tiempo real. Está dividido en los distintos componentes que se explican a continuación, y que a su vez utilizan zookeeper y postgreSQL para tareas de sincronización.
- **Druid Coordinator**: gestiona y asigna las tareas de almacenamiento y procesamiento de segmentos de información procesada por Druid, en función de su importancia y cronología.
- **Druid Overlord**: asigna y reparte tareas de procesamiento en tiempo real a los MiddleManagers, utilizando Zookeeper como medio de sincronización.
- **Druid MiddleManager**: procesa tareas de indexado y procesado de datos en tiempo real, asignadas por el Overlord.
- **Druid Historical**: cuando finaliza el periodo de tiempo real, la información se almacena en un sistema de almacenamiento profundo (como S3). Los Historicals cargan esa información para hacerla disponible a las aplicaciones cliente, realizando además las operaciones oportunas sobre los datos.
- **Druid Broker**: proporciona una API de acceso a los datos y se encarga de localizar y servir la información que se encuentra repartida por todo el clúster de Druid.
- **Samza / rb-enrich** [7]: obtiene un flujo de datos de Kafka y realizan un proceso de enriquecimiento[10] de éstos, para devolverlos a otra cola de Kafka ya enriquecidos.
- **Hadoop** [8]: sistema de procesamiento distribuido, escalable y de alta disponibilidad, que incluye tanto herramientas para el procesado de información como para su almacenamiento distribuido. Es utilizado por varios servicios de redBorder para ejecutar tareas procesamiento de eventos.
- **S3**: servicio de almacenamiento de objetos fiable y de alta disponibilidad, proveído por la aplicación Riak [9]. Es utilizado sobre todo para el almacenamiento profundo en Druid y para el almacenamiento de plantillas de Chef.
- **PostgreSQL** [10]: base de datos relacional, utilizado para el almacenamiento de metadados de la web y de otros servicios.
- **Memcached** [11] : servicio de caché distribuido, que almacena en RAM parejas clave/valor para su rápida disponibilidad por parte de otros servicios y sistemas.
- **Web**: es el Front-end del sistema, donde los usuarios se conectan para ver los datos recolectados por el sistema. Utiliza NGINX[11] para gestionar peticiones y PUMA[12] como servidor web, estando la aplicación desarrollada en Ruby On Rails[13].
- **Chef Server** [12]: almacena y gestiona las plantillas, recetas y cookbooks, de forma que se puede configurar de forma centralizada todo un clúster de redBorder. Se explica más detalladamente en el punto 5.2.1.
- **Chef Client**: configura de forma automática los servicios de cada nodo de redBorder en función de la configuración idicada en el Chef Server.

Cabe destacar que redBorder implementa más servicios de los mencionados aquí, pero se ha optado por

[10] Enriquecimiento: proceso por el cual se añade información útil a un evento a partir de la información original. Por ejemplo, si un evento contiene una IP de origen, se puede añadir un campo de locacalización de esa IP de origen, añadiendo información útil al evento.
[11] NGINX: proxy inverso y de http. Más información en su documentación [41].
[12] PUMA: servidor web optimizado para aplicaciones escritas utilizando Ruby on Rails. Más información en su documentación [42].
[13] Ruby on Rails: framework para la creación de aplicaciones web basada en el lenguaje de programación Ruby. Más información en su documentación [43].

introducir sólo los más relevantes para comprender su funcionamiento. También es importante tener en cuenta que determinados servicios podrían ser externalizados, de forma que no sean gestionados por redBorder, como por ejemplo, el servicio S3 de Amazon.

3.4 Arquitectura propuesta en entornos cloud

Para aprovechar la escalabilidad de la nube, se ha creado un arquitectura basada en roles para los managers de redBorder, de forma que los servicios se reparten entre distintos managers con diferentes roles, que cumplen distintas funciones en el sistema.

La arquitectura queda representada en la siguiente figura:

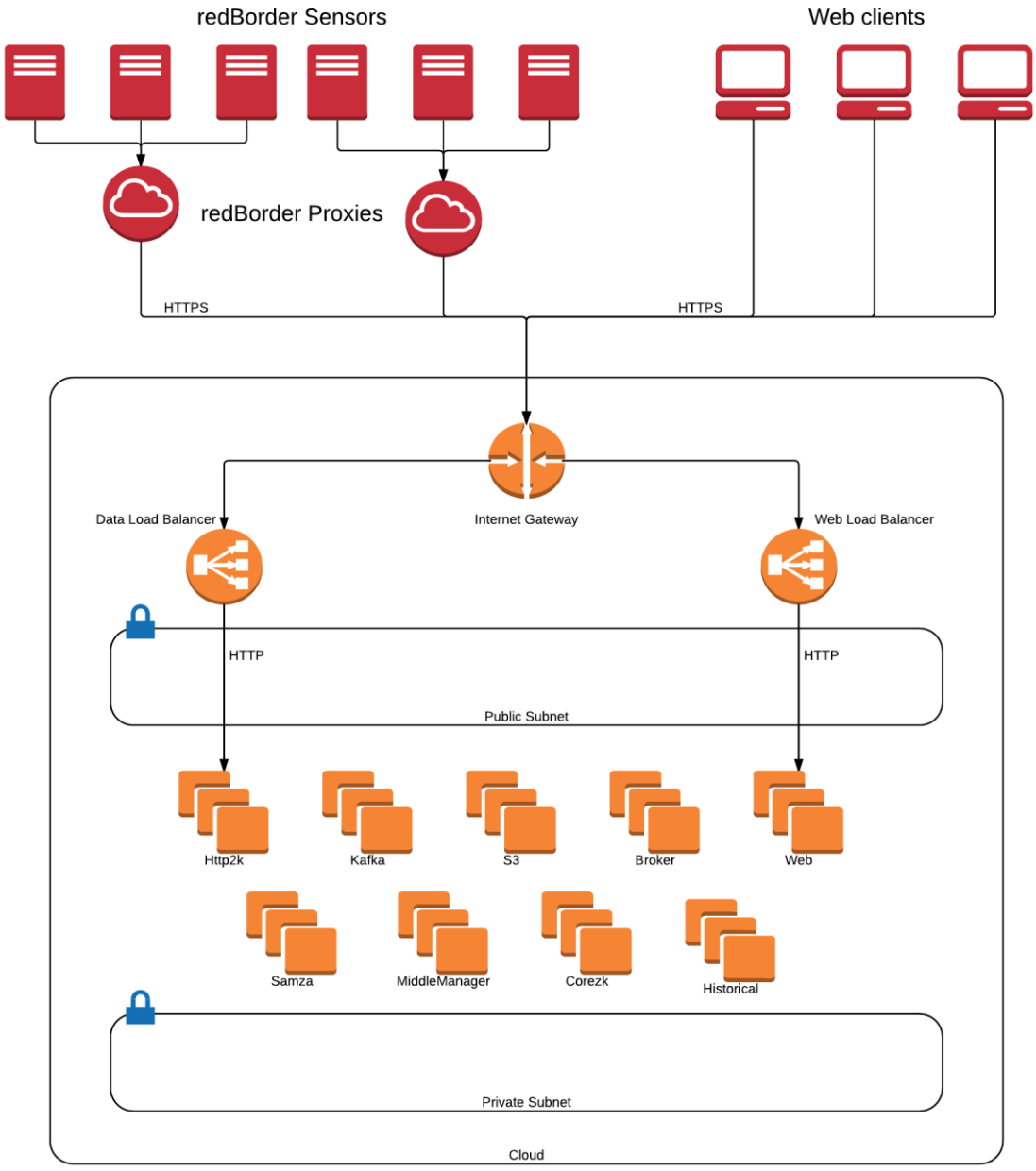

Figura 4 Arquitectura propuesta para entornos cloud

Se deben definir dos subredes en la cloud, una con acceso a internet, por la que llegaran los datos de los sensores y las peticiones de los clientes, y otra que se utiliza para el sincronismo entre los nodos del clúster.

Cabe destacar que la entrada de datos, ya sea por parte de los sensores o de los clientes web, siempre utiliza una comunicación HTTPS segura que es terminada en los balanceadores de carga. A las instancias que procesan las peticioines llega el tráfico ya sin cifrar.

Los roles definidos para las instancias de redBorder son los siguientes:

- **Corezk**: implementa los servicios de gestión y coordinación del clúster. Ejecuta zookeeper, Druid Coordinator, PostgreSQL y Chef Server, entre otros servicios.
- **Kafka**: implementa el clúster de Apache Kafka, que permite la comunicación indirecta entre entre múltiples componentes del sistema.
- **Http2k**: rol dedicado al servicio http2k y que se dedica exclusivamente a recibir información de los clientes y pasarla a kafka. Es un rol que está preparado para escalar rápidamente conforme haya más clientes y eventos.
- **Web**: ofrece el servicio web de redBorder. Al igual que el rol Http2k, también debe estar preparado para escalar rápidamente en función de las peticiones de los clientes.
- **Broker**: implementa el servicio Druid Broker, haciendo de intermediario entre Druid y la web.
- **MiddleManager**: rol dedicado al procesamiento en tiempo real de la información, implementando el servicio Druid Middlemanager.. Son los nodos que necesitan más recursos en cuanto a hardware debido a la gran carga de procesamiento que soportan. Deben estar preparados para escalar rápido si aumenta el número de clientes.
- **Historical**: implementan el servicio Druid Historical. Puede haber dos tipos, Hot y Long Term. Los primeros sirvne información con poca antigüedad que suele ser muy solicitida, mientras que los segundos sirven información más antigua y menos solicitada, por lo que pueden procesar un intervalo de tiempo más amplio.
- **Enrichment / Samza**: implementan el servicio rb-enrich o samza (en función del rol). Ambos servicios se dedican al enriquecimiento de los eventos recibidos.
- **S3**: implementa el servicio de almacenamiento profundo S3. Si es implementado por redBorder, utilizará la aplicación Riak, pero también se podría utilizar un servicio externo, ya que muchos proveedores de entornos cloud ofrecen servicios de S3 gestionados.

Hay que tener en cuenta que en todos los nodos, independientemente de su configuración, debe estar habilitado el servicio Chef Client para su configuración automática.

4 PREPARACIÓN DE LA IMAGEN DE REDBORDER PARA ENTORNOS CLOUD

The most powerful tool we hace as developers is automation.

Scott Hanselman

El sistema redBorder se puede instalar mediante su disco de instalación, que es proveído en forma de imagen ISO. Sin embargo en entornos cloud esta no es una buena manera de desplegar el software debido a la lentitud del proceso de instalación, por lo que se tardaría demasiado tiempo en desplegar nuevas instancias del sistema.

Dado que los entornos cloud se basan en la virtualización, podemos utilizar una instalación en una máquina virtual cualquiera para ejecutarla en las instancias de la cloud.

Además, es fundamental incorporar en la imagen del sistema métodos de automatización de la configuración de las instancias, con el fin de que puedan ser puestas en producción sin intervención manual.

En este capítulo se muestra cómo se prepara una imagen de redBorder para su correcto despliegue y funcionamiento en entornos cloud.

4.1 Creación de la imagen cloud de redBorder

Para desplegar un sistema operativo en la nube lo que se hace es crear una imagen de disco donde se encuentra instalado el sistema operativo y el software básico necesario. Para crearla basta con instalar el sistema en una máquina virtual cualquiera y convertir el disco virtual al formato RAW (o IMG).

Una vez obtenida la imagen del sistema operativo, se puede subir al servicio de cloud computing correspondiente utilizando las herramientas que cada proveedor ofrece. En el caso de Openstack, se puede desplegar directamente la imagen RAW, pero por ejemplo, en Amazon EC2 es necesario utilizar la CLI de AWS (Command Line Tool) para importar la imagen y convertirla a un formato compatible.

En este proyecto se ha trabajado sobre todo con Amazon Web Services por lo que a continuación se explica el procedimiento que se sigue para esta plataforma

4.1.1 Importación de imagen a AWS y creación de la Amazon Machine Image (AMI)

En el caso de AWS, la imagen que contiene el sistema operativo se denomina AMI (Amazon Machine Image). Se puede generar a partir de instancias de Amazon EC2, por lo que el procedimiento para generarla consistirá en importar una máquina virtual para disponer de una copia en Amazon en forma de instancia, y a partir de esta generar la AMI, que podrá ser desplegada en tantas instancias como se desee.

Para ello es necesario utilizar la Command-Line Interface de Amazon EC2 y la CLI de AWS. Son herramientas que por línea de comandos nos permiten realizar operaciones con los recursos de Amazon Web Services[14]. La importación de la máquina virtual se realiza con los siguientes comandos:

- **ec2-import-instance**: crea la tarea de importación de la máquina virtual y sube la imagen del disco

[14] En el anexo II se encuentran las guías de instalación de las herramientas necesarias.

duro en formato RAW a Amazon EC2. Una vez finalizado el proceso de subida, hay que esperar a que Amazon finalice la conversión y genere la instancia.

- **ec2-resume-import**: permite continuar el proceso de subida en caso de que se interrumpa el comando anterior.
- **aws ec2 describe-conversion-tasks**: nos permite ver el estado de la conversión y de la generación de la instancia.
- **aws ec2 create-image**: crea la AMI a partir de la instancia cuando ya ha finalizado el proceso de conversión y generación de ésta.

En el Anexo III se muestran unos scripts que automatizan el proceso de creación de la máquina virtual, subida y creación de la AMI.

4.2 Cloud-Init

Cloud-Init es un paquete software que gestiona la inicialización de instancias en la cloud, realizando operaciones de configuración necesarias para el correcto funcionamiento del sistema en este tipo de entornos. Inicialmente estuvo disponible en las Ubuntu Cloud Images[15], pero actualmente puede instalarse en cualquier distribución de Linux

No es obligatoria la implementación de este paquete en las imágenes cloud, pero sí es necesario implementar de una forma u otra las siguientes operaciones:

- **Soporte para añadir la clave SSH de usuario mediante la interfaz del proveedor Cloud**. Esto es necesario para poder tener acceso seguro a la instancia sin que haya ningún periodo en el que ésta tenga una clave por defecto.
- **Redimensión del sistema de ficheros**. Dado que el sistema ya está instalado en una imagen de disco de un tamaño fijo, pero el disco que después se asigna a la instancia cloud puede tener un tamaño diferente, es necesario que la instancia pueda redimensionar el sistema de ficheros de forma automática, para aprovechar el tamaño completo del disco.
- **Elección del hostname de la instancia**. Dado que inicialmente la imagen es común para muchas instancias en la cloud, es importante poder elegir un hostname que identifique unívocamente a la instancia.
- **Soporte para User-Data**. Es fundamental poder permitir insertar contenido personalizado para cada instancia, con el fin de que esta pueda configurarse al arrancar de distintas formas aunque tengan la misma imagen instalada. Todos los proveedores de IaaS soportan el traspaso de User-Data de una forma u otra a las instancias, por lo que la imagen cloud debe estar preparada para procesar esa información y realizar las operaciones oportunas.

Cloud-Init implementa todas estas características, además de otras muchas, siendo compatible con los principales proveedores de IaaS del mercado (AWS y Openstack, entre otros).

4.2.1 Servicios de Cloud-Init

Cloud-Init provee de 4 servicios diferentes que entran en acción en distintas fases de la configuración de la instancia. Son los siguientes:

- **cloud-init-local**: ejecuta módulos de cloud-init antes de que se configure la red en la instancia.
- **cloud-init**: ejecuta módulos de cloud-init una vez que se ha realizado la configuración de red en la instancia.
- **cloud-config**: ejecuta módulos de cloud-init para la configuración de servicios una vez que se ha finalizado el proceso de arranque del sistema operativo.

[15] Ubuntu Cloud Images: imágenes de la distribución Linux Ubuntu preparada para su despliegue en entornos cloud.

- **cloud-final**: ejecuta módulos de cloud-init una vez que se ha finalizado la configuración de los servicios que provee la instancia.

Esta división de los servicios de Cloud-Init permite elegir qué módulos ejecutar en cada fase de arranque y configuración de la instancia.

4.2.2 Módulos de Cloud-Init

Cloud-Init permite ejecutar módulos, que son scripts que resuelven distintos problemas comunes de configuración de instancias en la cloud. Aquí se hará referencia a algunos de ellos, pero hay muchos más que pueden ser utilizados y cuyo funcionamiento y configuración está recogido en la documentación de cloud-init [13].

Los principales módulos utilizados por redBorder son:

- **runcmd**: para ejecutar un script al arrancar la instancia.
- **final_message**: para poner un mensaje de bienvenida en el arranque.
- **users-groups**: para la creación del usuario para acceder por ssh a la instancia.
- **ssh**: configura el servicio ssh con la clave RSA que pone el proveedor de IaaS en el arranque de la instancia.

Hay muchos otros módulos en el fichero de configuración de Cloud-Init, aunque no vamos a entrar en detalles sobre su funcionamiento.

4.2.3 Configuración de Cloud-Init

Cloud-Init se configura mediante ficheros de configuración ubicados en /etc/cloud-init. En ese directorio se deben situar tanto el fichero de configuración general cloud-init.cfg como el directorio cloud-init.d.

El fichero de configuración de Cloud-Init sigue el formato YAML[16], y se divide en 4 partes que coinciden con los 4 servicios que ofrece. En cada parte se eligen los módulos que serán ejecutados en cada fase del arranque de la instancia, siendo posible además configurar el comportamiento de cada módulo, tanto en el propio fichero cloud-init.cfg como en ficheros dedicados para cada módulo y colocados en el directorio /etc/cloud-init.d (opción recomendada).

En redBorder se utiliza el siguiente fichero de configuración para Cloud-Init:

[16] YAML: formato de serialización de datos fácilmente interpretable por el ser hunano, muy utilizado para ficheros de configuración por su simplicidad. Más información en su web [44].

```
users:
 - default

disable_root: 1
ssh_pwauth:   0

locale_configfile: /etc/sysconfig/i18n
resize_rootfs_tmp: /dev
ssh_deletekeys:   0
ssh_genkeytypes:  ~
syslog_fix_perms: ~

final_message: "Welcome to redBorder Horama Cloud 3.1.50-1"

runcmd:
 - /opt/rb/bin/rb_cloud_init.sh

cloud_init_modules:
 - migrator
 - bootcmd
 - write-files
 - growpart
 - resizefs
 - update_etc_hosts
 - rsyslog
 - users-groups
 - ssh

cloud_config_modules:
 - runcmd
 - mounts
 - locale
 - set-passwords
 - timezone
 - yum-add-repo
 - package-update-upgrade-install
 - chef
 - disable-ec2-metadata

cloud_final_modules:
 - rightscale_userdata
 - scripts-per-once
 - scripts-per-boot
 - scripts-per-instance
 - scripts-user
 - ssh-authkey-fingerprints
 - keys-to-console
 - phone-home
 - final-message

system_info:
  default_user:
    name: ec2
    lock_passwd: true
    gecos: redBorder Cloud User
    groups: [wheel, adm]
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
    shell: /bin/bash
  distro: rhel
  paths:
    cloud_dir: /var/lib/cloud
    templates_dir: /etc/cloud/templates
  ssh_svcname: sshd

datasource:
  Ec2:
    timeout: 10
    max_wait: 30

# vim:syntax=yaml
```

5 DESPLIEGUE DE REDBORDER EN ENTORNOS CLOUD GENÉRICOS

There should be two tasks for human being to perform to deploy software into a development, test, or production environment: to pick the version and environment and to press the "deploy" button."

David Farley, 2010

Una de las principales ventajas de los entornos Cloud es la facilidad de despliegue de aplicaciones y servicios, permitiéndose la automatización de estas operaciones mediante llamadas a la API de los proveedores de IaaS o PaaS.

En este capítulo se explica el procedimiento de despliegue del sistema redBorder utilizando un servicio de IaaS genérico, de tal forma que se consiga disponer en el entorno cloud de un clúster preparado para entrar en producción.

5.1 Claves del despliegue en entornos Cloud

Normalmente los sistemas que se despliegan en la nube suelen estar formados por múltiples instancias que forman clústers y que pueden crecer indefinidamente. Además, se suelen utilizar servicios del proveedor del entorno cloud, por lo que finalmente podemos llegar a necesitar una gran cantidad de recursos, que deben ser gestionados.

Debido a este problema, a la hora de desplegar y operar infraestructuras cloud vamos a centrarnos muchísimo en la **automatización**. Este concepto es el que permite que un gran sistema muy complejo pueda ser gestionado por pocos ingenieros. La nube además nos lo pone fácil porque todas las operaciones pueden hacerse mediante software, lo que posibilita su automatización.

Esta automatización se divide en dos partes fundamentales

- **Automatización de la configuración de los nodos de la cloud**. Es fundamental la autoconfiguración de los distintos servicios que se quieren ejecutar en las instancias desplegadas, a fin de reducir al máximo o incluso eliminar la intervención manual en la puesta en servicio del sistema.
- **Automatización del despliegue de recursos de la cloud**. Se puede automatizar el despliegue de la infraestructura sobre la que se ejecuta el sistema como las instancias, la configuración de redes, sistemas de monitorización, balanceadores de carga, etc.

El primer punto es común sea cual sea la infraestructura cloud que se utiliza, ya que sólo depende del sistema que se está desplegando, pero no del proveedor, salvo que se quieran utilizar servicios específicos de una cloud concreta.

Sin embargo, el segundo es dependiente de cómo estén implementados los servicios de automatización de despliegue o la API que controla los recursos de cada Cloud, por lo que en este capítulo no se explicará cómo automatizar ese despliegue de recursos, sino que se darán instrucciones genéricas sobre cómo realizarlo. Más adelante sí se verá cómo automatizarlo en el caso de que se utilice Amazon Web Services como proveedor.

5.2 Automatización de la configuración de los nodos

Es fundamental automatizar la preparación de cada instancia y la configuración de ellas para que funcionen conjuntamente, ya que debido a la elasticidad de la nube, pueden estar levantándose y apagándose instancias y clústers enteros constantemente. Además, facilita enormemente la labor del despliegue de infraestructuras y si se hace correctamente, puede evitar muchos errores de configuración.

Existen varias formas de automatizar la configuración en instancias, como utilizando scripts, servicios de autoconfiguración como Chef, o incluso utilizando servicios de PaaS, donde no es necesario configurar servicios, sino sólo aplicaciones. En redBorder se ha optado por el uso de Chef, del cuál se hará una pequeña introducción.

5.2.1 Introducción a Chef

Chef es una plataforma de automatización que nos permite manejar una infraestructura como si fuera código, permitiendo configurar, desplegar y gestionar a través de la red todos los elementos de la infraestructura, independientemente de su tamaño.

Para lograrlo, Chef se basa en una descripción del estado deseado de una infraestructura (incluyendo la arquitectura y configuración de elementos), y realiza las operaciones sobre cada módulo de la infraestructura hasta alcanzar dicho estado.

La arquitectura general de Chef se representa en la siguiente figura:

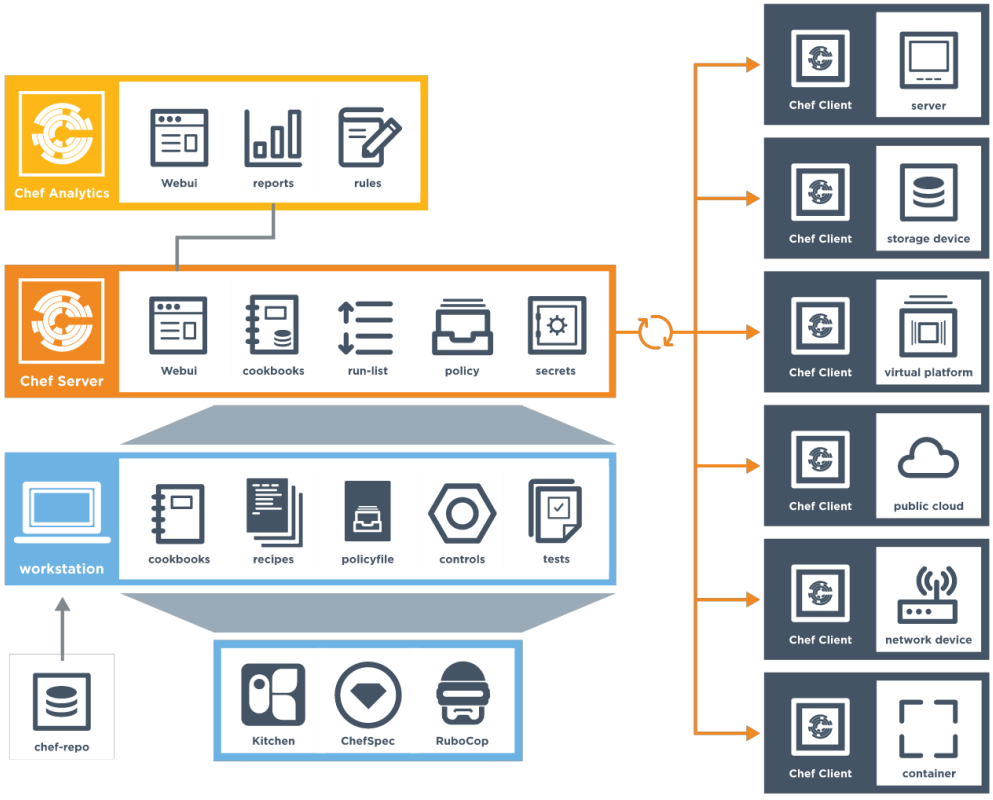

Figura 5 Arquitectura general de Chef

La configuración de la infraestructura se describe en Chef en forma de código en Ruby. Hay varios tipos de documentos que contienen la descripción:

- **Recetas**: contienen declaraciones sobre el estado de elementos de la infraestructura. Por ejemplo, una receta puede indicar que se desea tener instalado un servidor web, que esté habilitado en el arranque del sistema y que utilice la configuración indicada en una plantilla.
- **Plantillas**: son ficheros de configuración a los que se puede añadir código en Ruby, a fin de que el resultado final de la plantilla dependa de ciertos parámetros y sea adaptable.
- **Databags**: son contendores de información necesarios para la configuración de ciertos elementos. Por

ejemplo, pueden contener claves de acceso encriptadas o detalles sobre las opciones disponibles en la configuración de cierto parámetro.

Estos documentos se organizan en una estructura de directorios denominados "cookbooks", que contienen toda la información necesaria para la configuración de un nodo determinado de la infraestructura.

Estos documentos son tratados por distintos servicios de Chef para alcanzar el estado deseado de la arquitectura. Los servicios principales son los siguientes:

- **Chef Client**: actúa sobre los elementos gestionados por Chef y realiza las operaciones necesarias para que el elemento gestionado por este servicio alcance el estado deseado. Puede gestionar todo tipo de recursos, desde servidores y equipos de red hasta aplicaciones y contenedores de éstas.
- **Chef Server**: contiene la información necesaria para la configuración de todo el sistema. Los Chef Clients solicitan la configuración de cada elemento al Chef Server, y en función de las plantillas que éste gestiona, actúan sobre los nodos de una forma u otra.
- **Workstation**: los gestores de la arquitectura no trabajan directamente sobre Chef Server, sino que desarrollan los "cookbooks" en sus equipos de desarrollo y utilizan la herramienta **Knife**[17] para gestionar la configuración almacenada en el Chef Server.

En la figura anterior aparecen más elementos que aportan características adicionales, pero no serán explicadas en esta introducción[18].

Por último, destacar que Chef permite gestionar roles de nodos, y en función del rol que se elija para cada nodo, configurará los servicios que sean necesarios para el desempeño de dicho rol. Cada rol tiene asociado una **run list**, en la que se especifíca que cookbooks debe utilizar para su configuración.

5.2.2 rb_discover

Como se ha visto en el punto anterior, Chef permite la autoconfiguración de nodos en un clúster. Sin embargo, no provee de mecanismos automáticos para detectar nuevos nodos y añadirlos a éste.

Para poder hacer eso, el nuevo nodo debería conocer información sobre el clúster al que quiere unirse, y donde encontrar el servicio Chef Server con las plantillas necesarias. En entornos cloud las instancias al arrancar no conocen esta información por lo que la deben auto-descubrir por otros medios. Para solventar este problema se utiliza el servicio rb_discover.

rb_discover se ejecuta en todos los nodos de redBorder que se despliegan en un clúster en la nube. Para ello, averigua qué equipos están conectados a la subred de sincronismo del clúster, y les pregunta quién es el nodo maestro de éste. De esta forma, puede preguntar a este nodo la información sobre el clúster para que pueda unirse.

En las redes de la mayoría de los proveedores de IaaS no está soportada ni los mensajes a difusión ni en multicast, pero sí está soportado el protocolo ARP. Es por ello que rb-discover lo utiliza para averiguar qué Ips hay activas en la subred y posteriormente preguntarles una a una para obtener información del clúster. Si las Ips a las que pregunta pertenecen a nodos de redBorder con el servicio rb-discover activo, podrá obtener de ellos la información necesaria.

Es importante tener en cuenta que si se utiliza este servicio sólo puede haber un clúster de redBorder en esa subred de sincronismo. Sin embargo, dado que en entornos cloud se pueden crear las redes que sean necesarias, esto no representa un problema. Además, todos los miembros de un clúster deberán estar conectados a la red de sincronismo, o no podrán unirse al clúster de forma automática.

5.2.3 Uso de Chef en redBorder

Todos los servicios asociados de redBorder, tanto en los managers como en los sensores son configurados de forma automática utilizando Chef. Dado que lo que se implementa en la nube son los managers, nos vamos a centrar en éstos.

[17] Knife: herramienta de línea de comandos para la gestión de Chef Server.
[18] Para más información consultar la documentación de Chef [12].

Todos los managers tienen implementado el servicio Chef Client, que se encarga de autoconfigurar cada uno de los nodos de un clúster de redBorder. Además, en todo clúster debe existir al menos un nodo que implemente también el servicio Chef Server, donde se encuentra toda la configuración del el clúster. De esta forma, se puede gestionar la configuración de todo el clúster simplemente modificando las recetas y plantillas que gestiona el Chef Server. Cabe destacar que estos documentos se encuentran almacenados en un servicio de almacenamiento S3[19], de forma que si hay varios Chef Servers, todos cogen la configuración de ese sistema de almacenamiento compartido.

Al arrancar, los nodos utilizan el servicio rb-discover explicado anteriormente para averiguar si algún nodo de redBorder implementa el servicio Chef Server, para así registrarse en el servicio como miembro del clúster y poder acceder a las plantillas para su configuración.

La configuración que realiza el servicio Chef Client también es influida por los parámetros del User data de cada instancia. De esta forma podemos indicar parámetros de configuración concretos y el rol que queremos que cada instancia coja. En función del rol elegido, Chef client configurará distintos servicios para cada instancia.

Por último, indicar que los detalles de configuración de Chef en redBorder no son objetivo de este proyecto, pero se ha realizado la introducción para comprender cómo se realiza la configuración de los servicios del sistema a lo largo de las instancias que despliegan en la nube.

5.3 Despliegue de recursos de la cloud

En el punto anterior se veía cómo se realiza la autoconfiguración de los nodos, pero para poder realizar esta operación tenemos que tener instancias desplegadas en la nube, donde ejecutar el sistema.

Cada proveedor de IaaS nos facilita un procedimiento para realizar este despliegue, por lo que no se hará una guía paso a paso de cómo realizar el despliegue de redBorder, sino que se darán instrucciones generales, que se deberán adaptar en función del proveedor.

El procedimiento además podría ser automatizado, creando scripts o aplicaciones que utilicen las APIs de los proveedores, pero de nuevo, cada proveedor ofrece una API diferente. En el capítulo 0 sí se explicará una manera de automatizar el procedimiento para el caso de Amazon Web Services.

5.3.1 Instrucciones de despliegue de redBorder en un servicio IaaS

Antes de poder desplegar las instancias es necesario preparar la configuración de red que necesitan para su funcionamiento. redBorder necesita dos subredes, una con acceso a internet y otra aislada, utilizada la comunicación entre los nodos del clúster.

Cada instancia de redBorder debe tener dos interfaces de red, asociadas a las subredes anteriormente mencionadas.

Las instancias deben ser desplegadas según el procedimiento indicado en el siguiente diagrama, con el objetivo de que se realice correctamente la autoconfiguración del clúster. Para ello además deberán especificarse unos User-Data concretos, que son indicados después.

[19] S3: Simple Storage Service. Puede ser proveído mediante la aplicación Riak [9] o mediante el servicio web de Amazon S3.

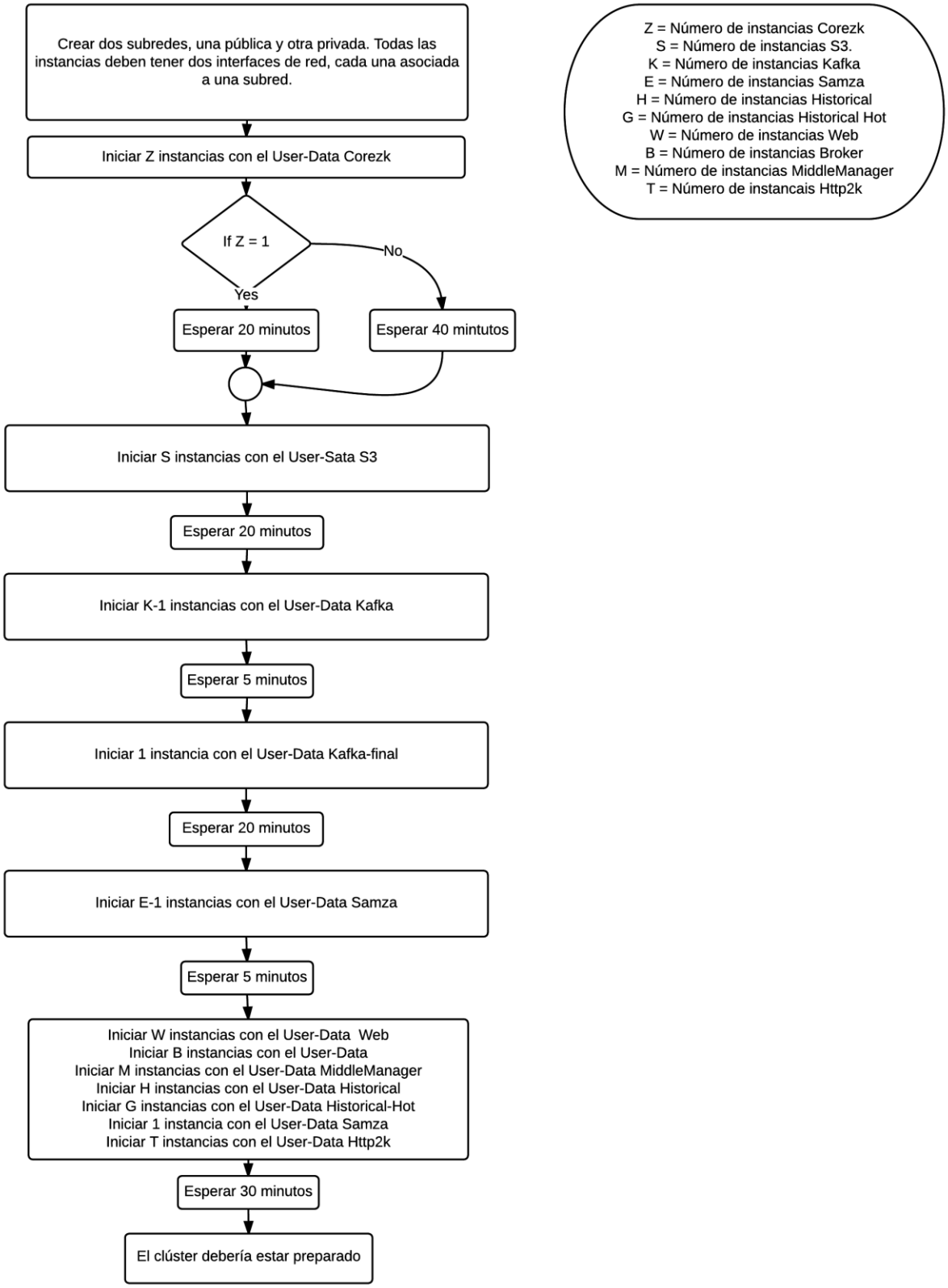

Figura 6 Procedimiento de despliegue manual de redBorder en un servicio IaaS

Los tiempos de espera de 20 minutos se utilizan para esperar a que el nodo finalice su configuración. Existen métodos para detectarlo y no tener que utilizar tiempos de espera, pero no serán explicados en este proyecto. Sin embargo, estos métodos son utilizados en el despliegue automático utilizando AWS CloudFormation que se explica en el punto 6.12.

Las esperas de 5 minutos se utilizan para garantizar que las instancias desplegadas después de la espera van a terminar después, ya que tienen que realizar operaciones adicionales (de hecho tienen User-Data distintos).

Los User-Data necesarios se muestran a continuación.

Corezk

```
#!/bin/bash

NODEROLE=corezk
NODESERVICES=""
MODULES="flow:true location:true ips:false monitor:true malware:false"
ENRICHMODE=samza
CMDFINISH=”/opt/rb/bin/rb_druid_rules.rb -t _default -d P3M -r 1 -p P1W -i 1”
```

S3

```
#!/bin/bash

NODEROLE=s3
NODESERVICES=""
```

Kafka

```
#!/bin/bash

NODEROLE=kafka
NODESERVICES=""
```

Kafka-final

```
#!/bin/bash

NODEROLE=kafka
NODESERVICES=""

CMDFINISH="/opt/rb/bin/rb_reassign_partitions.sh -de -p 4 -t
rb_flow,rb_flow_post,rb_nmsp,rb_loc,rb_loc_post,rb_state,samza-nmsp-m-log,samza-nmsp-i-
log,samza-loc-log"
```

Samza

```
#!/bin/bash

NODEROLE=samza
NODESERVICES=""
```

Samza-final

```
#!/bin/bash

NODEROLE=samza
NODESERVICES=""
CMDFINISH="/opt/rb/bin/rb_samza.sh -e"
```

Web

```
#!/bin/bash

NODEROLE=web
NODESERVICES=""
```

Broker

```
#!/bin/bash

NODEROLE=broker
NODESERVICES=""
```

Http2k

```
#!/bin/bash

NODEROLE=http2k
NODESERVICES=""
```

Historical

```
#!/bin/bash

NODEROLE=historical
NODESERVICES=""
```

Historical Hot

```
#!/bin/bash

NODEROLE=historical
NODESERVICES=""

cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_
{
        "redBorder" : {
                "druid" : {
                        "historical" : {
                                "tier" : "hot"
                        }
                }
        }
}
_RBEOF2_
```

MiddleManager

```
#!/bin/bash

NODEROLE=middleManager
NODESERVICES=""
```

Siguiendo los procedimientos indicados se obtiene un clúster completamente funcional, sin ningún servicio externo.

6 INTEGRACIÓN CON AMAZON WEB SERVICES

Based on our experience, I believe that we can be even more secure in the AWS cloud than in our own data centers

Tom Soderstrom, NASA CTO

En este proyecto se ha adaptado el sistema redBorder para su correcto funcionamiento en cualquier entorno Cloud, pero prestando especial atención a su integración con Amazon Web Services y a los servicios que este proveedor ofrece, ya que es sin duda el mayor proveedor de IaaS y PaaS que existe actualmente.

AWS, además de ofrecer el servicio de IaaS, proporciona otros muchos servicios gestionados, facilitando la implementación, despliegue y mantenimiento de éstos para permitirnos centrarnos en nuestra aplicación, al tener que mantener una menor cantidad de servicios.

Varios ejemplos de este tipo de servicios son AWS RDS (para bases de datos relacionales), AWS Elasticache (para clústers de caché) o AWS S3 (para almacenamiento de datos), que nos permiten emplear en nuestros sistemas estos servicios sin la necesidad de preocuparnos de su implementación y mantenimiento.

De ahí la importancia de la externalización de servicios para la gestión de un sistema complejo en la nube. Cuantos menos servicios se tengan que gestionar, menos problemas habrá que ir solucionando en el ciclo de vida del sistema, mejorando tanto la rentabilidad como la calidad de servicio ofrecido por el sistema.

En redBorder, gran parte de la integración con AWS consiste precisamente en esa externalización de servicios, reduciendo así la complejidad de la gestión del sistema y facilitando su despliegue. En los siguientes apartados se irán explicando los servicios utilizados y cómo se ha adaptado redBorder para funcionar con ellos.

6.1 Estructura de Amazon Web Services

Amazon Web Services dispone de una serie de centros de datos repartidos por todo el mundo. De esta forma, organiza su servicio en **regiones,** en función de dónde se sitúen los centros de datos que ofrecen el servicio. A su vez, cada región tiene varias **zonas de disponibilidad**, que son distintos centros de datos separados pero que pertenecen a una misma región, y que que están conectados mediante enlaces dedicados de alta capacidad y muy baja latencia, a fin de ofrecer una mayor fiabilidad frente a posibles desastres.

Los servicios de AWS se gestionan de forma independiente en función de la región, por lo que se debe especificar en qué región se trabaja al utilizarlos.

6.2 AWS Elastic Compute Cloud (EC2)

6.2.1 Introducción al servicio

EC2 es el principal servicio de Cloud Computing que provee Amazon. Es un servicio de IaaS que nos permite

crear tantas instancias como sean necesarias, con sistemas operativos tanto Windows como Linux.

A continuación veremos las principales características y funcionalidades de este servicio:

- Ejecución de tantas instancias (máquinas virtuales) con distintos recursos de hardware, permitiéndose ejecutar AMIs[20], tanto proveídas por Amazon como personalizadas y configuradas por el usuario, siendo posible ponerlas en ejecución, pararlas, o destruirlas en minutos mediante llamadas a la API de EC2.
- Configuración de seguridad y de acceso a redes por parte de la instancia, mediante su integración con Amazon Virtual Private Cloud (VPC), que será explicada posteriormente.
- Posibilidad de ejecutar las instancias en distintas regiones y zonas de disponibilidad.
- Mantener una misma dirección IP aunque se cambie la instancia, mediante el uso de Elastic IPs[21].
- Posibilidad de añadir o eliminar interfaces de red que pueden ser asignadas a distintas subredes.
- Almacenamiento persistente mediante el servicio Elastic Block Store (EBS). Son volúmenes de almacenamiento accesibles mediante red que pueden asignarse a instancias como unidades de almacenamiento, sin que se destruyan sus datos aunque se detenga la ejecución de la instancia. Pueden utilizarse como volumen raíz donde se encuentra el sistema operativo instalado, permitiéndose hacer copias de seguridad e incluso compartirlas para su ejecución en otras instancias de otros usuarios.
- Almacenamiento no persistente de acceso rápido (sólo en algunos tipos de instancia).
- Integración con grupos de autoescalado (Autoscaling Groups), balanceadores de carga (Elastic Load Balancers) y sistemas de monitorización (Cloudwatch).
- Servicio de metadatos, a través del cual la instancia puede obtener información sobre ella misma, incluyendo el User-Data.
- Servicio de Importación / Exportación de máquinas virtuales.

6.2.1.1 Amazon Machine Image (AMI)

Es una plantilla que contiene la configuración de software (sistemas operativos, aplicaciones, herramientas, etc), que se quiere desplegar en una instancia de Amazon EC2. La plantilla incluye:

- Una imagen de la unidad raíz de la instancia, donde se encuentra el software.
- Información sobre permisos de uso de la AMI, que indica qué cuentas pueden utilizarla para desplegar instancias.
- El mapa de dispositivos de almacenamiento predeterminado de la instancia, que especifica qué volúmenes deben ser asociados a las instancias.

Las AMIs, pueden ser generadas tanto a partir de otras AMIs modificadas (Amazon provee de plantillas básicas con varios sistemas operativos), como a partir de máquinas virtuales que se importan a Amazon.

Existen dos tipos de AMI según el medio de almacenamiento utilizado:

- **EBS-Backed AMI:** la plantilla de software se almacena en una unidad EBS[22]. Para desplegar una instancia se crea una copia de ese volumen EBS y se asigna a la instancia como unidad raíz.
- **Instance Store-Backed AMI**: la plantilla de software se almacena en Amazon S3[23]. Para desplegar la instancia se hace una copia del software en un volumen asociado a la instancia (no EBS). Cuando se apaga la instancia toda la información de la unidad se pierde.

---

[20] AMI: Amazon Machine Image. Imagen del disco duro de una instancia con un sistema operativo y software preinstalado, que puede ser desplegado una o varias instancias de Amazon EC2.

[21] Elastic IP: Es una dirección IP que nos reserva Amazon y que podemos asociarla a distintas instancias, permitiéndose enmascarar fallos cambiando la instancia a la que se asigna la IP.

[22] EBS: Elastic Block Store. Unidad de almacenamiento de alta fiabilidad y disponibilidad que puede ser asociada a instancias de EC2.

[23] Amazon S3: Amazon Simple Storage Service. Servicio de almacenamiento de objetos de AWS. Más información en su documentación [22]

También se pueden clasificar según su visibilidad:

- **Public AMI**: todas las cuentas y usuarios de AWS pueden acceder a ellas y desplegarlas de forma gratuita. Amazon provee de muchas AMIs públicas con sistemas operativos básicos o con software preinstalado, así como AMIs preparadas por otros muchos desarrolladores.
- **Private AMI**: sólo pueden ser desplegadas en la cuenta que ha creado la AMI. Pueden ser creadas a partir de AMIs públicas y también pueden compartirse o hacerse públicas posteriormente.
- **Shared AMI**: sólo las cuentas permitidas por el dueño de la AMI pueden acceder y usarlas.
- **Paid AMI**: son accesibles públicamente siempre y cuando se pague por su uso. Se acceden a ellas a través del servicio **AWS Marketplace**[24], y es una de las formas que tienen los desarrolladores de vender sus productos en Amazon Web Services.

