Facultad de Ingeniería

TRABAJO FINAL DE GRADO

EPA
EVENTS PIPELINES ARCHITECTURE

Apellido: Prósperi

Nombres: Nicolás Alfredo

Legajo: 0113381

Teléfono: 11-2255-5644

e-mail: nicolasprosperi@gmail.com

Carrera: Licenciatura en tecnología de la Información

Turno: Noche

Docente: Diego Esteve

Fecha de entrega: 23/11/2021

Proyecto: EPA - Events Pipelines Architecture

N.A. Prósperi Devops Manager, Buenos Aires, Argentina

Abstract

El objetivo principal del proyecto es ofrecer una solución modularizada, estandarizada y securizada que organiza los principales servicios de una plataforma Cloud en una arquitectura orientada a eventos.
Con el fin de construir este Stack tecnológico se crea un flujo de trabajo que inicia con la automatización por código de la infraestructura, para luego seguir con la automatización del despliegue del software base.
El resultado se puede ver como dos productos en uno, una arquitectura funcional que permita el ingreso de datos , procesamiento y posterior almacenamiento de información.
Yel segundo un template de desarrollo sobre una plataforma de integración y despliegue continuo.

Palabras Clave:

Amazon Web Services, Devops, Despliegue continuo, DevSecOps, Integración Continua, Microservicios, Serverless.

Nomenclatura:

SAS - Software como servicio.
PAS - Plataforma como servicio.
AWS - Amazon Web Services, plataforma cloud utilizada para construir la arquitectura.
CloudFront - Servicio de AWS para publicar contenido estático en internet.
S3 - Servicio de almacenamiento en AWS.
EC2 - Servicio de máquinas virtuales.

1 Introducción

El siguiente documento describe el proceso de investigación y desarrollo del proyecto “Event Oriented Architecture Pipelines” sponsoreado por la empresa NAP Technology. N. A. Prosperi Arquitecto de Soluciones en NAP Technology, C.A.B.A., Argentina (nicolasprosperi@gmail.com)

En el mundo de los negocios, cada vez se escucha mas fuerte el concepto de “Transformación Digital” Y sin duda en el contexto mundial actual (Pandemia y pos pandemia Covid-19) el marketing y las empresas tuvieron que adaptarse rápidamente para poder seguir funcionando.
De este cambio surge la digitalización de muchos procesos que antes eran presenciales, como por ejemplo hacer una compra de supermercado, compra en una farmacia, pedir comida, realizar todo tipo de trámites, movimientos bancarios, etc. Y por supuesto el surgimiento del eCommerce más fuerte que nunca desplazando a los centros comerciales del foco principal del comprador.
No es novedad que los eventos tan excepcionales como las guerras, o en este caso una Pandemia , aceleran los descubrimientos tecnológicos como el caso de las vacunas, o generan grandes cambios como el caso del eComerce, Marketing Digital y empresas Digitales.
En este proyecto vamos a abordar un proceso digital que inicia con un ingreso de datos, pasa por un procesamiento de los mismos y termina generando información, que al final, es lo que siempre buscamos, dar un valor a los datos convirtiéndolos en información.
El Worflow generado podrá utilizarse como marco de trabajo o “Framework” para un sinnúmero de procesos. Pero nos vamos a enfocar en un ejemplo actual que es el de un formulario para anotarse en una vacunación de covid, el cual procesa la información mediante validaciones y servicios externos, obteniendo al final un turno presencial.

2 Análisis de Mercado

Para la mayoría de los requerimientos de software el mercado ofrece una amplia gama de servicios SAS, como por ejemplo los CRM, CMS, dentro de los que se encuentra Saleforce, SAP, Microsoft Dynamics, e incluso AWS con soluciones para salud, marketing, etc.

Pero nos vamos a centrar en las soluciones que aceleran el desarrollo a medida que pueden mejorar los tiempos de una software factory.

Dentro de los cuales Redhat se encuentra dentro de los mejores en el mercado, ofreciendo la plataforma Openshift Runtimes, que básicamente nos permite elegir qué lenguaje de código vamos a utilizar, el caso de uso, y con esta información se genera un proceso de CICD completo que acelera el time to market de la solución.

Pero como desventaja tenemos el vendor lock-in, osea esta solución solo aplica a la tecnología Openshift, que aún está lejos de ser líder en el mercado de los Cloud Providers, lo que implica un menor uso de la plataforma comparado con la elegida AWS.

Y por otro lado, esta plataforma posee un sistema de pago anticipado por uso estimado de procesamiento, dejando la solución muy por encima del costo en AWS que para ambientes de desarrollo apenas puede superar la capa gratuita.

El proyecto no viene a competir con las soluciones SAS, sino a estandarizar tecnologías de desarrollo, integración continua y delivery continuo dentro de una Software Factory, buscando siempre el menor costo y la mayor performance en los equipos de trabajo.

2.1 Fortalezas de EPA

Para desarrollar la arquitectura y el proceso automatizado nos basamos en las últimas tecnologías que se están utilizando en el mercado (Microservicios Serverless).
Esta tecnología se está utilizando en los principales servicios de streaming, en donde se debe tener arquitecturas escalables, que se adapten a cualquier nuevo requerimiento de los usuarios.
Principalmente estamos ahorrando tiempo de investigación, desarrollo y configuración en el kickoff de un nuevo proyecto de software.

2.2 Oportunidad en el mercado actual

Creemos que en este último tiempo y sumado a la Pandemia Covid-19 el mercado genera una oportunidad de ofrecer productos y servicios al mundo, sin tener que tener un lugar de atención al cliente físico. Si existía alguna duda que el futuro iba por ese camino, quedó demostrado que se puede, pero la competencia es grande, y se debe llegar lo más rápido posible a una solución que guste, y que además pueda convivir con el cambio constante.

Por esto creemos que este tipo de soluciones de arquitecturas flexibles, escalables, automatizadas y seguras, son ideales para el mercado que se viene.

2.3 Debilidad

Uno de los puntos más importantes y tal vez más difíciles es la complejidad de la arquitectura ofrecida, que aunque esté resuelta, depende mucho del equipo de desarrollo y devops para su implementación y puesta a punto en un entorno productivo.
Por esto ofrecemos con el producto una capacitación en las herramientas que se utilizaron para desarrollarlo. La idea es terminar la implementación en un ambiente de desarrollo / testing pasando por todos los servicios y herramientas utilizadas, intentando mitigar la posible falta de conocimiento en el equipo que a futuro tendrá el mantenimiento y evolución del producto.

2.4 Amenazas

En el mercado la competencia se da por los servicios de consultoría, que generalmente se abocan a diseñar una arquitectura a partir de un requerimiento, llegan a