6.2.1.2 Instancias

Las instancias son máquinas virtuales que se despliegan en la infraestructura cloud con el software proveído mediante una AMI. Se caracterizan por tener asignados 4 tipos de recursos principales:

- **vCPUs**: son núcleos de CPU virtuales que se asignan a la instancia. Tienen asociado un valor ECU[25], que mide el rendimiento en cuanto a capacidad de ejecución y procesado de la instancia.
- **RAM**: la memoria RAM asignada a la máquina virtual de la instancia.
- **Almacenamiento**: unidades de almacenamiento asignadas a las instancias. Pueden ser unidades asociadas al tipo de instancia (Ephemeral storage), o volúmenes EBS.
- **Interfaces de red**: las instancias tienen asociadas al menos una interfaz de red. El tipo de instancia determina el rendimiento que pueden alcanzar estas interfaces de red, que en AWS son denominadas Elastic Network Interfaces (ENI).

Conviene aclarar que un tipo de instancia tiene unos recursos de CPU y RAM fijos, pero tanto la capacidad de almacenamiento como las interfaces de red pueden cambiar incluso en caliente (mientras la instancia se encuentra en ejecución).

6.2.1.2.1 Tipos de instancia

La forma de elegir qué recursos de CPU, RAM, almacenamiento de instancia y capacidad de red tiene una instancia en Amazon Web Services es a través de la elección de un tipo de instancia. Dado que los tipos de instancia van cambiando contínuamente, no los vamos a dejar reflejados en este documento, pero se pueden consultar en la documentación de Amazon EC2 [14].

6.2.1.2.2 Tarificación

En Amazon Web Services por defecto se paga únicamente por los recursos utilizados. En el caso de Amazon EC2, se tarifica por horas de uso, siendo el coste por hora dependiente del tipo de instancia utilizado.

Sin embargo, para abaratar los costes se puede optar por otros tipos de tarificación, obteniendo una nueva clasificación de instancias:

- **Instancias bajo demanda**: son instancias que se rigen por el tipo de tarificación normal, en el que las instancias tienen un coste por hora y pueden ser creadas, paradas y eliminadas cuando sea necesario.
- **Instancias puntuales**: disponen de un método de tarificación relativo al uso de los recursos en AWS. En este proyecto no vamos a entrar en su funcionamiento. Para más información, consultar la documentación de Amazon EC2 [14].
- **Instancias reservadas**: se puede reservar un tipo de instancia durante un tiempo, obligándote a pagar por ese tiempo utilices o no la instancia, a cambio de una reducción de su coste. En la reserva además puedes elegir entre pagar el coste al principio, pagarlo mes a mes, o pagarlo parcialmente al principio

[24] AWS Marketplace: tienda online de software desplegable en Amazon Web Services. Más información en su documentación [34]
[25] ECU: EC2 Compute Unit. Es una unidad de medida de la capacidad de procesamiento que tienen las vCPU de una instancia de Amazon EC2. 1 ECU equivale aproximadamente a la capacidad de un Intel Xeon de 2007 a 1 GHz.

y el resto mes a mes. En función de la elección, el precio por hora de la instancia varía. Cabe destacar, que lo que se reserva es un tipo de instancia y una una instancia concreta, por lo que si se despliega una instancia de ese tipo, será facturada utilizando el precio de la instancia reservada.

6.2.1.2.3 Ciclo de vida de las instancias

Las instancias pasan por una serie de estados desde que son creadas hasta que son destruidas. En el siguiente diagrama se muestra el denominado ciclo de vida de una instancia, que será descrito a continuación.

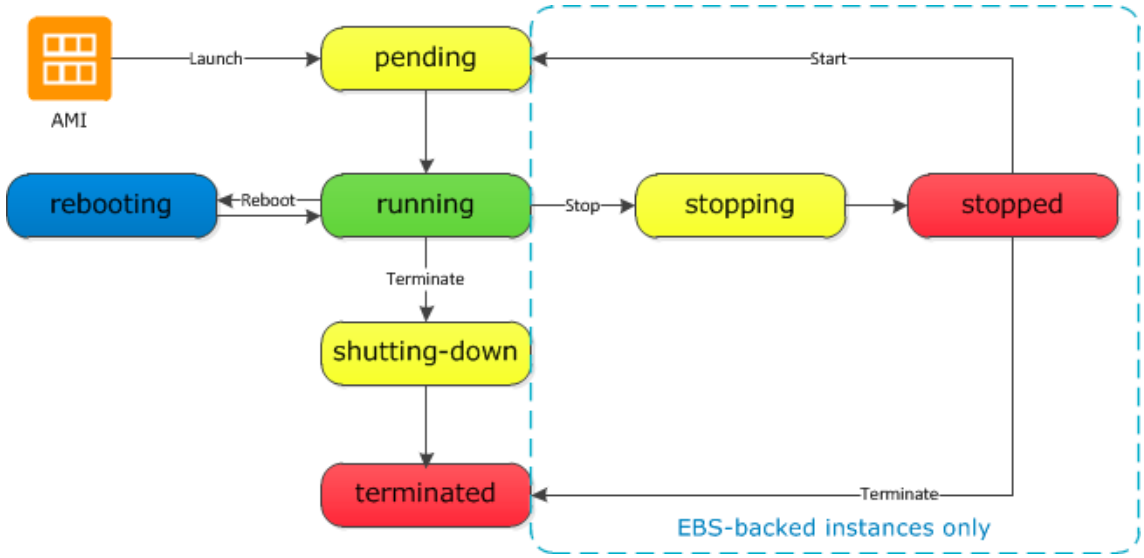

Figura 7 Ciclo de vida de una instancia de Amazon EC2 [14]

En Amazon EC2 toda instancia es creada a partir de una AMI, como ya se ha mencionado. En el momento que se crea la instancia pasa al estado **Pending**. En este estado Amazon EC2 se encarga de aprovisionar los recursos necesarios para la ejecución de la instancia y despliega la AMI.

Cuando finaliza el proceso la instancia cambia al estado **Running**, y comienza la ejecución de la máquina virtual. La instancia permanecerá en este estado siempre y cuando no se realice ninguna acción de gestión sobre ella y no se produzca ningún error, tanto en la infraestructura de Amazon como en la propia instancia.

A partir de este estado, pueden ocurrir varios sucesos:

- **La instancia se reinicia**: tanto si el sistema operativo de la instancia solicita un reinicio como si se reinicia por petición a la API de Amazon EC2, el sistema comienza el proceso de apagado para volver a arrancar. Cuando se apaga la instancia pasa al estado **Rebooting**, para volver al estado **Running**. Cabe destacar que la instancia sigue ejecutándose en el mismo host, conserva su Instance-Id[26], su dirección IP no cambia, los volúmenes de instancia conservan los datos y la tarificación no se ve afectada.
- **La instancia se para**: la instancia pasa al estado **Stopping** y se apaga la instancia. Cuando finaliza el proceso se pasa al estado **Stopped**. Sin embargo, la instancia no pierde ni su Instance-Id ni sus volúmenes EBS, pero sí su IP (a menos que sea una Elastic IP[27]), y los datos de sus volúmenes de instancia, ya podría ejecutarse de nuevo en otro host. La instancia podría volver a iniciarse, en cuyo caso pasaría primero a estado **Pending** y después a **Running**.
- **La instancia se termina**: la instancia pasa al estado **Shutting-down** y se apaga la instancia. Cuando finaliza el proceso se pasa al estado **Terminated**. La instancia es destruida y ya no se puede recuperar.

6.2.1.2.4 Metadatos y User-Data

Amazon EC2 provee a las instancias de un servicio de Metadatos que permite a éstas obtener información relacionada con las características de la propia instancia. El servicio es de tipo Web y es accedido mediante la

[26] Instance-Id: identificador único que Amazon le asigna a cada instancia de EC2.
[27] Elastic IP: dirección IP reservada en Amazon VPC que puede ser asignada a cualquier interfaz de red de la misma VPC. Es explicado más detalladamente en el punto 6.3

dirección IP reservada 169.254.169.254. Cada instancia accede a su propio servicio de metadatos.

Ejemplos de datos que se obtienen son la Instance-Id, IP pública asignada, tipo de instancia, etc.

Además se utiliza este servicio de metadatos para obtener el User-Data, ya explicado en el capítulo 4.

6.2.1.3 Balanceadores de carga

Amazon permite balancear tráfico entre instancias de Amazon EC2, sin necesidad de preocuparnos de escalar el sistema de balanceo, gracias al servicio AWS Elastic Load Balancing.

El servicio soporta los siguientes protocolos:

- HTTP o HTTPS
- TCP o SSL (TCP seguro)

Lo único que hay que hacer para configurar el servicio es asociar las instancias a las que se quiere balancear el tráfico y configurar los siguientes elementos:

- **Listeners**: permiten indicar qué protocolo y en qué puerto escuchar, y a qué protocolo y puerto enviar el tráfico de forma balanceada entre las instancias registradas. Sólo se puede hacer reenvío de protocolos del mismo nivel. Una práctica muy utilizada, es hacer que el balanceador termine el cifrado SSL de una conexión, siendo enviada la información a las instancias en claro. Para ello hay que registrar un certificado en el balanceador.
- **Health Checks**: son comprobaciones de que las instancias están funcionando, y que se les puede enviar tráfico. De esta forma, el tráfico sólo es enviado a instancias que están en funcionamiento, obteniendo así un servicio en alta disponibilidad. El load balancer envía peticiones cada cierto tiempo para comprobar que el servicio está en funcionamiento en las instancias, asignándoles el estado InService si están activas u OutOfService en caso contrario.

Normalmente, estos balanceadores de carga reciben tráfico de Internet, pero también pueden crearse balanceadores internos, que reciben tráfico proveniente de instancias de EC2 y que es dirigido a otras instancias.

6.2.1.4 Grupos de autoescalado

Cuando hay varias instancias con las mismas características que cumplen un mismo rol, se pueden agrupar en grupos de autoescalado. De esta forma se puede controlar fácilmente la creación y destrucción de instancias similares, incluso programando el crecimiento o decrecimiento del grupo cuando ocurren ciertos eventos.

En un grupo de autoescalado hay tres valores importantes que determinan su tamaño:

- **Tamaño mínimo**: determina el número mínimo de instancias que debe tener el grupo de autoescalado. Nunca puede haber menos instancias que las indicadas por este parámetro.
- **Tamaño máximo**: determina el número máximo de instancias que puede llegar a tener el grupo de autoescalado. Nunca puede haber más instancias que las indicadas por este parámetro.
- **Capacidad deseada**: indica el número de instancias que debe tener el grupo de autoescalado en un momento concreto. Si el número de instancias no coincide con este parámetro, el grupo debe crecer o decrecer hasta alcanzarlo. La capacidad deseada se mantiene incluso cuando una instancia deja de funcionar, siendo sustituida de forma automática por otra.

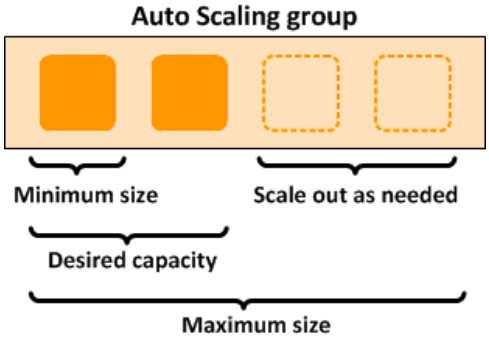

Figura 8 Esquema representativo de los valores que definen el tamaño del grupo de autoescalado [15]

6.2.1.4.1 Configuración de arranque

Todas las instancias de un grupo de autoescalado utilizan la misma configuración, proveída mediante una plantilla denominada Launch Configuration. A continuación se enumeran los campos más importantes:

- Image Id (AMI).
- Grupos de seguridad.
- Tipo de instancia.
- Mapa de unidades de disco asociados.
- User-Data.

Es importante tener en cuenta que todas las instancias tendrán exactamente el mismo User-Data.

6.2.1.4.2 Políticas de autoescalado

Las políticas de autoescalado definen cuando y cómo debe crecer o decrecer el grupo de autoescalado, siempre que se cumplan las restricciones de tamaño mínimo y máximo del grupo.

Lo que hacen estas políticas es cambiar el parámetro "capacidad deseada" del grupo de autoescalado para provocar la creación o destrucción de instancias dentro del grupo. Este cambio se puede producir por dos motivos:

- **Autoescalado programado**: se puede programar en el tiempo un cambio en la capacidad deseada del grupo de autoescalado. Por ejemplo, se puede programar que a cierta hora en la que se sabe que va a haber más actividad crezca el grupo de autoescalado y que a otra hora en la que se sabe que apenas va a haberla, disminuya.
- **Autoescalado provocado por alarmas**: se puede configurar que cuando se produzca una alarma en Amazon Cloudwatch[28], crezca o disminuya el grupo de autoescalado. Por ejemplo, podemos hacer que el grupo crezca si el uso medio de la CPU en el grupo es mayor que una cierta cantidad, y que disminuya si es menor que otra cierta cantidad. Cuando se produce el escalado mediante este método, se deja un tiempo de espera (Cooldown) entre cada acción de autoescalado, para dejar que se estabilice la métrica que se está monitorizando tras la realización del proceso de escalado.

Cabe destacar que un grupo de autoescalado no tiene por qué tener políticas, ya que siempre se puede modificar su tamaño de forma manual modificando el parámetro "Capacidad Deseada".

Cuando se disminuye el tamaño del grupo de autoescalado modificando el parámetro "Capacidad Deseada", la elección de la instancia a destruir se realiza en función de la política de terminación elegida. AWS soporta las siguientes políticas:

- **Instancia más antigua**: elimina la instancia del grupo más antigua, siendo útil si se quiere ir cambiando el tipo de instancia.
- **Instancias más nueva**: elimina la instancia del grupo más nueva, siendo útil si se quiere probar nuevas configuraciones sin mantenerlas en producción.

[28] Amazon Cloudwatch: servicio de monitorización de AWS. Se explicará con más profundidad en el punto 6.8.

- **Configuración de arranque más antigua**: elimina la instancia del grupo con la configuración de arranque más antigua, siendo útil si se quiere ir actualizando la configuración del grupo.
- **Instancia más cercana a la finalización de la hora de facturación** (política por defecto): elimina la instancia que está más cerca de terminar el ciclo de facturación, optimizando el coste de las instancias.

Es importante tener en cuenta que no se pueden parar instancias en un grupo de autoescalado. O se crean o se eliminan, pero no se pueden parar a menos que se quiten del grupo de autoescalado.

6.2.1.4.3 Integración con balanceadores de carga

El servicio de autoescalado en Amazon está integrado con AWS Elastic Load Balancer, de forma que cuando el grupo crece o decrece, las instancias son registradas y desregistradas de forma automática en el balanceador, permitiendo repartir la carga entre las instancias deseadas de forma sencilla.

6.2.1.4.4 Ciclo de vida de las instancias de un grupo de autoescalado

Las instancias que pertenecen a un grupo de autoescalado tienen un ciclo de vida un poco diferente a las que no, ya que deben adaptarse a los procesos de escalado.

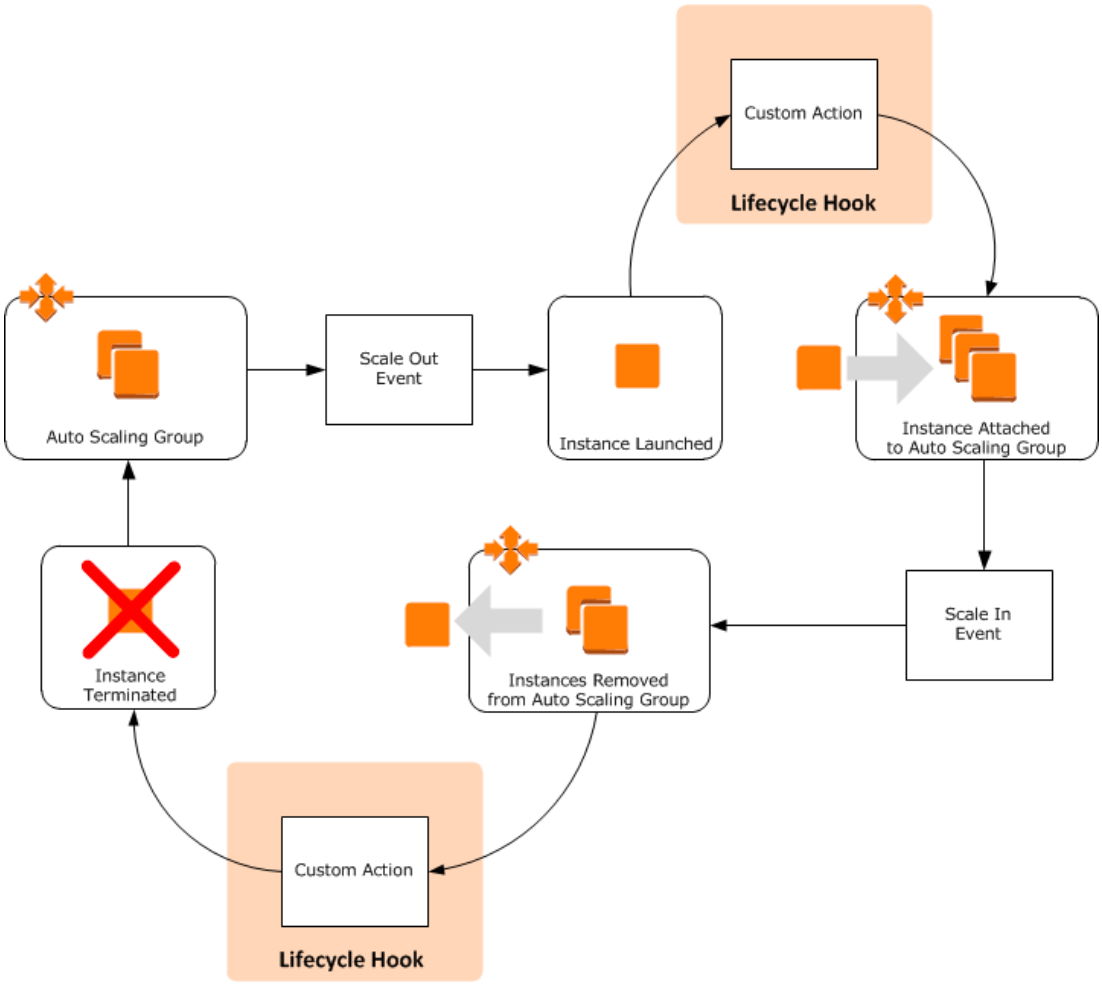

Figura 9 Ciclo de vida de las instancias en un grupo de autoescalado [15]

Las instancias sólo pueden ser iniciadas y destruidas, pero no paradas, por lo que en principio sólo podemos tener instancias en los estados estables InService (Running) o Terminated, y los estados de transición Pending y Terminating. Sin embargo, dado que puede que no sepamos cuando se van a crear o destruir instancias, se pueden introducir otros dos estados pertenecientes a lo que Amazon llama **Lifecycle Hooks**:

- **Pending:Wait**: permite realizar acciones en las instancias antes de que sean puestas en servicio.
- **Terminating:Wait**: permite realizar acciones en las instancias antes de que sean destruidas.

Cuando una instancia entra en esos estados especiales, se manda un mensaje a Amazon SNS[29] o a Amazon SQS[30] para que la instancia sea notificada de ese cambio de estado y pueda realizar las acciones que sean necesarias.

6.2.2 Integración con redBorder

En el capítulo 5 se explicaba el proceso de adaptación de redBorder a entornos Cloud genéricos. En éste, continuaremos añadiendo nuevas integraciones específicas para Amazon EC2.

El primer paso para la ejecución de redBorder en este entorno es la creación de la Amazon Machine Image (AMI) de redBorder, del cuál también se había hablado anteriormente en el capítulo 4.

Tanto el servicio Cloud-Init como otros scripts que incluye el sistema redBorder utilizan el servicio de metadatos de Amazon EC2 para configurar la instancia:

- Se configura el hostname de la instancia para que coincida con su Instance Id.
- Se añade el certificado SSH elegido en EC2 a la instancia.
- Se ejecuta el script pasado mediante el User-Data

El User-Data nos va a permitir definir qué rol ejecuta cada instancia y pasar información necesaria para que se puedan integrar con otros servicios de Amazon.

Por cada rol se ha creado un grupo de autoescalado distinto, ya que todas las instancias de un mismo rol comparten la misma configuración, definida en el Launch Configuration del grupo de autoescalado. La variable NODEROLE del User-data es la que le dice a cada instancia que rol debe tomar dentro del clúster, como se puede observar en el ejemplo:

```
NODEROLE=corezk
```

Como ya se ha explicado en el capítulo 3 , para formar un clúster altamente escalable en la nube se han definido los siguientes roles:

- Corezk
- Kafka
- Http2K*
- Web*
- Broker
- MiddleManager*
- Historical
- Historical Hot
- Samza / Enrichment

Los roles marcados con un asterisco (*) son los que están preparados para autoescalar en caso necesario. Los detalles del autoescalado de las instancias se explicarán más profundamente en el capítulo 7.

6.2.2.1 Tipos de instancia por rol

Cada rol tiene unas necesidades de hardware específicas, por lo que se han elegido los tipos de instancia de Amazon EC2 más adecuados, que quedan representadas en la siguiente tabla:

[29] Amazon Simple Notification Service (SNS): servicio de envío de notificaciones de Amazon. Será explicado en el punto 6.10
[30] Amazon Simple Queue Service (SQS): servicio de colas de mensajes de Amazon. Será explicado en el punto 6.9

Tabla 2 Tipos de instancia por rol

Role	Tipo de instancia	vCPUs	RAM (GB)	Tamaño disco EBS (GB)	Tamaño disco instancia (GB)	Razón de decisión	Tipo optimizado para
Corezk	i2.xlarge	4	30,5	50	800	Uso intenso de E/S y alto consumo de RAM	E/S
Kafka	i2.xlarge	4	30,5	50	800	Uso intenso de E/S	E/S
Http2K	c4.2xlarge	8	15	8*	No tiene	Predominio de uso de CPU	CPU
Web	c4.2xlarge	8	15	8*	No tiene	Predominio de uso de CPU	CPU
Broker	c4.2xlarge	8	15	50	No tiene	Predominio de uso de CPU	CPU
Middle Manager	r3.4xlarge	16	122	30	No se asigna**	Altísimos usos de CPU y RAM	RAM
Historical	c4.4xlarge	16	30	500	No tiene	Muy alto uso de CPU y gran capacidad de almacenamiento	CPU
Historical Hot	c4.4xlarge	16	30	50	No tiene	Muy alto uso de CPU	CPU
Samza	c4.4xlarge	16	30	50	No tiene	Muy alto uso de CPU	CPU

* En estos roles se mantiene el tamaño de disco definido en la AMI para evitar el reparticionado del disco en el arranque de la instancia y aumentar la velocidad de arranque, ya que son instancias que no necesitan mucho almacenamiento y deben de arrancar muy rápido porque están preparadas para autoescalar.

** El tipo de instancia r3.4xlarge viene con un disco duro de instancia de 320 GB, pero no es asignado para evitar su formateo y acelerar el proceso de arranque.

Estos son los tipos de instancias que se recomiendajn para el correcto funcionamiento del clúster en AWS, pero podrían ser cambiados en función de las necesidades del servicio.

6.2.2.2 Uso de balanceadores en redBorder

El sistema redBorder utiliza 2 balanceadores, uno para la interfaz web y otro para datos de los sensores asociados. En ambos casos, el tráfico recibido es HTTPS, siendo terminado el cifrado en el balanceador, de forma que a las instancias llega tráfico HTTP.

6.2.2.2.1 Balanceador de carga para la Web

Este balanceador sólo tiene asociadas las instancias que ofrecen el servicio de la Web de redBorder, es decir, las que tienen rol Web. Por lo tanto, lo que se ha hecho es configurar el grupo de autoescalado del rol Web para que todas las instancias asociadas a ese grupo queden registradas en el balanceador. Además, se asocia el balanceador a la Virtual Private Cloud (VPC), de las cuales se hablará más adelante.

En la siguiente tabla podemos ver la configuración de los listeners del Load Balancer:

Tabla 3 Listeners de balanceador de la web

Protocolo a escuchar	Puerto a escuchar	Protocolo a reenviar	Puerto a reenviar
HTTP	80	HTTP	80
HTTPS	443	HTTP	7979

El objetivo del balanceador es repartir la carga de tráfico sobre la web, que debe ser HTTPS siempre. Sin embargo, también se balancean las peticions HTTP al puerto 80 porque AWS Elastic Load Balancer no soporta redirección a HTTPS, por lo que la debe realizar la instancia. En nuestro caso esta redirección la hace el servicio NGINX[31]. El balanceador además termina el tráfico HTTPS, por lo que se debe configurar también el certificado utilizado para la conexión segura mediante SSL, cuyo procedimiento será explicado posteriormente. Una vez finalizada esa conexión, el tráfico es enviado en claro mediante HTTP al puerto 7979, donde NGINX procesa las peticiones y las reenvía a distintos servicios de redBorder (Web, API, etc.).

Además se deben configurar las comprobaciones de estado de la instancia (Health Checks). En el caso de la web, el balanceador se conectará cada cierto tiempo al Path /users/login en el puerto 7979 mediante HTTP. Si la comprobación falla demasiadas veces seguidas, el balanceador dejará de enviar tráfico hacia esa instancia. En la siguiente tabla, se muestra la configuración utilizada para la web de redBorder:

Tabla 4 Health Checks del balanceador de la web

Protocolo	HTTP
Puerto	7979
Path	/users/login
Límite de tiempo de espera de respuestas (si es superado, se considera respuesta errónea)	25 segundos
Intervalo entre comprobaciones	30 segundos
Límite de comprobaciones erróneas seguidas	5
Comprobaciones correctas necesarias para volver a poner en servicio a la instancia	3

[31] NGINX: proxy inverso y de http. Más información en su documentación [41].

Por último, se ha asociado un certificado al balanceador para que pueda terminar las conexions HTTPS. Debe proveerse en formato PEM la siguiente información:

- Private Key
- Public Key Certificate
- Certificate Chain (opcional).
- Nombre (para identificarlo en la cuenta de AWS).

Una vez subido a AWS[32], podemos utilizarlo en los balanceadores que necesitemos, simplemente seleccionándolo mediante su nombre asociado.

6.2.2.2.2 Balanceador de carga para datos de sensores

La información de los sensores llega a los managers de redBorder de la Cloud mediante el protocolo HTTPS también. Este balanceador es muy parecido al de la web, por lo que sólo serán explicadas sus diferencias

En primer lugar, el grupo de autoescalado asociado será el del rol Http2K, que se encarga de pasar la información que viene mediante HTTP al sistema de colas Kafka, desde el cuál se hace disponible dicha información al resto del clúster. Además, en este caso no es necesario redireccionar peticiones desde HTTP a HTTPS, por lo que la configuración de los listeners quedaría de la siguiente forma:

Tabla 5 Listeners del balanceador de datos

Protocolo a escuchar	Puerto a escuchar	Protocolo a reenviar	Puerto a reenviar
HTTPS	443	HTTP	7979

Las comprobaciones de estado sí que cambian más, ya que el servicio NGINX que corre en los nodos Http2K en principio sólo soportan peticiones POST (envío de datos), y AWS Elastic Load Balancer sólo permite realizar comprobaciones con GET. Por ello se ha habilitado un Path especial (/nginx_status) para saber si el servicio está activo o no. Así quedaría la configuración de las comprobaciones:

Tabla 6 Health Checks del balanceador de datos

Protocolo	HTTP
Puerto	7979
Path	/nginx_status
Límite de tiempo de espera de respuestas (si es superado, se considera respuesta errónea)	25 segundos
Intervalo entre comprobaciones	30 segundos
Límite de comprobaciones erróneas	5

[32] Existen varios procedimientos para subir a AWS el certificado SSL para AWS Elastic Load Balancer. En este documento no son explicados, pero se pueden consultar en la documentación de AWS Elastic Load Balancer [45].

seguidas	
Comprobaciones correctas necesarias para volver a poner en servicio a la instancia	3

Finalmente, mencionar que el certificado para terminar SSL se instala de forma similar al explicado para el balanceador de la Web.

6.3 AWS Virtual Private Cloud (VPC)

6.3.1 Introducción al servicio

Amazon Virtual Private Cloud es el servicio de gestión de redes en AWS. Nos permite crear un entorno de red privado y controlar el acceso a este tanto desde internet como por parte de las instancias que se ejecutan en dicho entorno, como si fuera nuestro propio centro de datos, pero con las ventajas de escalabilidad de AWS.

6.3.1.1 Elementos de una VPC

Una VPC puede contener los siguientes elementos:

- **Interfaz de red**: son asociadas a instancias y les permite acceder a una subred, que puede tener o no acceso a otras redes o internet. Estas interfaces siempre tienen al menos una dirección privada del rango de la subred.
- **Dirección IP elástica**: son direcciones IP que se pueden asociar a distintas interfaces de red, permitiéndose mantener una misma IP aunque se cambie de interfaz o instancia.
- **Subred**: se puede dividir la VPC en distintas subredes y controlar en cada una de ellas el acceso a otras redes o a internet. Todas las instancias con interfaces de red asociadas a la misma subred tienen visibilidad directa entre sí, como si estuvieran en una misma LAN[33]
- **Tabla de rutas**: cada subred tiene asociada una tabla de rutas en la que se define a qué redes tiene acceso cada subred.
- **Puerta de enlace a Internet (Internet Gateway)**: es un router virtual que permite el acceso a Internet desde una subred. Para ello simplemente hay que meter en la tabla de rutas el identificador de la puerta de enlace y las interfaces asociadas a la subred dispondrán de acceso a Internet.
- **Conexión VPN**: se establece entre una puerta de enlace para VPN de Amazon y un router del cliente que soporte VPN basada en IPSec.
- **Puerta de enlace para** VPN (Virtual Private Gateway): es un router virtual que permite el acceso a la red del cliente mediante una conexión VPN basada en IPSec, permitiendo extender la red corporativa a AWS.
- **Puerta de enlace de cliente (Customer Gateway)**: es un recurso lógico que identifica al router del cliente que termina la conexión VPN.
- **Security Group**: se asignan a interfaces de red y permiten establecer reglas de filtrado de tráfico de nivel 3 y 4. Los parámetros que se pueden indicar para el filtrado son el protocolo, el puerto y un bloque de IPs de origen. Además se permite la creación tanto de reglas de salida como de entrada.
- **Listas de control de acceso (ACL)**: se asigna a una subred y limita el tráfico permitido dentro de esa subred. A diferencia de los Security Groups, las reglas de la ACL son sin estado, por lo que se deben definir tanto para tráfico entrante como para tráfico saliente.

[33] Local Area Network (LAN): red de área local.

- **VPC Endpoint**: permite acceder a ciertos servicios de Amazon a través de una conexión interna de la VPC, en vez de a través de Internet.
- **VPC Peering connection:** conexión entre dos VPC distintas. Pueden ser incluso de distintas cuentas de AWS.

En la siguiente figura se muestra la estructura básica de una Virtual Private Network.

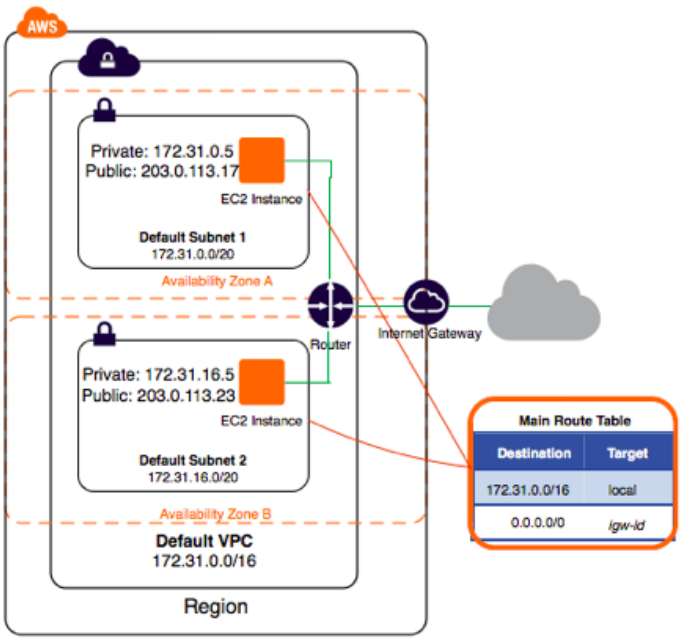

Figura 10 Estructura básica de una VPC [16]

Las VPC deben ser creadas en una región de AWS, pero pueden abarcar varias zonas de disponibilidad. Las subredes sin embargo, sí deben estar dentro de una zona de disponibilidad concreta, por lo que si se desea tener una arquitectura Multi-AZ (con varias zonas de disponibilidad), deben existir varias subredes. Estas subredes tienen asociadas unas tablas de rutas, que definen la conexiones con otras subredes y con el resto de internet. Finalmente, la conexión a Internet se realiza mediante el Internet Gateway. Las subredes que tengan en su tabla de rutas asociado el Internet Gateway de la VPC, y además tengan una IP pública asociada, tendrán acceso a Internet.

Cabe destacar que en AWS no hay que configurar dispositivos físicos de red como Routers o Switches, sino que se trabaja a un nivel más alto de abstracción: subredes y tablas de rutas, ya que no se conoce la localización física de las instancias en los centros de datos de Amazon.

6.3.2 Integración con redBorder

Cada clúster de redBorder es creado en una VPC distinta, de forma que están completamente aislados entre sí, incluso si hay varios clústers en una misma cuenta de AWS.

6.3.2.1 Arquitectura de red

En la siguiente figura se ilustra la arquitectura de red de redBorder en la VPC:

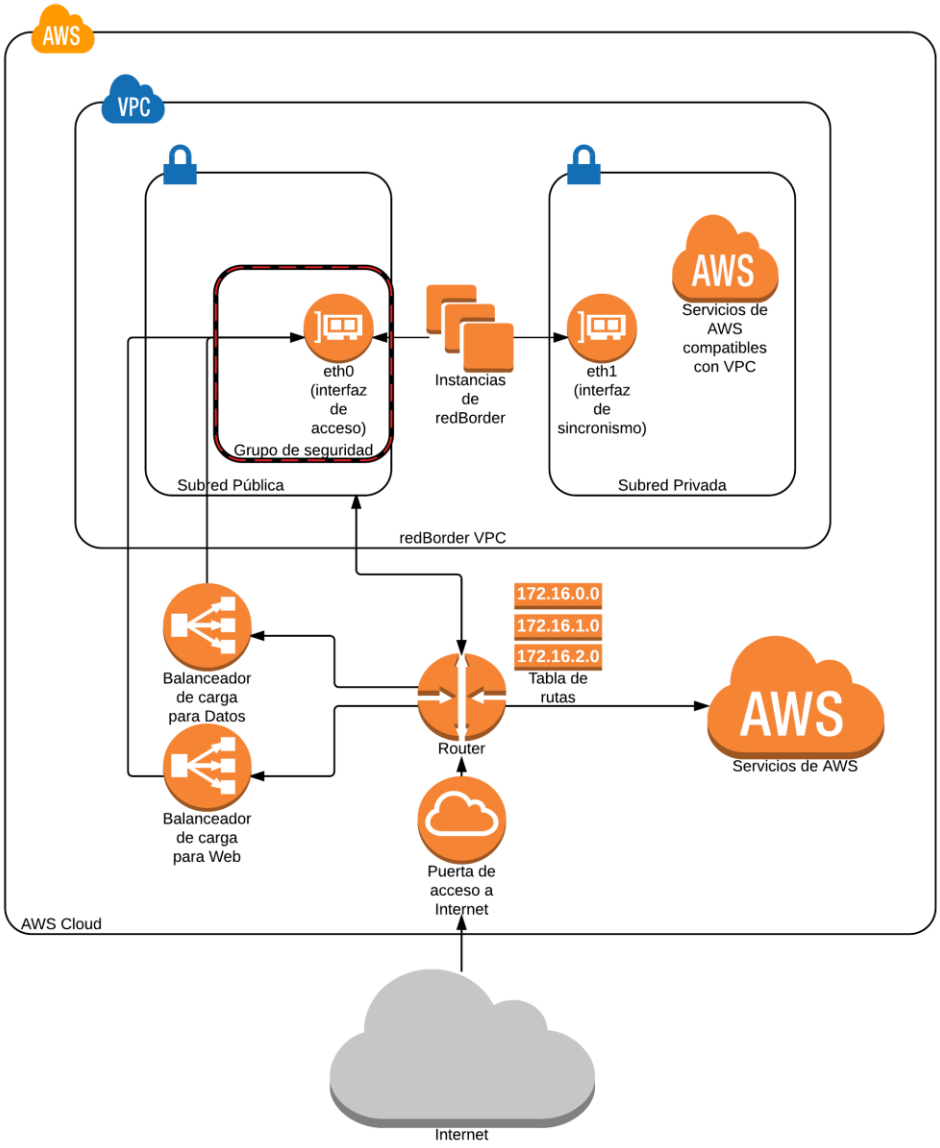

Figura 11 Arquitectura de la VPC de redBorder

La VPC está formada por 2 subredes, una subred pública y otra privada, con el objetivo de disponer de una subred de acceso desde el exterior y una subred de sincronismo, utilizada para la comunicación entre los distintos nodos del clúster.

Cada instancia de redBorder en Amazon EC2 tendrá asociadas dos interfaces de red (Elastic Network Interfaces), estando conectada cada una a una subred distinta. El resto de servicios de Amazon utilizados por redBorder y que son compatibles con VPC, se implementan siempre en la subred privada, a fin de que sólo las instancias puedan acceder a ellos.

Las instancias de redBorder son desplegadas en grupos de autoescalado y éstos no soportan la creación de instancias con más de una interfaz. Como en redBorder es necesario el uso de dos interfaces, cada instancia debe crear una interfaz de red (Elastic Network Interface) y asignársela a sí misma mediante el uso de la CLI de Amazon Web Services. En la imagen de redBorder se ha incluido el script **rb_set_aws_interface.sh** que realiza estas operaciones y que puede ser consultado en el Anexo VIII.

El script se ejecuta por indicación en el User-Data cuando sea necesario, mediante la siguiente línea:

```
. /opt/rb/bin/rb_set_aws_interface.sh
```

En cuanto a la seguridad, la interfaz de acceso (eth0) tiene asignado un grupo de seguridad con la siguiente configuración:

Tabla 7 Grupo de seguridad de las instancias de redBorder

Protocolo	Puerto	Origen
TCP	22	IP de los gestores de redBorder (sólo ellos deben tener acceso por SSH)
TCP	7979	IP del balanceador de carga
TCP	80	IP del balanceador de carga

Este grupo de seguridad otorga acceso a la web de redBorder (puerto 7979 y 80) únicamente a los balanceadores de carga, ya que las peticiones HTTP son gestionadas por éste y a la instancia de redBorder le llega con esa IP de origen. El acceso para gestión mediante SSH se permite únicamente a la IP que se especifique, que será la de la sede de redBorder.

En la interfaz de sincronismo no se ha asignado ningún grupo de seguridad, ya que sólo será accesible por otras instancias de redBorder, que necesitan acceso total entre ellas para su correcto funcionamiento.

6.4 AWS Route 53

6.4.1 Introducción al servicio

Route 53 es el servicio de gestión avanzada de DNS que ofrece Amazon Web Services. Nos permite trabajar alrededor de dos conceptos fundamentales:

- **Dominios**: AWS permite tanto registrar dominios de distinto tipo como utilizar dominios ya existentes para su gestión de DNS.
- **Hosted Zones**: es una colección de registros DNS que se gestionan bajo un mismo nombre. Pueden estar asociados a un dominio o no.

Cuando se crea una Hosted Zone, se le asignan dos entradas:

- **Registro NS**: contiene los 4 servidores de nombres asociados a la Hosted Zone.
- **Registro SOA**: proporciona información sobre el servidor DNS primario de la Hosted Zone.

Posteriormente se pueden crear los registros DNS que sean necesarios[34].

Existen dos tipos de Hosted Zones:

- Públicas: contienen información sobre cómo se quiere enrutar el tráfico de un dominio **en Intenet**.
- Privadas: contienen información sobre cómo se quiere enrutar el tráfico de un dominio **en una VPC**.

De esta forma, se puede gestionar la resolución de nombres de forma diferente para equipos de Internet y para equipos de la VPC asociada a la Hosted Zone privada.

6.4.1.1 Características avanzadas

Route 53, además de comportarse como un servicio de DNS normal, ofrece una serie de características avanzadas para enrutar el tráfico de un dominio, llamadas políticas de enrutamiento. Soporta las siguientes:

- **Enrutamiento simple**: funciona como un servicio de resolución de nombres tradicional. Asigna un nombre a un recurso (como una IP).

[34] Los tipos de registros soportados en AWS Route 53 se encuentran listados en la documentación de AWS Route 53 [25].

- **Enrutamiento balanceado y ponderado**: se utiliza para balancear el tráfico entre varios recursos en la proporción deseada. Por ejemplo se pueden enviar el 60% de peticiones a un servidor web y el 40% a otro.
- **Enrutamiento basado en latencia**: permite enrutar el tráfico hacia el recurso que se encuentre en el centro de datos que ofrece una menor latencia al usuario que realiza la solicitud.
- **Enrutamiento basado en conmutación por error**: permite asignar una comprobación de estado (Health check) a un recurso. Todo el tráfico se enrutaría hacia ese recurso excepto en caso de fallo, siendo entonces enrutado hacia otro recurso de reserva (backup).
- **Enrutamiento basado en geolocalización**: permite enrutar el tráfico hacia distintos recursos en función de la localización del cliente.

6.4.2 Integración con redBorder

En redBorder se utiliza el servicio AWS Route 53 para lo siguiente:

- Asignar nombres a los balanceadores de carga.
- Asignar nombres a las IPs públicas de las instancias de redBorder.
- Asignar nombres a las IPs privadas de las interfaces de sincronismo de las instancias de redBorder, que sólo son resueltos por las propias instancias.

En el caso de los balanceadores, se asigna de forma manual o mediante el servicio de despliegue automático AWS Cloudformation, del cuál se hablará más adelante. En el caso de las instancias, utilizarán un script que utilizando la API de Route 53 registre automáticamente el nombre de cada instancia en el servicio.

El nombre que registran tiene el siguiente formato:

Instance-id.cdomain

Donde:

- Instance-id es el identificador de la instancia de Amazon EC2 (obtenido mediante el servicio de metadatos de Amazon).
- Cdomain es el nombre de la Hosted Zone en el que se registra la entrada, y que debe pasársele a la instancia mediante el User-Data, quedando de la siguiente manera;

  ```
  CDOMAIN="example.com"
  ```

En el Anexo VI se encuentra el código de este script y documentación sobre su funcionamiento.

6.5 AWS Simple Storage Service (S3)

6.5.1 Introducción al servicio

Es un servicio web de almacenamiento de objetos con capacidad virtualmente ilimitada, de alta disponibilidad y fiabilidad, completamente gestionado por AWS. Funciona como un servicio web en el que se pueden subir, eliminar y modificar ficheros, y se puede utilizar para cualquier aplicación que necesite almacenamiento de ficheros.

S3 se organiza en Buckets, que son directorios que se gestionan de manera independiente. Se puede limitar el acceso a un bucket para que sólo ciertos usuarios tengan acceso, o se puede publicar para su acceso a través de Internet, pudiéndose utilizar para servir contenido estático o simplemente para permitir descargas de ficheros a clientes.

6.5.2 Integración con redBorder

redBorder utiliza un bucket en Amazon S3 para almacenar información de varios servicios que necesitan de

almacenamiento fiable o que necesitan que la información sea accesible por múltiples instancias. Para que las instancias puedan acceder al bucket, deben conocer una serie de parámetros pasados por user-data. Son los siguientes.

- S3TYPE: le indica a la instancia que debe utilizar el servicio S3 de Amazon. En caso contrario utilizará el servicio Riak implementado por redBorder. Los valores admitidos son aws y riak.
- S3BUCKET: nombre del bucket que deben utilizar los nodos del clúster. Toda la información será almacenada y leída de ese bucket.
- S3HOST: url que deben configurar los servicios como punto de acceso al servicio de S3.

A continuación se explicará el uso que le da cada servicio.

6.5.2.1 Chef

Chef Server utiliza S3 para almacenar los cookbooks y las plantillas. De esta forma se pueden tener varias instancias de Chef Server en alta disponibilidad sin problemas en el sincronismo de las plantillas, ya que se encuentran en un medio compartido de alta disponibilidad.

Cuando se necesita actualizar la configuración del clúster, Chef Server actualiza las plantillas en S3 y Chef client configurará los nodos con esa configuración accesible por todos los nodos del clúster.

6.5.2.2 Druid

Cuando los MiddleManagers terminan de procesar los eventos más recientes y pasan a no ser de tiempo real, los eventos son almacenados en S3 y pasan a ser gestionados por los nodos Historicals. De esta forma, la información de monitorización de los clientes pasa a este medio de alta disponibilidad y fiabilidad, evitando posibles pérdidas de información incluso en caso de que se produjera un desastre que hiciera caer el clúster de redBorder.

Además, la capacidad virtualmente infinita de S3 nos permite almacenar información con toda la antigüedad que sea necesaria, garantizando que nunca va a faltar espacio para almacenarla.

6.5.2.3 Logs

Algunos logs de aplicaciones son también enviados a S3 para poder analizarlos en caso de que hubiera problemas con la instancia en la que corre el servicio, y así poder determinar el error.

6.6 AWS Relational Database Service (RDS)

6.6.1 Introducción al servicio

RDS es un servicio de base de datos relacionadas gestionado por Amazon. Permite desplegar instancias con distintos motores de bases de datos, ofreciendo funcionalidades de gestión automatizadas.

A continuación mencionamos las características principales que ofrece este servicio:

- Autoconfiguración de la base de datos y automatización de operaciones de gestión como actualizaciones, copias de seguridad y recuperación automática ante errores.
- Posibilidad de realizar un despliegue multi-AZ, es decir, en varias zonas de disponibilidad, siguiendo el paradigma de alta disponibilidad Activo/Pasivo.
- Soporte para los siguientes motores de bases de datos: PostgreSQL, MySQL, Oracle, Microsoft SQL Server y Amazon Aurora DB Engine (MySQL-compatible).
- Monitorización detallada con métricas específicas en Amazon Cloudwatch (explicado en el punto 0).

El servicio está compuesto por los siguientes elementos:

- **Instancias de base de datos**: como su nombre indica, son las instancias que implementan la base de

datos. AWS no permite el acceso por SSH a estas instancias, ya se encarga de su gestión. Las aplicaciones pueden acceder a la base de datos mediante una url como si estuviera implementado en un servidor cualquiera.

- **Grupos de seguridad**: pueden ser asociados a las bases de datos para permitir el acceso únicamente a ciertas IPs.
- **Grupos de parámetros**: contiene la configuración de la instancia de la base de datos. Se provee al servicio en el momento de la creación y puede ser modificada en cualquier momento, encargándose el servicio de realizar los cambios pertinentes.
- **Grupos de opciones**: opciones extra que soportan algunos motores de bases de datos para facilitar su gestión y mantenimiento.

RDS simplemente automatiza las labores de gestión de la base de datos. El uso de ésta cae bajo responsabilidad del usuario, que la maneja como cualquier base de datos relacional convencional.

6.6.2 Integración con redBorder

Varios servicios redBorder utilizan la base de datos PostgreSQL para almacenar tablas de valores y metadatos. El sistema implementa su propia base de datos, pero se puede elegir utilizar la base de datos gestionada por AWS. Para ello se utilizan los siguientes parámetros del User-Data:

- SQLHOST: para indicar la url y el puerto de la base de datos a la que debe conectarse.
- SQLUSER: usuario con permisos de administración de la base de datos, para que la instancia pueda configurar las tablas, permisos y usuarios que utilizarán las aplicaciones de redBorder.
- SQLPASSWORD: contraseña del usuario con permisos de administración.
- SQLDB: nombre de la base de datos.

Chef configura los servicios de redBorder para utilizar esta base de datos como si fuera cualquier base de datos PostgreSQL. Los principales servicios que la usan son la Web, Druid y el propio Chef.

6.7 AWS Elasticache

6.7.1 Introducción al servicio

Elasticache es un servicio de caché distribuido completamente gestionado por Amazon, que permite almacenar y acceder a parejas clave/valor rápidamente, ya que se almacenan en la memoria RAM de la instancia. La gestión del servicio es similar a la de AWS RDS, ya que permite desplegar instancias con servicios de caché preconfigurados y que son gestionados por AWS.

Los motores de caché en memoria soportados son Memcached y Redis. En ambos casos, se puede disponer de un clúster de nodos, permitiéndose la adición o eliminación de ellos en cualquier momento, siendo la configuración de los nodos automática.

El servicio está compuesto por los siguientes elementos:

- **Nodos de caché**: son las instancias que implementan el servicio de caché en memoria. No se permite el acceso por SSH, ya que están gestionadas por AWS.
- **Clúster de Elasticache**: es el conjunto de nodos de caché que ofrecen el servicio de forma coordinada. Todos los nodos deben tener la misma configuración y utilizar el mismo motor de caché.
- **Parámetros del grupo**: configuración del clúster. Permite especificar el número de nodos deseados y el tipo de nodo (similar a los tipos de instancias de EC2).

En el caso de Memcached, AWS ofrece un cliente especial que es capaz de detectar cambios en el clúster de Memcached, y reconfigurarse para utilizar distintos nodos. Esto lo hace siguiendo el procedimiento que se indica en la figura:

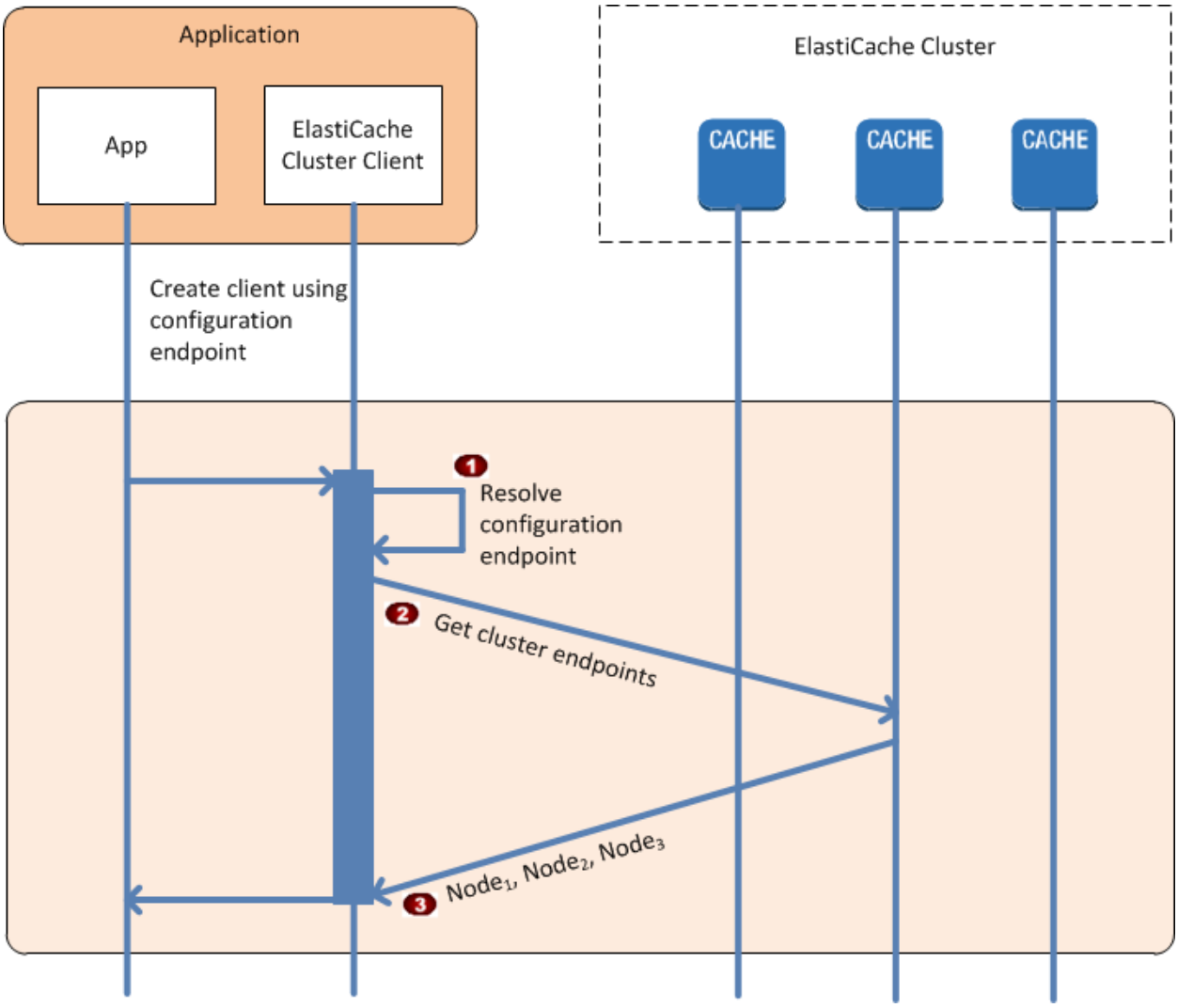

Figura 12 Autodescubrimiento de nodos de Memcached en AWS Elasticache

El cliente utiliza una url de configuración para averiguar qué nodos del clúster están activos, para así dar soporte a posibles cambios en el clúster.

6.7.2 Integración con redBorder

redBorder utiliza Memcached como caché distribuido. Al igual que con PostgreSQL, el propio sistema de redBorder implementa el servicio, pero se ha adaptado para que también pueda utilizar el servicio AWS Elasticache.

Sin embargo, dado que muchos servicios de redBorder implementan su propio cliente de Memcached, no se ha podido utilizar el cliente que ofrece AWS, teniéndose que utilizar un método alternativo para el descrubrimiento de los nodos del clúster de caché. Lo que se hace es consultar a la API de AWS las url de los nodos de Memcached existentes en un clúster, haciendo que Chef lo configure de forma automática.

La consulta se realiza utilizando la CLI de AWS, con el siguiente comando:

```
aws elasticache describe-cache-clusters –show-cache-node-info –cache-
cluster-id (ID DE CLUSTER) | jq –r
.CacheClusters[].CacheNodes[].Endpoint.Address | sed
'a;N;$!ba;s/\n/,/g')
```

El comando devuelve las url de los nodos del clúster separadas por comas, para que Chef pueda configurar los servicios que requieren Memcached. Para que el comando funcione hay que pasarle el identificador del clúster, utilizando el User-Data para ello:

```
ELASTICCACHECLUSTERID=(Identificador del clúster)
```

Cabe destacar que este método de obtención de los nodos del clúster no soporta el añadir o quitar instancias del clúster de Memcached, porque no es un método de configuración dinámico.

6.8 AWS Cloudwatch

6.8.1 Introducción al servicio

AWS Cloudwatch es el servicio de monitorización de Amazon Web Services. Nos basaremos en el siguiente esquema para ver su arquitectura y funcionalidad:

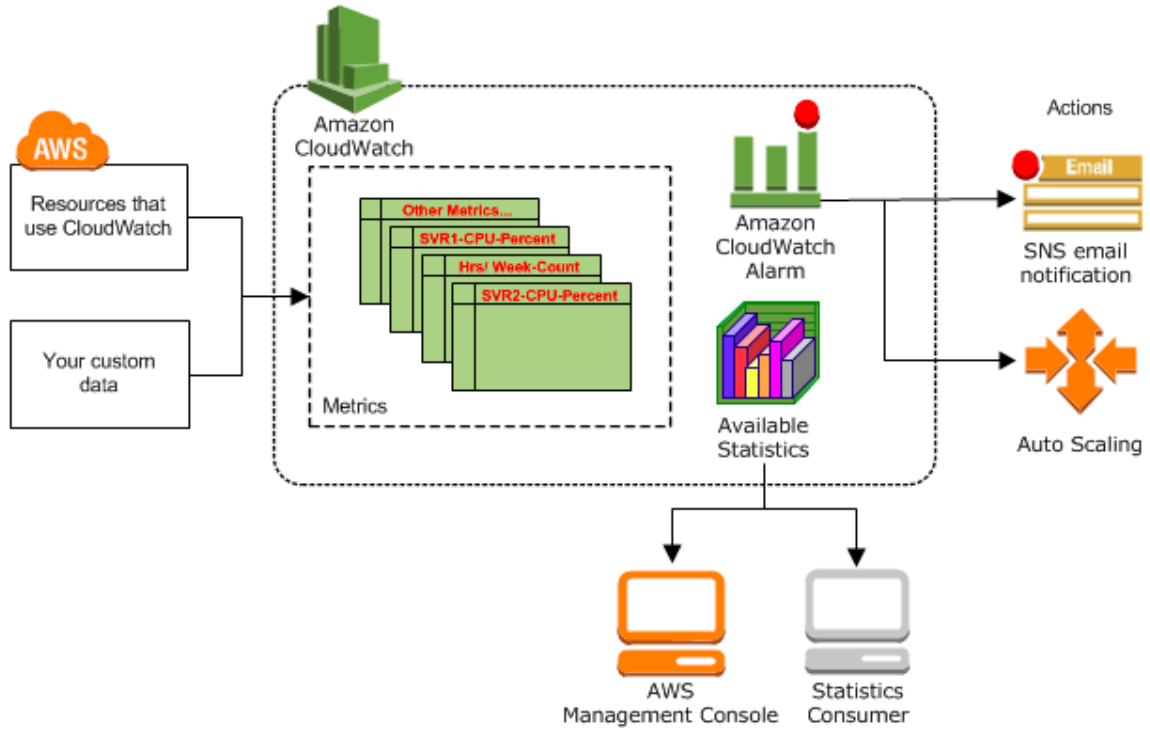

Figura 13 Arquitectura de AWS Cloudwatch [17]

Cloudwatch registra y almacena información de monitorización, clasificada en métricas, tanto de servicios de Amazon Web Services (como EC2, RDS, S3, etc.), como de origen personalizado enviado a través de la API de Amazon Cloudwatch.

Nada más registrar la información, queda disponible para su visualización tanto desde la consola de AWS como desde aplicaciones de terceros que utilicen la API de Cloudwatch.

Finalmente, también se ofrece un servicio de alarmas que monitoriza las métricas seleccionadas y permite automatizar acciones para distintos estados de dicha alarma.

6.8.1.1 Conceptos básicos

Explicamos ahora los conceptos básicos que se manejan en este servicio:

- **Métrica**: representa un conjunto de muestras ordenadas con respecto al tiempo. Los datos pueden provenir tanto de otros servicios de Amazon como por publicaciones del usuario mediante la API.
- **Namespace**: es un contenedor de métricas, que permite organizarlas y aislarlas de otros contenedores.
- **Dimensión**: son parejas clave/valor que ayudan a la identificación unívoca de la métrica.
- **Marca de tiempo**: cada muestra tiene asociada una marca de tiempo que identifica a qué momento corresponde. Esta marca puede señalar un instante entre dos semanas en el pasado y 2 horas en el futuro. En caso contrario, la muestra se elimina.

- **Unidad**: representa la unidad de las muestras en una métrica.
- **Estadística**: son agregaciones de datos de una métrica durante ciertos periodos de tiempo[35].
- **Periodo**: es la longitud de tiempo asociada a una estadística. Las muestras que se encuentran en ese periodo se resumen en una sola, cuyo valor depende del tipo de estadística que sea.
- **Alarma**: permite iniciar acciones cuando una métrica cruza cierto umbral durante un periodo de tiempo.

6.8.1.2 Acciones provocadas por alarmas

Una de las características más valiosas de Cloudwatch es que permite realizar una serie de acciones cuando se produce una alarma. Las acciones soportadas son las siguientes:

- Envío de notificaciones a un Topic de AWS SNS[36].
- Realizar acciones sobre instancias de Amazon EC2. Las acciones soportadas son parar, terminar, reiniciar o recuperar una instancia.
- Realizar acciones de autoescalado. Se puede hacer que el grupo de autoescalado crezca o disminuya cuando se produzca una alarma.

6.8.2 Integración con redBorder

redBorder ofrece tanto la posibilidad de monitorizar el clúster mediante su propia web como la posibilidad de monitorizarlo a través de AWS Cloudwatch.

Cloudwatch monitoriza de forma automática algunos parámetros de las instancias, como el uso de CPU, uso de E/S, etc., ya que es información que puede ser extraída desde fuera de la instancia. Sin embargo hay otros parámetros, como el uso de RAM o disco que deben ser enviados por la propia instancia, utilizando los scripts que provee Amazon[37].

6.8.2.1 Scripts de monitorización de Amazon

Amazon provee unos scripts que envían métricas internas de una instancia de EC2 de forma automática. Están escritos en Python y utilizan la API de Amazon Cloudwatch para el envío y recepción de las métricas.

redBorder sólo emplea la opción de envío de métricas a la plataforma de Cloudwatch, y utiliza la herramienta CRON de Linux para ejecutar el script cada 5 minutos. Se usa para monitorizar los siguientes parámetros:

- Memoria RAM usada
- Tamaño de los siguientes directorios: /, /opt/rb/data/aggregated, /opt/rb/data/raw
- Espacio utilizado de disco

La línea de configuración de cron sería la siguiente:

```
*/5 * * * * root /opt/rb/var/aws-scripts-mon/mon-put-instance-data.pl
    --mem-util --disk-path=/ --disk-path=/opt/rb/data/aggregated
    --disk-path=/opt/rb/data/raw --disk-space-util --from-cron
```

6.8.2.1.1 Envío de métricas propias de redBorder

redBorder también soporta el envío de métricas propias a Cloudwatch, para lo cual se ha desarrollado una herramienta: **rb_cloudwatch**.

---

[35] Las agregaciones de datos soportadas por Cloudwatch pueden ser consultados en su documentación [17].
[36] AWS SNS (Simple Notification Service): servicio de notificaciones de Amazon Web Services. Será explicado en el punto 6.10.
[37] Amazon explica el procedimiento para la instalación y configuración de los scripts de monitorización de instancias EC2 en la documentación de AWS Cloudwatch [17].

Para entender su funcionamiento haremos una introducción al sistema de monitorización interno de redBorder, llamado rb_monitor, basándonos en el siguiente esquema.

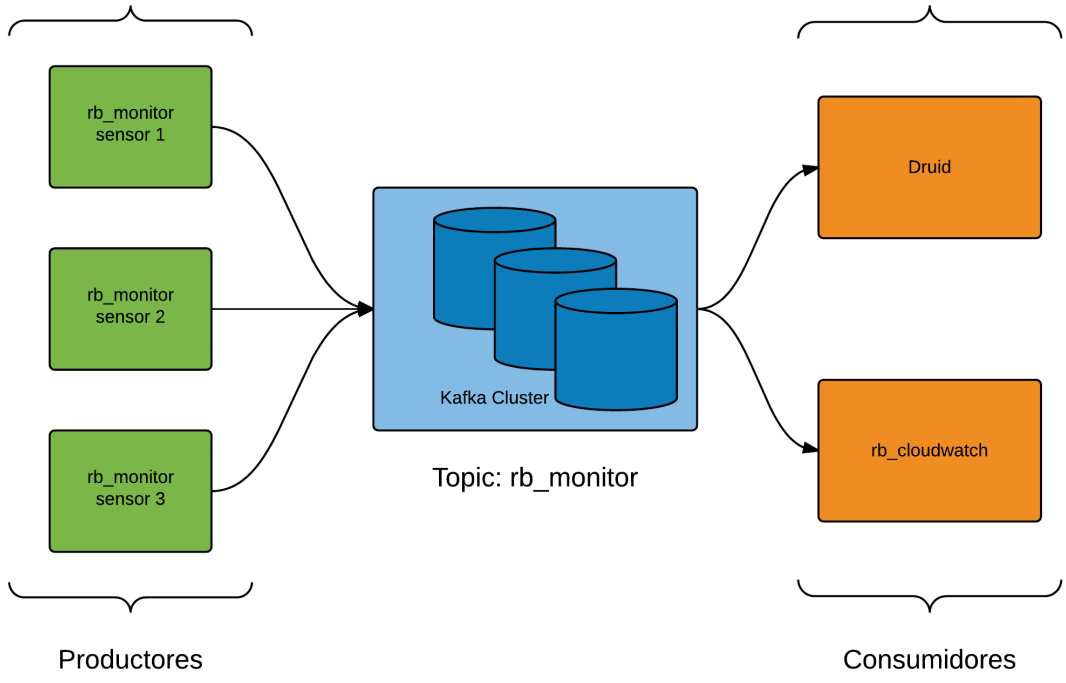

Figura 14 Arquitectura de rb_monitor

Rb_monitor consta de una serie de sensores que registran información de monitorización de distintos elementos, sistemas y servicios, y envían mensajes en formato JSON a un Topic de Apache Kafka, servicio que se encuentra implementado en el sistema redBorder.

Una vez almacenada en el sistema de colas, los eventos de monitorización son consumidos por distintos servicios. Los principales son **Druid**, que indexa los eventos y organiza la información para su posterior presentación en la web de redBorder, y **rb_cloudwatch**, que se encargará de enviar la información a Amazon Cloudwatch.

El servicio rb_cloudwatch tiene la siguiente arquitectura:

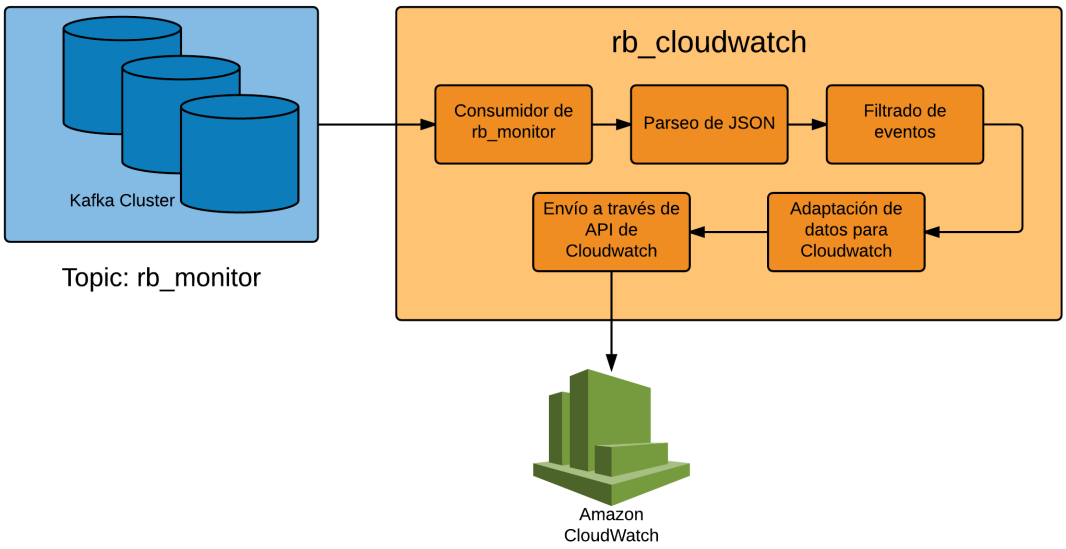

Figura 15 Estructura básica de AWS Cloudwatch

El servicio simplemente consume los mensajes que se envían al topic rb_monitor, los filtra para enviar a AWS

sólo las métricas que se necesiten y adapta la información para poder enviarla a través de la API de Amazon Cloudwatch. Este proceso se ejecuta en varios hilos y pueden incluso ejecutarse varias instancias de rb_cloudwatch en varias máquinas para garantizar alta disponibilidad.

En el Anexo V se encuentra la documentación y el código de este servicio.

6.9 AWS Simple Queue Service (SQS)

6.9.1 Introducción al servicio

SQS es un servicio de colas de mensajes rápido, fiable, escalable y totalmente gestionado por Amazon, que nos permite comunicar procesos y servicios de forma asíncrona. Permite gestionar cualquier volumen de mensajería, sin necesidad de preocuparnos por mantener un clúster ni escalarlo.

A continuación mostramos las principales características del servicio:

- Garantía de entrega del mensaje: SQS ofrece garantía de entrega de tipo "at least once", que significa que garantiza la entrega de al menos una copia del mensaje al destinatario, y también ofrece alta disponibilidad de envío y recepción de mensajes.
- Escritores y lectores múltiples: pueden haber varios clientes enviando y recibiendo mensajes en la misma cola al mismo tiempo.
- Configuración por cola: se permite configurar cada cola creada de forma independiente.
- Tamaño de mensaje de hasta 256 KB.
- Control de acceso mediante Amazon IAM.
- Liberación retardada de mensajes: se permite asignar un retraso a la liberación de mensajes.

6.9.1.1 Parámetros de configuración de una cola SQS

En función de la configuración de la cola varía el tratamiento de los mensajes que se envían y se reciben de ésta. Los parámetros que se pueden configurar son los siguientes:

- **Tiempo de visibilidad por defecto**: cuando un mensaje es solicitado por un cliente, éste se oculta el tiempo indicado por este parámetro para evitar que sea procesado más de una vez. El cliente que envía el mensaje a la cola puede elegir este tiempo, y en caso de no hacerlo, se aplica el valor indicado en el parámetro por defecto.
- **Periodo de retención de mensajes**: es el tiempo máximo que se mantiene un mensaje en la cola. Si un mensaje no ha sido borrado cuando ha pasado este tiempo, se elimina.
- **Tamaño máximo de mensaje**: Indica el tamaño máximo permitido en esta cola. En ningún caso puede superar los 256 KB.
- **Retardo de liberación**: es el tiempo que se espera cuando se ha publicado un mensaje para que esté disponible para los consumidores.
- **Tiempo de espera para la recepción de mensajes**: es máximo tiempo que un cliente puede estar esperando a que haya mensajes en la cola en una sola solicitud a la API.

6.9.1.2 Ciclo de vida de los mensajes

La siguiente figura representa el ciclo de vida de los mensajes en SQS, que se explica a continuación:

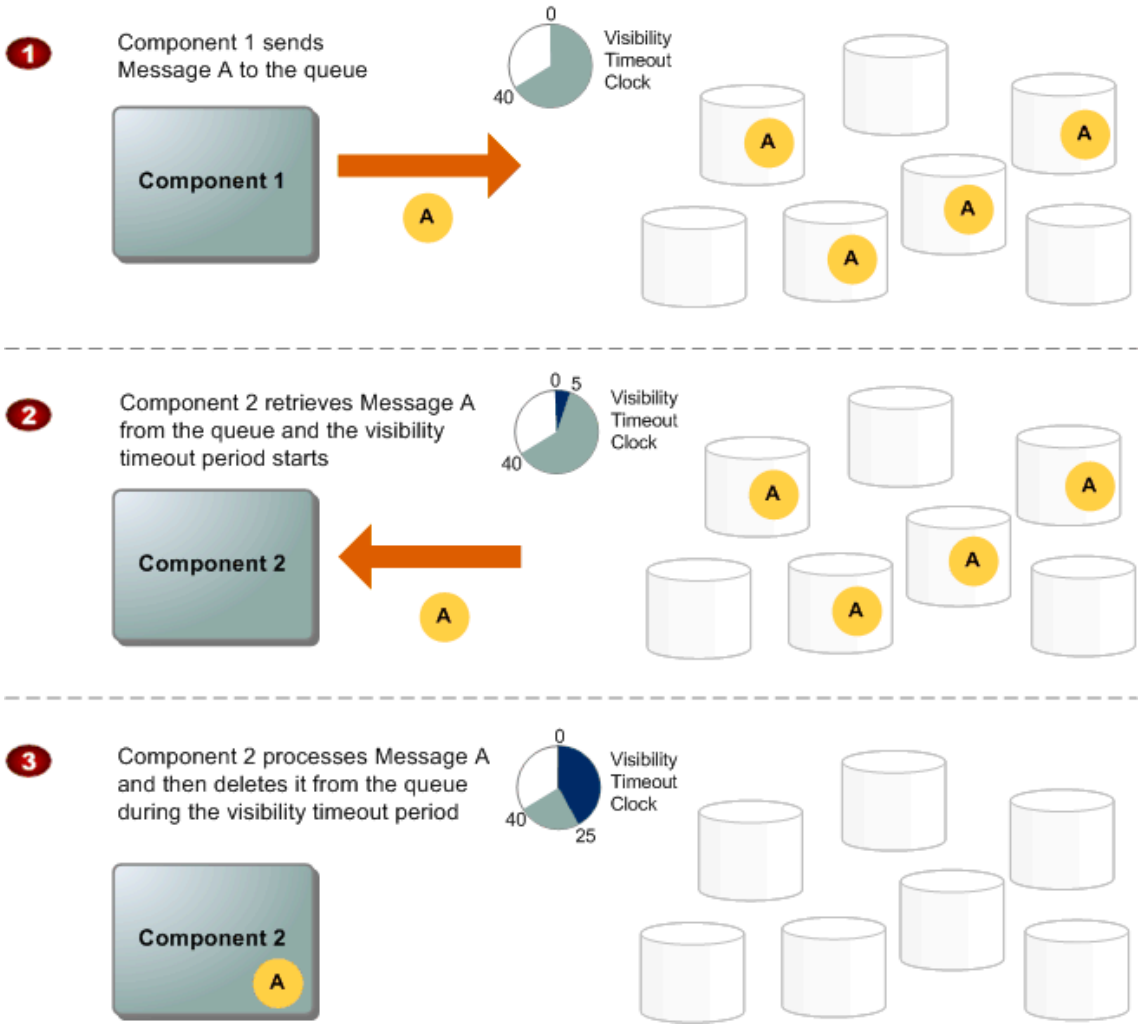

Figura 16 Ciclo de vida de los mensajes SQS

- El componente 1 (productor) envía un mensaje a la cola SQS y se almacena en el sistema de forma redundante.
- El componente 2 (consumidor) obtiene un mensaje de la cola SQS y comienza el periodo de visibilidad, quedando el mensaje oculto.
- El componente 2 (consumidor), procesa el mensaje y lo borra del sistema antes de que termine el periodo de visibilidad. En caso contrario, podría haber sido procesado más de una vez.

6.9.2 Integración con redBorder

El sistema redBorder tiene su propio sistema de colas de mensajes (Apache Kafka), por lo que no necesita SQS para las funciones principales del sistema. Sin embargo, redBorder utiliza una cola SQS para recibir mensajes de otros servicios de AWS que son necesarios para la gestión del clúster en AWS. Concretamente, se utiliza SQS para:

- Recibir mensajes generados por grupos de autoescalado que tengan los Lifecycle Hooks activados y poder realizar las operaciones necesarias.
- Recibir mensajes de alarma de Amazon Cloudwatch (a través de AWS SNS, explicado más adelante), que provocan la realización de operaciones de gestión del autoescalado para ciertos roles de redBorder.

Para ello se ha desarrollado un servicio que consume esos mensajes y, en función del mensaje recibido, ejecuta una serie de operaciones. El servicio se llama **rb_sqsld**, y está implementado en Ruby.

Se utiliza el AWS SDK for Ruby, que contiene una serie de herramientas para facilitar el uso y la

comunicación con los servicios de Amazon Web Services. A continuación mostramos un esquema básico de su funcionamiento:

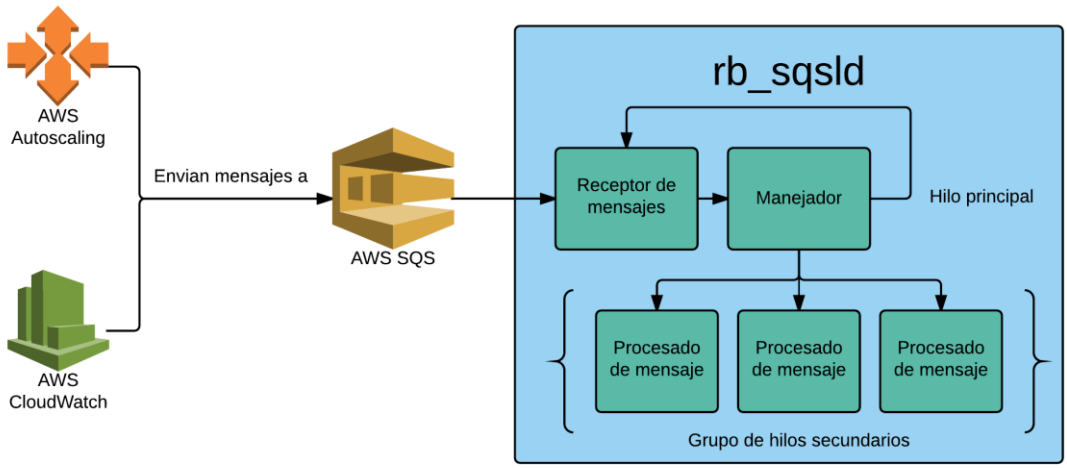

Figura 17 Esquema básico del funcionamiento de rb_sqsld

El servicio rb_sqsld corre en varios hilos. El hilo principal se encarga de leer constantemente de la cola SQS. Cuando llega un mensaje, lo parsea (están en formato JSON), y en función de su contenido decide cómo se debe procesar el mensaje. El procesado de los mensajes se realiza en otros hilos para que el hilo principal pueda seguir a la espera de mensajes inmediatamente. Estos hilos secundarios realizan la tarea necesaria en función del mensaje y mueren cuando finalizan el procesado.

En el capítulo 7, se explica más profundamente el funcionamiento de este servicio y se explica por qué es necesario. Además, en el Anexo VII se encuentra la documentación del servicio.

6.10 AWS Simple Notification Service (SNS)

6.10.1 Introducción al servicio

Amazon SNS es un servicio web que coordina y gestiona el envío de mensajes y notificaciones a los subscritores del servicio. Hay dos tipos de clientes de SNS:

- **Publicadores**: envían mensajes al servicio SNS para que sea difundido a todos los subscritores del servicio.
- **Subscritores**: reciben los mensajes que los publicadores han enviado al Topic de SNS.

De esta forma, SNS permite desacoplar el envío y la recepción de los mensajes, convirtiéndolo en un proceso asíncrono. En la figura se ilustra este proceso y se muestran los tipos de subscritores que son soportados por el servicio.

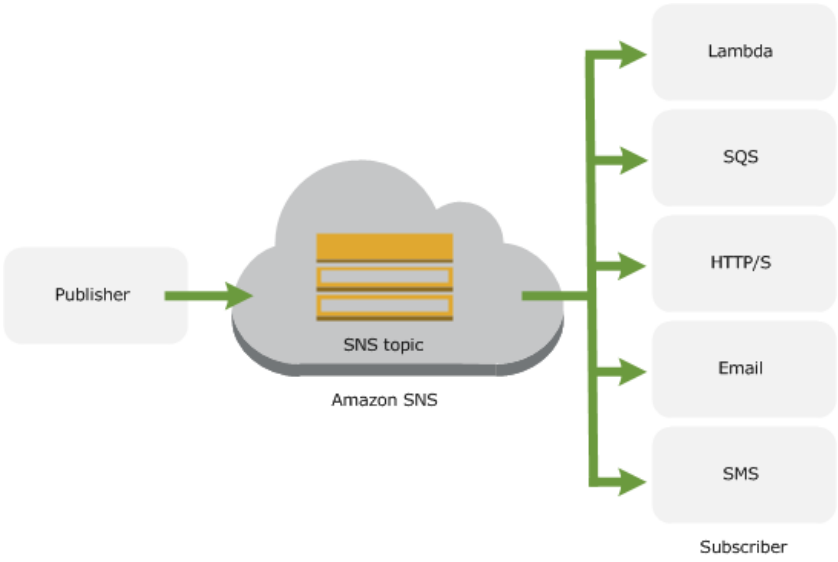

Figura 18 Arquitectura de AWS SNS [18]

A continuación se hace referencia a los subscritores soportados:

- **Lambda**: es un servicio de AWS que permite ejecutar funciones en Node.js o Java sin necesidad de mantener instancias para su ejecución. De esta tarea ya se encarga AWS Lambda, así como de su escalabilidad. Uno de los disparadores que provocan la ejecución de una función de Lambda es un mensaje de SNS.
- **SQS**: se pueden subscribir colas SQS a las notificaciones de SNS, de tal forma que el mensaje es encolado para el posterior procesado por parte de instancias que lean de la cola.
- **HTTP/S**: se pueden enviar notificaciones a endpoints HTTP o HTTPS que procesen el mensaje.
- **Email**: las notificaciones pueden ser enviadas por email, para avisar a los gestores de eventos que se produzcan en la cloud.
- **SMS**: también válido para el envío de notificaciones.

Además también se puede utilizar SNS para el envío de notificaciones Push a distintas plataformas, mostradas en la figura:

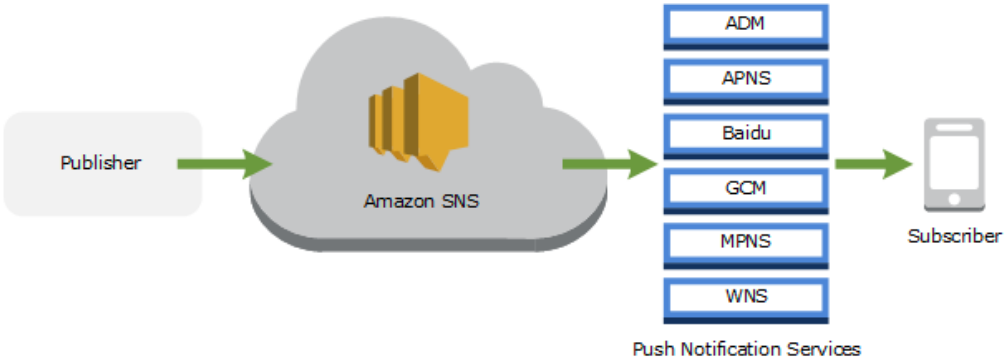

Figura 19 Notificaciones Push de SNS [18]

6.10.2 Integración con redBorder

En redBorder se utiliza AWS SNS para lo siguiente:

- Enviar notificaciones de alarmas del AWS Cloudwatch a los gestores del clúster. Cabe destacar que gracias al servicio rb_cloudwatch, se pueden configurar alarmas y configuraciones sobre métricas personalizadas del clúster.
- Enviar notificaciones a una cola SQS, para que las instancias que leen de la cola ejecuten acciones en

respuesta a la notificación (utilizando el servicio rb_sqsld).

Para ello, se debe crear primero un Topic de SNS, que no requiere ninguna configuración salvo elegir un nombre. Una vez creado, se asignan los subscritores.

Para poder enviar notificaciones a e-mails, es necesario registrar las direcciones de e-mail deseadas como subscritoras de un Topic de SNS. El servicio enviará un e-mail de confirmación de subscritor, y si se acepta, comenzará a recibir las notificaciones enviadas por ese Topic.

Por otro lado, para el envío de las notificaciones a SQS, se deben configurar ambos servicios. En primer lugar se debe configurar SQS para que acepte mensajes entrantes del servicio SNS, mediante la asignación de una política de permisos a la cola SQS. Para ello se debe tener en cuenta lo siguiente:

- Cuando SNS envía un mensaje no consta que se envía desde la cuenta de Amazon Web Services del creador del Topic, sino desde una cuenta especial para dicho servicio, cuyo identificador no es conocido.
- Lo único que necesita SNS es permiso para enviar mensajes a SQS, ya que no va a recibir ningún mensaje de la cola.
- Todo servicio o recurso de Amazon tiene asociado un ARN (Amazon Resource Name), que lo identifica unívocamente.

La política entonces sería la siguiente:

Tabla 8 Política de SNS para redBorder

Efecto	Permitir
Cuentas	Todas
Acciones	Enviar mensajes a SQS
Condiciones	ARN de origen = ARN del Topic de SNS

Dado que no conocemos la cuenta SNS desde la que se envía el mensaje, se permiten todas, pero con la condición de que el ARN del servicio que envía el mensaje sea el del Topic de SNS.

Una vez configurada la política, podemos añadir a SNS la subscrición de la cola SQS. Esto se hace especificando el ARN de la cola SQS, que la identifica unívocamente.

Resumiendo, lo que se consigue es que las notificaciones también puedan ser procesadas por el servicio **rb_sqsld**, ya que terminan siendo mensajes de la cola SQS.

6.11 AWS Identity and Access Management (IAM)

6.11.1 Introducción al servicio

Amazon IAM es un servicio web que ayuda a gestionar el control de acceso a los recursos de Amazon Web Services para los distintos usuarios de la cuenta. Permite controlar quién utiliza los recursos de AWS (autenticación) y de qué forma (autorización).

Para ello, este servicio cuenta con los siguientes elementos:

- **Usuarios**: IAM permite la creación de usuarios con credenciales de acceso propias, a los cuáles se les puede asociar los permisos que sean necesarios. Cuando se crea una cuenta de Amazon Web Services, automáticamente se crea un usuario raíz con acceso total a los recursos de la cuenta, aunque se

recomienda utilizarla únicamente para la creación de otras cuentas de usuario con las que manejar Amazon Web Services.

- **Permisos y políticas**: por defecto, un usuario no raíz no tiene acceso a nada en la cuenta. Es necesario otorgar permisos creando una política, que lista las acciones que el usuario puede realizar y sobre qué recursos.
- **Grupos**: es posible organizar a los usuarios en grupos, de forma que al asignar una política a dicho grupo se le asigna a su vez a todos los usuarios asociados.

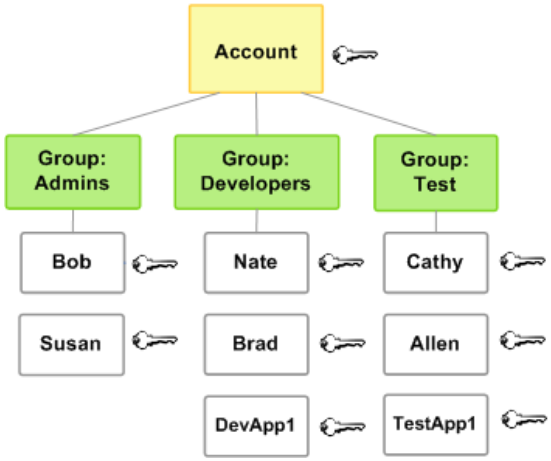

Figura 20 Organización por grupos de usuarios en AWS IAM [19]

- **Access Keys**: los usuarios pueden tener asociadas unas claves que permitan el acceso a los recursos utilizando la API de Amazon Web Services. Estas claves pueden ser regeneradas cuando sea necesario a fin de controlar su distribución, ya que pueden ser utilizadas por aplicaciones.
- **Roles**: pueden ser asociados a instancias u otras entidades para otorgarles permisos para la realización de operaciones con la API de AWS, sin necesidad de proveerles de claves de acceso.

6.11.1.1 Formato de las políticas

Las políticas se especifican mediante un documento en formato JSON. Cada política consta de los siguientes campos:

- Action: indica qué acciones están permitidas o denegadas por la política.
- Effect: indica si la acción es permitida o denegada.
- Resource: indica sobre qué recursos actúa la política. Para indicar el recurso se utiliza el ARN (Amazon Resource Number) único del recurso, o el símbolo '*' para indicar que se refiere a todos. Muchas acciones no soportan elegir el recurso sobre el que actúan.

Un ejemplo de política sería la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::example-bucket/*"
  }
}
```

Figura 21 Ejemplo de política de AWS IAM

En este ejemplo la política permite realizar cualquier acción de AWS S3 sobre el bucket indicado mediante su ARN.

6.11.1.2 Permisos a nivel de recursos

AWS IAM permite limitar las acciones que puede realizar un usuario, pero sin embargo, sólo se puede limitar el acceso a recursos para ciertas acciones. Poco a poco se va mejorando el soporte para limitar que un usuario sólo pueda acceder a ciertos recursos para una acción, pero aún está muy incompleto.

Por ello en las políticas, en muchas ocasiones, hay que elegir en el parámetro "resource" el símbolo '*' que indica "Todos los recursos".

Para ver las acciones y servicios que soportan permisos a nivel de recursos actualmente se puede consultar la documentación de AWS IAM [19].

6.11.2 Integración con redBorder

Las instancias de redBorder que se despliegan en Amazon EC2 necesitan permisos para realizar operaciones, sobre todo para la autoconfiguración de servicios de AWS asociados al clúster. Deben tener por lo tanto un usuario o un rol asociado con los permisos necesarios.

Dado el sistema tiene aplicaciones que no son compatibles con los roles de AWS IAM, se han proveído las credenciales mediante una Access Key y una Secret Key, que están asociados a un usuario. También se le asocia el rol con los mismo permisos, ya que se requieren en el proceso de configuración de redBorder antes de que se hayan configurado las credenciales en la instancia.

Además, al servicio Chef Server se le han proveído de otras credenciales aparte más restrictivas, ya que las utiliza para configurar sensores externos al sistema que está implementado en AWS.

Se le otorgan a las instancias EC2 los siguientes permisos, en el caso de ser desplegados utilizando el servicio AWS CloudFormation, que será explicado en el siguiente punto:

- Creación y asignado de interfaces de red, necesario para la creación de la segunda interfaz de red de las instancias en grupos de autoescalado.
- Obtención de información sobre instancias, para conocer su estado.
- Asignación y eliminación de Ips privadas secundarias para alta disponibilidad utilizando KeepAlive[38].
- Acceso a un bucket de S3 para almacenar información de distintos servicios de redBorder.
- Descripción de la pila de AWS CloudFormation en la que se ha creado la instancia.
- Envío de métricas al servicio AWS Cloudwatch.
- Obtención de estado de alarmas asociadas al clúster.
- Creación de de HostedZones y de entradas en ellas.
- Gestión de instancias en grupos de autoescalado.
- Recepción de mensajes de una cola SQS.
- Envío de logs a AWS Cloudwatch.

Esto se consigue mediante la siguiente política:

[38] KeepAlive: servicio para la monitorización de estado de otros servicios y conmutación activo/pasivo entre distintas instancias de éste.

```
{
  "Version" : "2012-10-17",
  "Statement": [ {
    "Action" : [
      "ec2:CreateNetworkInterface",
      "ec2:AttachNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute",
      "ec2:DescribeInstances",
      "ec2:AssignPrivateIpAddresses",
      "ec2:UnassignPrivateIpAddresses" ],
    "Effect" : "Allow",
    "Resource" : "*"
  }, {
    "Action" : "s3:*",
    "Effect" : "Allow",
    "Resource" : (ARN DEL BUCKET)
  }, {
    "Effect": "Allow",
    "Action": [ "cloudformation:DescribeStackResource" ],
    "Resource": (ARN DE LA PILA DE CLOUDFORMATION)
  },{
    "Effect": "Allow",
    "Action" : [ "cloudwatch:PutMetricData", "cloudwatch:DescribeAlarms" ],
    "Resource" : "*"
  },{
    "Effect": "Allow",
    "Action" : [ "route53:ChangeResourceRecordSets", "route53:ListHostedZones" ],
    "Resource" : (ARN DE LAS HOSTED ZONES)
  },{
    "Effect": "Allow",
    "Action" : [ "route53:ListHostedZones", "route53:ChangeResourceRecordSets",
"route53:GetHostedZone" ],
    "Resource" : "*"
  },{
    "Effect": "Allow",
    "Action" : [ "autoscaling:TerminateInstanceInAutoscalingGroup",
"autoscaling:DescribeAutoscalingGroups",
      "autoscaling:CompleteLifecycleAction" ],
    "Resource" : "*"
  },{
    "Effect": "Allow",
    "Action" : [ "sqs:ReceiveMessage", "sqs:DeleteMessage" ],
    "Resource" : (ARN DE LA COLA SQS)
  },{
    "Effect": "Allow",
    "Action" : [ "logs:*" ],
    "Resource" : "*"
  } ]
}
```

El usuario que se utiliza para Chef sólo debe tener acceso al directorio donde se encuentran las plantillas dentro bucket S3 del clúster, por lo que su política es sencilla:

```
{
  "Version" : "2012-10-17",
  "Statement": [ {
    "Action" : "s3:*",
    "Effect" : "Allow",
    "Resource" : (ARN DEL DIRECTORIO DEL BUCKET)
  ] }
}
```

Finalmente mencionar que las Access y Secret Keys son pasadas a las instancias mediante los parámetros del User-Data listados a continuación:

- AWS_ACCESS_KEY
- AWS_SECRET_KEY
- CHEF_AWS_ACCESS_KEY

- CHEF_AWS_SECRET_KEY

6.12 AWS CloudFormation

6.12.1 Introducción al servicio

Este servicio permite automatizar el despliegue de los recursos de AWS necesarios para el funcionamiento de cualquier sistema en la nube de una forma declarativa. Esto quiere decir que el servicio se encarga de desplegar los recursos mediante la descripción de éstos a través de una plantilla.

Con ello se consiguen principalmente tres cosas:

- Automatizar el despliegue de sistemas en Amazon Web Services.
- Posibilidad de replicar un mismo sistema tantas veces como sea necesario.
- Disponer de un documento descriptivo y trazable (plantilla) de toda la infraestructura

CloudFormation maneja dos conceptos fundamentales:

- **Plantilla**: documento en formato JSON que describe los recursos que CloudFormation debe desplegar, y la configuración de éstos.
- **Pila**: agrupación lógica de recursos creados mediante el servicio CloudFormation a partir de una plantilla dada.

Para hacer que Cloudformation cree una pila a partir de una plantilla se debe seguir el procedimiento ilustrado en la figura:

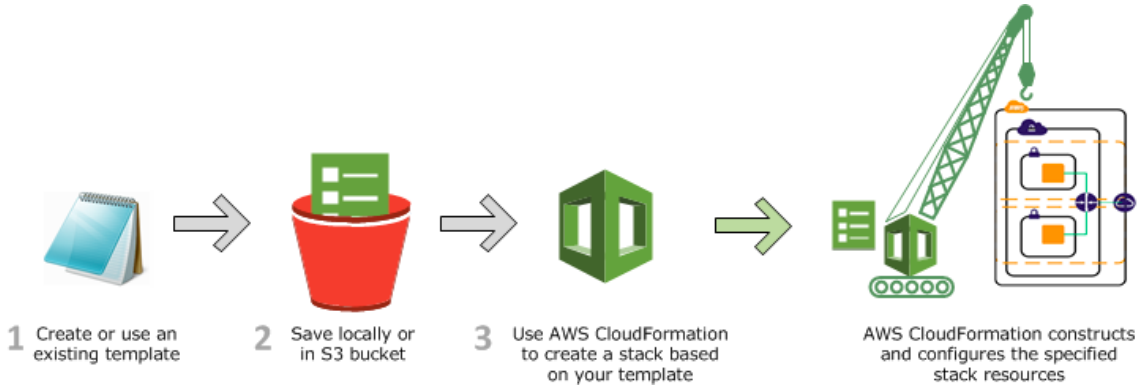

Figura 22 Procedimiento de creación de una pila en CloudFormation

Además de la creación de pilas, soporta también la actualización y eliminación de éstas. Para actualizarlas, simplemente hay que modificar la plantilla y Cloudformation realizará los cambios necesarios para que los recursos de la pila se configuren como se encuentra indicado en la nueva versión de la plantilla.

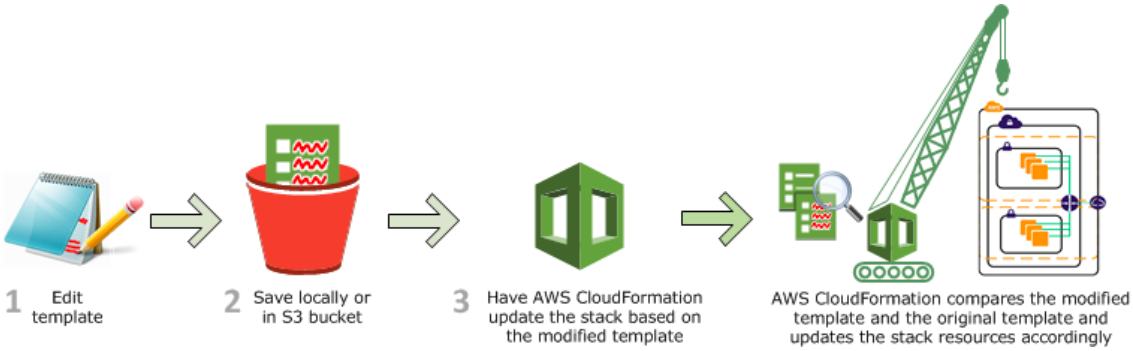

Figura *23 Procedimiento de actualización de una pila en CloudFormation*

Para su eliminación, ya no es necesario incluir ninguna plantilla, CloudFormation se encarga automáticamente

de eliminar los recursos existentes en la pila.

6.12.1.1 Estructura de las plantillas

Las plantillas de CloudFormation tienen la siguiente estructura:

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

Figura 24 Estructura de una plantilla de AWS CloudFormation [20]

A continuación describimos las distintas partes:

- **AWSTemplateFormatVersion**: especifica la versión del formato que se utiliza en la plantilla.
- **Description**: cadena de caracteres que contiene una descripción de la pila que se despliega a partir de la plantilla.
- **Metadata**: código en formato JSON que provee de información adicional sobre la plantilla.
- **Parameters**: especifica valores que se pueden pasar a la plantilla en el momento de la creación de la pila. Estos valores pueden ser referenciados en otras partes de la plantilla.
- **Mappings**: mapas de claves y valores que se usan para obtener valores que cambian en función de parámetros o recursos. Se pueden buscar valores en los mapas mediante funciones que se pueden integrar (se explicarán posteriormente).
- **Conditions**: define condiciones que deciden si se crean o no ciertos recursos, o si se asignan unos valores u otros a las propiedades de los recursos.
- **Recursos**: especifica los recursos que serán creados por CloudFormation y sus propiedades. Es la única parte de la plantilla que es obligatoria.
- **Outputs**: define los valores que serán devueltos como propiedades de la pila, pudiéndose referir a valores de características de recursos.

6.12.1.2 Gestión del ciclo de vida

CloudFormation gestiona el ciclo de vida tanto de la pila completa como de cada uno de los recursos creados, asignándoles en cada momento un estado. Los estados posibles son los siguientes:

- **CREATE_IN_PROGRESS**: indica que el recurso o la pila está en proceso de creación. Se mantendrá hasta que el recurso se encuentre listo para entrar en producción, o se produzca algún error.

- **CREATE_COMPLETE**: indica que el recurso o la pila ha sido creado correctamente y está preparado para entrar en funcionamiento.
- **CREATE_FAILED**: indica que ha habido un error en la creación de la pila o del recurso. Si cualquier recurso alcanza este estado, la pila automáticamente pasará también a este estado, y en función de su configuración iniciará el proceso de eliminación de los recursos ya creados, o se mantendrá en ese estado para analizar los motivos del error.
- **ROLLBACK_IN_PROGRESS**: si la pila alcanza el estado CREATE_FAILED y está configurada para hacer "Rollback", pasará a este estado y eliminará todos los recursos que habían sido creados, ya que la pila se gestiona como una unidad completa y para que esté en funcionamiento, todos los recursos deben haber sido creados correctamente.
- **ROLLBACK_COMPLETE**: estado en el que se mantiene la pila cuando ha finalizado el proceso de "Rollback", a la espera de su eliminación.
- **ROLLBACK_FAILED**: indica que no se ha podido realizar el "Rollback". La pila deja de poder ser actualizada y sólo se permite su eliminación.
- **UPDATE_IN_PROGRESS**: indica que el recurso o la pila está en proceso de actualización debido a un cambio en la plantilla.
- **UPDATE_COMPLETE**: indica que el recurso o la pila ha sido actualizado correctamente.
- **UPDATE_FAILED**: indica que Cloudformation no puedo alcanzar el nuevo estado definido en la plantilla.
- **ROLLBACK_UPDATE_IN_PROGRESS**: si la pila alcanza el estado UPDATE y está configurada para hacer "Rollback", pasará a este estado y eliminará los cambios que se hayan realizado en la nueva versión de la plantilla, dejando la pila como indicaba la versión anterior de ésta.
- **ROLLBACK_UPDATE_COMPLETE**: indica que la pila ha vuelto al estado anterior tras un intento de actualización.
- **DELETE_IN_PROGRESS**: indica que el recurso o la pila está en proceso de eliminación. Cuando se elimina una pila, todos los recursos asociados son eliminados también.
- **DELETE_COMPLETE**: indica que el recurso o la pila ha sido eliminado correctamente. Si la pila presenta este estado, todos sus recursos asociados también han sido eliminados correctamente.
- **DELETE_FAILED**: indica que el recurso o la pila no ha podido ser eliminado.

En el siguiente diagrama se ve cómo cambia el estado de una pila en su ciclo de vida:

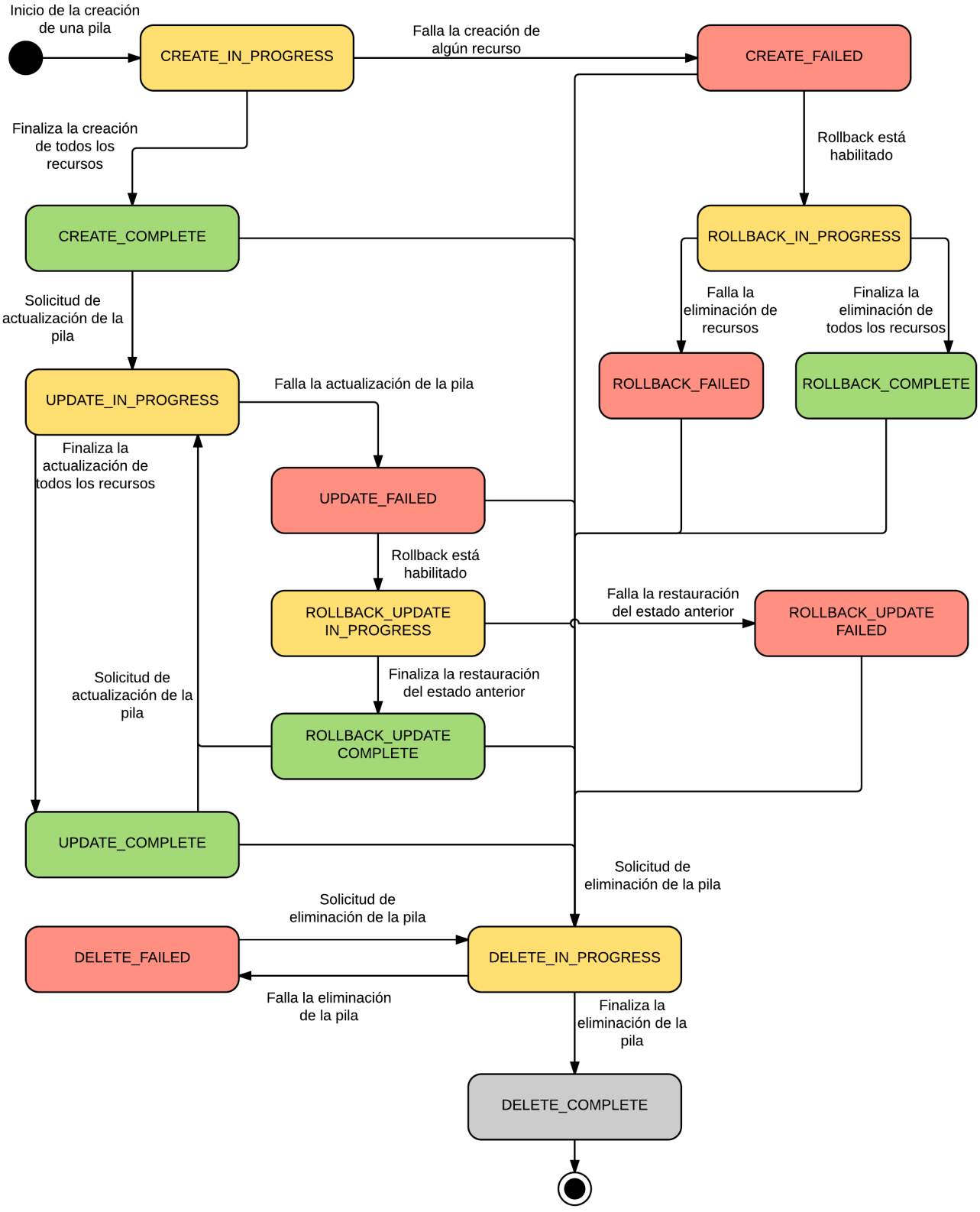

Figura 25 Diagrama de estados de una pila en CloudFormation

A los recursos también se les asigna estos estados. En el caso de los servicios de AWS, CloudFormation nos indica cuando están listos para ser utilizados asignándoles el estado CREATE_COMPLETE. Sin embargo, con las instancias no es tan sencillo, ya que CloudFormation a priori no puede saber cuándo una instancia está preparada.

Por ello, es necesario que la instancia le haga saber a CloudFormation de alguna forma que su configuración y puesta en producción ha finalizado. Esto se hace mediante los CloudFormation Helper Scripts, que

implementan comandos para llamar a la API de CloudFormation y le indican que el recurso ha sido creado y que puede ser pasado al estado CREATE_COMPLETE.

6.12.2 Integración con redBorder

6.12.2.1 Despliegue de redBorder usando CloudFormation

La plantilla de redBorder está diseñada para desplegar un clúster completo con todas las integraciones con AWS incluidas sin necesidad de intervención del usuario una vez elegidos los parámetros de configuración.

Para explicar cómo CloudFormation despliega un clúster de redBorder utilizaremos un esquema general de la arquitectura final en Amazon Web Services:

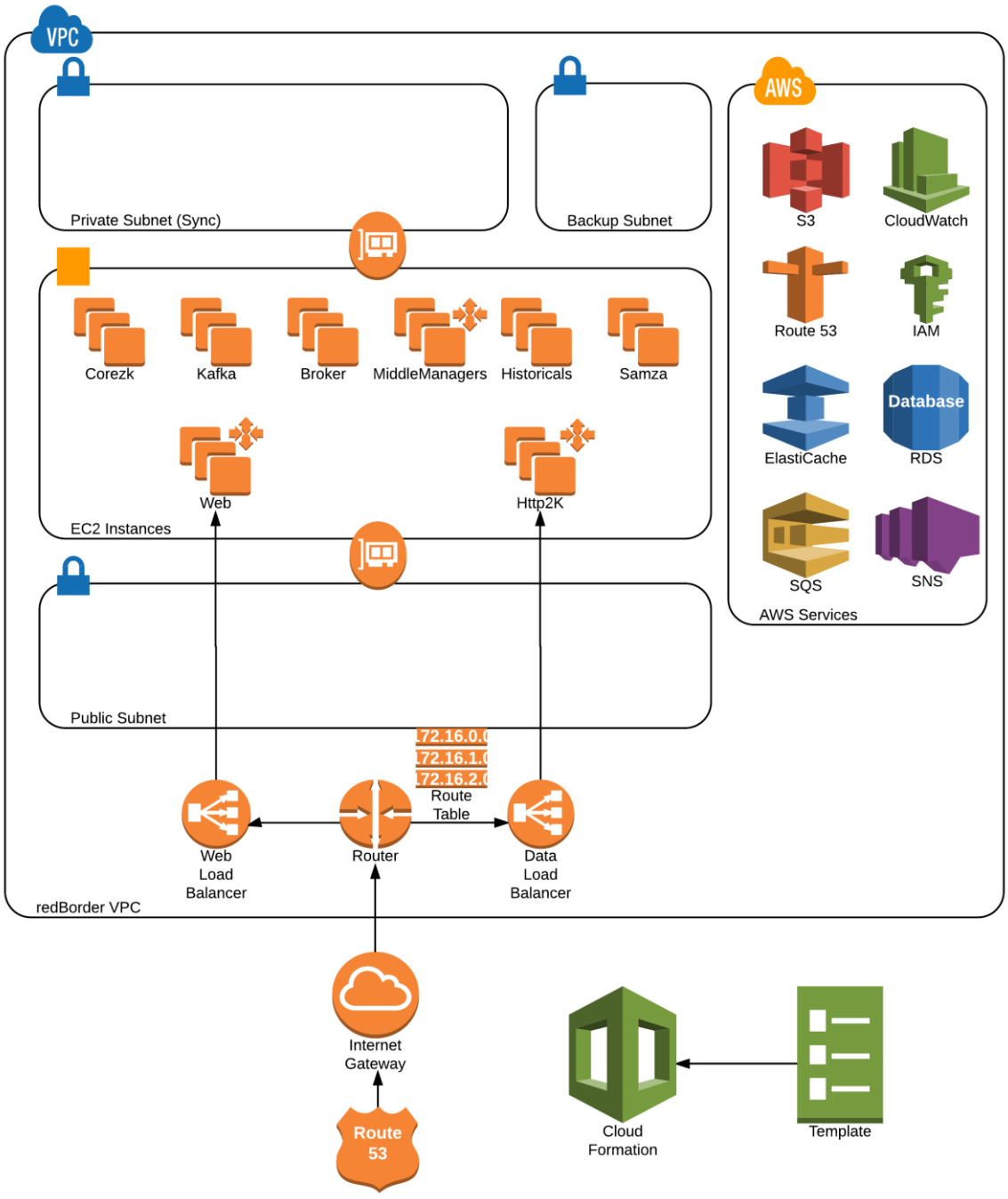

Figura 26 Arquitectura a desplegar por CloudFormation

En la plantilla se definen todos los recursos y servicios de Amazon Web Services que se van a utilizar y que aparecen en la figura. Además, crea los grupos de autoescalado de las instancias de redBorder y les provee de toda la configuración que necesitan para funcionar (Tipos de instancias, User-Data a utilizar, etc).

De esta forma, la infraestructura necesaria es creada por CloudFormation, dejando la configuración e integración de los servicios de redBorder a las instancias, que disponen de toda la información necesaria en el User-Data para terminar el trabajo.

Es importante destacar que las instancias de redBorder implementan los Helper Scripts de CloudFormation, y utilizan el siguiente comando para indicar que la instancia ha finalizado:

```
cfn-signal --success --logical-id (NOMBRE DEL RECURSO EN LA PLANTILLA)
--stack (NOMBRE DE LA STACK)
```

En el caso de los grupos de autoescalado, no pasarán a estado CREATE_COMPLETE hasta que todas las instancias hayan señalizado que han terminado su configuración. En ciertos roles de redBorder es necesario ejecutar algunas tareas al final de la creación de todos los nodos del rol, por lo que se debe detectar si todas las instancias han sido configuradas, mirando el estado de la pila de CloudFormation. Para ello se utiliza el siguiente comando:

```
aws cloudformation describe-stack-resource --stack  --logical-resource-id (IDENTIFICADOR
DEL GRUPO DE AUTOESCALADO EN LA PLANTILLA)
```

El resultado es que especificando sólo una serie de parámetros se automatiza totalmente la creación y la configuración del clúster, que en aproximadamente 1 hora y media puede estar listo para entrar en producción.

La plantilla utilizada para lograr estos resultados se encuentra en el Anexo IV, donde además se explican las configuraciones en todo el sistema en forma de comentarios de la plantilla.

7 MONITORIZACIÓN Y ESCALADO EN AMAZON WEB SERVICES

Scaling is only difficult when you grow your organization like a tower instead of a city

Jurgen Appelo, 2014

En este capítulo nos vamos centrar en la monitorización y autoescalado de ciertos roles críticos en redBorder, que deben ser lo suficientemente elásticos como adaptarse a cambios en la tasa de peticiones que reciben y en la cantidad de información que deben procesar.

Para ello es fundamental elegir unas buenas métricas de monitorización que nos indiquen cuando es necesario escalar horizontalmente, y cuando tenemos más capacidad de la necesaria. Estas métricas dependen del rol que vayamos a escalar, por lo que explicaremos uno a uno como se ha diseñado el plan de monitorización y autoescalado.

Los roles que vamos a tratar son los que soportan autoescalado, que son:

- Web
- Http2k
- MiddleManager

El resto de roles no soportan actualmente el autoescalado, por lo que se escalarán de forma manual si es necesario. No vamos a entrar en detalles de las métricas que hay que monitorizar para dichos roles, únicamente nos centraremos en los roles mencionados anteriormente.

7.1 Registrado y desregistrado de instancias

Antes de entrar en los detalles de configuración del autoescalado, debemos tener en cuenta dos cosas:

- Cuando se levanta una instancia nueva de un rol, Chef se encarga de autoconfigurarla y registrarla en el clúster, por lo que al finalizar el proceso, entrará automáticamente en funcionamiento con el resto del clúster.
- Dado que las instancias quedan registradas en la base de datos del servicio Chef Server, antes de eliminarlas, debemos proceder a su desregistrado. Para ello, la instancia que será eliminada debe ejecutar un script llamado rb_cluster_leave.sh, el cuál se encarga de detener los servicios correctamente para evitar pérdidas de datos, y desregistra la instancia del clúster. Una vez ejecutado el script, puede ser eliminada.

El primer punto no representa un problema pero el segundo sí, porque la instancia tiene que saber de alguna manera que va a ser eliminada.

Aquí entra en juego el concepto de **Lifecycle Hook** y el uso del servicio SQS, ambos explicados en el capítulo anterior. Cuando se va a eliminar una instancia de un grupo de autoescalado que tenga habilitado los Lifecycle Hooks, el grupo de autoescalado envía un mensaje de aviso a SQS, y la instancia a terminar, queda en estado de espera. El servicio rb_sqsld, que corre en los nodos corezk (que no autoescalan), se encarga de recibir el mensaje y de ejecutar en la instancia que va a ser terminada el script rb_cluster_leave. Una vez finalizado, se

notifica al grupo de autoescalado que ha finalizado el proceso de espera, y la instancia es eliminada.

7.2 Web

La web es uno de los servicios que escalan con más facilidad, ya que no necesitan estar sincronizados con otros nodos web. La alta disponibilidad y reparto de carga se consigue mediante un sistema externo: El balanceador de tráfico.

Cuando se levanta una instancia se autoconfiguran los servicios, y dado que las instancias pertenecen a un grupo de autoescalado que tiene asociado un balanceador de carga, cuando el servicio web de la nueva instancia esté disponible, automáticamente comenzará a recibir tráfico.

El factor que limita el rendimiento de las instancias web es la CPU, por lo que es la métrica que deberá ser monitorizada mediante AWS Cloudwatch, definiendo una alarma que provoque el autoescalado si la CPU alcanza más de un determinado valor. La configuración escogida sería la siguiente:

- Si la CPU media de los nodos web supera el 65% durante 5 minutos, levantamos 1 nodo.
- Si la CPU media de los nodos web supera el 75% durante 5 minutos, levantamos 2 nodos.
- Si la CPU media de los nodos web supera el 85% durante 5 minutos, levantamos 3 nodos.

Además, cuando se realiza una acción de autoescalado, se bloquean nuevas acciones durante 10 minutos, ya que es el tiempo que tarda una nueva instancia en estar preparada. Si cuando la nueva instancia entra en funcionamiento, no ha bajado el uso medio de CPU, se volvería a autoescalar.

Esta política permite mantener siempre un colchón de capacidad para soportar picos, y en el caso de recibir un pico demasiado grande, permite escalar rápidamente para alcanzar la capacidad necesaria como para que la experiencia de usuario no se vea demasiado afectada.

La política para reducir las instancias también se basa en la CPU, y es la siguiente:

- Si la CPU media de los nodos web baja del 40 % durante 15 minutos, quitamos 1 nodo
- Si la CPU media de los nodos web baja del 20 % durante 15 mintutos, quitamos 2 nodos.

Si la carga se reduce y se mantiene baja, se quitarán instancias hasta que la CPU se mantenga alrededor del 50% de su capacidad. Esto permite a la web funcionar con buen rendimiento, teniendo un margen para picos, pero sin tener demasiada capacidad desaprovechada.

7.3 Http2k

El autoescalado es muy similar al de la web, ya que es una aplicación con uso predominante de CPU, por lo que se han definido las mismas alarmas y las mismas políticas de autoescalado.

7.4 MiddleManager

Los MiddleManagers tienen un modelo de autoescalado más complejo que los nodos web, debido a que tienen un sistema de asignación de recursos muy diferente. Sin embargo, es necesario definir muy bien su autoescalado porque son críticos para el sistema, ya que procesan la información en tiempo real que llega a éste y que debe estar disponible rápidamente en la web.

Trabajan con tareas que tienen capacidad para procesar un número aproximado de eventos por segundo. En función de los recursos hardware asignados a cada instancia, soportará un número mayor o menor de tareas.

Estas tareas pueden estar en cuatro estados:

- **Running**: la tarea está en ejecución y procesando eventos.
- **Complete**: la tarea ha finalizado.
- **Pending**: no quedan huecos para soportar la tarea, por lo que los eventos que debería procesar se

pierden.

- **Failed**: la tarea ha tenido errores.

Es fundamental evitar a toda costa que llegue a haber una tarea en estado Pending, porque entonces se estarán perdiendo datos de clientes en tiempo real.

Las tareas tienen una duración de 1 hora. Cada hora, se crean tareas nuevas, pero las tareas anteriores continúan otros 20 minutos para procesar posibles eventos retrasados. Esto hace que durante los primeros 20 minutos el uso de capacidad sea mucho más elevado que en el resto del tiempo. Por ello, la métrica más representativa es el porcentaje de capacidad utilizada en ese periodo. Además, las nuevas tareas se preparan con 15 minutos de antelación para que en el caso de que no haya suficiente capacidad, se tenga tiempo para realizar el autoescalado.

Por lo tanto, tenemos un periodo de 35 minutos donde se utiliza (en condiciones normales) el doble de la capacidad que en los otros 25 minutos.

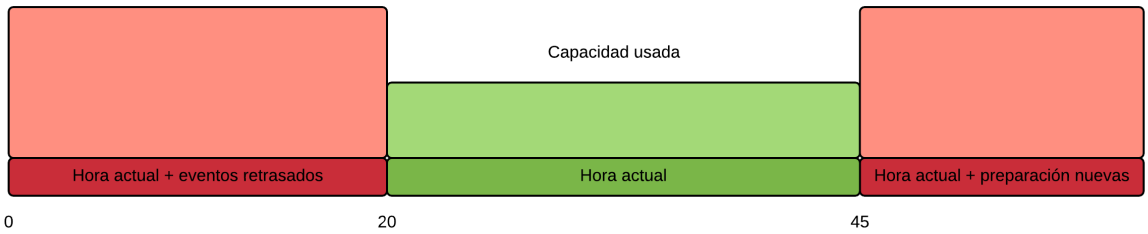

Figura 27 Capacidad utilizada en cada período de tiempo por los MiddleManagers

Para saber si vamos a tener suficiente capacidad en cualquier momento, se utilizará una métrica especial:

- Durante los 35 minutos en los que hay tanto tareas de la hora anterior como nuevas tareas, se indica el porcentaje de capacidad utilizado, es decir:

$$\frac{Número\ de\ tareas\ en\ funcionamiento^{39}}{Capacidad\ total} \cdot 100$$

- Durante los otros 25 minutos, se suman las tareas de la hora anterior a las de la hora actual para hacer el cálculo:

$$\frac{Número\ de\ tareas\ en\ funcionamiento + número\ de\ tareas\ de\ la\ hora\ anterior}{Capacidad\ total} \cdot 100$$

De esta forma conseguimos tener un valor representativo de la capacidad necesaria para que el clúster responda correctamente. La métrica es llamada Desired_capacity.

Otra métrica que se envía es el número de tareas Pending. No debe haber ninguna en ningún momento para evitar pérdida de datos, pero si en algún momento se detectara alguna, se debería autoescalar lo más rápidamente posible para evitar prolongar esa situación en el tiempo. El nombre de la métrica sería Pending_tasks.

Las métricas mencionadas se envían a la cola Kafka, para que rb_cloudwatch las envíe a Amazon Cloudwatch y poder así programar acciones de autoescalado.

Las políticas de autoescalado para el crecimiento del clúster basada en la métrica Desired_capacity serían las siguientes:

- Si la capacidad deseada es mayor del 65% durante 1 minuto, se añade 1 instancia.
- Si la capacidad deseada es mayor del 80 % durante 1 mintuo, se añaden 2 instancias.

[39] Incluye tanto tareas Running como Pending

- Si la capacidad deseada es mayor del 100% [40] durante 1 minuto, se añaden 3 instancias.
- Si la capacidad deseada es mayor del 120% durante 1 minuto, se añaden 4 instancias.

En el caso de que haya tareas en estado Pending, automáticamente se autoescalaría añadiendo 2 instancias, siendo los periodos de comprobación de 1 minuto también. En ambas políticas se espera 20 minutos antes de reaundar los procesos de autoescalado para esperar a que las nuevas instancias estén operativas.

Para la eliminación de instancias sobrantes, también se utiliza la métrica Desired_capacity, pero no se asigna ninguna política de autoescalado. Esto es así porque en el caso de los MiddleManagers, rb_cluster_leave debe esperar a que finalice la tarea para permitir la eliminación de la instancia. Esto conlleva un problema y es que si se produce una disminución de instancias debido al autoescalado, hasta que no finalice la tarea todas las operaciones de autoescalado quedarían bloqueadas, dejando al clúster sin respuesta ante un posible pico de eventos.

Por ello, la eliminación de las instancias no es gestionada por el grupo de autoescalado, sino por el servicio rb_sqsld. Se explicó en el punto 6.9.2 que este servicio procesaba tanto mensajes de autoescalado como mensajes de alarmas de Cloudwatch. Lo que se hace es utilizar la notificación de alarma para el caso de que la métrica Desired_capacity alance un nivel lo suficientemente bajo, de forma que cuando el servicio la recibe, ejecuta el script rb_cluster_leave sin que afecte al resto de operaciones de autoescalado.

El proceso se muestra en la siguiente figura:

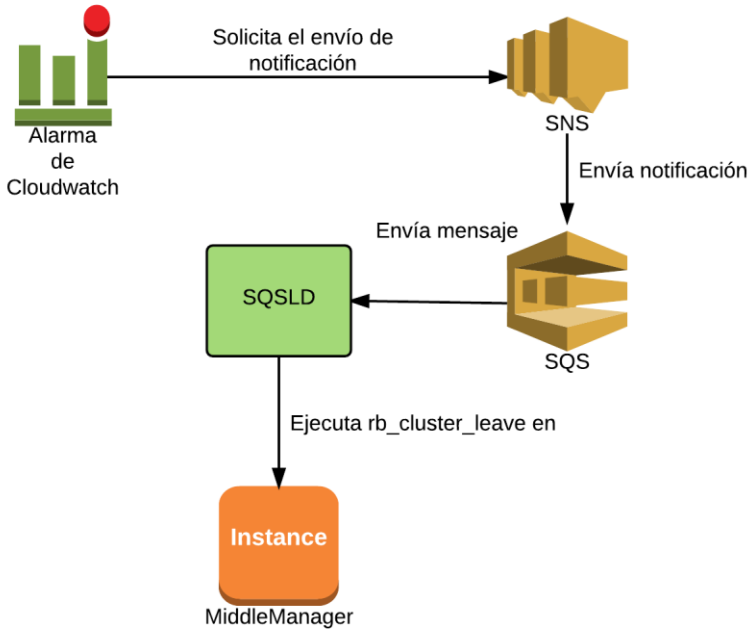

Figura 28 Proceso de eliminación de MiddleManagers mediante SQSLD

[40] Se soportan porcentajes mayores del 100%, ya que podrían haber más tareas que capacidad tiene el clúster.

8 VENTA DEL PRODUCTO

Great products sell themselves

Kevin Systrom, CEO of Instagram

En este capítulo veremos cómo se pretende vender redBorder a los usuarios, a fin de que puedan obtener los servicios que el sistema provee y de monetizar el producto, siempre utilizando los sistemas de Cloud Computing como medio para lograrlo.

8.1 Venta como servicio (SaaS)

La mayoría de los clientes de IaaS o PaaS utilizan estos servicios para ofrecer a su vez un servicio de SaaS. Existen muchísimos ejemplos en el mercado: Dropbox, Gmail, GitHub, etc. Incluso si nos acercamos más al área de mercado de redBorder, nos encontramos con otros SaaS como New Relic u OpenDNS.

redBorder también ofrecerá un SaaS para sus clientes, que consistirá en un servicio de redBorder Manager en la nube, completamente gestionado, al que se pueden asociar sensores por cliente y que tiene una web disponible para acceder a los datos de cada cliente.

Los clientes podrán comprar distintos tipos de licencia en función de sus necesidades, e incluso tendrán opción a una licencia de prueba temporal, para probar el servicio.

8.2 Venta como sistema

redBorder también ofrece la posibilidad a sus clientes de que desplieguen su propio clúster de managers en la nube y lo mantengan ellos mismos, de forma que tendrían un sistema dedicado para ellos.

Para ello se utilizará el servicio AWS Marketplace[41], que permite vender Amazon Machine Images (AMIs) publicadas en esta plataforma y ofrece la posibilidad de que Amazon gestione el cobro al cliente tanto por los propios recursos de Amazon empleados como por el coste del software que define el publicador del software.

Los métodos de pago[42] que soporta AWS Marketplace son los siguientes:

- Gestionar nuestra propia licencia: el vendedor del software gestiona el cobro a los clientes y les provee de una licencia que deben utilizar para activar el producto.
- Gratis: se pueden publicar AMIs gratuitas para que sean usadas libremente por los usuarios.
- Por horas: se aplica un precio por hora, dependiendo del tipo de instancia. Se puede aplicar a cada cliente también un tiempo de prueba gratuito para una única instancia.
- Por meses: se paga un precio al mes por el software, aunque en el caso de no cumplirse un mes entero, se pagaría la parte equivalente al tiempo usado.

Además, se pueden incluir la opción de comprar meses y años por adelantado a cambio de una reducción del

[41] AWS Marketplace: tiena online que ayuda a los clientes de Amazon Web Services a desplegar y utilizar servicios de distintos proveedores. Más información en su documentación [34].

[42] En la fecha de elaboración de este proyecto aún no se han definidos los precios del producto ni el método de cobro por parte de la empresa que vende redBorder, por lo que no se especifica el método de pago concreto.

coste.

Para facilitar el despliegue de la infraestructura por parte del cliente, junto con la AMI se proveerá de una plantilla de CloudFormation preparada para crear un clúster completamente operativo de forma automática, simplemente especificando una serie de parámetros básicos.

9 FUTURAS MEJORAS

Nothing works better than just improving your product.

Joel Spolsky, Stack Exchange

En el proyecto se han sentado las bases para desplegar redBorder en la cloud y más concretamente en Amazon Web Services, integrándolo con varios servicios. Las siguientes mejoras que se pueden abordar en Amazon son las siguientes:

- **Autoescalado de todos los roles**: actualmente, sólo esta soportado el autoescalado en los roles http2k, web y middleManager, pero sería deseable que todos los roles pudieran autoescalar en el futuro para aumentar la elasticidad y capacidad del clúster, optimizando a la vez los costes del sistema.
- **Multi-AZ y multi-región**: en este proyecto se ha trabajado para desplegar redBorder en una única zona de disponibilidad, por lo que en caso de que esa zona quedara fuera de servicio, caería el sistema. Por ello se trabajará en el futuro en desplegar el sistema a lo largo de varias zonas de disponibilidad y regiones, haciéndolo más resistente a desastres.
- **Uso de AWS CloudFront**: para mejorar el rendimiento de la web, se acercará el contenido estático de las web mediante el servicio de distribución de información estática CloudFront, que generará copias de este contenido en múltiples centros de datos. Cuando un cliente lo solicite, se le enviará desde el centro de datos más cercano.
- **Externalización de más servicios**: AWS ofrece servicios gestionados que podrían ser utilizados en redBorder en lugar de ser implementado y mantenido por el sistema. Ejemplos son AWS Kinesis[43], para análisis de datos en tiempo real, AWS EMR[44] para clústers de Hadoop o AWS DynamoDB[45] para bases de datos NoSQL. Al externalizar los servicios, se reducen los costos y dificultad de mantenimiento, permitiéndonos centrarnos en el software de redBorder que aporta más valor.
- **VPN**: añadir soporte para gestionar el clúster mediante una VPN reduciría el acceso público a las instancias, mejorando su seguridad y otorgando un mejor acceso a la red de AWS donde se encuentra el clúster.
- **Mejorar la integración continua**: actualmente, para sacar nuevas versiones de redBorder en la nube hay que realizar todo el procedimiento de importación de instancias a AWS y generar una AMI, o bien actualizar las instancias manualmente. Un gran punto de mejora sería definir un método automatizado de actualización y de creación de AMIs dentro del entorno de AWS, para reducir el tiempo de despliegue y de actualización.

Finalmente, otros de los objetivos futuros es el despligue en otros entornos Cloud, aprovechando los servicios y posibilidades que ofrecen, para no depender de un único proveedor.

---

[43] AWS Kinesis: servicio de análisis de flujos de datos en tiempo real de alto rendimiento, escalable y completamente gestionado por AWS. Más información en su documentación [27].

[44] AWS EMR (Elastic Map Reduce): permite desplegar un clúster de Hadoop con aplicaciones preinstaladas gestionado por AWS. Más información en su documentación [26].

[45] AWS DynamoDB: base de datos NoSQL de alto rendimiento, escalable y completamente gestionada por AWS. Más informaciónen su documentación [46].

10 CONCLUSIONES

Cloud Computing is really a no-brainer for any start-up because it allows you to test your business plan very quickly for little money. Every start-up, or even a division within a company that has an idea for something new, should be figuring out how to use cloud computing its plan.

Brad Jefferson, CEO of Animoto

Gracias a los entornos Cloud, muchas empresas, entre ellas la creadora de redBorder, pueden desarrollar, probar, desplegar y vender servicios y sistemas que antes sólo estaban al alcance de las empresas más potentes, que podían asumir una gran inversión inicial para disponer del equipo y la infraestructura necesaria.

Ahora, sin necesidad de inversión inicial, cualquier startup puede crear un pequeño sistema que sea capaz de escalar a medida que tiene más clientes, hasta convertirse en un gran sistema. Existen muchos casos de éxito en el mercado, como son el caso de las startups que crearon LucidChart, Candy Crash o AlienVault OSSIM.

Con este proyecto, se ha automatizado y facilitado el despliegue de redBorder, que puede aprovechar también las posibilidades de escalado y de crecimiento que ofrece la nube, tomando así impulso para su crecimiento a nivel mundial.

Las bases quedan sentadas, dando la posibilidad que el trabajo de los próximos años hagan grandes al producto y la empresa dentro del mundo de la seguridad y la monitorización de redes.

11 PRESUPUESTO

Descripción	Cantidad	Precio	Subtotal
Facturación AWS			
2014 Octubre	1	44 €	44 €
2014 Noviembre	1	137 €	137 €
2014 Diciembre	1	277 €	277 €
2015 Enero	1	280 €	280 €
2015 Febrero	1	448 €	448 €
2015 Marzo	1	584 €	584 €
2015 Abril	1	1.100 €	1.100 €
2015 Mayo	1	846 €	846 €
2015 Junio	1	339 €	339 €
2015 Julio	1	668 €	668 €
2015 Agosto	1	1.450 €	1.450 €
Recursos humanos			
Salario Ingeniero Junior	9	1.865 €	16.785 €
		Total	**22.958 €**

ANEXO I - DOCUMENTACIÓN DE PARÁMETROS DEL USER-DATA DE REDBORDER

En este Anexo se explican los parámetros admitidos en el User-Data de las instancias de redBorder, y para qué sirven.

- **NODEROLE**: permite indicar el rol que tomará la instancia de redBorder. Los valores admitidos son: corezk, brokerweb, middleManager, kafka, historical, samza, enrichment, http2k, webdruid, master y custom. Ejemplo: NODEROLE=corezk.
- **NODESERVICES**: permite añadir servicios extras no incluidos en el rol de la instancia. Sintaxis: NODESERVICES="nombreServicio1:1|0 nombreServicio2:(boolean) …"
  Ejemplo: NODESERVICES="rb-cloudwatch:1 sqsld:true"
  Los valores admitidos son los siguientes: druid_coordinator, druid_realtime, druid_historical, druid_broker, druid_overlord, druid_middleManager, kafka, zookeeper, zookeeper2, rb-webui, rb-workers, erchef, bookshelf, chef-server-webui, postgresql, nginx, riak, hadoop_historyserver, rb-monitor, nprobe, memcached, strom_ui, strom_nimbus, strom_supervisor, n2klocd, n2kmetricd, n2kmobiled, trap2kafka, freeradius, nmspd, rb-sociald, k2http, kafka-offset-monitor, gridgain, rb-discover, oozie, rb-sequence-oozie, rb-darklistd, rb-enrich, rb-cloudwatch, awslogs, newrelic-sysmond, sqsld.
- **CDOMAIN**: permite asociar un dominio al clúster. De esta forma los nodos se registran con su nombre de host en el servicio AWS Route 53. El dominio debe haber sido registrado con anterioridad por un proveedor de dominios. Además, se requieren credenciales de acceso a un usuario de Amazon Web Services autorizado para la gestión del servicio AWS Route 53. Para ello deben especificar las variables AWS_ACCESS_KEY y AWS_SECRET_KEY en el User-Data. También es necesario especificar la región en la que se utilizará el servicio AWS Route 53 mediante la variable REGION. El dominio registrado debe ser de segundo nivel como por ejemplo, redborder.net. En el caso de que se especifique un CDOMAIN de mayor nivel, se crearía una Hosted Zone en AWS Route 53 con el nombre del subdominio, con el fin de poder tener varios clústers asociados al mismo dominio y organizados en distintas Hosted Zones.
  Ejemplos:
  CDOMAIN="redborder.net"
  CDOMAIN="live.redborder.net"
- **PUBLIC_HOSTEDZONE_ID:** identificador de la hosted zone pública creada por AWS CloudFormation. Sólo necesario si el clúster está siendo creado por este servicio.
  Ejemplo: EXAMPLESHDKF
- **PRIVATE_HOSTEDZONE_ID:** identificador de la hosted zone privada creada por AWS CloudFormation: Sólo necesario si el clúster está siendo creado por este servicio.
  Ejemplo:EXAMPLESDHFS
- **MODULES**: permite elegir qué módulos de los servicios de redBorder serán habilitados.
  Sintaxis: MODULES="modulo1:(boolean) modulo2:(boolean) …"
  Ejemplo: MODULES="flow:true ips:false social:false monitor:true"
  Módulos admitidos: Flow, IPS, Social, Monitor, Malware.
- **REGION**: indica la región de AWS en la que corre el clúster y/o los servicios de AWS utilizados.

Ejemplo: REGION=eu-west-1
Nota: sólo se puede especificar una región y debe ser común para todos los nodos del clúster.
Valores admitidos: us-east-1, us-west-1, us-west-2, eu-west-1, eu-central-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, sa-east-1.

- **STACKNAME**: nombre de la pila de AWS CloudFormation en la que se está creando el clúster. Se utiliza para solicitar a la API de AWS información sobre el estado de la pila. Necesario sólo cuando el clúster es creado en AWS mediante las plantillas de AWS CloudFormation.
  Ejemplo: STACKNAME=Production-cluster
- **AWS_ACCESS_KEY y AWS_SECRET_KEY**: credenciales de acceso a servicios de AWS. Se recomienda crear un usuario que tenga los permisos estrictamente necesarios para el correcto funcionamiento del sistema. Ejemplo:
  AWS_ACCESS_KEY=”AKIAIOSFODNN7EXAMPLE”
  AWS_SECRET_KEY=”wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY”
- **CHEF_AWS_ACCESS_KEY y CHEF_AWS_SECRET_KEY**: credenciales que Chef enviará a los sensores de redBorder para que puedan acceder a un bucket de S3 que contiene las plantillas de configuración. El usuario asociado sólo debe tener permisos para acceder a dicho bucket. Sólo es necesario si se utiliza el servicio S3 de AWS.
- **AUTOSCALINGGROUPNAME**: nombre del grupo de autoescalado al que pertenece la instancia. Se utiliza para saber si se han creado todas las instancias asociadas al grupo de autoescalado. Necesario sólo cuando el clúster es creado en AWS mediante las plantillas de AWS CloudFormation.
  Ejemplo: AUTOSCALINGGROUPNAME=redBorderAutoscalingGroup
- **SQSQUEUEURL**: url de la cola de AWS SQS, necesaria para el servicio SQSLD. Ejemplo:
  SQSQUEUEURL=http://sqs.us-east-1.amazonaws.com/123456789012/queue1
- **SUBNET_ID:** identificador de la subred privada creada por AWS CloudFormation. Se utiliza para la creación de entradas en la Hosted Zone privada. Sólo necesario si el clúster está siendo creado por CloudFormation.
  Ejemplo: SUBNET_ID: subnet-asdfasdf
- **S3TYPE**: indica si se usa el servicio S3 de AWS o si se utiliza Riak en el propio clúster de redBorder. Valores admitidos: aws, riak. Por defecto es riak.
  Ejemplo: S3TYPE=aws
- **S3HOST**: dirección en la que se encuentra el servicio de almacenamiento S3. Sólo necesario si se utiliza el servicio de S3 de Amazon.
  Ejemplo S3HOST=s3.redbordercloud.com
- **S3BUCKET**: nombre del bucket de S3 en el que se almacenará la información relacionada con el clúster. Sólo necesario si se utiliza el servicio de S3 de Amazon.
  Ejemplo: S3BUCKET=rb-bucket
- **ELASTICCACHECLUSTERID**: permite indicar el identificador del clúster de AWS Elasticache. redBorder lo utiliza para obtener los nodos del clúster de memcached y poder así configurar sus aplicaciones para utilizarlo. Sólo necesario si se utiliza el servicio AWS Elasticache.
  Ejemplo: ELASTICCACHECLUSTERID=”redborder-elasticache”
- **SQLHOST**: dirección y puerto en la que se encuentra la base de datos PostgreSQL de AWS RDS. Sólo es necesario si se utiliza AWS RDS como base de datos. Si no se indica nada, redBorder creará y gestionará su propia base de datos PostgreSQL. Si se usa este parámetro también serán necesarios los parámetros SQLDB, SQLUSER y SQLPASSWORD.
  Ejemplo: SQLHOST=postgre.redbordercloud.com:5432
- **SQLDB**: nombre de la base de datos PostgreSQL externa.
  Ejemplo: SQLDB=clusterBBDD
- **SQLUSER**: usuario administrador de la base de datos PostgreSQL externa. Los usuarios admin y redborder no son válidos.
  Ejemplo: SQLUSER=”usuario”

- **SQLPASSWORD**: contraseña del usuario administrador de la base de datos PostgreSQL externa. Ejemplo: SQLPASSWORD=”redborder123”
- **ENRICHMODE**: sistema de enriquecimiento de datos a utilizar en el clúster. Sólo es obligatorio especificarlo en los nodos con rol corezk o master. Valores admitidos: samza, rb-enrich. Ejemplo: ENRICHMODE=samza
- **MMDOWNALARMNAME**: nombre de la alarma que hará que se eliminen instancias del rol MiddleManager. SQSLD utiliza este parámetro para monitorizar la alarma. Sólo es necesario si el clúster es desplegado usando AWS CloudFormation.
- **CMDFINISH**: permite indicar un comando para que sea ejecutado al finalizar la configuración de la instancia.
  Ejemplo: CMDFINISH=”echo \”Ejecucion de CMDFINISH\” > /root/cmdfinish.txt”
- **NEWRELICLICENSE**: licencia del servicio NewRelic[46] para monitorización del sistema. Sólo es necesario si se desea utilizar este servicio de monitorización.

Además hay cierta información que debe pasarse directamente a Chef para la correcta configuración del nodo. A continuación se muestra un ejemplo:

```
cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF_
{
   "redBorder": {
      "aws" : {
         "cloudwatch" : "true",
         "autoscaled" : "false"
      },
      "druid" : {
         "historical" : {
            "tier" : "hot"e
         }
      }

   }
}
```

- **redBorder – aws – cloudwatch:** puede tomar los valores “true” y “false”, permitiendo habilitar o deshabilitar el envío de métricas internas de instancias a Cloudwatch como el uso de RAM o disco. Sólo debe ser “true” si la instancia está siendo ejecutada en Amazon EC2.
- **redBorder – aws – autoscaled**: puede tomar los valores “true” y false”. Indica si esta instancia puede ser sometida a autoescalado (posible terminación automática). Las instancias con esta características serán eliminadas de la base de datos de Chef-Server si no responden en un periodo definido de tiempo.
- **druid – historical – tier**: puede tomar los valores hot y lt. Sólo debe configurarse si la instancia tiene el rol Historical, permitiendo seleccionar si se comportará como un nodo Historical Hot o como Historical Long Term.

[46] NewRelic: servicio de monitorización de aplicaciones. Más información en su documentación [48].

ANEXO II - INSTALACIÓN DE HERRAMIENTAS

Instalación de Cloud-Init

El paquete de software Cloud-Init es Open Source y su código fuente está disponible para su descarga, por lo que puede ser compilado e implementado en cualquier sistema operativo Linux. Además, también hay disponibles paquetes precompilados para facilitar su instalación.

redBorder está basado en la distribución de Linux CentOS 6.5, por lo que se ha realizado su instalación utilizando un paquete RPM de Cloud-Init compatible con este sistema operativo. En el caso de redBorder el paquete es instalado en el momento de la instalación del sistema operativo, por lo que se ha colaborado con los desarrolladores para la integración e instalación automática del paquete RPM.

http://rpmfind.net/linux/rpm2html/search.php?query=cloud-init

En el caso de que no estuviera preinstalado en el sistema basta con descargar el RPM e instalarlo con el siguiente comando:

```
yum –y install cloud-init
```

Instalación de AWS CLI

AWS provee de estas herramientas de línea de comandos para gestionar sus servicios, permitiendo así la automatización de operaciones mediante scripts.

En redBorder se ha integrado mediante un paquete RPM en la sdk que genera las ISOS, de forma que se incluye en la instalación del sistema.

Los detalles de instalación para cualquier sistema están descritos en la siguiente web: http://docs.aws.amazon.com/cli/latest/userguide/installing.html

Instalación de las EC2 Command Line Tools

Estas herramientas no son necesarias en el sistema redBorder, pero sí para el proceso de creación e importación de la AMI en Amazon Web Services.

Las instrucciones de instalación se pueden encontrar en el siguiente enlace: http://docs.aws.amazon.com/AWSEC2/latest/CommandLineReference/ec2-cli-get-set-up.html

ANEXO III - SCRIPTS DE CREACIÓN DE LA AMAZON MACHINE IMAGE DE REDBORDER

Para facilitar el despliegue de nuevas versiones de redBorder en Amazon Web Services se han creado una serie de script que dada una imagen ISO de redBorder, generan una AMI en AWS lista para ser desplegada en instancias de EC2.

Estos scripts son ejecutados utilizando el servicio Jenkins[47], el cuál está implementado en los servidores de la empresa desarrolladora de redBorder. Hay 4 tareas asociadas a la creación de la AMI, cada una de las cuales ejecuta un script en un equipo dedicado para ello.

- **Create Cloud IMG**: ejecuta el script **rb_create_cloud_img.sh**. Se encarga de descargar del repositorio de imágenes de la empresa la versión indicada de la ISO de redBorder, y realiza de forma automática la instalación en una máquina virtual para generar un disco duro virtual en formato IMG (RAW) con redBorder ya instalado. Ese disco virtual será el que se despliegue en entornos cloud.
- **AWS Create Import Task**: ejecuta el script **rb_aws_create_import_task.sh**. Crea una tarea de importación de la imagen a Amazon. Para ello se utilizan las ec2-command-line tools, que implementa las herramientas necesarias para la importación del IMG a Amazon EC2.
- **AWS Resume Import**: ejecuta el script **rb_aws_resume_import.sh** que a su vez llama al script **rb_aws_check_conversion.sh**. Continúa la tarea anterior y sube la imagen a Amazon Web Services. Cuando finaliza el proceso de subida, se obtiene una instancia de Amazon EC2 con el disco duro importado, a partir de la cuál se genera la AMI mediante un comando de la CLI de AWS.

A continuación mostramos el código de éstos scripts:

rb_create_cloud_img.sh

```
#!/bin/bash -e

release=$1
rm -f *.iso
wget -O $release.iso http://storage.redborder.lan/pandora/isos/redBorder-isos/$release.iso

if [ 'x$release' != 'x' ] ; then

   rm -f *.img
   rm -rf test
   mkdir test
   mount -o loop $release.iso test/
   rm -rf iso
   mkdir iso
   rsync -av test/ iso/
   umount test
   sed -i '/menu default/d' iso/isolinux/isolinux.cfg
   sed -i '/Manager Horama^/a menu default' iso/isolinux/isolinux.cfg
   sed -i 's/timeout 600/timeout 10/g' iso/isolinux/isolinux.cfg
   sed -i '/timezone --utc GMT/i rootpw 'redborder'' iso/ks-manager.cfg
   rm -f redBorder-*
   cd iso
   mkisofs -o ../cloud_$release.iso -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -
boot-load-size 4 -boot-info-table -J -R -V 'asdf' .
   cd ..
   rm -rf test
   rm -rf iso
   qemu-img create $release.img 8G
   qemu-system-x86_64 -boot d -cdrom cloud_$release.iso -m 1024 -hda $release.img -no-reboot -
enable-kvm -k es -vnc :5 -usbdevice tablet
```

[47] Jenkins: es servicio que monitoriza la ejecución de tareas repetibles, como la compilación de software o tareas programadas. Más información en su web .

```
   chown alberto:alberto $release.img
else
   echo 'Bad arguments, redBorder release is needed'
fi
```

Este script realiza las siguientes operaciones:

- Descarga del repositorio de redBorder la versión de la ISO pasada como parámetro.
- Modifica la iso para que instale el Manager de redBorder sin que solicite contraseña de usuario root en la instalación.
- Utiliza la herramienta de virtualización QEMU para instalar redBorder y generar un disco duro virtual en formato RAW con la instalación realizada.

rb_aws_create_import_task.sh

```
#!/bin/bash
if [ -r vauto-ami.txt ] ; then
   source vauto-ami.txt
fi
source aws_configuration

release=$1

if [ "x$release" != "x" -a "x$AWS_ACCESS_KEY" != "x" -a "x$AWS_SECRET_KEY" != "x" ] ; then
   IMPORT_RESULT=1
   COUNTER=1
   while [ $COUNTER -lt 11 -a "$IMPORT_RESULT" != "0" ] ; do
      DEBUG=$(ec2-import-instance ./$release.img -f RAW -t m3.medium -a x86_64 --bucket rbos \
             -o $AWS_ACCESS_KEY -w $AWS_SECRET_KEY --region eu-west-1 -p Linux --no-upload )
      IMPORT_RESULT=$?
      echo $DEBUG
      if [ "$IMPORT_RESULT" != "0" ] ; then
         echo 'Import instance failure, retrying... ($COUNTER/10)'
      fi
          let COUNTER=COUNTER+1
   done
   if [ "$IMPORT_RESULT" != 0 ] ; then
      echo 'Import instance failure, imposible create task'
      exit 1
   fi
   TASK_ID=$(echo "$DEBUG" | grep TaskId | cut -f 4)
   echo 'TASK_ID=$TASK_ID' > vauto-ami.txt
   echo 'TASK_ID=$TASK_ID'
else   echo 'USAGE: rb_aws_create_import_task.sh <img filename>'
fi
```

Este script realiza las siguientes operaciones:

- Lee las credenciales de AWS de un fichero externo (aws_configuration)
- Utiliza las ec2-command-line tools para crear la tarea de importación de la imagen a Amazon EC2.
- Almacena el identificador de la tarea en el fichero vauto-ami.txt.

rb_aws_resume_import.sh

```
if [ "x$release" != "x" -a "x$AWS_ACCESS_KEY" != "x" -a "x$AWS_SECRET_KEY" != "x" -a "x$TASK_ID"
!= "x" ] ; then
   ec2-resume-import -t $TASK_ID -o $AWS_ACCESS_KEY -w $AWS_SECRET_KEY --region eu-west-1
$release.img
   . rb_aws_check_conversion.sh $TASK_ID
   aws ec2 modify-instance-attribute --instance-id $INSTANCE_ID --sriov-net-support simple \
   --region eu-west-1

   COUNTER=1
   CREATE_STATUS=1
   while [ "$CREATE_STATUS" != "0"  -a $COUNTER -lt 21 ]; do
      echo "Trying with AMINAME = $release-V$COUNTER ($COUNTER/20)"
      DEBUG=$( aws ec2 create-image --instance-id $INSTANCE_ID  --name "$release-V$COUNTER" )
      CREATE_STATUS=$?

      let COUNTER=COUNTER+1
   done
   AMI_ID=$(echo "$DEBUG"| jq -r .ImageId )
   echo "AMI_ID=$AMI_ID"
   echo "AMI_ID=$AMI_ID" > AMI_ID.txt
   ./rb_pushbullet.sh aws $release
```

```
else
   echo 'Invalid arguments'
   exit 1
fi
```

Este script realiza las siguientes operaciones:

- Reanuda la tarea de importación de la imagen a Amazon EC2, subiéndola a Amazon S3.
- Llama al script rb_aws_check_conversion.sh, que devuelve el control cuando se haya finalizado la tarea de importación de la AMI.
- Registra una AMI a partir de la instancia creada por la tarea de importación.

rb_aws_check_conversion.sh

```
#!/bin/bash

#
#  Script para la comprobacion de si se ha terminado la importación de una imagen.
#

if [ 'x$TASK_ID' != 'x' ] ; then
   COUNTER=1
   while [ $COUNTER -lt 1000 ]; do
      CONVERSION_TASKS=$(aws ec2 describe-conversion-tasks --conversion-task-ids $TASK_ID --
region eu-west-1)
      CONVERSION_STATUS_MESSAGE='$(echo $CONVERSION_TASKS | jq -r
.ConversionTasks[0].StatusMessage)'
      CONVERSION_STATUS='$(echo $CONVERSION_TASKS | jq -r .ConversionTasks[0].State)'
           INSTANCE_ID='$(echo $CONVERSION_TASKS | jq -r
.ConversionTasks[0].ImportInstance.InstanceId)'
      echo 'Checking if conversion have finished  ($COUNTER/1000) => $CONVERSION_STATUS -
$CONVERSION_STATUS_MESSAGE'
        if [ '$CONVERSION_STATUS' = 'completed' ] ; then
          COUNTER=1001
          echo 'Conversion done succesfully'
           elif [ '$CONVERSION_STATUS' = 'cancelled' ] ; then
          COUNTER=1001
           echo 'Conversion failed'
            exit 1
       else
          let COUNTER=COUNTER+1
          sleep 5
       fi
   done
else
   echo 'USAGE: rb_aws_check_conversion.sh <TASK_ID>'
fi
```

Este script realiza las siguientes operaciones:

- Comprueba si la tarea ha finalizado. Devuelve el control cuando finaliza.

ANEXO IV - PLANTILLA DE CLOUDFORMATION DE REDBORDER

En este anexo mostramos la plantilla de CloudFormation utilizada para desplegar el sistema, explicando los recursos que aparecen y sus configuraciones.

Para ello se añadirán a la plantilla líneas de comentarios indicadas con la sintaxis de comentarios del lenguaje C, aunque se debe saber que el formato JSON no es compatible con este tipo de comentarios. Para disponer de una plantilla funcional se deben eliminar esas líneas.

redborder_template.json

```
1    {
2        "AWSTemplateFormatVersion" : "2010-09-09",
3        /* Versión del formato de la plantilla. Actualmente sólo existe como valor válido el
utilizado aquí */
4
5        "Description" : "AWS CloudFormation template to deploy redBorder Horama Manager",
6        /* Descripción de la plantilla */
7
8        "Parameters" : {
9        /* Parámetros que el usuario puede elegir cuando crea la pila (Stack) de CloudFormation
*/
10
11           "NumberOfNodes" : {
12               /* Número de nodos inicial que tendrá el clúster de redBorder. En función de
este parámetro
13                  se eligirán de forma automática los roles de las instancias. En el caso de
que haya menos
14                  instancias que roles definidos, se eligen unos roles especiales (master,
custom y webdruid) */
15               "Description" : "number of nodes for redBorder cluster. Roles will be configured
automatically",
16               "Type" : "Number",
17               "MinValue" : "1",
18               "MaxValue" : "20",
19               "Default" : "1",
20               "ConstraintDescription" : "number of nodes must between 1 and 20" //Si el
parámetro no cumple
21               // con la descripción, se muestra este mensaje.
22           },
23
24           "InstanceSize" : {
25               /* Tamaño y capacidad de las instancias a usar. Se permiten tres valores:
26                    - small: se utilizan instancias muy ajustadas en cuanto a recursos,
ofreciendo un coste
27                           bajo, pero con un rendimiento muy limitado
28                    - medium: se utiliza el tamaño de instancia recomendado para el correcto
funcionamiento de redBorder.
29                    - large: se utilizan instancias de gran tamaño, con un rendimiento muy
alto, pero un coste también
30                           muy elevado.
31               */
32               "Description" : "performance of each node (instance types). Instance types will
be choosen automatically in function of this parameter",
33               "Type" : "String",
34               "AllowedValues" : [ "small", "medium", "large" ],
35               "Default" : "medium"
36           },
37
38           "KeyName" : {
```

```
39              /* Nombre de la clave para el acceso mediante SSH a las instancias. Debe haber
sido creada o registrada
40                 anteriormente en Amazon EC2 */
41              "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
instances",
42              "Type" : "AWS::EC2::KeyPair::KeyName",
43              "ConstraintDescription" : "must be the name of an existing AWS EC2 KeyPair"
44          },
45
46          "CDOMAIN" : {
47              /* Nombre del dominio asociado al clúster de redBorder. Los nodos de redBorder
se registraran en Route 53
48                 utilizando su instance-id seguido de este parámetro. Además, la plantilla
creará HostedZones también
49                 con este nombre. Para que el nombre sea resuelto desde internet debe
registrarse el dominio
50                 previamente en Route 53 */
51              "Description" : "DNS Domain for cluster",
52              "Type" : "String",
53              "Default" : "redbordercloud.com"
54          },
55
56          "SSLCertificateName" : {
57              /* Nombre del certificado SSL a utilizar para terminar tráfico HTTPS en los
balanceadores de carga.
58                 Debe haber sido registrado anteriormente en AWS */
59              "Description" : "SSL Certificate for load balancer",
60              "Type" : "String"
61          },
62
63          "UseRDS" : {
64              /*  Permite elegir si utilizar el servicio AWS Relational Database Service (RDS)
para la base de datos
65                  PostgreSQL de redBorder. En el caso de que no se utilice, se levantará el
servicio en las instancias
66                  de redBorder. */
67              "Description" : "Indicates if you want to use AWS Relational Database Service
for PostgreSQL or you prefer to use PostgreSQL provided by redBorder",
68              "Type" : "String",
69              "AllowedValues" : [ "true", "false" ],
70              "Default" : "false"
71          },
72          "UseElasticache" : {
73              /*  Permite elegir si utilizar el servicio AWS Elasticache para el sistema de
caché distribuido
74                  Memcached de redBorder. En el caso de que no se utilice, se levantará el
servicio en las instancias
75                  de redBorder. */
76              "Description" : "Indicates if you want to use AWS Elasticache for Memcached or
you prefer to use Memcached provided by redBorder",
77              "Type" : "String",
78              "AllowedValues" : [ "true", "false" ],
79              "Default" : "false"
80          },
81          "DBPassword": {
82              /* Contraseña para la base de datos RDS. Solo es necesario en caso de que se
utilice RDS como base de datos */
83              "NoEcho": "true",
84              "Description" : "The database admin account password",
85              "Default" : "RDSpasswordAWS",
86              "Type": "String",
87              "MinLength": "8",
88              "MaxLength": "41",
89              "AllowedPattern" : "[a-zA-Z0-9]*",
90              "ConstraintDescription" : "must contain only alphanumeric characters and must
have a minium of 8 characters and a maximum of 41"
91          }
92      },
93
94      /* En esta sección se encuentran mapas de claves y valores que elegir distintos valores
de configuración
95      en función de parámetros o características de los recursos */
96      "Mappings" : {
97
98          /* En este mapping se la distribución de instancias en roles dependiendo del número
de instancias elegidas */
99          "RolesDistributionMap" : {
100             "1" : { "master": "1", "custom":"0", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
```

```
101             "2" : { "master": "1", "custom":"1", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
102             "3" : { "master": "1", "custom":"2", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
103             "4" : { "master": "1", "custom":"3", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
104             "5" : { "master": "1", "custom":"4", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
105             "6" : { "master": "1", "custom":"5", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
106             "7" : { "master": "1", "custom":"6", "corezk":"0", "kafka":"0", "http2k": "0",
"web":"0", "broker":"0", "middlemanager":"0", "historical":"0", "historicalhot":"0", "samza":"0"
},
107             "8" : { "master": "0", "custom":"0", "corezk":"1", "kafka":"1", "http2k": "0",
"web":"1", "broker":"1", "middlemanager":"1", "historical":"1", "historicalhot":"1", "samza":"1"
},
108             "9" : { "master": "0", "custom":"0", "corezk":"1", "kafka":"1", "http2k": "1",
"web":"1", "broker":"1", "middlemanager":"1", "historical":"1", "historicalhot":"1", "samza":"1"
}
109         },
110         /* En este mapping se encuentran los tipos de instancia elegidas en función del rol
y del tamaño que eligió el
111         usuario como parámetro a la hora de crear la pila */
112         "InstanceTypeMap" : {
113             "master" :        { "small":"m3.large",       "medium":"m3.2xlarge",
"large":"m3.2xlarge"       },
114             "custom" :        { "small":"m3.large",       "medium":"m3.2xlarge",
"large":"m3.2xlarge"       },
115             "corezk" :        { "small":"m3.large",       "medium":"i2.xlarge",
"large":"i2.xlarge"        },
116             "kafka" :         { "small":"m3.large",       "medium":"i2.xlarge",
"large":"i2.xlarge"        },
117             "http2k" :        { "small":"m3.large",       "medium":"c4.2xlarge",
"large":"c4.2xlarge"       },
118             "web" :           { "small":"c4.2xlarge",     "medium":"c4.2xlarge",
"large":"c4.2xlarge"       },
119             "broker" :        { "small":"c4.2xlarge",     "medium":"c4.2xlarge",
"large":"c4.2xlarge"       },
120             "middlemanager" : { "small":"m3.large",       "medium":"r3.4xlarge",
"large":"r3.4xlarge"       },
121             "historical" :    { "small":"m3.large",       "medium":"c4.4xlarge",
"large":"c4.4xlarge"       },
122             "historicalhot" : { "small":"m3.large",       "medium":"c4.4xlarge",
"large":"c4.4xlarge"       },
123             "samza" :         { "small":"m4.4xlarge",     "medium":"c4.4xlarge",
"large":"c4.4xlarge"       },
124             "rds" :           { "small":"db.m3.2xlarge",  "medium":"db.m3.2xlarge",
"large":"db.r3.large"      },
125             "memcached" :     { "small":"cache.t2.small", "medium":"cache.m3.xlarge",
"large":"cache.m3.2xlarge" }
126         },
127
128         /* Este mapping permite elegir la AMI que se desplegará para las instancias de
redBorder en función
129            de la región de AWS en la que se despliegue. Esto es necesario porque las AMIs
son únicas por región, por
130            lo que debe haber registrada una AMI de redBorder en cada region de AWS. */
131         "AWSRegion2AMI" : {
132             "eu-west-1"      : { "HVM64" : "ami-43e1c534" },
133             "eu-central-1"   : { "HVM64" : "ami-7e2c6309" },
134             "us-east-1"      : { "HVM64" : "ami-7e2c6309" },
135             "us-west-2"      : { "HVM64" : "ami-7e2c6309" },
136             "us-west-1"      : { "HVM64" : "ami-7e2c6309" },
137             "ap-northeast-1" : { "HVM64" : "ami-7e2c6309" },
138             "ap-southeast-1" : { "HVM64" : "ami-7e2c6309" },
139             "ap-southeast-2" : { "HVM64" : "ami-7e2c6309" },
140             "sa-east-1"      : { "HVM64" : "ami-7e2c6309" },
141             "cn-north-1"     : { "HVM64" : "ami-7e2c6309" }
142         }
143     },
144
145     "Conditions" : {
146     /* En esta sección se encuentran definidas una serie de condiciones que permiten decidir
si se crean o no ciertos recursos,
```

```
147         o si se asignan ciertas configuraciones.
148         En la sección de recurso se puede hacer referencia a estas condiciones definidas aquí
*/
149
150
151          /* Se ha definido una condicion por cada tipo de rol, ya que en función del número
de nodos elegidos habrá roles que
152             serán creados y otros que no. Para las condiciones utilizamos varias funciones
intrínsecas de cloudformation:
153             - Ref: permite hacer referencia al valor devuelto por un recurso o parámetro
154             - Fn::FindInMap: permite obtener un valor de un mapa definido en la sección
Mapping.
155             - Fn::Equals: permite comparar dos valores. Si son iguales, devuelve true. En
caso contrario, false.
156             - Fn::Not: operador Not sobre un valor booleano.
157              Las condiciones que se han definido como {rol}RoleCreation comprueban si en el
mapa de distribución de instancias
158              por rol el valor es 0 para el número de nodos definido mediante parámetro. Si lo
es, entonces no se crearan los
159              recursos asociados a ese rol.
160          */
161          "masterRoleCreation" : {
162
163              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "master" ] } ] } ]
164          },
165          "customRoleCreation" : {
166              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "custom" ] } ] } ]
167          },
168          "corezkRoleCreation" : {
169              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "corezk" ] } ] } ]
170          },
171          "kafkaRoleCreation" : {
172              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "kafka" ] } ] } ]
173          },
174          "http2kRoleCreation" : {
175              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "http2k" ] } ] } ]
176          },
177          "webRoleCreation" : {
178              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "web" ] } ] } ]
179          },
180          "brokerRoleCreation" : {
181              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "broker" ] } ] } ]
182          },
183          "middlemanagerRoleCreation" : {
184              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "middlemanager" ] } ] } ]
185          },
186          "historicalRoleCreation" : {
187              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "historical" ] } ] } ]
188          },
189          "historicalhotRoleCreation" : {
190              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "historicalhot" ] } ] } ]
191          },
192          "samzaRoleCreation" : {
193              "Fn::Not" : [ { "Fn::Equals" : [ "0",  { "Fn::FindInMap" : [
"RolesDistributionMap", { "Ref" : "NumberOfNodes" }, "samza" ] } ] } ]
194          },
195
196          /* Condicion para la creacion de recursos relacionados con AWS RDS, en función del
parámetro UseRDS
197             que define el usuario como parámetro */
198          "RDSCreation" : {
199              "Fn::Equals" : [ "true", { "Ref" : "UseRDS" } ]
200          },
201
202          /* Condicion para la creacion de recursos relacionados con AWS Elasticache, en
función del parámetro
203             UseElasticache que define el usuario como parámetro */
204          "ElasticacheCreation" : {
205              "Fn::Equals" : [ "true", { "Ref" : "UseElasticache" } ]
206          }
```

```
207      },
208
209      "Resources" : {
210      /* En la sección Resources es donde se definen los recursos a crear por Cloudformation y
las configuraciones de estos.
211         Es la parte más importante de la plantilla, y utiliza información de las secciones
anteriores. Dividiremos los
212         recursos en secciones para facilitar la compresión de la plantilla */
213
214
215     /*************************** CONFIGURACION DE RED ***************************/
216
217          "VPC" : {
218          /* Creacion de la Virtual Private Cloud en la que se desplegará el clúster de
redBorder */
219              "Type" : "AWS::EC2::VPC",
220              "Properties" : {
221                  "CidrBlock" : "172.16.0.0/16", //Rango de IPs asociado a la VPC
222                  "EnableDnsSupport" : "true", //Habilita compatibilidad con el uso de DNS de
Amazon, y por lo tanto con
223                  "EnableDnsHostnames" : "true", //Hosted Zones privadas que se crearán más
adelante
224                  "InstanceTenancy" : "default", //Define si las instancias deben tener
hardware dedicado o compartido en esta
225                                                 //VPC. Por defecto es compartido
226                  "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "VPC" ] ] } } ]
227                  /* Las tags son etiquetas que se asocian a recursos de Amazon y que permiten
organizar y localizar los recursos
228                     con más facilidad. Se ha utilizado para definirlas la función intrínseca
Fn::Join ,que permite unir
229                     cadenas de caracteres. Además se ha hecho referencia a un parámetro que
provee CloudFormation que es
230                     el nombre de la pila elegida por el usuario (AWS::StackName) */
231              }
232          },
233
234          "PublicSubnet" : {
235              /* Creación de la subred pública dentro la VPC */
236              "Type" : "AWS::EC2::Subnet",
237              "Properties" : {
238                  "VpcId" : { "Ref" : "VPC" }, //Referencia a la VPC en la que crear la subred
239                  "CidrBlock" : "172.16.1.0/24", //Rango de IPs asociado a la subred y que
debe estar dentro del rango de la VPC
240                  "AvailabilityZone" : { "Fn::Select" : [ 1, { "Fn::GetAZs" : { "Ref" :
"AWS::Region" } } ] },
241                  /* Zona de disponibilidad en la que crear la subred. Debe ser una zona de
disponibilidad que se encuentre en la
242                    region en la que se está creando la pila. Para ello se ha utilizado la
función Fn::GetAZs, que nos devuelve
243                    un array con las zonas de disponibilidad que hay en una región (obtenida
mediante el parámetro que provee
244                    CloudFomation AWS::Region). Además, se ha utilizado la función Fn::Select
para elegir un valor del array */
245                  "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "PublicSubnet" ] ] } } ]
246              }
247          },
248
249          "PrivateSubnet" : {
250              /* Creación de la subred privada dentro de la VPC. Los parámemtro son similares
a los de la subred pública
251              pero eligiendo un rango de IPs distinto */
252              "Type" : "AWS::EC2::Subnet",
253              "Properties" : {
254                  "VpcId" : { "Ref" : "VPC" },
255                  "CidrBlock" : "172.16.2.0/24",
256                  "AvailabilityZone" : { "Fn::GetAtt" : [ "PublicSubnet", "AvailabilityZone" ]
},
257                  /* Dado que las instancias van a tener dos interfaces de red conectadas cada
una a una subred, ambas deben
258                  estar en la misma zona de disponibilidad, ya que una instancia no puede
estar en dos zonas de disponibilidad
259                  al mismo tiempo. Por lo tanto se utiliza la función intrínseca Fn:GetAtt
para obtener una característica
260                  del recurso "PrivateSubnet", en este caso la zona de disponibilidad, y así
configurar la misma */
261                  "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "PrivateSubnet" ] ] } } ]
```

```
262             }
263         },
264
265         "BackupSubnet" : {
266             /* Algunos servicios de AWS necesitan otra subred en una zona de disponibilidad
distinta para alta disponibilidad */
267             "Type" : "AWS::EC2::Subnet",
268             "Properties" : {
269                 "VpcId" : { "Ref" : "VPC" },
270                 "CidrBlock" : "172.16.3.0/24",
271                 "AvailabilityZone" : { "Fn::Select" : [ 2, { "Fn::GetAZs" : { "Ref" :
"AWS::Region" } } ] },
272                 "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "BackupSunbet" ] ] } } ]
273             }
274         },
275
276         "InternetGateway" : {
277             /* Creación de una puerta de enlace a Internet. Primero se crea con este recurso
y después se asocia a la VPC */
278             "Type" : "AWS::EC2::InternetGateway",
279             "Properties" : {
280                 "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "InternetGateway" ] ] } } ]
281             }
282         },
283
284         /* Recurso que representa la asociación de una puerta de enlace a Internet con una
VPC */
285         "AttachGateway" : {
286             "Type" : "AWS::EC2::VPCGatewayAttachment",
287             "Properties" : {
288                 "VpcId" : { "Ref" : "VPC" },
289                 "InternetGatewayId" : { "Ref" : "InternetGateway" }
290             }
291         },
292
293         /* Tabla de rutas . Crea el recurso sin ningua entrada, que serán añadidas
294         en los siguientes recursos definidos y serán asociada a la subred pública. La subred
privada
295         no necesita tabla de rutas porque está aislada. */
296         "RouteTable" : {
297             "Type" : "AWS::EC2::RouteTable",
298             "Properties" : {
299                 "VpcId" : { "Ref" : "VPC" },
300                 "Tags" : [ { "Key" : "Name", "Value" : { "Fn::Join" : [ " - ", [ { "Ref" :
"AWS::StackName" }, "RouteTable" ] ] } } ]
301             }
302         },
303
304         /* Ruta por defecto asociada a la puerta de enlace a internet,
305         que provee de acceso a este. */
306         "Route" : {
307             "Type" : "AWS::EC2::Route",
308             "DependsOn" : "AttachGateway",
309             "Properties" : {
310                 "RouteTableId" : { "Ref" : "RouteTable" },
311                 "DestinationCidrBlock" : "0.0.0.0/0",
312                 "GatewayId" : { "Ref" : "InternetGateway" }
313             }
314         },
315
316         /* Añade a la tabla de rutas la subred pública, que junto con la ruta por defecto
otorga
317         acceso a Internet a las interfaces conectadas a esa subred. */
318         "PublicSubnetRouteTableAssociation" : {
319             "Type" : "AWS::EC2::SubnetRouteTableAssociation",
320             "Properties" : {
321                 "SubnetId" : { "Ref" : "PublicSubnet" },
322                 "RouteTableId" : { "Ref" : "RouteTable" }
323             }
324         },
325
326         /* Rol de AWS IAM asociado a las instancias de Amazon EC2 */
327         "IAMRole" : {
328             "Type" : "AWS::IAM::Role",
329             "Properties" : {
330                 "AssumeRolePolicyDocument" : {
331                     "Version" : "2012-10-17",
332                     "Statement" : [ {
```

```
333                         "Effect" : "Allow",
334                         "Principal" : {
335                             "Service" : [ "ec2.amazonaws.com" ]
336                         },
337                         "Action" : [ "sts:AssumeRole" ]
338                     } ]
339                 },
340                 "Path" : "/"
341             }
342         },
343
344         /* Recurso derivado del rol necesario para poder asignarse a las instancias EC2*/
345         "IAMInstanceProfile" : {
346             "Type" : "AWS::IAM::InstanceProfile",
347             "Properties" : {
348                 "Path" : "/",
349                 "Roles" : [ { "Ref" : "IAMRole" } ]
350             }
351         },
352
353         /* Usuario de IAM para las instancias de redBorder */
354         "IAMUser" : {
355             "Type" : "AWS::IAM::User",
356             "Properties" : { }
357         },
358
359         /* IAM Keys asociadas al usuario creado anteriormente. Se pasan a la instancia
360         por User-Data para su configuración en los servicios que necesiten de acceso a
361         recursos de EC2 */
362         "IAMAccessKey" : {
363             "Type" : "AWS::IAM::AccessKey",
364             "Properties" : {
365                 "Status" : "Active",
366                 "UserName" : { "Ref" : "IAMUser" }
367             }
368         },
369
370         /* Política asociada al usuario y al rol declarados anteriormente. Otorga los
permisos
371         necesarios para que la instancia realice las labores de autoconfiguración del
372         clúster, y utilice servicios proveídos por AWS.  */
373         "IAMPolicy" : {
374             "Type" : "AWS::IAM::Policy",
375             "Properties" : {
376                 "PolicyName" : "RB-CF-POLICY",
377                 "PolicyDocument" : {
378                     "Version" : "2012-10-17",
379                     "Statement": [ {
380                         "Action" : [ //Permite la creacion de la segunda interfaz de red de
los nodos
381                             "ec2:CreateNetworkInterface",
382                             "ec2:AttachNetworkInterface",
383                             "ec2:ModifyNetworkInterfaceAttribute",
384                             "ec2:DescribeInstances",
385                             //Permite asociar IPs flotantes para alta disponbilidad de
algunos servicios
386                             "ec2:AssignPrivateIpAddresses",
387                             "ec2:UnassignPrivateIpAddresses" ],
388                         "Effect" : "Allow",
389                         "Resource" : "*"
390                     }, { //Acceso al bucket S3 creado para el clúster. Hace referencia al
recurso
391                         //S3Bucket, indicado posteriormente.
392                         "Action" : "s3:*",
393                         "Effect" : "Allow",
394                         "Resource" : [ { "Fn::Join" :
395                             [ "", [ "arn:aws:s3:::",
396                             { "Ref" : "S3Bucket" }, "/*" ] ] },
397                             { "Fn::Join" :
398                             [ "", [ "arn:aws:s3:::",
399                             { "Ref" : "S3Bucket" } ] ] } ]
400                     }, { //Permite a la instancia conocer otros recursos creados en la
plantilla.
401                         "Effect": "Allow",
402                         "Action": [ "cloudformation:DescribeStackResource" ],
403                         "Resource": { "Fn::Join" : [ "", [ "arn:aws:cloudformation:", {
"Ref" : "AWS::Region" }, ":",
404                                     { "Ref" : "AWS::AccountId" }, ":stack/", { "Ref" :
"AWS::StackName" }, "/*"    ] ] }
```

```
405                     },{ //Permite a la instancia enviar métricas a AWS Cloudwatch. Este
permiso es requerido por
406                         //el servicio rb_cloudwatch y por los scripts de envío de métricas
de la instancia que provee
407                         //Amazon. Además, también se permite ver qué alarmas hay en AWS
Cloudwatch, necesario para el
408                         //funcionamiento del servicio SQSLD.
409                         "Effect": "Allow",
410                         "Action" : [ "cloudwatch:PutMetricData", "cloudwatch:DescribeAlarms"
],
411                         "Resource" : "*"
412                     },{  //Permite la creación de entradas en Route53.
413                         "Effect": "Allow",
414                         "Action" : [ "route53:ListHostedZones",
"route53:ChangeResourceRecordSets", "route53:GetHostedZone" ],
415                         "Resource" : "*"
416                     },{  //Permite a SQSLD gestionar la finalización de las instancias en
los grupos de autoescalado.
417                         "Effect": "Allow",
418                         "Action" : [ "autoscaling:TerminateInstanceInAutoscalingGroup",
"autoscaling:DescribeAutoscalingGroups",
419                                      "autoscaling:CompleteLifecycleAction" ],
420                         "Resource" : "*"
421                     },{  //Permite a SQSLD recibir mensajes de la cola SQS, y eliminarlos
cuando se finaliza su procesamiento.
422                         "Effect": "Allow",
423                         "Action" : [ "sqs:ReceiveMessage", "sqs:DeleteMessage" ],
424                         "Resource" : [ { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] } ]
425                     }, ]
426                 },
427                 //Roles y usuarios a los que se asocia la política
428                 "Roles" :  [ { "Ref" : "IAMRole" } ],
429                 "Users" : [ { "Ref" : "IAMUser" } ]
430             }
431         },
432
433         "IAMUserChef" : {
434             /* Se crea un segundo usuario específico para Chef, que únicamente tiene
permisos para acceder a la carpeta
435                donde se almacenan las plantillas. Esto es así porque los sensores también
necesitan leer las plantillas
436                almacenadas en S3, por lo que hay que restringir el acceso de estos a la
gestión del sistema. */
437             "Type" : "AWS::IAM::User",
438             "Properties" : { }
439         }, //Credenciales asociadas al usuario de chef
440         "IAMAccessKeyChef" : {
441             "Type" : "AWS::IAM::AccessKey",
442             "Properties" : {
443                 "Status" : "Active",
444                 "UserName" : { "Ref" : "IAMUserChef" }
445             }
446         },
447
448         "IAMPolicyChef" : {
449             /* Política para el usuario de Chef. Como ya se ha dicho, sólo se permite el
acceso a un directorio de S3. */
450             "Type" : "AWS::IAM::Policy",
451             "Properties" : {
452                 "PolicyName" : "RB-CF-POLICY",
453                 "PolicyDocument" : {
454                     "Version" : "2012-10-17",
455                     "Statement": [ {
456                         "Action" : "s3:*",
457                         "Effect" : "Allow",
458                         "Resource" : [ { "Fn::Join" :
459                             [ "", [ "arn:aws:s3:::",
460                             { "Ref" : "S3Bucket" },
461                             "/organization-00000000000000000000000000000000/*" ] ] } ]
462                     } ]
463                 },
464                 "Users" : [ { "Ref" : "IAMUserChef" } ]
465             }
466         },
467
468         "IAMRoleLifecycleHook" : {
469         /* Rol que se asocia a los grupos de autoescalado para que puedan
470           enviar mensajes a SQS con información relacionada con los lifecycle hooks */
471             "Type" : "AWS::IAM::Role",
472             "Properties" : {
```

```
473                 "AssumeRolePolicyDocument" : {
474                     "Version" : "2012-10-17",
475                     "Statement" : [ {
476                         "Effect" : "Allow",
477                         "Principal" : {
478                             "Service" : [ "ec2.amazonaws.com", "autoscaling.amazonaws.com" ]
479                         },
480                         "Action" : [ "sts:AssumeRole" ]
481                     } ]
482                 },
483                 "Path" : "/"
484             }
485         },
486
487         "IAMPolicyLifecycleHook" : {
488         //Política asociada a los grupos de autoescalado.
489             "Type" : "AWS::IAM::Policy",
490             "Properties" : {
491                 "PolicyName" : "RB-CF-POLICY-FOR-LIFECYCLE-HOOKS",
492                 "PolicyDocument" :  {
493                     "Version" : "2012-10-17",
494                     "Statement": [ {
495                         "Effect": "Allow",
496                         "Action" : [ "sqs:*" ],
497                         "Resource" : [ { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] } ]
498                     } ]
499                 },
500                 "Roles" :  [ { "Ref" : "IAMRoleLifecycleHook" } ]
501             }
502         },
503
504         "S3Bucket" : {
505         //Creación del bucket de S3 que utilizará redBorder.
506             "Type" : "AWS::S3::Bucket",
507             "Properties" : {
508                 "AccessControl" : "Private"
509             },
510             "DeletionPolicy" : "Delete"
511         },
512
513         "S3BucketPolicy" : {
514         /* Habilita la posibilidad de publicar información accesible desde internet en la
carpeta
515            Public del bucket. */
516             "Type" : "AWS::S3::BucketPolicy",
517             "Properties" : {
518                 "Bucket" : { "Ref" : "S3Bucket" },
519                 "PolicyDocument" : {
520                     "Statement" : [ {
521                         "Effect" : "Allow",
522                         "Principal" : "*",
523                         "Action" : [ "s3:GetObject" ],
524                         "Resource" : { "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" :
"S3Bucket" },
525                              "/Public/*" ] ] }
526                     } ]
527                 }
528             }
529         },
530
531         "SQSQueue" : {
532         /* Cola SQS utilizada para la recepción por parte del servicio SQSLD de los mensajes
de
533            autoescalado y la gestión de alarmas. */
534             "Type" : "AWS::SQS::Queue",
535             "Properties" : {
536                 "MessageRetentionPeriod" : "1800", //Los mensajes se retendrán en la cola
durante
537                                                    //un máximo de media hora.
538                 "ReceiveMessageWaitTimeSeconds" : "20",
539                 "VisibilityTimeout" : "30" //Cuando un mensaje es entregado, queda oculto
durante 30 segundos al resto
540                                            //de clientes que leen de la cola.
541             }
542         },
543
544
545         "SQSPolicy" : {
546         /* Política que permite a SNS enviar los mensajes a la cola SQS.*/
```

```
547             "Type" : "AWS::SQS::QueuePolicy",
548             "Properties" : {
549                 "Queues" : [ { "Ref" : "SQSQueue" } ],
550                 "PolicyDocument" : {
551                     "Version":"2012-10-17",
552                     "Statement":[ {
553                         "Sid": { "Fn::Join" : [ "", [ { "Fn::GetAtt" : [ "SQSQueue",
"QueueName"  ] }, "-Policy" ] ] } ,
554                         "Effect":"Allow",
555                         "Principal":"*",
556                         "Action":"sqs:SendMessage",
557                         "Resource": { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] },
558                         "Condition":{
559                             "ArnEquals": { "aws:SourceArn" : { "Ref" : "SNSTopic" } }
560                         }
561                     } ]
562                 }
563             }
564         },
565
566         "SNSTopic" : {
567         /* Topic de SNS utilizado para el envío de notificaciones de alarmas de Cloudwatch.
Además,
568            se configura la subscripción de la cola SQS, para que el servicio SQSLD pueda
realizar
569            acciones cuando se produce una alarma. */
570             "Type" : "AWS::SNS::Topic",
571             "Properties" : {
572                 "Subscription" : [ {
573                     "Protocol" : "sqs",
574                     "Endpoint" : { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] }
575                 } ]
576             }
577         },
578
579         "DBRDS" : {
580         /* Instancia de la base de datos RDS. La configuración es la siguiente:
581              - Almacenamiento: 50 GB.
582              - Tipo de instancia: se elige en función del parámetro Instance Size, mediante
el uso del
583                  mapa InstanceTypeMap.
584              - Motor de base de datos: PostgreSQL.
585              - Usuario de la cuenta de administrador: rdsredborder.
586              - Contraseña: se elige con el parámetro DBPassword cuando se crea la Pila de
Cloudformation.
587              - Acceso restringido únicamente desde la subred privada.
588              - Las copias de seguridad automáticas se mantendrán durante 7 días.
589              - Se habilita el despliegue en múltiples zonas de disponibilidad para mayor
fiabilidad del servicio
590              - El tipo de almacenamiento utilizado es en discos SSD
591              - Se asigna un conjunto de subredes y de grupos de seguridad
592
593              Cabe destacar que la instancia únicamente será creada si se habilita su
creación en el parámetro
594              UseRDS
595          */
596             "Type" : "AWS::RDS::DBInstance",
597             "Condition" : "RDSCreation",
598             "Properties" : {
599                 "AllocatedStorage" : "50",
600                 "DBInstanceClass" : { "Fn::FindInMap" : [ "InstanceTypeMap", "rds", { "Ref"
: "InstanceSize" } ] },
601                 "Engine" : "postgres",
602                 "DBName" : "redborderRDS",
603                 "MasterUsername" : "rdsredborder",
604                 "MasterUserPassword" : { "Ref" : "DBPassword" },
605                 "PubliclyAccessible" : "false",
606                 "BackupRetentionPeriod" : "7",
607                 "MultiAZ" : "true",
608                 "StorageType" : "gp2",
609                 "DBSubnetGroupName" : { "Ref" : "DBSubnetGroup" },
610                 "VPCSecurityGroups" : [ { "Ref" : "DataBaseSecurityGroup" } ]
611             }
612         },
613
614         "DBSubnetGroup" : {
615         /* Subredes en las que se desplegaran las instancias de RDS. Son necesarias dos,
616         la subred privada (por defecto) y para alta disponibilidad otra subred de backup */
617             "Type" : "AWS::RDS::DBSubnetGroup",
618             "Condition" : "RDSCreation",
```

```
619             "Properties" : {
620                 "DBSubnetGroupDescription" : "redBorder RDS subnet group",
621                 "SubnetIds" : [ { "Ref" : "PrivateSubnet" }, { "Ref" : "BackupSubnet" } ]
622             }
623         },
624
625         "DataBaseSecurityGroup" : {
626         /* Grupo de seguridad de las instancias RDS. Sólo se permite el acceso desde
627             la VPC y al puerto donde se encuentra el servicio PostgreSQL. */
628             "Type" : "AWS::EC2::SecurityGroup",
629             "Properties" : {
630                 "GroupDescription" : "Enable access to DataBase for VPC instances",
631                 "VpcId" : { "Ref" : "VPC" },
632                 "SecurityGroupIngress" : [ {
633                     "IpProtocol" : "tcp",
634                     "FromPort" : "5432",
635                     "ToPort" : "5432",
636                     "CidrIp" : "172.16.0.0/16"
637                 } ]
638             }
639         },
640
641         "Memcached" : {
642         /* Cluster de Elasticache con la siguiente configuración:
643              - Motor de cache: Memcached.
644              - Zona de disponibilidad preferida: La de la subred privada.
645              - Numero de nodos: 1
646              - Tipo de nodo: dependerá del parámetro InstanceSize.
647              - Se utilizaran grupos de seguridad y una serie de subredes
648                para el servicio.
649             Este recurso sólo se desplegará si se indica en el parámetro
650             UseElasticache
651         */
652             "Type" : "AWS::ElastiCache::CacheCluster",
653             "Condition" : "ElasticacheCreation",
654             "Properties" : {
655                 "Engine" : "memcached",
656                 "PreferredAvailabilityZone" : { "Fn::GetAtt" : [ "PrivateSubnet",
"AvailabilityZone" ] },
657                 "NumCacheNodes" : "1",
658                 "CacheNodeType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "memcached", {
"Ref" : "InstanceSize" } ] },
659                 "CacheSubnetGroupName" : { "Ref" : "MemcachedSubnetGroup" },
660                 "VpcSecurityGroupIds" : [ { "Ref" : "MemcachedSecurityGroup" } ]
661             }
662         },
663
664         "MemcachedSubnetGroup" : {
665         /* Conjunto de subredes en las que desplegar el clúster de Elasticache.
666            Se utilizara la subred privada y la de backup, al igual que con RDS */
667             "Type" : "AWS::ElastiCache::SubnetGroup",
668             "Condition" : "ElasticacheCreation",
669             "Properties" : {
670                 "Description" : "Subnet Group for Memcached",
671                 "SubnetIds" : [ { "Ref" : "PrivateSubnet" }, { "Ref" : "BackupSubnet" } ]
672             }
673         },
674
675         "MemcachedSecurityGroup" : {
676         /* Grupo de seguridad para los nodos de Elasticache. Sólo se permite el acceso desde
677         la VPC del clúster y al puerto del servicio de Memcached. */
678             "Type" : "AWS::EC2::SecurityGroup",
679             "Condition" : "ElasticacheCreation",
680             "Properties" : {
681                 "GroupDescription" : "Enable access to DataBase for VPC instances",
682                 "VpcId" : { "Ref" : "VPC" },
683                 "SecurityGroupIngress" : [ {
684                     "IpProtocol" : "tcp",
685                     "FromPort" : "11211",
686                     "ToPort" : "11211",
687                     "CidrIp" : "172.16.0.0/16"
688                 } ]
689             }
690         },
691
692         "redBorderSecurityGroup" : {
693         /* Grupo de seguridad de las instancias de redBorder con las siguientes
características:
```

```
694          - Se permite el acceso desde internet al puerto 6666 (SSH) para la gestión de las
instancias.
695          - Acceso al puerto 80 para el balanceador (subred pública). Utilizado para
peticiones HTTP a la web
696             que serán redirigidas para que utilicen HTTPS
697          - Acceso al puerto 7979 para el Load Balancer. En este puerto escucha Nginx, que
gestiona
698            las peticiones y las redirige al servicio correspondiente */
699              "Type" : "AWS::EC2::SecurityGroup",
700              "Properties" : {
701                  "GroupDescription" : "Enable SSH, HTTP and HTTPS",
702                  "VpcId" : { "Ref" : "VPC" },
703                  "SecurityGroupIngress" : [ {
704                      "IpProtocol" : "tcp",
705                      "FromPort" : "6666",
706                      "ToPort" : "6666",
707                      "CidrIp" : "0.0.0.0/0"
708                  }, {
709                      "IpProtocol" : "tcp",
710                      "FromPort" : "80",
711                      "ToPort" : "80",
712                      "CidrIp" : "172.16.1.0/24"
713                  }, {
714                      "IpProtocol" : "tcp",
715                      "FromPort" : "7979",
716                      "ToPort" : "7979",
717                      "CidrIp" : "172.16.1.0/24"
718                  } ]
719              }
720          },
721
722          "WebLoadBalancerSecurityGroup" : {
723          /* Grupo de seguridad del balanceador. Permite sólo el acceso a tráfico HTTP y HTTPS
724              (puertos TCP 80 y 443) desde internet. */
725              "Type" : "AWS::EC2::SecurityGroup",
726              "Properties" : {
727                  "GroupDescription" : "Enable HTTP and HTTPS",
728                  "VpcId" : { "Ref" : "VPC" },
729                  "SecurityGroupIngress" : [ {
730                      "IpProtocol" : "tcp",
731                      "FromPort" : "443",
732                      "ToPort" : "443",
733                      "CidrIp" : "0.0.0.0/0"
734                  }, {
735                      "IpProtocol" : "tcp",
736                      "FromPort" : "80",
737                      "ToPort" : "80",
738                      "CidrIp" : "0.0.0.0/0"
739                  } ]
740              }
741          },
742
743          "PublicHostedZone" : {
744          /* Hosted Zone de Route 53 para acceso público desde internet. El nombre de la
hostedzone es el CDOMAIN indicado
745             como parámetro de la Pila. Las instancias utilizan el script rb_route53.sh
746             para añadir sus ips públicas a la Hosted Zone, utilizando su instance id como
nombre. De esta forma,
747             se puede acceder a las instancias desde internet utilizado un nombre con el
formato instanceid.cdomain
748          */
749              "Type" : "AWS::Route53::HostedZone",
750              "Properties" : {
751                  "HostedZoneConfig" : {
752                      "Comment" : { "Fn::Join" : [ "", [ "Public Hosted Zone for stack ", {
"Ref" : "AWS::StackName" } ] ] }
753                  },
754                  "Name" : { "Fn::Join" : [ "", [ { "Ref" : "CDOMAIN" }, "." ] ] }
755              }
756          },
757
758          "PrivateHostedZone" : {
759          /* Hosted Zone de Route 53 para acceso exclusivo de la VPC. Las instancias registran
el mismo nombre que
760             en la Hosted Zone pública, pero con la IP de sus interfaces privadas, para
resolución de nombre en las
761             comunicaciones internas del clúster. */
762              "Type" : "AWS::Route53::HostedZone",
763              "Properties" : {
764                  "HostedZoneConfig" : {
```

```
765                     "Comment" : { "Fn::Join" : [ "", [ "Private Hosted Zone for stack ", { "Ref" : "AWS::StackName" } ] ] }
766                 },
767                 "Name" : { "Fn::Join" : [ "", [ { "Ref" : "CDOMAIN" }, "." ] ] },
768                 "VPCs" : [ {
769                     "VPCId" : { "Ref" : "VPC" },
770                     "VPCRegion" : { "Ref" : "AWS::Region" }
771                 } ]
772             }
773         },
774         "WebLoadBalancerDNSName" : {
775         /* Entrada en la hosted zone pública para que el nombre indicado en la variable CDOMAIN resuelva al balanceador
776             de la web */
777             "Type" : "AWS::Route53::RecordSet",
778             "Properties" : {
779                 "HostedZoneId" : { "Ref" : "PublicHostedZone" },
780                 "Name" : { "Ref" : "CDOMAIN" },
781                 "AliasTarget" : {
782                     "HostedZoneId" : { "Fn::GetAtt" : [ "WebElasticLoadBalancer", "CanonicalHostedZoneNameID" ] },
783                     "DNSName" : { "Fn::GetAtt" : [ "WebElasticLoadBalancer", "DNSName" ] }
784                 },
785                 "Type" : "A"
786             }
787         },
788
789         "DataLoadBalancerDNSName" : {
790         /* Entrada en la hosted zone pública para que el nombre data.CDOMAIN resuelva al balanceador de datos de los sensores */
791             "Type" : "AWS::Route53::RecordSet",
792             "Condition" : "corezkRoleCreation",
793             "DependsOn" : "corezkGroup",
794             "Properties" : {
795                 "HostedZoneId" : { "Ref" : "PublicHostedZone" },
796                 "Name" : { "Fn::Join" : [ "", [ "data.", { "Ref" : "CDOMAIN" } ] ] },
797                 "AliasTarget" : {
798                     "HostedZoneId" : { "Fn::GetAtt" : [ "WebElasticLoadBalancer", "CanonicalHostedZoneNameID" ] },
799                     "DNSName" : { "Fn::GetAtt" : [ "WebElasticLoadBalancer", "DNSName" ] }
800                 },
801                 "Type" : "A"
802             }
803         },
804         "WebElasticLoadBalancer" : {
805             /* Creación del balanceador para la web. Se explicarán los parámetros en los siguientes comentarios */
806             "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
807             "Properties" : {
808                 "CrossZone" : "true", //habilita el balanceo entre varias zonas de disponibilidad
809                 "SecurityGroups" : [ { "Ref" : "WebLoadBalancerSecurityGroup" } ], //Grupo de seguridad
810
811 //asociado al balanceador
812                 "Subnets" : [ { "Ref" : "PublicSubnet" } ], //El trafico del balanceador es enviado por la
813                                                             //subred pública
814                 "ConnectionDrainingPolicy" : { //Cuando se quiere desregistrar una instancia del balanceador,
815                     "Enabled" : "true", //se espera un máximo de 30 segundos a que la instancia finalice las
816                     "Timeout" : "30"    //conexiones abiertas. Cuando las finaliza o expira el temporizador,
817                                         //se completa el registro
818                 },
819                 "Listeners" : [ { //El loadbalancer recibirá tráfico por el puerto 443 con el protocolo HTTPS.
820                     "LoadBalancerPort" : "443", //Terminará el tráfico HTTPS con el certificado especificado
821                     "InstancePort" : "7979", //en el parámetro SSLCertificateName y lo reenviará sin cifrar (HTTP)
822                     "Protocol" : "HTTPS", //al puerto 7979 donde escucha el servicio NGINX de redBorder
823                     "InstanceProtocol" : "HTTP",
824                     "SSLCertificateId" : { "Fn::Join" : [ "", [ "arn:aws:iam::", { "Ref" : "AWS::AccountId" },
                            ":server-certificate/", { "Ref" : "SSLCertificateName" } ] ] }
```

```
825                 }, { //El tráfico HTTP al puerto 80 se redirige al puerto 80 de las
instancias para que
826                     "LoadBalancerPort" : "80", //sea la instancia la que redirija para el
uso de HTTPS
827                     "InstancePort" : "80",
828                     "Protocol" : "HTTP",
829                     "InstanceProtocol" : "HTTP"
830                 } ],
831                 "HealthCheck" : { //Comprobación del estado de las instancias. Si se reciben
tres respuestas 2XX
832                     "Target" : "HTTP:7979/users/login", //de la instancia seguidas, la
instancia es puesta en
833                     "HealthyThreshold" : "3", //servicio. Si en 5 comprobaciones seguidas no
se reciben respuestas
834                     "UnhealthyThreshold" : "5", //2XX, la instancia es puesta como fuera de
servicio. La comprobación
835                     "Interval" : "30", //se realiza cada 30 segundos, y se espera la
respuesta a cada comprobación
836                     "Timeout" : "25" //un máximo de 25 segundos
837                 }
838             }
839         },
840
841         "DataElasticLoadBalancer" : { //La configuración del balanceador para los datos es
muy similar al anterior, s
842             "Type" : "AWS::ElasticLoadBalancing::LoadBalancer", // salvo porque sólo escucha
en el puerto 443
843             "Properties" : {
844                 "CrossZone" : "true",
845                 "SecurityGroups" : [ { "Ref" : "WebLoadBalancerSecurityGroup" } ],
846                 "Subnets" : [ { "Ref" : "PublicSubnet" } ],
847                 "ConnectionDrainingPolicy" : {
848                     "Enabled" : "true",
849                     "Timeout" : "30"
850                 },
851                 "Listeners" : [ {
852                     "LoadBalancerPort" : "443",
853                     "InstancePort" : "7979",
854                     "Protocol" : "HTTPS",
855                     "InstanceProtocol" : "HTTP",
856                     "SSLCertificateId" : { "Fn::Join" : [ "", [ "arn:aws:iam::", { "Ref" :
"AWS::AccountId" },
857                         ":server-certificate/", { "Ref" : "SSLCertificateName" } ] ] }
858                 } ],
859                 "HealthCheck" : {
860                     "Target" : "HTTP:7979/nginx_status",
861                     "HealthyThreshold" : "3",
862                     "UnhealthyThreshold" : "5",
863                     "Interval" : "30",
864                     "Timeout" : "25"
865                 }
866             }
867         },
868
869         "masterGroup" : {
870         /*Grupo de autoescalado para el rol especial master. Este rol se utiliza cuando el
número de instancias
871           es menor que el número de roles estándar, y se caracteriza por implementar todos
los servicios de redBorder.
872
873           La configuración que se debe proveer en este y en el resto de grupos de
autoescalado es la siguiente:
874           - Subred en la que desplegar la instancia (interfaz eth0), que provoca que sea
creada en la zona
875                de disponibilidad de la subred.
876           - Configuración de arranque asociada (se hace referencia a otro recurso que se
explicará después)
877           - Balanceador de carga asociado (en el caso de que el rol lo requiera)
878           - Mínimo número de instancias que debe tener (depende del número de nodos
desplegados, encontrándose
879                en el mapa RolesDistributionMap)
880           - Máximo número de instancias que puede llegar a tener. En todos los grupos se ha
elegido 10.
881           - Etiquetas con el role, el nombre del clúster y un nombre para la instancia
creado apartir de las dos
882             etiquetas anteriores.
883           - Se aplican políticas de creación y actualización, indicándose que para que el
recurso pase al
884                estado CREATE_COMPLETE deben recibirse tantas señales de CloudFormation como
mínimo número
```

```
885                 de instancias tenga el grupo de autoescalado.
886             */
887               "Type" : "AWS::AutoScaling::AutoScalingGroup",
888               "DependsOn" : "AttachGateway",
889               "Condition" : "masterRoleCreation",
890               "Properties" : {
891                   "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
892                   "LaunchConfigurationName" : { "Ref" : "masterLaunch" },
893                   "LoadBalancerNames" : [ { "Ref" : "WebElasticLoadBalancer" } ],
894                   "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "master" ] },
895                   "MaxSize" : "10",
896                   "Tags" : [ {
897                       "Key" : "Role",
898                       "Value" : "master",
899                       "PropagateAtLaunch" : "true"
900                   },{
901                       "Key" : "Cluster",
902                       "Value" : { "Ref" : "AWS::StackName" },
903                       "PropagateAtLaunch" : "true"
904                   },{
905                       "Key" : "Name",
906                       "Value" : { "Fn::Join" : [ "", [
907                           { "Ref" : "AWS::StackName" }, " - master" ] ] },
908                       "PropagateAtLaunch" : "true"
909                   } ]
910               },
911               "CreationPolicy" : {
912                   "ResourceSignal" : {
913                       "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "master" ] },
914                       "Timeout" : "PT200M"
915                   }
916               },
917               "UpdatePolicy": {
918                   "AutoScalingRollingUpdate": {
919                       "MinInstancesInService": "1",
920                       "MaxBatchSize": "1",
921                       "PauseTime" : "PT40M",
922                       "WaitOnResourceSignals": "true"
923                   }
924               }
925           },
926
927           "masterLaunch" : {
928           /* Configuración de arranque de las instancias del grupo de autoescalado para el rol
master.
929              Se especifica los siguiente:
930              - AssociatePublicIpAddress: Si se desea asociar una IP pública a las instancias.
931                  En redBorder siempre se asignarán,
932                  para poder acceder a ellas mediante SSH.
933              - ImageID: AMI a utilizar para el despliegue. En función de la región se elige la
AMI de
934                  redBorder adecuada, que
935                  se encuentra en el mapa AWSRegion2AMI.
936              - SecurityGroups: Grupos de seguridad asociado a las instancias. Todos los roles
937                  tienes el mismo grupo de seguridad
938                  asociado, que ya se ha descrito anteriormente.
939              - KeyName: clave para el acceso por SSH a la instancia.
940              - InstanceType: tipo de instancia para el rol. Se elige del mapa InstanceSize.
941              - IamInstanceProfile: identifica al rol que se asocia a la instancia, dándole
permisos
942                  para realizar operaciones con la API de AWS.
943              - InstanceMonitoring: permite elegir si se quiere monitorización detallada para
la instancia
944                  en Cloudwatch
945              - BlockDeviceMappings: permite elegir las unidades de almacenamiento que serán
asignadas a la instancia.
946                  en función del rol se eligen distintos tamaños de almacenamiento.
947              - User-Data: permite especificar el user data para las instancias del rol.
948           */
949               "Type" : "AWS::AutoScaling::LaunchConfiguration",
950               "Condition" : "masterRoleCreation",
951               "Properties" : {
952                   "AssociatePublicIpAddress" : "true",
953                   "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
954                   "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
955                   "KeyName"        : { "Ref" : "KeyName" },
```

```
956                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "master", { "Ref"
: "InstanceSize" } ] },
957                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
958                 "InstanceMonitoring" : "true",
959                 "BlockDeviceMappings" : [ {
960                     "DeviceName" : "/dev/sda1",
961                     "Ebs" : {
962                         "VolumeSize" : "500",
963                         "VolumeType" : "gp2",
964                         "DeleteOnTermination" : "true"
965                     }
966                 },  {
967                     "DeviceName" : "/dev/sdb",
968                         "VirtualName" : "ephemeral0"
969                 },  {
970                     "DeviceName" : "/dev/sdc",
971                         "VirtualName" : "ephemeral1"
972                 } ],
973                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
974                     "#!/bin/bash\n",
975                     "NODEROLE=master\n",
976                     "#NODESERVICES=\n",
977                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
978                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
979                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
980                     "REGION=", { "Ref" : "AWS::Region" }, "\n",
981                     "VPCID=", { "Ref" : "VPC" }, "\n",
982                     "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
983                     "AUTOSCALINGGROUPNAME=masterGroup\n",
984                     "LOGWATCHG=", { "Ref" : "AWS::StackName" }, "\n",
985                     "SQSQUEUEURL=", { "Ref" : "SQSQueue" }, "\n",
986                     "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
987                     "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
988                     "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
989                     "S3TYPE=aws\n",
990                     "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
991                     "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
992                     "CHEF_AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKeyChef" }, "\"\n",
993                     "CHEF_AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKeyChef",
"SecretAccessKey" ] }, "\"\n",
994                     //Solo se pasan los parámetros de configuración de Elasticache en el
caso de que el parámetro
995                     //UseElasticache lo indique. Para ello se utiliza la función intrínseca
Fn::If
996                     { "Fn::If" : [ "ElasticacheCreation", { "Fn::Join" : [ "", [
997                         "ELASTICCACHECLUSTERID=", { "Ref" : "Memcached" }, "\n",
998                         "ELASTICCACHEENDPOINT=", { "Fn::Join" : [ "", [ { "Fn::GetAtt" : [
"Memcached", "ConfigurationEndpoint.Address" ] },
999                         ":", { "Fn::GetAtt" : [ "Memcached", "ConfigurationEndpoint.Port" ]
} ] ] }, "\n"
1000                         ] ] },
1001                     "" ] },
1002                     //Solo se pasan los parámetros de configuración de RDS en el caso de que
el parámetro
1003                     //UseRDS lo indique. Para ello se utiliza la función intrínseca Fn::If
1004                     { "Fn::If" : [ "RDSCreation", { "Fn::Join" : [ "", [
1005                         "SQLUSER=rdsredborder\n",
1006                         "SQLPASSWORD=", { "Ref" : "DBPassword" }, "\n",
1007                         "SQLDB=redborderRDS\n",
1008                         "SQLHOST=", { "Fn::Join" : [ "", [ { "Fn::GetAtt" : [ "DBRDS",
"Endpoint.Address" ] }, ":",
1009                             { "Fn::GetAtt" : [ "DBRDS", "Endpoint.Port" ] } ] ] }, "\n"
1010                         ] ] },
1011                     "" ] },
1012                     "ENRICHMODE=samza\n",
1013                     "\n\n",
1014                     "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1015                     "{\n",
1016                     "  \"redBorder\" : {\n",
1017                     "    \"aws\" : {\n",
1018                     "      \"cloudwatch\" : \"true\"\n",
1019                     "    }\n",
1020                     "  }\n",
1021                     "}\n",
1022                     "_RBEOF2_\n",
1023                     ". /opt/rb/bin/rb_set_aws_interface.sh",
1024                     "\n"
1025                 ] ] } }
1026             }
```

```
1027         },
1028
1029         "customGroup" : {
1030             /*Grupo de autoescalado con el rol custom*/
1031             "Type" : "AWS::AutoScaling::AutoScalingGroup",
1032             "Condition" : "customRoleCreation",
1033             "DependsOn" : "masterGroup",
1034             "Properties" : {
1035                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1036                 "LaunchConfigurationName" : { "Ref" : "customLaunch" },
1037                 "LoadBalancerNames" : [ { "Ref" : "WebElasticLoadBalancer" } ],
1038                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "custom" ] },
1039                 "MaxSize" : "10",
1040                 "Tags" : [ {
1041                     "Key" : "Role",
1042                     "Value" : "custom",
1043                     "PropagateAtLaunch" : "true"
1044                 },{
1045                     "Key" : "Cluster",
1046                     "Value" : { "Ref" : "AWS::StackName" },
1047                     "PropagateAtLaunch" : "true"
1048                 },{
1049                     "Key" : "Name",
1050                     "Value" : { "Fn::Join" : [ "", [
1051                         { "Ref" : "AWS::StackName" }, " - custom" ] ] },
1052                     "PropagateAtLaunch" : "true"
1053                 } ]
1054             },
1055             "CreationPolicy" : {
1056                 "ResourceSignal" : {
1057                     "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "custom" ] },
1058                     "Timeout" : "PT200M"
1059                 }
1060             },
1061             "UpdatePolicy": {
1062                 "AutoScalingRollingUpdate": {
1063                     "MinInstancesInService": "1",
1064                     "MaxBatchSize": "1",
1065                     "PauseTime" : "PT40M",
1066                     "WaitOnResourceSignals": "true"
1067                 }
1068             }
1069         },
1070         "customLaunch" : {
1071             /*Configuración de arranque para el rol custom*/
1072             "Type" : "AWS::AutoScaling::LaunchConfiguration",
1073             "Condition" : "customRoleCreation",
1074             "Properties" : {
1075                 "AssociatePublicIpAddress" : "true",
1076                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1077                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1078                 "KeyName"        : { "Ref" : "KeyName" },
1079                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "custom", { "Ref"
: "InstanceSize" } ] },
1080                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1081                 "InstanceMonitoring" : "true",
1082                 "BlockDeviceMappings" : [ {
1083                     "DeviceName" : "/dev/sda1",
1084                     "Ebs" : {
1085                         "VolumeSize" : "500",
1086                         "VolumeType" : "gp2",
1087                         "DeleteOnTermination" : "true"
1088                     }
1089                 },  {
1090                     "DeviceName" : "/dev/sdb",
1091                         "VirtualName" : "ephemeral0"
1092                 },  {
1093                     "DeviceName" : "/dev/sdc",
1094                         "VirtualName" : "ephemeral1"
1095                 } ],
1096                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1097                     "#!/bin/bash\n",
1098                     "NODEROLE=custom\n",
1099                     "#NODESERVICES=\n",
1100                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1101                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
```

```
1102                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1103                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1104                    "VPCID=", { "Ref" : "VPC" }, "\n",
1105                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1106                    "AUTOSCALINGGROUPNAME=customGroup\n",
1107                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1108                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1109                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1110                    "S3TYPE=aws\n",
1111                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1112                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1113                    "\n\n",
1114                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1115                    "{\n",
1116                    "  \"redBorder\" : {\n",
1117                    "    \"aws\" : {\n",
1118                    "      \"cloudwatch\" : \"true\"\n",
1119                    "    }\n",
1120                    "  }\n",
1121                    "}\n",
1122                    "_RBEOF2_\n",
1123                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1124                    "\n"
1125                ] ] } }
1126            }
1127        },
1128
1129        "webdruidGroup" : {
1130            /* Grupo de autoescalado para el rol webdruid*/
1131            "Type" : "AWS::AutoScaling::AutoScalingGroup",
1132            "Condition" : "masterRoleCreation",
1133            "Properties" : {
1134                "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1135                "LaunchConfigurationName" : { "Ref" : "webdruidLaunch" },
1136                "LoadBalancerNames" : [ { "Ref" : "WebElasticLoadBalancer" } ],
1137                "MinSize" : "0",
1138                "MaxSize" : "10",
1139                "Tags" : [ {
1140                    "Key" : "Role",
1141                    "Value" : "webdruid",
1142                    "PropagateAtLaunch" : "true"
1143                },{
1144                    "Key" : "Cluster",
1145                    "Value" : { "Ref" : "AWS::StackName" },
1146                    "PropagateAtLaunch" : "true"
1147                },{
1148                    "Key" : "Name",
1149                    "Value" : { "Fn::Join" : [ "", [
1150                        { "Ref" : "AWS::StackName" }, " - webdruid" ] ] },
1151                    "PropagateAtLaunch" : "true"
1152                } ]
1153            }
1154        },
1155
1156        "webdruidLaunch" : {
1157            /* Configuración de arranque para el rol webdruid */
1158            "Type" : "AWS::AutoScaling::LaunchConfiguration",
1159            "Condition" : "masterRoleCreation",
1160            "Properties" : {
1161                "AssociatePublicIpAddress" : "true",
1162                "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1163                "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1164                "KeyName"        : { "Ref" : "KeyName" },
1165                "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "custom", { "Ref"
: "InstanceSize" } ] },
1166                "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1167                "InstanceMonitoring" : "true",
1168                "BlockDeviceMappings" : [ {
1169                    "DeviceName" : "/dev/sda1",
1170                    "Ebs" : {
1171                        "VolumeSize" : "500",
1172                        "VolumeType" : "gp2",
1173                        "DeleteOnTermination" : "true"
1174                    }
1175                },  {
1176                    "DeviceName" : "/dev/sdb",
1177                        "VirtualName" : "ephemeral0"
1178                },  {
```

```
1179                    "DeviceName" : "/dev/sdc",
1180                        "VirtualName" : "ephemeral1"
1181                } ],
1182                "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1183                    "#!/bin/bash\n",
1184                    "NODEROLE=webdruid\n",
1185                    "#NODESERVICES=\n",
1186                    "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1187                    "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1188                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1189                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1190                    "VPCID=", { "Ref" : "VPC" }, "\n",
1191                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1192                    "AUTOSCALINGGROUPNAME=webdruidGroup\n",
1193                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1194                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1195                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1196                    "S3TYPE=aws\n",
1197                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1198                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1199                    "\n\n",
1200                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1201                    "{\n",
1202                    "  \"redBorder\" : {\n",
1203                    "    \"aws\" : {\n",
1204                    "      \"autoscaled\" : \"true\",\n",
1205                    "      \"cloudwatch\" : \"true\"\n",
1206                    "    }\n",
1207                    "  }\n",
1208                    "}\n",
1209                    "_RBEOF2_\n",
1210                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1211                    "\n"
1212                ] ] } }
1213            }
1214        },
1215        /* Dado que webdruid es uno de los roles que pueden autoescalar, se habilitan los
1216        Lifecycle Hooks. Cuando se vaya a terminar una instancia, se envía a SQS un mensaje
1217        de notificación */
1218        "webdruidLifecycleHook" : {
1219            "Type" : "AWS::AutoScaling::LifecycleHook",
1220            "Condition" : "masterRoleCreation",
1221            "Properties" : {
1222                "AutoScalingGroupName" : { "Ref" : "webdruidGroup"  },
1223                "LifecycleTransition" : "autoscaling:EC2_INSTANCE_TERMINATING",
1224                "NotificationTargetARN" : { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] },
1225                "RoleARN" : { "Fn::GetAtt" : [ "IAMRoleLifecycleHook", "Arn" ] }
1226            }
1227        },
1228
1229        /* Grupo de autoescalado para el rol corezk */
1230        "corezkGroup" : {
1231            "Type" : "AWS::AutoScaling::AutoScalingGroup",
1232            "Condition" : "corezkRoleCreation",
1233            "DependsOn" : "AttachGateway",
1234            "Properties" : {
1235                "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1236                "LaunchConfigurationName" : { "Ref" : "corezkLaunch" },
1237                "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "corezk" ] },
1238                "MaxSize" : "10",
1239                "Tags" : [ {
1240                    "Key" : "Role",
1241                    "Value" : "corezk",
1242                    "PropagateAtLaunch" : "true"
1243                },{
1244                    "Key" : "Cluster",
1245                    "Value" : { "Ref" : "AWS::StackName" },
1246                    "PropagateAtLaunch" : "true"
1247                },{
1248                    "Key" : "Name",
1249                    "Value" : { "Fn::Join" : [ "", [
1250                        { "Ref" : "AWS::StackName" }, " - corezk" ] ] },
1251                    "PropagateAtLaunch" : "true"
1252                } ]
1253            },
1254            "CreationPolicy" : {
1255                "ResourceSignal" : {
```

```
1256                    "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "corezk" ] },
1257                    "Timeout" : "PT200M"
1258                }
1259            },
1260            "UpdatePolicy": {
1261                "AutoScalingRollingUpdate": {
1262                    "MinInstancesInService": "1",
1263                    "MaxBatchSize": "1",
1264                    "PauseTime" : "PT40M",
1265                    "WaitOnResourceSignals": "true"
1266                }
1267            }
1268        },
1269
1270        "corezkLaunch" : {
1271        /* Configuración de arranque para el rol corezk*/
1272            "Type" : "AWS::AutoScaling::LaunchConfiguration",
1273            "Condition" : "corezkRoleCreation",
1274            "Properties" : {
1275                "AssociatePublicIpAddress" : "true",
1276                "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1277                "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1278                "KeyName"        : { "Ref" : "KeyName" },
1279                "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "corezk", { "Ref"
: "InstanceSize" } ] },
1280                "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1281                "InstanceMonitoring" : "true",
1282                "BlockDeviceMappings" : [ {
1283                    "DeviceName" : "/dev/sda1",
1284                    "Ebs" : {
1285                        "VolumeSize" : "50",
1286                        "VolumeType" : "gp2",
1287                        "DeleteOnTermination" : "true"
1288                    }
1289                },  {
1290                    "DeviceName" : "/dev/sdb",
1291                        "VirtualName" : "ephemeral0"
1292                },  {
1293                    "DeviceName" : "/dev/sdc",
1294                        "VirtualName" : "ephemeral1"
1295                } ],
1296                "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1297                    "#!/bin/bash\n",
1298                    "NODEROLE=corezk\n",
1299                    "NODESERVICES=\"rb-cloudwatch:1\"\n",
1300                    "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1301                    "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1302                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1303                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1304                    "VPCID=", { "Ref" : "VPC" }, "\n",
1305                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1306                    "AUTOSCALINGGROUPNAME=corezkGroup\n",
1307                    "LOGWATCHG=", { "Ref" : "AWS::StackName" }, "\n",
1308                    "SQSQUEUEURL=", { "Ref" : "SQSQueue" }, "\n",
1309                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1310                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1311                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1312                    "S3TYPE=aws\n",
1313                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1314                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1315                    "CHEF_AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKeyChef" }, "\"\n",
1316                    "CHEF_AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKeyChef",
"SecretAccessKey" ] }, "\"\n",
1317                    { "Fn::If" : [ "ElasticacheCreation", { "Fn::Join" : [ "", [
1318                        "ELASTICCACHECLUSTERID=", { "Ref" : "Memcached" }, "\n",
1319                        "ELASTICCACHEENDPOINT=", { "Fn::Join" : [ "", [ { "Fn::GetAtt" : [
"Memcached", "ConfigurationEndpoint.Address" ] },
1320                        ":", { "Fn::GetAtt" : [ "Memcached", "ConfigurationEndpoint.Port" ]
} ] ] }, "\n"
1321                        ] ] },
1322                    "" ] },
1323                    { "Fn::If" : [ "RDSCreation", { "Fn::Join" : [ "", [
1324                        "SQLUSER=rdsredborder\n",
1325                        "SQLPASSWORD=", { "Ref" : "DBPassword" }, "\n",
1326                        "SQLDB=redborderRDS\n",
1327                        "SQLHOST=", { "Fn::Join" : [ "", [ { "Fn::GetAtt" : [ "DBRDS",
"Endpoint.Address" ] }, ":",
```

```
1328                            { "Fn::GetAtt" : [ "DBRDS", "Endpoint.Port" ] } ] ] }, "\n"
1329                        ] ] },
1330                    "" ] },
1331                    "ENRICHMODE=samza\n",
1332                    "MMDOWNALARMNAME=\"", { "Ref" : "middlemanagerLowCapacityAlarm" },
"\"\n",
1333                    "\n\n",
1334                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1335                    "{\n",
1336                    "  \"redBorder\" : {\n",
1337                    "    \"aws\" : {\n",
1338                    "      \"cloudwatch\" : \"true\"\n",
1339                    "    }\n",
1340                    "  }\n",
1341                    "}\n",
1342                    "_RBEOF2_\n",
1343                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1344                    "\n"
1345                ] ] } }
1346            }
1347        },
1348
1349        "kafkaGroup" : {
1350        /*Grupo de autoescalado para el rol kafka*/
1351            "Type" : "AWS::AutoScaling::AutoScalingGroup",
1352            "Condition" : "kafkaRoleCreation",
1353            "DependsOn" : "corezkGroup",
1354            "Properties" : {
1355                "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1356                "LaunchConfigurationName" : { "Ref" : "kafkaLaunch" },
1357                "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "kafka" ] },
1358                "MaxSize" : "10",
1359                "Tags" : [ {
1360                    "Key" : "Role",
1361                    "Value" : "kafka",
1362                    "PropagateAtLaunch" : "true"
1363                },{
1364                    "Key" : "Cluster",
1365                    "Value" : { "Ref" : "AWS::StackName" },
1366                    "PropagateAtLaunch" : "true"
1367                },{
1368                    "Key" : "Name",
1369                    "Value" : { "Fn::Join" : [ "", [
1370                        { "Ref" : "AWS::StackName" }, " - kafka" ] ] },
1371                    "PropagateAtLaunch" : "true"
1372                } ]
1373            },
1374            "CreationPolicy" : {
1375                "ResourceSignal" : {
1376                    "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "kafka" ] },
1377                    "Timeout" : "PT200M"
1378                }
1379            },
1380            "UpdatePolicy": {
1381                "AutoScalingRollingUpdate": {
1382                    "MinInstancesInService": "1",
1383                    "MaxBatchSize": "1",
1384                    "PauseTime" : "PT40M",
1385                    "WaitOnResourceSignals": "true"
1386                }
1387            }
1388        },
1389
1390        "kafkaLaunch" : {
1391        /*Configuración de arranque para el rol kafka*/
1392            "Type" : "AWS::AutoScaling::LaunchConfiguration",
1393            "Condition" : "kafkaRoleCreation",
1394            "Properties" : {
1395                "AssociatePublicIpAddress" : "true",
1396                "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1397                "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1398                "KeyName"        : { "Ref" : "KeyName" },
1399                "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "kafka", { "Ref" :
"InstanceSize" } ] },
1400                "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1401                "InstanceMonitoring" : "true",
```

```
1402                "BlockDeviceMappings" : [ {
1403                    "DeviceName" : "/dev/sda1",
1404                    "Ebs" : {
1405                        "VolumeSize" : "50",
1406                        "VolumeType" : "gp2",
1407                        "DeleteOnTermination" : "true"
1408                    }
1409                },  {
1410                    "DeviceName" : "/dev/sdb",
1411                        "VirtualName" : "ephemeral0"
1412                },  {
1413                    "DeviceName" : "/dev/sdc",
1414                        "VirtualName" : "ephemeral1"
1415                } ],
1416                "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1417                    "#!/bin/bash\n",
1418                    "NODEROLE=kafka\n",
1419                    "#NODESERVICES=\n",
1420                    "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1421                    "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1422                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1423                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1424                    "VPCID=", { "Ref" : "VPC" }, "\n",
1425                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1426                    "AUTOSCALINGGROUPNAME=kafkaGroup\n",
1427                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1428                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1429                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1430                    "S3TYPE=aws\n",
1431                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1432                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1433                    "\n\n",
1434                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1435                    "{\n",
1436                    "  \"redBorder\" : {\n",
1437                    "    \"aws\" : {\n",
1438                    "      \"cloudwatch\" : \"true\"\n",
1439                    "    }\n",
1440                    "  }\n",
1441                    "}\n",
1442                    "_RBEOF2_\n",
1443                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1444                    "\n"
1445                ] ] } }
1446            }
1447        },
1448
1449        "http2kGroup" : {
1450            /*Grupo de autoescalado para el rol http2k */
1451            "Type" : "AWS::AutoScaling::AutoScalingGroup",
1452            "DependsOn" : "kafkaGroup",
1453            "Condition" : "http2kRoleCreation",
1454            "Properties" : {
1455                "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1456                "LoadBalancerNames" : [ { "Ref" : "DataElasticLoadBalancer" } ],
1457                "LaunchConfigurationName" : { "Ref" : "http2kLaunch" },
1458                "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "http2k" ] },
1459                "MaxSize" : "10",
1460                "Tags" : [ {
1461                    "Key" : "Role",
1462                    "Value" : "http2k",
1463                    "PropagateAtLaunch" : "true"
1464                },{
1465                    "Key" : "Cluster",
1466                    "Value" : { "Ref" : "AWS::StackName" },
1467                    "PropagateAtLaunch" : "true"
1468                },{
1469                    "Key" : "Name",
1470                    "Value" : { "Fn::Join" : [ "", [
1471                        { "Ref" : "AWS::StackName" }, " - http2k" ] ] },
1472                    "PropagateAtLaunch" : "true"
1473                } ]
1474            },
1475            "CreationPolicy" : {
1476                "ResourceSignal" : {
1477                    "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "http2k" ] },
1478                    "Timeout" : "PT200M"
```

```
1479                 }
1480             },
1481             "UpdatePolicy": {
1482                 "AutoScalingRollingUpdate": {
1483                     "MinInstancesInService": "1",
1484                     "MaxBatchSize": "1",
1485                     "PauseTime" : "PT40M",
1486                     "WaitOnResourceSignals": "true"
1487                 }
1488             }
1489         },
1490
1491         "http2kLaunch" : {
1492         /* Configuración de arranque para el rol http2k */
1493             "Type" : "AWS::AutoScaling::LaunchConfiguration",
1494             "Condition" : "http2kRoleCreation",
1495             "Properties" : {
1496                 "AssociatePublicIpAddress" : "true",
1497                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1498                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1499                 "KeyName"        : { "Ref" : "KeyName" },
1500                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "http2k", { "Ref"
: "InstanceSize" } ] } },
1501                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1502                 "InstanceMonitoring" : "true",
1503                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1504                     "#!/bin/bash\n",
1505                     "NODEROLE=http2k\n",
1506                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1507                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1508                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1509                     "REGION=", { "Ref" : "AWS::Region" }, "\n",
1510                     "VPCID=", { "Ref" : "VPC" }, "\n",
1511                     "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1512                     "AUTOSCALINGGROUPNAME=http2kGroup\n",
1513                     "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1514                     "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1515                     "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1516                     "S3TYPE=aws\n",
1517                     "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1518                     "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1519                     "\n\n",
1520                     "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1521                     "{\n",
1522                     "  \"redBorder\" : {\n",
1523                     "    \"aws\" : {\n",
1524                     "      \"cloudwatch\" : \"true\",\n",
1525                     "      \"autoscaled\" : \"true\"\n",
1526                     "    }\n",
1527                     "  }\n",
1528                     "}\n",
1529                     "_RBEOF2_\n",
1530                     ". /opt/rb/bin/rb_set_aws_interface.sh",
1531                     "\n"
1532                 ] ] } }
1533             }
1534         },
1535         /* Dado que http2k es uno de los roles que pueden autoescalar, se habilitan los
1536         Lifecycle Hooks. Cuando se vaya a terminar una instancia, se envía a SQS un mensaje
1537         de notificación */
1538         "http2kLifecycleHook" : {
1539             "Type" : "AWS::AutoScaling::LifecycleHook",
1540             "Condition" : "http2kRoleCreation",
1541             "Properties" : {
1542                 "AutoScalingGroupName" : { "Ref" : "http2kGroup"  },
1543                 "LifecycleTransition" : "autoscaling:EC2_INSTANCE_TERMINATING",
1544                 "NotificationTargetARN" : { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] },
1545                 "RoleARN" : { "Fn::GetAtt" : [ "IAMRoleLifecycleHook", "Arn" ] }
1546             }
1547         },
1548
1549
1550
1551         "http2kHighCPUAlarm" : {
1552         /* Alarma de Cloudwatch que indica cuándo se ha superado el 65% del uso de la CPU
1553            en las instancias http2k. */
1554             "Type" : "AWS::CloudWatch::Alarm",
```

```
1555             "Condition" : "http2kRoleCreation",
1556             "Properties" : {
1557                 "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
http2kHighCPUAlarm" ] ] },
1558                 "AlarmDescription" : "Alarm for web autoscaling. When CPU is so high,
triggers a scaling out",
1559                 "Namespace" : "AWS/EC2", //Namespace de la métrica, en este caso es proveída
por AWS
1560                 "MetricName" : "CPUUtilization", //Nombre de la métrica
1561                 "Dimensions" : [ {
1562                     "Name" : "AutoScalingGroupName", //Dimensión que identifica a la métrica
1563                     "Value" : { "Ref" : "http2kGroup" }
1564                 } ],
1565                 "EvaluationPeriods" : "1",  //Condiciones para que salte la métrica: más del
65% de CPU
1566                 "Period" : "300", //durante 5 minutos
1567                 "ComparisonOperator" : "GreaterThanThreshold",
1568                 "Threshold" : "65",
1569                 "Statistic" : "Average",
1570                 "ActionsEnabled" : "true",
1571                 "AlarmActions" : [ { "Ref" : "http2kScaleUpPolicy" } ] //Cuando se produce
la alarma se
1572             }                                                          //entra en acción la
política de
1573                                                                        //autoescalado
1574         },
1575         /* Política para escalar cuando salta la alarma, asignando 900 segundo de bloqueo
después del
1576         autoescalado */
1577         "http2kScaleUpPolicy" : {
1578             "Type" : "AWS::AutoScaling::ScalingPolicy",
1579             "Properties" : {
1580                 "AdjustmentType" : "ChangeInCapacity",
1581                 "AutoScalingGroupName" : { "Ref" : "http2kGroup" },
1582                 "Cooldown" : "900",
1583                 "ScalingAdjustment" : "1"
1584             }
1585         },
1586         "http2kLowCPUAlarm" : {
1587             "Type" : "AWS::CloudWatch::Alarm",
1588             "Condition" : "http2kRoleCreation",
1589             "Properties" : {
1590                 "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
http2kLowCPUAlarm" ] ] },
1591                 "AlarmDescription" : "Alarm for web autoscaling. When CPU is so low,
triggers a scaling in",
1592                 "Namespace" : "AWS/EC2",
1593                 "MetricName" : "CPUUtilization",
1594                 "Dimensions" : [ {
1595                     "Name" : "AutoScalingGroupName",
1596                     "Value" : { "Ref" : "http2kGroup" }
1597                 } ],
1598                 "EvaluationPeriods" : "3",
1599                 "Period" : "300",
1600                 "ComparisonOperator" : "LessThanThreshold",
1601                 "Threshold" : "40",
1602                 "Statistic" : "Average",
1603                 "ActionsEnabled" : "true",
1604                 "AlarmActions" : [ { "Ref" : "http2kScaleDownPolicy" } ]
1605             }
1606         },
1607         /* Política para escalar cuando salta la alarma, asignando 900 segundo de bloqueo
después del
1608         autoescalado */
1609         "http2kScaleDownPolicy" : {
1610             "Type" : "AWS::AutoScaling::ScalingPolicy",
1611             "Properties" : {
1612                 "AdjustmentType" : "ChangeInCapacity",
1613                 "AutoScalingGroupName" : { "Ref" : "http2kGroup" },
1614                 "Cooldown" : "300",
1615                 "ScalingAdjustment" : "-1"
1616             }
1617         },
1618
1619         "webGroup" : {
1620             /*Grupo de autoescalado para el rol web */
1621             "Type" : "AWS::AutoScaling::AutoScalingGroup",
1622             "Condition" : "webRoleCreation",
1623             "DependsOn" : "kafkaGroup",
1624             "Properties" : {
```

```
1625                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1626                 "LaunchConfigurationName" : { "Ref" : "webLaunch" },
1627                 "LoadBalancerNames" : [ { "Ref" : "WebElasticLoadBalancer" } ],
1628                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "web" ] },
1629                 "MaxSize" : "10",
1630                 "Tags" : [ {
1631                     "Key" : "Role",
1632                     "Value" : "web",
1633                     "PropagateAtLaunch" : "true"
1634                 },{
1635                     "Key" : "Cluster",
1636                     "Value" : { "Ref" : "AWS::StackName" },
1637                     "PropagateAtLaunch" : "true"
1638                 },{
1639                     "Key" : "Name",
1640                     "Value" : { "Fn::Join" : [ "", [
1641                         { "Ref" : "AWS::StackName" }, " - web" ] ] },
1642                     "PropagateAtLaunch" : "true"
1643                 } ]
1644             },
1645             "CreationPolicy" : {
1646                 "ResourceSignal" : {
1647                     "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "web" ] },
1648                     "Timeout" : "PT200M"
1649                 }
1650             },
1651             "UpdatePolicy": {
1652                 "AutoScalingRollingUpdate": {
1653                     "MinInstancesInService": "1",
1654                     "MaxBatchSize": "1",
1655                     "PauseTime" : "PT40M",
1656                     "WaitOnResourceSignals": "true"
1657                 }
1658             }
1659         },
1660         "webLaunch" : {
1661             /*Configuración de arranque para el rol web*/
1662             "Type" : "AWS::AutoScaling::LaunchConfiguration",
1663             "Condition" : "webRoleCreation",
1664             "Properties" : {
1665                 "AssociatePublicIpAddress" : "true",
1666                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1667                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1668                 "KeyName"        : { "Ref" : "KeyName" },
1669                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "web", { "Ref" :
"InstanceSize" } ] },
1670                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1671                 "InstanceMonitoring" : "true",
1672                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1673                     "#!/bin/bash\n",
1674                     "NODEROLE=web\n",
1675                     "#NODESERVICES=\n",
1676                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1677                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1678                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1679                     "REGION=", { "Ref" : "AWS::Region" }, "\n",
1680                     "VPCID=", { "Ref" : "VPC" }, "\n",
1681                     "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1682                     "AUTOSCALINGGROUPNAME=webGroup\n",
1683                     "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1684                     "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1685                     "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1686                     "S3TYPE=aws\n",
1687                     "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1688                     "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1689                     "\n\n",
1690                     "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1691                     "{\n",
1692                     "  \"redBorder\" : {\n",
1693                     "    \"aws\" : {\n",
1694                     "      \"autoscaled\" : \"true\",\n",
1695                     "      \"cloudwatch\" : \"true\"\n",
1696                     "    }\n",
1697                     "  }\n",
1698                     "}\n",
```

```
1699                    "_RBEOF2_\n",
1700                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1701                    "\n"
1702                ] ] } }
1703            }
1704        },
1705        /* lifecycle hook para nodos web similar al de http2k */
1706        "webLifecycleHook" : {
1707            "Type" : "AWS::AutoScaling::LifecycleHook",
1708            "Condition" : "webRoleCreation",
1709            "Properties" : {
1710                "AutoScalingGroupName" : { "Ref" : "webGroup"  },
1711                "LifecycleTransition" : "autoscaling:EC2_INSTANCE_TERMINATING",
1712                "NotificationTargetARN" : { "Fn::GetAtt" : [ "SQSQueue", "Arn" ] },
1713                "RoleARN" : { "Fn::GetAtt" : [ "IAMRoleLifecycleHook", "Arn" ] }
1714            }
1715        }, /* Alarma similar a la de http2k */
1716        "webHighCPUAlarm" : {
1717            "Type" : "AWS::CloudWatch::Alarm",
1718            "Condition" : "webRoleCreation",
1719            "Properties" : {
1720                "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
webHighCPUAlarm" ] ] },
1721                "AlarmDescription" : "Alarm for web autoscaling. When CPU is so high,
triggers a scaling out",
1722                "Namespace" : "AWS/EC2",
1723                "MetricName" : "CPUUtilization",
1724                "Dimensions" : [ {
1725                    "Name" : "AutoScalingGroupName",
1726                    "Value" : { "Ref" : "webGroup" }
1727                } ],
1728                "EvaluationPeriods" : "1",
1729                "Period" : "300",
1730                "ComparisonOperator" : "GreaterThanThreshold",
1731                "Threshold" : "65",
1732                "Statistic" : "Average",
1733                "ActionsEnabled" : "true",
1734                "AlarmActions" : [ { "Ref" : "webScaleUpPolicy" } ]
1735            }
1736        }, /* política de autoescalado similar a la de http2k*/
1737        "webScaleUpPolicy" : {
1738            "Type" : "AWS::AutoScaling::ScalingPolicy",
1739            "Properties" : {
1740                "AdjustmentType" : "ChangeInCapacity",
1741                "AutoScalingGroupName" : { "Ref" : "webGroup" },
1742                "Cooldown" : "900",
1743                "ScalingAdjustment" : "1"
1744            }
1745        },/* alarma similar a las de http2k */
1746        "webLowCPUAlarm" : {
1747            "Type" : "AWS::CloudWatch::Alarm",
1748            "Condition" : "webRoleCreation",
1749            "Properties" : {
1750                "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
webLowCPUAlarm" ] ] },
1751                "AlarmDescription" : "Alarm for web autoscaling. When CPU is so low,
triggers a scaling in",
1752                "Namespace" : "AWS/EC2",
1753                "MetricName" : "CPUUtilization",
1754                "Dimensions" : [ {
1755                    "Name" : "AutoScalingGroupName",
1756                    "Value" : { "Ref" : "webGroup" }
1757                } ],
1758                "EvaluationPeriods" : "3",
1759                "Period" : "300",
1760                "ComparisonOperator" : "LessThanThreshold",
1761                "Threshold" : "40",
1762                "Statistic" : "Average",
1763                "ActionsEnabled" : "true",
1764                "AlarmActions" : [ { "Ref" : "webScaleDownPolicy" } ]
1765            }
1766        }, /* política de autoescalado similar a la de http2k */
1767        "webScaleDownPolicy" : {
1768            "Type" : "AWS::AutoScaling::ScalingPolicy",
1769            "Properties" : {
1770                "AdjustmentType" : "ChangeInCapacity",
1771                "AutoScalingGroupName" : { "Ref" : "webGroup" },
1772                "Cooldown" : "300",
1773                "ScalingAdjustment" : "-1"
1774            }
```

```
1775         },
1776
1777         /* Grupo de autoescalado para el rol broker */
1778         "brokerGroup" : {
1779             "Type" : "AWS::AutoScaling::AutoScalingGroup",
1780             "Condition" : "brokerRoleCreation",
1781             "DependsOn" : "kafkaGroup",
1782             "Properties" : {
1783                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1784                 "LaunchConfigurationName" : { "Ref" : "brokerLaunch" },
1785                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "broker" ] },
1786                 "MaxSize" : "10",
1787                 "Tags" : [ {
1788                     "Key" : "Role",
1789                     "Value" : "broker",
1790                     "PropagateAtLaunch" : "true"
1791                 },{
1792                     "Key" : "Cluster",
1793                     "Value" : { "Ref" : "AWS::StackName" },
1794                     "PropagateAtLaunch" : "true"
1795                 },{
1796                     "Key" : "Name",
1797                     "Value" : { "Fn::Join" : [ "", [
1798                         { "Ref" : "AWS::StackName" }, " - broker" ] ] },
1799                     "PropagateAtLaunch" : "true"
1800                 } ]
1801             },
1802             "CreationPolicy" : {
1803                 "ResourceSignal" : {
1804                     "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "broker" ] },
1805                     "Timeout" : "PT200M"
1806                 }
1807             },
1808             "UpdatePolicy": {
1809                 "AutoScalingRollingUpdate": {
1810                     "MinInstancesInService": "1",
1811                     "MaxBatchSize": "1",
1812                     "PauseTime" : "PT40M",
1813                     "WaitOnResourceSignals": "true"
1814                 }
1815             }
1816         },
1817         /* Configuración de arranque para el rol broker */
1818         "brokerLaunch" : {
1819             "Type" : "AWS::AutoScaling::LaunchConfiguration",
1820             "Condition" : "brokerRoleCreation",
1821             "Properties" : {
1822                 "AssociatePublicIpAddress" : "true",
1823                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1824                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1825                 "KeyName"        : { "Ref" : "KeyName" },
1826                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "broker", { "Ref"
: "InstanceSize" } ] },
1827                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1828                 "InstanceMonitoring" : "true",
1829                 "BlockDeviceMappings" : [ {
1830                     "DeviceName" : "/dev/sda1",
1831                     "Ebs" : {
1832                         "VolumeSize" : "50",
1833                         "VolumeType" : "gp2",
1834                         "DeleteOnTermination" : "true"
1835                     }
1836                 },  {
1837                     "DeviceName" : "/dev/sdb",
1838                         "VirtualName" : "ephemeral0"
1839                 },  {
1840                     "DeviceName" : "/dev/sdc",
1841                         "VirtualName" : "ephemeral1"
1842                 } ],
1843                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1844                     "#!/bin/bash\n",
1845                     "NODEROLE=broker\n",
1846                     "#NODESERVICES=\n",
1847                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1848                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1849                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
```

```
1850                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1851                    "VPCID=", { "Ref" : "VPC" }, "\n",
1852                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1853                    "AUTOSCALINGGROUPNAME=brokerGroup\n",
1854                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1855                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1856                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1857                    "S3TYPE=aws\n",
1858                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1859                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1860                    "\n\n",
1861                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1862                    "{\n",
1863                    "  \"redBorder\" : {\n",
1864                    "    \"aws\" : {\n",
1865                    "      \"autoscaled\" : \"true\",\n",
1866                    "      \"cloudwatch\" : \"true\"\n",
1867                    "    }\n",
1868                    "  }\n",
1869                    "}\n",
1870                    "_RBEOF2_\n",
1871                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1872                    "\n"
1873                ] ] } }
1874            }
1875        },
1876        /* Grupo de autoescalado para el rol Middlemanager */
1877        "middlemanagerGroup" : {
1878            "Type" : "AWS::AutoScaling::AutoScalingGroup",
1879            "DependsOn" : "kafkaGroup",
1880            "Condition" : "middlemanagerRoleCreation",
1881            "Properties" : {
1882                "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
1883                "LaunchConfigurationName" : { "Ref" : "middlemanagerLaunch" },
1884                "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "middlemanager" ] },
1885                "MaxSize" : "10",
1886                "Tags" : [ {
1887                    "Key" : "Role",
1888                    "Value" : "middlemanager",
1889                    "PropagateAtLaunch" : "true"
1890                },{
1891                    "Key" : "Cluster",
1892                    "Value" : { "Ref" : "AWS::StackName" },
1893                    "PropagateAtLaunch" : "true"
1894                },{
1895                    "Key" : "Name",
1896                    "Value" : { "Fn::Join" : [ "", [
1897                        { "Ref" : "AWS::StackName" }, " - middlemanager" ] ] },
1898                    "PropagateAtLaunch" : "true"
1899                } ]
1900            },
1901            "CreationPolicy" : {
1902                "ResourceSignal" : {
1903                    "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "middlemanager" ] },
1904                    "Timeout" : "PT200M"
1905                }
1906            },
1907            "UpdatePolicy": {
1908                "AutoScalingRollingUpdate": {
1909                    "MinInstancesInService": "1",
1910                    "MaxBatchSize": "1",
1911                    "PauseTime" : "PT40M",
1912                    "WaitOnResourceSignals": "true"
1913                }
1914            }
1915        },
1916        /* Configuración de arranque para el rol middlemanager*/
1917        "middlemanagerLaunch" : {
1918            "Type" : "AWS::AutoScaling::LaunchConfiguration",
1919            "Condition" : "middlemanagerRoleCreation",
1920            "Properties" : {
1921                "AssociatePublicIpAddress" : "true",
1922                "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
1923                "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
1924                "KeyName"        : { "Ref" : "KeyName" },
```

```
1925                "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "middlemanager", {
"Ref" : "InstanceSize" } ] },
1926                "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
1927                "InstanceMonitoring" : "true",
1928                "BlockDeviceMappings" : [ {
1929                    "DeviceName" : "/dev/sda1",
1930                    "Ebs" : {
1931                        "VolumeSize" : "50",
1932                        "VolumeType" : "gp2",
1933                        "DeleteOnTermination" : "true"
1934                    }
1935                },  {
1936                    "DeviceName" : "/dev/sdb",
1937                        "VirtualName" : "ephemeral0"
1938                },  {
1939                    "DeviceName" : "/dev/sdc",
1940                        "VirtualName" : "ephemeral1"
1941                } ],
1942                "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
1943                    "#!/bin/bash\n",
1944                    "NODEROLE=middleManager\n",
1945                    "#NODESERVICES=\n",
1946                    "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
1947                    "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
1948                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
1949                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
1950                    "VPCID=", { "Ref" : "VPC" }, "\n",
1951                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
1952                    "AUTOSCALINGGROUPNAME=middlemanagerGroup\n",
1953                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
1954                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
1955                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
1956                    "S3TYPE=aws\n",
1957                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
1958                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
1959                    "\n\n",
1960                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
1961                    "{\n",
1962                    "  \"redBorder\" : {\n",
1963                    "    \"aws\" : {\n",
1964                    "      \"cloudwatch\" : \"true\",\n",
1965                    "      \"autoscaled\" : \"true\"\n",
1966                    "    }\n",
1967                    "  }\n",
1968                    "}\n",
1969                    "_RBEOF2_\n",
1970                    ". /opt/rb/bin/rb_set_aws_interface.sh",
1971                    "\n"
1972                ] ] } }
1973            }
1974        },
1975        /*Alarma para el autoescalado del MiddleManager */
1976        "middlemanagerHighCapacityAlarm" : {
1977            "Type" : "AWS::CloudWatch::Alarm",
1978            "Condition" : "middlemanagerRoleCreation",
1979            "Properties" : {
1980                "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
middlemanagerHighCapacityAlarm" ] ] },
1981                "AlarmDescription" : "Alarm for middleManager autoscaling. When capacity is
so high, triggers a scaling out",
1982                //En este caso el namespace es personalizado de redBorder, ya que la métrica
no es proveída por AWS,
1983                //sino a través del servicio rb_cloudwatch
1984                "Namespace" : { "Fn::Join" : [ "", [ "rB/", { "Ref" : "AWS::StackName" } ] ]
},
1985                "MetricName" : "desired_capacity",
1986                "Dimensions" : [ {
1987                    "Name" : "InstanceId",
1988                    "Value" : "overlord"
1989                } ],
1990                "EvaluationPeriods" : "1",
1991                "Period" : "60",
1992                "ComparisonOperator" : "GreaterThanOrEqualToThreshold",
1993                "Threshold" : "60",
1994                "Statistic" : "Average",
1995                "ActionsEnabled" : "true",
1996                "AlarmActions" : [ { "Ref" : "middleManagerScaleUpPolicy" } ]
1997            }
```

```
1998         },
1999         /* Alarma para autoescalar en caso de tareas pendientes */
2000         "middlemanagerPendingTasksAlarm" : {
2001             "Type" : "AWS::CloudWatch::Alarm",
2002             "Condition" : "middlemanagerRoleCreation",
2003             "Properties" : {
2004                 "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
middlemanagerPendingTasksAlarm" ] ] },
2005                 "AlarmDescription" : "Alarm for middleManager autoscaling. When capacity is
so high, triggers a scaling out",
2006                 "Namespace" : { "Fn::Join" : [ "", [ "rB/", { "Ref" : "AWS::StackName" } ] ]
},
2007                 "MetricName" : "pending_tasks",
2008                 "Dimensions" : [ {
2009                     "Name" : "InstanceId",
2010                     "Value" : "overlord"
2011                 } ],
2012                 "EvaluationPeriods" : "1",
2013                 "Period" : "60",
2014                 "ComparisonOperator" : "GreaterThanOrEqualToThreshold",
2015                 "Threshold" : "1",
2016                 "Statistic" : "Average",
2017                 "ActionsEnabled" : "true",
2018                 "AlarmActions" : [ { "Ref" : "middleManagerScaleUpPolicy" } ]
2019             }
2020         },
2021
2022         /*Política para autoescalar el middleManager. Cabe destacar que sólo se ha definido
2023         para añadir instancias, ya que para eliminarlas se utiliza el servicio SQSLD*/
2024         "middleManagerScaleUpPolicy" : {
2025             "Type" : "AWS::AutoScaling::ScalingPolicy",
2026             "Condition" : "middlemanagerRoleCreation",
2027             "Properties" : {
2028                 "AdjustmentType" : "ChangeInCapacity",
2029                 "AutoScalingGroupName" : { "Ref" : "middlemanagerGroup" },
2030                 "Cooldown" : "900",
2031                 "ScalingAdjustment" : "2"
2032             }
2033         },
2034         /* Alarma para la eliminación de instancias de MiddleManager */
2035         "middlemanagerLowCapacityAlarm" : {
2036             "Type" : "AWS::CloudWatch::Alarm",
2037             "Condition" : "middlemanagerRoleCreation",
2038             "Properties" : {
2039                 "AlarmName" : { "Fn::Join" : [ "", [ { "Ref" : "AWS::StackName" }, " -
middlemanagerLowCapacityAlarm" ] ] },
2040                 "AlarmDescription" : "Alarm for middleManager autoscaling. When capacity is
so low, triggers a scaling in",
2041                 "Namespace" : { "Fn::Join" : [ "", [ "rB/", { "Ref" : "AWS::StackName" } ] ]
},
2042                 "MetricName" : "desired_capacity",
2043                 "Dimensions" : [ {
2044                     "Name" : "InstanceId",
2045                     "Value" : "overlord"
2046                 } ],
2047                 "EvaluationPeriods" : "10",
2048                 "Period" : "60",
2049                 "ComparisonOperator" : "LessThanOrEqualToThreshold",
2050                 "Threshold" : "40",
2051                 "Statistic" : "Average",
2052                 "AlarmActions" : [ { "Ref" : "SNSTopic" } ] //La acción de la alarma en este
caso es enviar
2053                 //una notificación al topic de SNS, para que SQSLD la reciba y procese la
terminación
2054                 //de la instancia.
2055             }
2056         },
2057         /* Grupo de autoescalado para el rol historical */
2058         "historicalGroup" : {
2059             "Type" : "AWS::AutoScaling::AutoScalingGroup",
2060             "Condition" : "historicalRoleCreation",
2061             "DependsOn" : "kafkaGroup",
2062             "Properties" : {
2063                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
2064                 "LaunchConfigurationName" : { "Ref" : "historicalLaunch" },
2065                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "historical" ] },
2066                 "MaxSize" : "10",
2067                 "Tags" : [ {
2068                     "Key" : "Role",
```

```
2069                    "Value" : "historical",
2070                    "PropagateAtLaunch" : "true"
2071                },{
2072                    "Key" : "Cluster",
2073                    "Value" : { "Ref" : "AWS::StackName" },
2074                    "PropagateAtLaunch" : "true"
2075                },{
2076                    "Key" : "Name",
2077                    "Value" : { "Fn::Join" : [ "", [
2078                        { "Ref" : "AWS::StackName" }, " - historical" ] ] },
2079                    "PropagateAtLaunch" : "true"
2080                } ]
2081            },
2082            "CreationPolicy" : {
2083                "ResourceSignal" : {
2084                    "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "historical" ] },
2085                    "Timeout" : "PT200M"
2086                }
2087            },
2088            "UpdatePolicy": {
2089                "AutoScalingRollingUpdate": {
2090                    "MinInstancesInService": "1",
2091                    "MaxBatchSize": "1",
2092                    "PauseTime" : "PT40M",
2093                    "WaitOnResourceSignals": "true"
2094                }
2095            }
2096        },
2097        /*Configuración de arranque para el rol historical, configurado en modo Long Term*/
2098        "historicalLaunch" : {
2099            "Type" : "AWS::AutoScaling::LaunchConfiguration",
2100            "Condition" : "historicalRoleCreation",
2101            "Properties" : {
2102                "AssociatePublicIpAddress" : "true",
2103                "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
2104                "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
2105                "KeyName"        : { "Ref" : "KeyName" },
2106                "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "historical", {
"Ref" : "InstanceSize" } ] },
2107                "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
2108                "InstanceMonitoring" : "true",
2109                "BlockDeviceMappings" : [ {
2110                    "DeviceName" : "/dev/sda1",
2111                    "Ebs" : {
2112                        "VolumeSize" : "500",
2113                        "VolumeType" : "gp2",
2114                        "DeleteOnTermination" : "true"
2115                    }
2116                },  {
2117                    "DeviceName" : "/dev/sdb",
2118                        "VirtualName" : "ephemeral0"
2119                },  {
2120                    "DeviceName" : "/dev/sdc",
2121                        "VirtualName" : "ephemeral1"
2122                } ],
2123                "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
2124                    "#!/bin/bash\n",
2125                    "NODEROLE=historical\n",
2126                    "#NODESERVICES=\n",
2127                    "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
2128                    "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
2129                    "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
2130                    "REGION=", { "Ref" : "AWS::Region" }, "\n",
2131                    "VPCID=", { "Ref" : "VPC" }, "\n",
2132                    "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
2133                    "AUTOSCALINGGROUPNAME=historicalGroup\n",
2134                    "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
2135                    "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
2136                    "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
2137                    "S3TYPE=aws\n",
2138                    "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
2139                    "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
2140                    "\n\n",
2141                    "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
2142                    "{\n",
2143                    "  \"redBorder\" : {\n",
```

```
2144                     "    \"aws\" : {\n",
2145                     "      \"cloudwatch\" : \"true\"\n",
2146                     "    }\n",
2147                     "  }\n",
2148                     "}\n",
2149                     "_RBEOF2_\n",
2150                     ". /opt/rb/bin/rb_set_aws_interface.sh",
2151                     "\n"
2152                 ] ] } }
2153             }
2154         },
2155         /*Grupo de autoescalado para el rol historical, configurado en modo Hot*/
2156         "historicalhotGroup" : {
2157             "Type" : "AWS::AutoScaling::AutoScalingGroup",
2158             "Condition" : "historicalhotRoleCreation",
2159             "DependsOn" : "kafkaGroup",
2160             "Properties" : {
2161                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
2162                 "LaunchConfigurationName" : { "Ref" : "historicalhotLaunch" },
2163                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "historicalhot" ] },
2164                 "MaxSize" : "10",
2165                 "Tags" : [ {
2166                     "Key" : "Role",
2167                     "Value" : "historicalhot",
2168                     "PropagateAtLaunch" : "true"
2169                 },{
2170                     "Key" : "Cluster",
2171                     "Value" : { "Ref" : "AWS::StackName" },
2172                     "PropagateAtLaunch" : "true"
2173                 },{
2174                     "Key" : "Name",
2175                     "Value" : { "Fn::Join" : [ "", [
2176                         { "Ref" : "AWS::StackName" }, " - historicalhot" ] ] },
2177                     "PropagateAtLaunch" : "true"
2178                 } ]
2179             },
2180             "CreationPolicy" : {
2181                 "ResourceSignal" : {
2182                     "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "historicalhot" ] },
2183                     "Timeout" : "PT200M"
2184                 }
2185             },
2186             "UpdatePolicy": {
2187                 "AutoScalingRollingUpdate": {
2188                     "MinInstancesInService": "1",
2189                     "MaxBatchSize": "1",
2190                     "PauseTime" : "PT40M",
2191                     "WaitOnResourceSignals": "true"
2192                 }
2193             }
2194         },
2195         /*Configuración de arranque para el rol historical en modo Hot*/
2196         "historicalhotLaunch" : {
2197             "Type" : "AWS::AutoScaling::LaunchConfiguration",
2198             "Condition" : "historicalhotRoleCreation",
2199             "Properties" : {
2200                 "AssociatePublicIpAddress" : "true",
2201                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
2202                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
2203                 "KeyName"        : { "Ref" : "KeyName" },
2204                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "historicalhot", {
"Ref" : "InstanceSize" } ] },
2205                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
2206                 "InstanceMonitoring" : "true",
2207                 "BlockDeviceMappings" : [ {
2208                     "DeviceName" : "/dev/sda1",
2209                     "Ebs" : {
2210                         "VolumeSize" : "50",
2211                         "VolumeType" : "gp2",
2212                         "DeleteOnTermination" : "true"
2213                     }
2214                 },  {
2215                     "DeviceName" : "/dev/sdb",
2216                         "VirtualName" : "ephemeral0"
2217                 },  {
2218                     "DeviceName" : "/dev/sdc",
2219                         "VirtualName" : "ephemeral1"
```

```
2220                 } ],
2221                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
2222                     "#!/bin/bash\n",
2223                     "NODEROLE=historical\n",
2224                     "#NODESERVICES=\n",
2225                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
2226                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
2227                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
2228                     "REGION=", { "Ref" : "AWS::Region" }, "\n",
2229                     "VPCID=", { "Ref" : "VPC" }, "\n",
2230                     "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
2231                     "AUTOSCALINGGROUPNAME=historicalhotGroup\n",
2232                     "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
2233                     "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
2234                     "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
2235                     "S3TYPE=aws\n",
2236                     "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
2237                     "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
2238                     "\n\n",
2239                     "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
2240                     "{\n",
2241                     "  \"redBorder\" : {\n",
2242                     "    \"aws\" : {\n",
2243                     "      \"cloudwatch\" : \"true\"\n",
2244                     "     },\n",
2245                     "    \"druid\" : {\n",
2246                     "      \"historical\" : {\n",
2247                     "        \"tier\" : \"hot\"\n",
2248                     "     }\n",
2249                     "    }\n",
2250                     "  }\n",
2251                     "}\n",
2252                     "_RBEOF2_\n",
2253                     ". /opt/rb/bin/rb_set_aws_interface.sh",
2254                     "\n"
2255                 ] ] } }
2256             }
2257         },
2258         /*Grupo de autoescalado para el rol samza*/
2259         "samzaGroup" : {
2260             "Type" : "AWS::AutoScaling::AutoScalingGroup",
2261             "Condition" : "samzaRoleCreation",
2262             "DependsOn" : "kafkaGroup",
2263             "Properties" : {
2264                 "VPCZoneIdentifier" : [ { "Ref" : "PublicSubnet" } ],
2265                 "LaunchConfigurationName" : { "Ref" : "samzaLaunch" },
2266                 "MinSize" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "samza" ] },
2267                 "MaxSize" : "10",
2268                 "Tags" : [ {
2269                     "Key" : "Role",
2270                     "Value" : "samza",
2271                     "PropagateAtLaunch" : "true"
2272                 },{
2273                     "Key" : "Cluster",
2274                     "Value" : { "Ref" : "AWS::StackName" },
2275                     "PropagateAtLaunch" : "true"
2276                 },{
2277                     "Key" : "Name",
2278                     "Value" : { "Fn::Join" : [ "", [
2279                         { "Ref" : "AWS::StackName" }, " - samza" ] ] },
2280                     "PropagateAtLaunch" : "true"
2281                 } ]
2282             },
2283             "CreationPolicy" : {
2284                 "ResourceSignal" : {
2285                     "Count" : { "Fn::FindInMap" : [ "RolesDistributionMap", { "Ref" :
"NumberOfNodes" }, "samza" ] },
2286                     "Timeout" : "PT200M"
2287                 }
2288             },
2289             "UpdatePolicy": {
2290                 "AutoScalingRollingUpdate": {
2291                     "MinInstancesInService": "1",
2292                     "MaxBatchSize": "1",
2293                     "PauseTime" : "PT40M",
2294                     "WaitOnResourceSignals": "true"
2295                 }
```

```
2296             }
2297         },
2298         /*Configuración de arranque para el rol samza */
2299         "samzaLaunch" : {
2300             "Type" : "AWS::AutoScaling::LaunchConfiguration",
2301             "Condition" : "samzaRoleCreation",
2302             "Properties" : {
2303                 "AssociatePublicIpAddress" : "true",
2304                 "ImageId" : { "Fn::FindInMap" : [ "AWSRegion2AMI", { "Ref" : "AWS::Region"
}, "HVM64" ] },
2305                 "SecurityGroups" : [ { "Ref" : "redBorderSecurityGroup" } ],
2306                 "KeyName"        : { "Ref" : "KeyName" },
2307                 "InstanceType" : { "Fn::FindInMap" : [ "InstanceTypeMap", "samza", { "Ref" :
"InstanceSize" } ] },
2308                 "IamInstanceProfile" : { "Ref" : "IAMInstanceProfile" },
2309                 "InstanceMonitoring" : "true",
2310                 "BlockDeviceMappings" : [ {
2311                     "DeviceName" : "/dev/sda1",
2312                     "Ebs" : {
2313                         "VolumeSize" : "50",
2314                         "VolumeType" : "gp2",
2315                         "DeleteOnTermination" : "true"
2316                     }
2317                 },  {
2318                     "DeviceName" : "/dev/sdb",
2319                         "VirtualName" : "ephemeral0"
2320                 },  {
2321                     "DeviceName" : "/dev/sdc",
2322                         "VirtualName" : "ephemeral1"
2323                 } ],
2324                 "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
2325                     "#!/bin/bash\n",
2326                     "NODEROLE=samza\n",
2327                     "NODESERVICES=\n",
2328                     "CDOMAIN=", { "Ref" : "CDOMAIN" }, "\n",
2329                     "PUBLIC_HOSTEDZONE_ID=", { "Ref" : "PublicHostedZone" }, "\n",
2330                     "PRIVATE_HOSTEDZONE_ID=", { "Ref" : "PrivateHostedZone" }, "\n",
2331                     "REGION=", { "Ref" : "AWS::Region" }, "\n",
2332                     "VPCID=", { "Ref" : "VPC" }, "\n",
2333                     "STACKNAME=", { "Ref" : "AWS::StackName" }, "\n",
2334                     "AUTOSCALINGGROUPNAME=samzaGroup\n",
2335                     "SUBNET_ID=", { "Ref" : "PrivateSubnet" }, "\n",
2336                     "S3BUCKET=", { "Ref" : "S3Bucket" }, " \n",
2337                     "S3HOST=s3-", { "Ref" : "AWS::Region" }, ".amazonaws.com\n",
2338                     "S3TYPE=aws\n",
2339                     "AWS_ACCESS_KEY=\"", { "Ref" : "IAMAccessKey" }, "\"\n",
2340                     "AWS_SECRET_KEY=\"", { "Fn::GetAtt" : [ "IAMAccessKey",
"SecretAccessKey" ] }, "\"\n",
2341                     "\n\n",
2342                     "cat > /opt/rb/etc/chef/initialdata.json <<- _RBEOF2_\n",
2343                     "{\n",
2344                     "  \"redBorder\" : {\n",
2345                     "    \"aws\" : {\n",
2346                     "      \"cloudwatch\" : \"true\"\n",
2347                     "    }\n",
2348                     "  }\n",
2349                     "}\n",
2350                     "_RBEOF2_\n",
2351                     ". /opt/rb/bin/rb_set_aws_interface.sh",
2352                     "\n"
2353                 ] ] } }
2354             }
2355         }
2356     }
2357 }
2358
```

ANEXO V - SERVICIO RB-CLOUDWATCH

En este Anexo se adjunta el código de rb_cloudwatch y la documentación asociada al mismo.

Los paquetes que forman el programa se muestran en el siguiente diagrama:

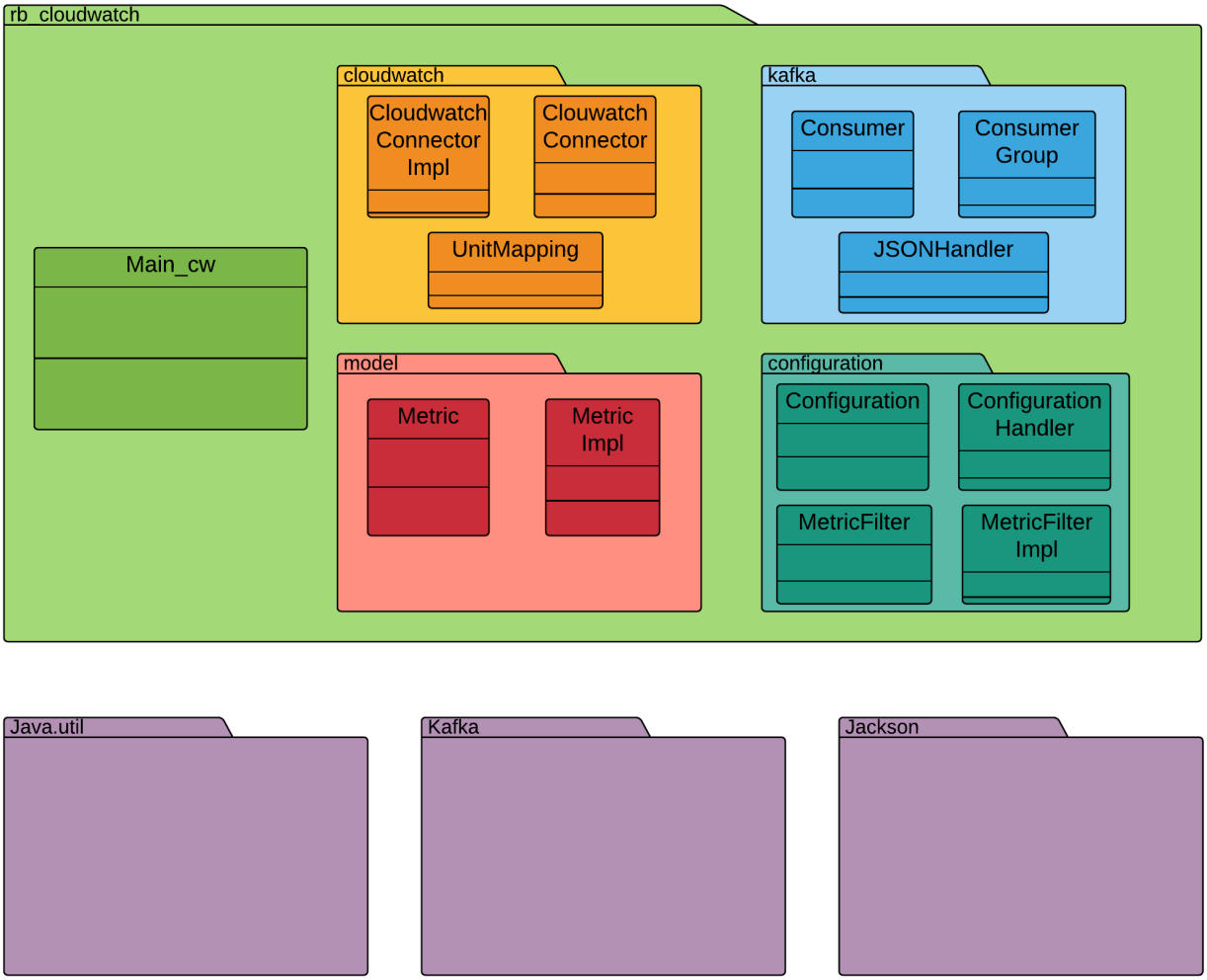

Figura 29 Diagrama de paquetes general de rb_cloudwatch

Configuración

rb_cloudwatch utiliza un fichero de configuración en formato JSON con los siguientes campos:

- **zk_connect**: host y puerto donde se encuentra el servicio Apache Zookeeper, necesario para poder obtener mensajes de Apache Kafka.
- **kafka_consumer_group_id**: identificador del grupo de consumidores entre los que se balancea el envío de mensajes kafka.
- **zookeeper_session_timeout**: tiempo máximo de espera antes de cerrar una conexión con Zookeeper.
- **zookeeper_sync_time**: tiempo má.ximo entre sincronizaciones con Zookeeper
- **autocommit_interval**: intervalo de tiempo entre cada confirmación de lectura a Kafka.
- **kafka_topic**: topic del que se leen mensajes de Kafka.
- **thread_number**: número de hilos que consumiran mensajes de la cola Kafka.
- **region**: region del servicio Amazon Cloudwatch al que se enviarán las métricas.
- **accesskey y secretkey**: credenciales de acceso a Amazon Web Services para poder enviar métricas. El usuario asociado debe tener permisos para enviar métricas.
- **namespace**: contenedor de Amazon Cloudwatch asociado a las métricas enviadas por la aplicación.
- **filter**: configuración de filtros. Sólo se enviarán métricas que cumplan con los filtros especificados. Se puede filtrar por nombre de métrica (**allowedMetricNames**) y por identificador de instancia que envía la métrica (**allowedInstanceIds**). Si la cadena con los valores permitidos contiene el símbolo '*', se reenvía todo. Para que una métrica sea enviada debe estar permitida en ambos tipos de filtro.

A continuación se muestra un ejemplo de configuración para rb_cloudwatch

```
{
  "zk_connect" : "localhost:2181",
  "kafka_consumer_group_id" : "1",
  "zookeeper_session_timeout" : "400",
  "zookeeper_sync_time" : "200",
  "autocommit_interval" : "1000",
  "kafka_topic" : "rb_monitor",
  "thread_number" : "2",
  "region" : "eu-west-1",
  "accesskey" : "example12345",
  "secretkey" : "example54321",
  "namespace" : "testNamespace",
  "filter" : {
    "allowedMetricNames" : [
      "desired_capacity", "pending_tasks"
    ],
    "allowedInstanceIds" : [
      "*"
    ]
  }
}
```

Código del programa

Paquete rb_cloudwatch

Main_cw.java

```
package rb_cloudwatch;

import rb_cloudwatch.cloudwatch.CloudwatchConnector;
import rb_cloudwatch.cloudwatch.CloudwatchConnectorImpl;
import rb_cloudwatch.configuration.Configuration;
import rb_cloudwatch.configuration.ConfigurationHandler;
import rb_cloudwatch.kafka.ConsumerGroup;

/**
 * Main class with main method of rb_cloudwatch.
 */
public class Main_cw {
    /**
     *  rb_cloudwatch main method. Starts reading configuration and then initiate Kafka consumer
     *  to get events.
     *  @author Alberto Rodriguez de la Cruz
     */
    public static void main (String[] args) {
        System.out.println(args.length);
        if(args.length == 1) {
            //Applying configuration...
            Configuration configuration = ConfigurationHandler.readConfiguration(args[0]);
            //Creating class to connect with AWS
            CloudwatchConnector cloudwatchConnector = new CloudwatchConnectorImpl(configuration);
            //Creating kafka consumer threads
            ConsumerGroup consumerGroup = new ConsumerGroup(configuration, cloudwatchConnector);
            consumerGroup.run();
        } else {
            System.out.println("Arguments: [ Configfile ]");
        }
    }
}
```

Paquete rb_cloudwatch.configuration

Configuration.java

```
package rb_cloudwatch.configuration;

/**
 * POJO Class with configuration parameters
 * @author Alberto Rodriguez de la Cruz
 */
public class Configuration {

    private String zk_connect;
    private String kafka_consumer_group_id;
    private String zookeeper_session_timeout;
    private String zookeeper_sync_time;
    private String autocommit_interval;
    private String kafka_topic;
    private String thread_number;
    private String region;
    private String namespace;
    private String accesskey;
    private String secretkey;
    private MetricFilter filter;

    public Configuration() {
        this.zk_connect = zk_connect;
        this.kafka_consumer_group_id = null;
        this.zookeeper_session_timeout = null;
        this.zookeeper_sync_time = null;
        this.autocommit_interval = null;
        this.kafka_topic = null;
        this.thread_number = null;
        this.region = null;
        this.namespace = null;
        this.accesskey = null;
        this.secretkey = null;
        this.filter = null;
```

```
}

public String getAccesskey() {
    return accesskey;
}
public void setAccesskey(String accesskey) {
    this.accesskey = accesskey;
}
public String getSecretkey() {
    return secretkey;
}
public void setSecretkey(String secretkey) {
    this.secretkey = secretkey;
}
public String getZk_connect() {
    return zk_connect;
}
public void setZk_connect(String zk_connect) {
    this.zk_connect = zk_connect;
}
public String getKafka_consumer_group_id() {
    return kafka_consumer_group_id;
}
public void setKafka_consumer_group_id(String kafka_consumer_group_id) {
    this.kafka_consumer_group_id = kafka_consumer_group_id;
}
public String getZookeeper_session_timeout() {
    return zookeeper_session_timeout;
}
public void setZookeeper_session_timeout(String zookeeper_session_timeout) {
    this.zookeeper_session_timeout = zookeeper_session_timeout;
}
public String getZookeeper_sync_time() {
    return zookeeper_sync_time;
}
public void setZookeeper_sync_time(String zookeeper_sync_time) {
    this.zookeeper_sync_time = zookeeper_sync_time;
}
public String getAutocommit_interval() {
    return autocommit_interval;
}
public void setAutocommit_interval(String autocommit_interval) {
    this.autocommit_interval = autocommit_interval;
}
public String getKafka_topic() {
    return kafka_topic;
}
public void setKafka_topic(String kafka_topic) {
    this.kafka_topic = kafka_topic;
}
public String getThread_number() {
    return thread_number;
}
public void setThread_number(String thread_number) {
    this.thread_number = thread_number;
}
public String getRegion() {
    return region;
}
public void setRegion(String region) {
    this.region = region;
}
public String getNamespace() {
    return namespace;
}
public void setNamespace(String namespace) {
    this.namespace = namespace;
}
public MetricFilter getFilter() {
    return filter;
}
public void setFilter(MetricFilter filter) {
    this.filter = filter;
}

@Override
public String toString() {
    return "Configuration{" +
            "zk_connect='" + zk_connect + '\'' +
            ", kafka_consumer_group_id='" + kafka_consumer_group_id + '\'' +
            ", zookeeper_session_timeout='" + zookeeper_session_timeout + '\'' +
            ", zookeeper_sync_time='" + zookeeper_sync_time + '\'' +
            ", autocommit_interval='" + autocommit_interval + '\'' +
```

```
                ', kafka_topic='' + kafka_topic + '\'' +
                ', thread_number='' + thread_number + '\'' +
                ', region='' + region + '\'' +
                ', namespace='' + namespace + '\'' +
                ', accesskey='' + accesskey + '\'' +
                ', secretkey='' + secretkey + '\'' +
                ', filter=' + filter.toString() +
                '}';
    }
}
```

ConfigurationHandler.java

```
package rb_cloudwatch.configuration;

import kafka.consumer.ConsumerConfig;
import org.codehaus.jackson.JsonParseException;

import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Gets and process configuartion from JSON configuration fles, and generate an
 * Configuration POJO object.
 * @author Alberto Rodriguez de la Cruz
 */
public class ConfigurationHandler {

    private static final Logger logger =
            Logger.getLogger(ConfigurationHandler.class.getName());

    /**
     * Static method that read configuration file (in JSON) indicated
     * in srcPath variable and generates a POJO object with
     * this configuration
     * Uses Jackson library to parse JSON
     * @param srcPath
     * @return
     */
    public static Configuration readConfiguration(String srcPath) {

        logger.info("Reading configuration file " + srcPath );
        Map<String, String> map = new HashMap<String, String>();    //Hash map with JSON values
        //Object that process JSON data
        org.codehaus.jackson.map.ObjectMapper jacksonMapper =
                new org.codehaus.jackson.map.ObjectMapper();

        File src; //configuration file
        Configuration config = new Configuration(); //Configuration POJO object

        if(srcPath == null || srcPath.isEmpty()) {
            src = new File('./config.json'); //Default file config
        } else {
            src = new File(srcPath);
        }
        try {
            config = jacksonMapper.readValue(src, Configuration.class);
            config.getFilter().createAllowedInstanceIdsHashSet();
            config.getFilter().createAllowedMetricNamesHashSet();
            System.out.println(config.toString());

        //if there is an error reading configuration, program finish
        } catch (JsonParseException e) {
            logger.log(Level.SEVERE,
                    'Error reading JSON configuration file, exiting...', e);
            e.printStackTrace();
            System.exit(1);
        } catch (IOException e) {
            logger.log(Level.SEVERE,
                    'Input/Output Error when try to read ' +
                            'configuration file, exiting...', e);
            e.printStackTrace();
            System.exit(1);
        } catch (Exception e) {
```

```
            logger.log(Level.SEVERE,
                    "Error reading JSON configuration file, exiting...", e);
            e.printStackTrace();
            System.exit(1);
        }
        return config;
    }
    /**
     * Static method that creates a ConsumerConfig object with the configuration
     * contained in a Configuration object.
     * readConfiguration method must be executed before execute this method.
     * @param config
     * @return
     */
    public static ConsumerConfig createConsumerConfig(Configuration config) {
        Properties props = new Properties(); //Configuration for
        props.put("zookeeper.connect", config.getZk_connect());
        props.put("group.id", config.getKafka_consumer_group_id());
        props.put("zookeeper.session.timeout.ms",
            config.getZookeeper_session_timeout());
        props.put("zookeeper.sync.time.ms", config.getZookeeper_session_timeout());
        props.put("auto.commit.interval.ms", config.getAutocommit_interval());
        return new ConsumerConfig(props);
    }
}
```

MetricFilter.java

```
package rb_cloudwatch.configuration;

import org.codehaus.jackson.map.annotate.JsonDeserialize;
import rb_cloudwatch.model.Metric;

import java.util.List;

/**
 * Interface with methods to manage metric filtering
 * @author Alberto Rodriguez de la Cruz
 */
@JsonDeserialize(as=MetricFilterImpl.class)
public interface MetricFilter {
    public List<String> getAllowedInstanceIds();
    public void setAllowedInstanceIds(List<String> allowedInstanceIds);
    public List<String> getAllowedMetricNames();
    public void setAllowedMetricNames(List<String> allowedMetricNames);

    /**
     * Checks if a metric must be filtered. If returns true,
     * Metric musn't be filtered. In other case, yes.
     */
    public boolean checkMetric(Metric metric);

    /**
     * Creates Hash Set from list to quick search of filters
     */
    public void createAllowedInstanceIdsHashSet();

    /**
     * Creates Hash Set from list to quick search of filters
     */
    public void createAllowedMetricNamesHashSet();

    /**
     * toString method for debug purposes
     * @return Descriptive string of object
     */
    public String toString();
}
```

MetricFilterImpl

```
package rb_cloudwatch.configuration;

import rb_cloudwatch.model.Metric;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

/**
```

```
 * Implementation of MetricFilter interface
 * @author Alberto Rodriguez de la Cruz
 */
public class MetricFilterImpl implements MetricFilter {

    private List<String> allowedMetricNames;
    private List<String> allowedInstanceIds;
    private Set<String> allowedMetricNamesSet;
    private Set<String> allowedInstaceIdsSet;

    /*Constructors*/
    public MetricFilterImpl() {
        this.allowedInstanceIds = null;
        this.allowedMetricNames = null;
        this.allowedInstaceIdsSet = null;
        this.allowedMetricNamesSet = null;
    }
    public MetricFilterImpl(List<String> allowedInstanceIds, List<String> allowedMetricNames) {
        this.allowedInstanceIds = allowedInstanceIds;
        this.allowedMetricNames = allowedMetricNames;
        this.allowedInstaceIdsSet = null;
        this.allowedMetricNamesSet = null;
    }

    public List<String> getAllowedInstanceIds() {
        return allowedInstanceIds;
    }
    public void setAllowedInstanceIds(List<String> allowedInstanceIds) {
        this.allowedInstanceIds = allowedInstanceIds;
    }
    public List<String> getAllowedMetricNames() {
        return allowedMetricNames;
    }
    public void setAllowedMetricNames(List<String> allowedMetricNames) {
        this.allowedMetricNames = allowedMetricNames;
    }

    @Override
    public String toString() {
        return 'MetricFilterImpl{' +
                'allowedMetricNames=' + allowedMetricNames +
                ', allowedInstanceIds=' + allowedInstanceIds +
                ', allowedMetricNamesSet=' + allowedMetricNamesSet +
                ', allowedInstaceIdsSet=' + allowedInstaceIdsSet +
                '}';
    }

    /**
     * Method that checks if this Metric must be forwarded
     */
    public void createAllowedInstanceIdsHashSet() {
        allowedInstaceIdsSet = new HashSet<>(allowedInstanceIds);
    }
    public void createAllowedMetricNamesHashSet() {
        allowedMetricNamesSet = new HashSet<>(allowedMetricNames);
    }
    public boolean checkMetric(Metric metric) {
        boolean resultInstanceIds = false;
        boolean resultMetricNames = false;
        if(allowedInstaceIdsSet.contains(metric.getSensor_name()) ||
allowedInstaceIdsSet.contains("*")) {
            resultInstanceIds = true;
        }
        if(allowedMetricNamesSet.contains(metric.getMonitor()) ||
allowedMetricNamesSet.contains("*")) {
            resultMetricNames = true;
        }
        return resultInstanceIds && resultMetricNames;
    }
}
```

Paquete rb_cloudwatch.kafka

ConsumerGroup.java

```
package rb_cloudwatch.kafka;

import kafka.consumer.KafkaStream;
import kafka.javaapi.consumer.ConsumerConnector;
import rb_cloudwatch.cloudwatch.CloudwatchConnector;
import rb_cloudwatch.configuration.Configuration;
import rb_cloudwatch.configuration.ConfigurationHandler;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.logging.Logger;
/**
 * Manages a pool of Consumers that runs in multiple threads.
 * @author alberto
 */
public class ConsumerGroup {
    private Configuration config;
    private ConsumerConnector consumerConnector;
    private CloudwatchConnector cloudwatchConnector;
    private ExecutorService executor;
    private static final Logger logger = Logger.getLogger
            (ConsumerGroup.class.getName());
    public ConsumerGroup(Configuration config,
                         CloudwatchConnector cloudwatchConnector) {
        this.cloudwatchConnector = cloudwatchConnector;
        this.config = config;
        consumerConnector =
                kafka.consumer.Consumer.createJavaConsumerConnector
                (ConfigurationHandler.createConsumerConfig(config));
    }
    public void run() {
        Map<String, Integer> topicCountMap = new HashMap<String, Integer>();
        //Select Topic for consume and the number of threads
        // which will consume in this topic.
        topicCountMap.put(config.getKafka_topic(),
                Integer.parseInt(config.getThread_number()));
        Map<String, List<KafkaStream<byte[], byte[]>>> consumerMap =
                consumerConnector.createMessageStreams(topicCountMap);
        List<KafkaStream<byte[], byte[]>> streams =
                consumerMap.get(config.getKafka_topic());
        //Launch all the threads. Number of thread is obtained from configuration.
        executor = Executors.newFixedThreadPool
                (Integer.parseInt(config.getThread_number()));
        logger.info("Created Thread pool to consume kafka streams");
        //Create object to consume messages
        int threadNumber = 0;
        for (KafkaStream stream : streams) {
            executor.submit(new Consumer(cloudwatchConnector, config.getFilter(),
                    stream, threadNumber));
            logger.info("Started thread number " + threadNumber);
            threadNumber++;
        }
    }
}
```

Consumer.java

```
package rb_cloudwatch.kafka;

import kafka.consumer.ConsumerIterator;
import kafka.consumer.KafkaStream;
import rb_cloudwatch.cloudwatch.CloudwatchConnector;
import rb_cloudwatch.configuration.MetricFilter;
import rb_cloudwatch.model.Metric;

import java.util.logging.Logger;

/**
 * Kafka consumer that runs in one thread. Connect with Kafka cluster, get new messages and
 * process it using Cloudwatch management classes
 * @author Alberto Rodriguez de la Cruz
 */
class Consumer implements Runnable {

    private KafkaStream kafkaStream;
    private JSONHandler jsonHandler;
    private CloudwatchConnector cloudwatchConnector;
    private Integer threadNumber;
    MetricFilter metricFilter;

    private static final Logger logger = Logger.getLogger(Consumer.class.getName());

    public Consumer(CloudwatchConnector cloudwatchConnector,
                    MetricFilter metricFilter, KafkaStream kafkaStream, int threadNumber) {
        jsonHandler = new JSONHandler();
        this.cloudwatchConnector = cloudwatchConnector;
        this.kafkaStream = kafkaStream;
        this.threadNumber = threadNumber;
        this.metricFilter = metricFilter;
    }

    @Override
    public void run() {
        ConsumerIterator<byte[], byte[]> iterator = kafkaStream.iterator();
        while(iterator.hasNext()) {
            try {
                logger.fine('New message detected in thread ' + threadNumber.toString());
                //First, consume a message and generate a Metric
                // object with information consumed
                String message = new String(iterator.next().message());
                Metric metric = jsonHandler.processJSON(message);
                logger.fine('Thread ' + threadNumber.toString() +
                        ' consumed a message. Metric object generated\n' +
                        metric.toString());
                //Filtering metrics
                if (metricFilter.checkMetric(metric)) {
                    //Then, send this metric to AWS Cloudwatch service
                    cloudwatchConnector.sendMetric(metric);
                }
            } catch (Exception e) {
                logger.warning('Message parsing error');
                e.printStackTrace();
            }
        }
    }
}
```

JSONHandler.java

```
package rb_cloudwatch.kafka;

import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.type.TypeReference;
import rb_cloudwatch.model.Metric;
import rb_cloudwatch.model.MetricImpl;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Parse rb_monitor messages and creates a Metric POJO object
 * @author Alberto Rodríguez de la Cruz
 */
class JSONHandler  {

    private static final Logger logger = Logger.getLogger
            (JSONHandler.class.getName());

    private Metric metric;
    private Map<String, String> map;
    private org.codehaus.jackson.map.ObjectMapper jacksonMapper;

    public JSONHandler() {
        map = new HashMap<String, String>(); //Construct map
        jacksonMapper = new org.codehaus.jackson.map.ObjectMapper();
    }

    public Metric processJSON(String json) throws Exception {

        try {
            map = jacksonMapper.readValue(json,
                    new TypeReference<HashMap<String, String>>() {
            });
        } catch (JsonParseException e) {
            logger.log(Level.SEVERE, 'Error reading JSON message', e);
            throw new Exception ('JSON error');
        } catch (IOException e) {
            logger.log(Level.SEVERE, "No input to Jackson parser?", e);
            throw new Exception ('JSON input error');
        }
        metric = new MetricImpl(map.get('timestamp'), map.get('sensor_name'),
                map.get('monitor'), map.get('value'), map.get('type'),
                map.get('unit'));
        return metric;
    }
}
```

Paquete rb_cloudwatch.cloudwatch

CloudwatchConnector.java

```
package rb_cloudwatch.cloudwatch;
import rb_cloudwatch.model.Metric;
/**
 * Interface that provides methos to manages metric sending to AWS Cloudwatch
 * @author Alberto Rodriguez de la Cruz
 */
public interface CloudwatchConnector {
    /**
     * Send data stored in Metric class to AWS Cloudwatch.
     * @param metric
     * @return Return true if metric has been sended correctly, and false in other case.
     */
    public boolean sendMetric(Metric metric);
}
```

CloudwatchConnectorImpl.java

```
package rb_cloudwatch.cloudwatch;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.BasicAWSCredentials;
```

```
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.cloudwatch.AmazonCloudWatch;
import com.amazonaws.services.cloudwatch.AmazonCloudWatchClient;
import com.amazonaws.services.cloudwatch.model.Dimension;
import com.amazonaws.services.cloudwatch.model.MetricDatum;
import com.amazonaws.services.cloudwatch.model.PutMetricDataRequest;
import rb_cloudwatch.configuration.Configuration;
import rb_cloudwatch.model.Metric;

import java.util.Collection;
import java.util.LinkedList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Implementation of Cloudwatch Connector.
 * @author Alberto Rodriguez de la Cruz
 */
public class CloudwatchConnectorImpl implements CloudwatchConnector {

    /* Attributes */
    AmazonCloudWatch client;
    UnitMapping unitMapping;
    Configuration configuration;
    //Logger for this class
    private static final Logger logger = Logger.getLogger
            (CloudwatchConnectorImpl.class.getName());

    /* Constructors */
    public CloudwatchConnectorImpl(Configuration configuration) {
        BasicAWSCredentials awsCreds = new BasicAWSCredentials
                (configuration.getAccesskey(), configuration.getSecretkey());
        this.client = new AmazonCloudWatchClient(awsCreds);
        client.setRegion(Region.getRegion
                (Regions.fromName(configuration.getRegion())));
        unitMapping = new UnitMapping();
        this.configuration = configuration;
    }

    @Override
    public synchronized boolean sendMetric(Metric metric) {
        boolean error = false; //Variable to indicate error status

        Collection<MetricDatum> l = new LinkedList<MetricDatum>();
        MetricDatum awsmetric = new MetricDatum();
        PutMetricDataRequest metricRequest = new PutMetricDataRequest();
        logger.fine("Created AWS Metric data structures");

        metricRequest.setNamespace(configuration.getNamespace());

        awsmetric.setMetricName(metric.getMonitor());
        awsmetric.setUnit(unitMapping.getMappedUnit(metric.getUnit()));
        awsmetric.setTimestamp(metric.getTimestamp());
        awsmetric.setValue(metric.getValue());
        Dimension instanceId = new Dimension();
        instanceId.setName("InstanceId");
        instanceId.setValue(metric.getSensor_name());
        awsmetric.withDimensions(instanceId);

        l.add(awsmetric);
        metricRequest.setMetricData(l);
        logger.fine(awsmetric.toString());

        try {
     client.putMetricData(metricRequest);
            logger.log(Level.FINE, "putMetricData executed");
        } catch (AmazonServiceException e) {
            logger.log(Level.SEVERE, "Amazon Server Exception", e);
            error = true;
        } catch (AmazonClientException e) {
            logger.log(Level.SEVERE, "Amazon Client Exception", e);
            error = true;
        }
        return error;
    }
}
```

UnitMapping.java

```
package rb_cloudwatch.cloudwatch;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Logger;

/**
 * Class with methos to convert units provided via Kafka to AWS metrics.
 * @author Alberto Rodriguez de la Cruz
 */
public class UnitMapping{

    private Map<String, String> map;
    private static final Logger logger = Logger.getLogger
            (CloudwatchConnectorImpl.class.getName());

    UnitMapping() {
        map = new HashMap<String, String>();
        map.put('%', 'Percent');
        map.put('ms', 'Milliseconds');
        map.put('', 'None');
        map.put('msgs', 'Count');
        map.put('tasks', 'Count');
        map.put('task%', 'None');
    }

    /**
     * Converts an unit from rb_monitor to AWS Unit. If it not possible,
     * returns AWS Unit None
     * @param unit from rb_monitor
     * @return Unit in AWS Cloudwatch Unit format.
     */
    public String getMappedUnit(String unit) {
        String awsUnit = map.get(unit);
        if(awsUnit == null) {
            logger.warning('Unit ' + unit +
                    ' not found in map, using AWS unit None');
            awsUnit = 'None';
        }
        return awsUnit;
    }
}
```

Paquete rb_cloudwatch.model

Metric.java

```
package rb_cloudwatch.model;

import java.util.Date;

/**
 * Interface of POJO object with Metric information
 * @author Alberto Rodriguez de la Cruz
 */
public interface Metric {

    public Date getTimestamp();
    public String getSensor_name();
    public String getMonitor();
    public Double getValue();
    public String getType();
    public String getUnit();

    public void setTimestamp(Date timestamp);
    public void setSensor_name(String sensorName);
    public void setMonitor(String monitor);
    public void setValue(Double value);
    public void setType(String type);
    public void setUnit(String unit);

    public String toString();
}
```

MetricImpl.java

```
package rb_cloudwatch.model;

import java.util.Date;
/**
 * Implementation of Metric Interface
 * @author Alberto Rodríguez de la Cruz
 */
public class MetricImpl implements Metric {

    /* Attributes */
    private Date timestamp;
    private String sensor_name;
    private String monitor;
    private Double value;
    private String type;
    private String unit;

    /* Constructors */
    public MetricImpl(String timestamp, String sensor_name,
                      String monitor, String value, String type, String unit) {

        this.timestamp = new Date(Long.parseLong(timestamp)*1000);
        this.sensor_name = sensor_name;
        this.monitor = monitor;
        this.value = Double.parseDouble(value);
        this.type = type;
        this.unit = unit;
    }

    /* Methods */
    public Date getTimestamp() {
        return timestamp;
    }
    public String getSensor_name() {
        return sensor_name;
    }
    public String getMonitor() {
        return monitor;
    }
    public Double getValue() {
        return value;
    }
    public String getType() {
        return type;
    }
    public String getUnit() {
```

```
        return unit;
    }

    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }
    public void setSensor_name(String sensor_name) {
        this.sensor_name = sensor_name;
    }
    public void setMonitor(String monitor) {
        this.monitor = monitor;
    }
    public void setValue(Double value) {
        this.value = value;
    }
    public void setType(String type) {
        this.type = type;
    }
    public void setUnit(String unit) {
        this.unit = unit;
    }

    @Override
    public String toString() {
        return 'MetricImpl{' +
                '\ntimestamp=' + timestamp +
                ', \nsensor_name='' + sensor_name + '\'' +
                ', \nmonitor='' + monitor + '\'' +
                ', \nvalue=' + value +
                ', \ntype='' + type + '\'' +
                ', \nunit='' + unit + '\'' +
                '\n}';
    }
}
```

ANEXO VI - SCRIPT RB-ROUTE53

rb_route53 es un script que se utiliza para registrar el nombre de las instancias en las Hosted Zones de Route 53 asociadas con el clúster. Para ello utiliza la variable pasada por user-data CDOMAIN, que le indica cual es el dominio donde tiene que registrar los nombres, puediendo registrar el nombre tanto en Hosted Zones públicas como privadas.

Además, el script crea las Hosted Zones necesarias en caso de que no existieran. Sin embargo, hay que tener en cuenta que para que los nombres puedan ser resueltos desde internet, debe estar registrado el dominio de segundo nivel en Route 53 con anterioridad.

Opciones

A continuación se describen las opciones del script

- -d : indica el dominio (cdomain) en el que registrar los nombres.
- -i : dirección IP que se asociará al nombre registrado en la Hosted Zone pública. Si no se indica nada, el script averigua la IP pública de la instancia.
- -p: dirección IP que se asociará al nombre registrado en la Hosted Zone privada. Si no se indica nada, el script averigua la IP privada de la instancia.
- -n: nombre del host que se quiere registrar. De esta forma, el registro tendría el formato hostname.cdomain. En el caso de que no se indicara nada, el script averiguaría el hostname de la instancia.
- -v: VPC a la que asociar la Hosted Zone privada.
- -r: región de la VPC a la que se asocia la Hosted Zone privada.
- -a: permite especificar la Hosted Zone pública si se había creado con anterioridad.
- -b: permite especificar la Hosted Zone privada si se había creado con anterioridad.
- -h: ayuda del comando

Código

rb_route53.sh

```
1    #!/bin/bash
2
3    source /opt/rb/bin/rb_manager_functions.sh
4    source /etc/profile
5
6    SCRIPTNAME=$(basename $0)
7    USAGE="Usage: $SCRIPTNAME -d <DOMAIN> [ -v <VPC_ID> -r <REGION> -a
<PUBLIC_HOSTED_ZONE_ID> -b <PRIVATE_HOSTED_ZONE_ID> (only in AWS)] [ -i
<PUBLIC_IP_HOST> -p <PRIVATE_IP_HOST> -n <HOST_NAME> -c <CLUSTER_NAME> -h
help ]"
```

```
8     RESULT=""
9
10    function examples() {
11        echo
12        echo "Examples: "
13        echo "-> $SCRIPTNAME -n rbhost -i 10.10.10.10 -d redbordercloud.com"
14        echo "      will create rbhost.redbordercloud.com"
15        echo "-> $SCRIPTNAME -n rbhost -i 10.10.10.10 -d redbordercloud.com
-c cluster01"
16        echo "      will create rbhost.cluster01.redbordercloud.com"
17    }
18    #This function looks for a HostedZone with a name and with a type
(public or private) and
19    #returns its id.
20    function getHostedZoneId() {
21        DOMAIN_TO_SEARCH=$1
22        TYPE_TO_SEARCH=$2
23        if [ "x$DOMAIN_TO_SEARCH" = "x" ] ; then
24            echo "Error in function getHostedZoneId(), Domain to search
required"
25            exit 1
26        elif [ "x$TYPE_TO_SEARCH" != "xprivate" -a "x$TYPE_TO_SEARCH" !=
"xpublic" ] ; then
27            echo "getHostedZoneId: Type to search not found, setting to
public"
28            TYPE_TO_SEARCH="public"
29        fi
30        if [ "$TYPE_TO_SEARCH" = "public" ] ; then
31            TYPE_IS_PRIVATE="false"
32        else
33            TYPE_IS_PRIVATE="true"
34        fi
35        HOSTED_ZONE_LIST=$(aws route53 list-hosted-zones)
36        COUNTER=0
37        EXIT=0
38        RESULT=""
39        SEARCHNAME="0"
40        while [ "x$SEARCHNAME" != "xnull" -a "x$SEARCHNAME" != "x" -a $EXIT
-eq 0 ] ; do
41            SEARCHNAME=$(echo $HOSTED_ZONE_LIST | jq -r
.HostedZones[$COUNTER].Name)
42            SEARCHTYPE=$(echo $HOSTED_ZONE_LIST | jq -r
.HostedZones[$COUNTER].Config.PrivateZone)
43            if [ "$DOMAIN_TO_SEARCH." = "$SEARCHNAME" ] ; then
44                if [ "$TYPE_IS_PRIVATE" = "$SEARCHTYPE" ] ; then
45                    RESULT=$(echo $HOSTED_ZONE_LIST | jq -r
.HostedZones[$COUNTER].Id)
46                    EXIT=1
47                fi
48            fi
49            let COUNTER=COUNTER+1
50            if [ $COUNTER -eq 10000 ] ; then
51                echo "Infinite loop detected"
52                exit 1
53            fi
54        done
55    }
56    function setNameServers() {
57        DOMAIN_FIRST_LEVEL=$(echo $DOMAIN | sed -r
's/.*\.([^.]+\.[^.]+)$/\1/')
58        NAMESERVERS_HOSTED_ZONE=$(aws route53 get-hosted-zone --id
$HOSTED_ZONE_ID)
59        getHostedZoneId $DOMAIN_FIRST_LEVEL public
60        DOMAIN_HOSTED_ZONE_ID=$RESULT
```

```
61        if [ "x$DOMAIN_HOSTED_ZONE_ID" != "x" -a "x$DOMAIN_HOSTED_ZONE_ID"
!= "xnull" ] ; then
62            NAMESERVERS_UPSERT=$(aws route53 change-resource-record-sets --
hosted-zone-id $DOMAIN_HOSTED_ZONE_ID --change-batch \
63            "{ \"Changes\": [
\
64                {
\
65                    \"Action\":\"UPSERT\",
\
66                    \"ResourceRecordSet\": {
\
67                        \"Name\":\"$CLUSTER_NAME$DOMAIN\",
\
68                        \"Type\":\"NS\",
\
69                        \"TTL\":300,
\
70                        \"ResourceRecords\" : [
\
71                            {\"Value\":\"$(echo $NAMESERVERS_HOSTED_ZONE |
jq -r .DelegationSet.NameServers[0])\"}, \
72                            {\"Value\":\"$(echo $NAMESERVERS_HOSTED_ZONE |
jq -r .DelegationSet.NameServers[1])\"}, \
73                            {\"Value\":\"$(echo $NAMESERVERS_HOSTED_ZONE |
jq -r .DelegationSet.NameServers[2])\"}, \
74                            {\"Value\":\"$(echo $NAMESERVERS_HOSTED_ZONE |
jq -r .DelegationSet.NameServers[3])\"}  \
75                        ]
\
76                    }
\
77                }
\
78            ] }")
79            NAMESERVERS_UPSERT_ID=$(echo $NAMESERVERS_UPSERT | jq -r
.ChangeInfo.Id)
80        else
81            NAMESERVERS_UPSERT_ID=""
82        fi
83    }
84
85    function configureHostedZone() {
86        HOSTED_ZONE_TYPE=$1
87        HOSTED_ZONE_ID=$2
88        if [ "x$HOSTED_ZONE_TYPE" != "xpublic" -a "x$HOSTED_ZONE_TYPE" !=
"xprivate" ] ; then
89            echo "configureHostedZone: Invalid Hosted Zone type"
90        else
91            if [ "x$HOSTED_ZONE_ID" = "x" ] ; then
92                getHostedZoneId $CLUSTER_NAME$DOMAIN $HOSTED_ZONE_TYPE
93                HOSTED_ZONE_ID=$RESULT
94            fi
95            if [ "xnull" == "x$HOSTED_ZONE_ID" -o "x" == "x$HOSTED_ZONE_ID"
] ; then
96                echo -n "$HOSTED_ZONE_TYPE HostedZone $CLUSTER_NAME$DOMAIN
doesn't exists, creating"
97                if [ "$HOSTED_ZONE_TYPE" = "public" ] ; then
98                    HOSTED_ZONE_DATA=$(aws route53 create-hosted-zone --name
$DOMAIN --caller-reference "public$HOST_NAME" )
99                else
```

```
100                 HOSTED_ZONE_DATA=$(aws route53 create-hosted-zone --name
$DOMAIN --caller-reference "private$HOST_NAME" --vpc
VPCRegion=$REGION,VPCId=$VPC_ID )
101             fi
102             HOSTED_ZONE_ID=$(echo $HOSTED_ZONE_DATA | jq -r
.HostedZone.Id)
103             if [ "x" = "x$HOSTED_ZONE_ID" ]; then
104                 print_result 1
105                 echo "Error creating a new $HOSTED_ZONE_TYPE hosted zone
$CLUSTER_NAME$DOMAIN"
106                 echo $HOSTED_ZONE_DATA
107                 exit 1
108             else
109                 print_result 0
110             fi
111         else
112             echo -n "$HOSTED_ZONE_TYPE HostedZone already exists
($HOSTED_ZONE_ID)"
113             print_result 0
114         fi
115
116         if [ "xnull" = "x$HOSTED_ZONE_ID" -o "x" = "x$HOSTED_ZONE_ID" ];
then
117             print_result 1
118             echo "Error obtaining HostedZone ID"
119             echo $HOSTED_ZONE_LIST
120             exit 1
121         fi
122
123         if [ "x$HOSTED_ZONE_TYPE" = "xpublic" ] ; then
124             echo -n "Creating NS entry for $HOSTED_ZONE_TYPE HostedZone"
125             setNameServers
126             if [ "x$NAMESERVERS_UPSERT_ID" = "x" -o
"x$NAMESERVERS_UPSERT_ID" = "xnull" ] ; then
127                 print_result 1
128                 echo "NS entry not created, maybe domain is not
registered in AWS Route 53"
129                 echo $NAMESERVERS_UPSERT
130             else
131                 print_result 0
132             fi
133         fi
134
135         #CREATING RECORDSET
136         echo -n "Creating $HOST_NAME$CLUSTER_NAME$DOMAIN"
137         if [ "x$HOSTED_ZONE_TYPE" = "xprivate" ] ; then
138             IP_HOST=$PRIVATE_IP_HOST
139         else
140             if [ "x$VPC_ID" != "x" -a "x$REGION" != "x" ] || [
"x$PRIVATE_HOSTED_ZONE_ID" != "x" ] ; then
141                 IP_HOST=$PUBLIC_IP_HOST
142             else
143                 IP_HOST=$PRIVATE_IP_HOST
144             fi
145         fi
146         RECORD_SETS_DATA=$(aws route53 change-resource-record-sets --
hosted-zone-id $HOSTED_ZONE_ID --change-batch  \
147             "{ \"Changes\": [
\
148                 {
\
149                     \"Action\":\"UPSERT\",
\
```

```
150                     \"ResourceRecordSet\": {
\
151                         \"Name\": \"$HOST_NAME$CLUSTER_NAME$DOMAIN\",
\
152                         \"Type\": \"A\",
\
153                         \"TTL\": 300,
\
154                         \"ResourceRecords\": [
\
155                             {
\
156                                 \"Value\" : \"$IP_HOST\"
\
157                             }
\
158                         ]
\
159                     }
\
160                 }
\
161             ] }" )
162         CHANGE_INFO_ID=$(echo $RECORD_SETS_DATA | jq -r .ChangeInfo.Id)
163         if [ "x" = "x$CHANGE_INFO_ID" ] ; then
164             print_result 1
165             echo "Error creating record set for this host"
166             echo $RECORD_SETS_DATA
167             exit 1
168         else
169             print_result 0
170         fi
171     fi
172 }
173
174 while getopts "hc:n:d:i:p:v:r:a:b:" opt ; do
175     case $opt in
176         h) echo $USAGE; examples; HELP="YES";;
177         c) CLUSTER_NAME=$OPTARG.;;
178         n) HOST_NAME=$OPTARG;;
179         i) PUBLIC_IP_HOST=$OPTARG;;
180         p) PRIVATE_IP_HOST=$OPTARG;;
181         d) DOMAIN=$OPTARG;;
182         v) VPC_ID=$OPTARG;;
183         r) REGION=$OPTARG;;
184         a) PUBLIC_HOSTED_ZONE_ID=$OPTARG;;
185         b) PRIVATE_HOSTED_ZONE_ID=$OPTARG;;
186     esac
187 done
188
189 #Obtaining hostname if it is not set
190 [ "x$HOST_NAME" == "x" ] && HOST_NAME=$(hostname -s)
191
192 #Obtaining Host IPs
193 if [ "x$PUBLIC_IP_HOST" == "x" ]; then
194     if [ -d /sys/class/net/bond0 ]; then
195         PUBLIC_IP_HOST=$(curl -s -S http://169.254.169.254/latest/meta-
data/public-ipv4)
196     fi
197 fi
198 if [ "x$PRIVATE_IP_HOST" == "x" ]; then
199     if [ -d /sys/class/net/bond1 ]; then
```

```
200          PRIVATE_IP_HOST=$(ip a s bond1 2>/dev/null |grep inet|grep
brd|awk '{print $2}'|head -n 1|tr '/' ' '|awk '{print $1}')
201      fi
202  fi
203
204  if [ "x" != "x$DOMAIN" -a "x" != "x$HOST_NAME" -a "x" !=
"x$PUBLIC_IP_HOST" -a "x$HELP" != "xYES" ] ; then
205
206      echo $HOST_NAME | grep -q "\.$"
207      [ $? -ne 0 ] && HOST_NAME="${HOST_NAME}."
208
209      HOSTED_ZONE_ID=""
210      configureHostedZone public $PUBLIC_HOSTED_ZONE_ID
211
212      if [ "x$PRIVATE_IP_HOST" != "x" ] ; then
213          if [ "x$REGION" != "x" -a "x$VPC_ID" != "x" ] || [
"x$PRIVATE_HOSTED_ZONE_ID" != "x" ] ; then
214              HOSTED_ZONE_ID=""
215              configureHostedZone private $PRIVATE_HOSTED_ZONE_ID
216          fi
217      fi
218
219  elif [ "$HELP" != "YES" ] ; then
220      echo "ERROR: Invalid Parameters"
221      echo $USAGE
222  fi
223
```

ANEXO VII - SERVICIO RB-SQSLD

rb_sqsld es un script en ruby que está configurado como servicio en redBorder. Se encarga de obtener los mensajes de la cola SQS asociada al clúster de redBorder.

En el caso de que sea un mensaje de los Lifecycle Hooks, el script se encarga de ejecutar rb_cluster_leave en las instancias para que terminen correctamente, y a continuación completa el ciclo de vida de la instancia.

Si es una alarma de Cloudwatch que avisa de que hay más MiddleManagers de los necesarios, se encarga de gestionar el terminado de los MiddleManagers, sin influir en las acciones de autoescalado del grupo.

Cabe destacar que el servicio es capaz de procesar mensajes de forma concurrente y que en caso de recibir señales SIGINT o SIGTERM, espera a la finalización de las operaciones de los hilos para terminar el programa.

Código

```
#!/usr/bin/ruby

require 'aws-sdk'
require 'yaml'
require 'time'

#SIGNAL HANDLING###################################################################
def terminate_script
  p "Waiting threads..."
  threads = $thread_group.list
  threads.each do |thread|
    thread.join
  end
  p "Threads finished"
  exit
end

#CLOUDWATCH ALARM HANDLING#########################################################
def alarm_handler(alarm_body, config, cloudwatch_alarm)
  p "Alarm notification detected"
  if alarm_body["AlarmName"] == config["MMDOWNALARMNAME"]
    p "Alarm name = #{alarm_body["AlarmName"]}"
    time_to_sleep =
2*Integer(alarm_body["Trigger"]["Period"])*Integer(alarm_body["Trigger"]["EvaluationPeriods"])
    $thread_group.add Thread.new {
    #Only one thread can manage this alarm, using semaphore
      if !$semaphore.locked?
        $semaphore.synchronize {
          if alarm_body["NewStateValue"] == "ALARM"
            p "NewStateValue = ALARM"
            while cloudwatch_alarm.state_value == "ALARM"
              p "Cloudwatch alarm is in ALARM state"
              instance_termination_handler(config["CDOMAIN"])
              p "Waiting #{time_to_sleep} seconds for possible Alarm change"
              sleep(time_to_sleep)
              p "Finish wait"
            end
          end
        }
      end
    }
  else
    p "Unknown alarm with name #{alarm_body["AlarmName"]}"
  end
end
#INSTANCE TERMINATION HANDLER#####################################################
```

```
def instance_termination_handler(cdomain)

  #Obtaining number of registered middleManagers
  numInstances = (`/opt/rb/bin/rb_get_druid_middleManagers.rb`).split.count
  instance = `/opt/rb/bin/rb_get_druid_middleManagers.rb -p`
  instance_id = instance.split('.')[0]
  minSize = Integer(`trap '' SIGINT && trap '' SIGTERM && \
            /opt/rb/bin/rb_manager_ssh.sh #{instance_id}.#{cdomain} \
            'source /opt/rb/etc/externals.conf && \
              aws autoscaling describe-auto-scaling-groups \
                --auto-scaling-group-names $AUTOSCALINGGROUPID | \
                  jq -r .AutoScalingGroups[0].MinSize' 2>&1`)
  if numInstances > minSize
    p "Instance #{instance_id} will be terminated"
      file = File.open("/var/log/sqsld/#{instance_id}", 'w')
      if !file.nil?
        file.sync = true
        p "Executing rb_cluster_leave in instance #{instance_id}"
        file.write("#{`trap '' SIGINT && trap '' SIGTERM && \
          /opt/rb/bin/rb_manager_ssh.sh #{instance_id}.#{cdomain} \
          '/opt/rb/bin/rb_cluster_leave.sh -f' 2>&1`}")
        p "Alarm processed"
      else
        p "Unable to create file /var/log/sqsld/#{instance_id}"
      end
  else
    p "Min size of autoscaling group reached, ignoring alarm"
  end
end

#TERMINATION LIFECYCLE HOOK HANDLING#############################################
def termination_lifecyclehook_handler(message_body, cdomain)
  if !message_body.has_key? "EC2InstanceId"
    p "Error: instance-id not found"
  else
    message_instance_id = message_body["EC2InstanceId"]
    p "Instance-id: #{message_instance_id}"
    #SSH Execution (must be multi-thread)
    $thread_group.add Thread.new {
      if !message_body.has_key? "LifecycleActionToken" or
          !message_body.has_key? "LifecycleHookName" or
          !message_body.has_key? "AutoScalingGroupName"
        p "Error: No enough parameters in message"
      else
        message_lifecycle_action_token = message_body["LifecycleActionToken"]
        message_lifecycle_hook_name = message_body["LifecycleHookName"]
        message_autoscaling_group_name = message_body["AutoScalingGroupName"]
        p "Processing message for instance #{message_instance_id}"
        file = File.open("/var/log/sqsld/#{message_instance_id}", 'w')
        file.write("#{`trap '' SIGINT && trap '' SIGTERM && \
          /opt/rb/bin/rb_manager_ssh.sh #{message_instance_id}.#{cdomain} \
          '/opt/rb/bin/rb_cluster_leave.sh -f' 2>&1`}")
        file.write("#{`trap '' SIGINT && trap '' SIGTERM && \
          /opt/rb/bin/rb_manager_ssh.sh #{message_instance_id}.#{cdomain} \
          'aws autoscaling complete-lifecycle-action \
            --lifecycle-action-token #{message_lifecycle_action_token} \
            --lifecycle-hook-name #{message_lifecycle_hook_name} \
            --auto-scaling-group-name #{message_autoscaling_group_name} \
            --lifecycle-action-result CONTINUE' 2>&1`}")
        file.close
        p "Message for instance #{message_instance_id} processed"
      end
    }
  end
end

################################################################################
# MAIN EXECUTION                                                               #
################################################################################


Signal.trap("TERM") { terminate_script }
Signal.trap("INT") { terminate_script }

#CONFIGURATION###################################################################
p "Reading externals.yml"
if !File.exists? "/opt/rb/etc/externals.yml"
  p "ERROR: /opt/rb/etc/externals.yml doesn't exists"
else
  config = YAML.load_file("/opt/rb/etc/externals.yml")
  if !config.has_key?("SQSQUEUEURL")
    p "ERROR: SQSQUEUEURL not found"
  elsif !config.has_key?("CDOMAIN")
```

```
    p "ERROR: CDOMAIN not found"
  elsif !config.has_key?("REGION")
    p "ERROR: REGION not found"
  else
#AWS CONFIGURATION
    AWS.config({:region => config["REGION"]})
#SQS CLIENT PREPARATION#####################################################
    sqs_queue_url = config["SQSQUEUEURL"]
    cdomain = config["CDOMAIN"]
    p "Creating SQS client"
    sqs   = AWS::SQS.new()
    queue = sqs.queues[sqs_queue_url]
#CLOUDWATCH CLIENT PREPARATION##############################################
    cloudwatch_alarm=nil
    if !config["MMDOWNALARMNAME"].nil? and !config["MMDOWNALARMNAME"].to_s.empty?
      p "Creating Cloudwatch client"
      cloudwatch = AWS::CloudWatch.new
      cloudwatch_alarm=cloudwatch.alarms[config["MMDOWNALARMNAME"]]
      p "Cloudwatch alarm ARN: #{cloudwatch_alarm.arn}"
    end
#MULTITHREAD PREPARATION####################################################
    $thread_group = ThreadGroup.new
    $semaphore = Mutex.new
#MESSAGE PROCESSING#########################################################
    p "Waiting for messages..."
    queue.poll do |msg|
      p "Got message"
      p "#{msg.body}"
      begin
        message_body = JSON.parse(msg.body)
        #First check if is a Cloudwatch alarm
        if message_body.has_key? "Message" and !cloudwatch_alarm.nil?
          alarm_body = JSON.parse(message_body["Message"])
          p message_body["Message"]
        #First check if is a Cloudwatch alarm
          if alarm_body.has_key? "AlarmName"
              alarm_handler(alarm_body, config, cloudwatch_alarm)
          end
        #If is not an alarm, check if is a Service (like autoscaling)
        elsif !message_body.has_key? "Service"
          p "Error: Service property not found"
        else
          message_service = message_body["Service"]
          p "Service #{message_service}"
          if message_body["Service"] == "AWS Auto Scaling"
                termination_lifecyclehook_handler(message_body, cdomain)
          else
            p "Unknown service"
          end
        end
      rescue JSON::ParserError
        p "JSON parse error"
      end
    end
  end
end
```

ANEXO VIII - SCRIPT RB_SET_AWS_INTERFACE

Este script crea una Elastic Network Interface de Amazon utilizando la CLI de AWS, y la asigna a la propia instancia, para poder así tener acceso a la red de sincronismo. Además, modifica las propiedades de la interfaz para que en caso de que la instancia sea eliminada, automáticamente se elimine la interfaz de red también. El código del script es el siguiente:

rb_set_aws_interface.sh

```
1    #!/bin/bash
2
3    source /opt/rb/bin/rb_manager_functions.sh
4    source /etc/profile
5
6    #
7    #   This script creates and attaches an Elastic Network Interface (ENI)
in AWS VPC. In addition
8    #   the script modifies the ENI properties for delete on termination.
9    #
10   if [ ! -d /sys/class/net/eth1 ] && [ "x$SUBNET_ID" != "x" ] ; then
11
12       cat > /etc/sysconfig/network-scripts/ifcfg-bond1 <<- _RBEOF2_
13   DEVICE=bond1
14   ONBOOT=yes
15   BOOTPROTO=dhcp
16   BONDING_OPTS='mode=1 miimon=100'
17   PEERDNS=no
18   _RBEOF2_
19
20       cat > /etc/sysconfig/network-scripts/ifcfg-eth1 <<- _RBEOF2_
21   DEVICE=eth1
22   NM_CONTROLLED=no
23   ONBOOT=yes
24   BOOTPROTO=none
25   MASTER=bond1
26   SLAVE=yes
27   _RBEOF2_
28
29       echo -n "Creating Elastic Network Interface (AWS)"
30       NetworkInterfaceId=$( /opt/rb/bin/aws ec2 create-network-interface -
-subnet-id $SUBNET_ID --description CF_PRIVATE_SUBNET_INTERFACE --region
$REGION | /opt/rb/bin/jq .NetworkInterface.NetworkInterfaceId -r ) ;
31       if [ "x$NetworkInterfaceId" = "x" ]; then
32           print_result 1
33       else
34           print_result 0
35           echo -n "Attaching Elastic Network Interface (AWS) to instance"
36           InterfaceAttachmentId=$( /opt/rb/bin/aws ec2 attach-network-
interface --network-interface-id $NetworkInterfaceId --instance-id $(wget -q
-O - http://169.254.169.254/latest/meta-data/instance-id) --device-index 1 --
region $REGION | /opt/rb/bin/jq .AttachmentId -r ) ;
```

```
37
38          if [ "x$InterfaceAttachmentId" = "x" ]; then
39              print_result 1
40          else
41              print_result 0
42
43              echo -n "Modifying Elastic Network Interface (AWS)
attributes"
44              /opt/rb/bin/aws ec2 modify-network-interface-attribute --
network-interface-id $NetworkInterfaceId --attachment
AttachmentId=$InterfaceAttachmentId,DeleteOnTermination=true --region $REGION
;
45              print_result $?
46
47              counter=0;
48              while [ ! -f /sys/class/net/bond1/address -a $counter -lt 60
]; do
49                  sleep 1
50                  counter=$(($counter +1));
51              done
52
53              ifup eth1 &>/dev/null
54              ifup bond1 &>/dev/null
55
56              counter=0
57              while [ $counter -lt 60 ]; do
58                  ipbond1=$(ip a s bond1 2>/dev/null|grep inet|grep
bond1|awk '{print $2}')
59                  if [ "x$ipbond1" == "x" ]; then
60                      echo "INFO: IP on bond1 is not ready yet
($counter/60)"
61                      sleep 1
62                      counter=$(($counter +1))
63                  else
64                      counter=1000
65                  fi
66              done
67      fi
68      fi
69  fi
70
71
```

REFERENCIAS

[1] National Institute of Standards and Technology (EEUU), «The NIST Definition of Cloud Computing» [En línea]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[2] J. Bort, «Business Insider» 2015. [En línea]. Available: http://www.businessinsider.com/synergy-research-amazon-dominates-16-billion-cloud-market-2015-2.

[3] Openstack, «Documentación de Openstack» 2015. [En línea]. Available: http://docs.openstack.org/.

[4] Apache Foundation, "Documentación de Apache Zookeeper" 2015. [Online]. Available: https://zookeeper.apache.org/doc/trunk/.

[5] Apache Foundation, "Documentación de Apache Kafka" 2015. [Online]. Available: http://kafka.apache.org/documentation.html.

[6] Druid, «Documentación de Druid» 2015. [En línea]. Available: http://druid.io/docs/latest/design/index.html.

[7] Apache Foundation, «Documentación de Apache Samza» 2015. [En línea]. Available: https://samza.apache.org/learn/documentation/0.7.0/.

[8] Apache Foundation, «Documentación de Apache Hadoop» 2015. [En línea]. Available: https://hadoop.apache.org/docs/r2.6.0/.

[9] Basho, «Documentación de Riak» 2015. [En línea]. Available: http://docs.basho.com/.

[10] PostgreSQL Global Development Group, «Documentación de PostgreSQL» 2015. [En línea]. Available: http://www.postgresql.org/docs/.

[11] Dormando, «Documentación de Memcached» 2015. [En línea]. Available: http://memcached.org/.

[12] Opscode, «Documentación de Chef» 2015. [En línea]. Available: http://docs.chef.io/.

[13] Read the docs, "Documentación de Cloud-Init" 2015. [Online]. Available: https://cloudinit.readthedocs.org/en/latest/.

[14] Amazon Web Services, "Documentación de Amazon Elastic Compute Cloud (EC2)" 2015. [Online]. Available: https://aws.amazon.com/es/documentation/ec2/.

[15] Amazon Web Services, "Documentación de Amazon Autoscaling" 2015. [Online]. Available: https://aws.amazon.com/es/documentation/autoscaling/.

[16] Amazon Web Services, "Documentación de Amazon Virtual Private Cloud (VPC)" 2015. [Online]. Available: https://aws.amazon.com/es/documentation/vpc/.

[17] Amazon Web Services, “Documentación de Amazon Cloudwatch” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/cloudwatch/.

[18] Amazon Web Services, “Documentación de Amazon Simple Notification Service (SNS)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/sns/.

[19] Amazon Web Services, “Documentación de Amazon Identity and Access Management (IAM)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/iam/.

[20] Amazon Web Services, “Documentación de Amazon Cloudformation” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/cloudformation/.

[21] Amazon Web Services, “Documentación de Amazon Lambda” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/lambda/.

[22] Amazon Web Services, “Documentación de Amazon Simple Storage Service (S3)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/s3/.

[23] Amazon Web Services, “Documentación de Amazon Elasticache” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/elasticache/.

[24] Amazon Web Services, “Documentación de Amazon Relational Database Service (RDS)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/rds/.

[25] Amazon Web Services, “Documentación de Amazon Route 53” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/route53/.

[26] Amazon Web Services, “Documentación de Amazon Elastic Mapreduce (EMR)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/elastic-mapreduce/.

[27] Amazon Web Services, “Documentación de Amazon Kinesis” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/kinesis/.

[28] Amazon Web Services, “Documentación de Amazon Simple Queue Service (SQS)” 2015. [Online]. Available: https://aws.amazon.com/es/documentation/sqs/.

[29] B. Golden, Amazon Web Services for dummies, John Wiley & Sons, Inc, 2013.

[30] P. Kuppusamy and U. Vyas, “AWS development essentials: design and build flexible, highly scalable, and cost-effective applications using Amazon Web Services” Safari Books Online, 2014.

[31] T. Rosner, “Learning AWS OpsWorks: learn how to exploit advanced technologies to deploy and auto-scale web stacks” Packt Pub., 2013.

[32] N. Antonopoulos and L. Gillam, Cloud Computing: Principles, Systems and Applications, Springer, 2010.

[33] Canonical, “Documentación de Ubuntu: Cloud-init” 2014. [Online]. Available: https://help.ubuntu.com/community/CloudInit.

[34] Amazon Web Services, “Documentación de Amazon Marketplace” 2015. [Online]. Available: http://docs.aws.amazon.com/marketplace/latest/controlling-access/what-is-marketplace.html.

[35] Amazon Web Services, «Documentación de Amazon Beanstalk» 2015. [En línea]. Available: https://aws.amazon.com/es/documentation/elastic-beanstalk/.

[36] Google, «Documentación de Google App Engine» 2015. [En línea]. Available: https://cloud.google.com/appengine/docs.

[37] Red Hat Enterprise Linux, «Documentación de Openshift Origin» 2015. [En línea]. Available: https://docs.openshift.org/.

[38] M. Delgado, «Blog de Manuel Delgado» 2015. [En línea]. Available: http://manueldelgado.com/que-es-el-analisis-del-sentimiento/.

[39] redBorder, «Web de redBorder» 2015. [En línea]. Available: http://redborder.net/.

[40] Jenkins, «Documentación de Jenkins» 2015. [En línea]. Available: https://jenkins-ci.org/.

[41] Nginx, «Documentación de Nginx» 2015. [En línea]. Available: http://nginx.org/en/docs/.

[42] E. Phoenix, «Web del servidor Puma» 2013. [En línea]. Available: http://puma.io/.

[43] Ruby on Rails, «Documentación de Ruby on Rails» 2015. [En línea]. Available: http://rubyonrails.org/documentation/.

[44] C. C. Evans, «Web de YAML» [En línea]. Available: http://yaml.org/.

[45] Amazon Web Services, «Documentación de Amazon Elastic Load Balancer» 2015. [En línea]. Available: https://aws.amazon.com/es/documentation/elastic-load-balancing/.

[46] Amazon Web Services, «Documentación de Amazon DynamoDB» 2015. [En línea]. Available: https://aws.amazon.com/es/documentation/dynamodb/.

[47] A. G. Ferrer, «Análisis y enriquecimiento, en tiempo real, de flujos masivos de datos utilizando Apache Storm» Universidad de Sevilla, Sevilla, 2014.

[48] New Relic, «Documentación de New Relic» 2015. [En línea]. Available: https://docs.newrelic.com/.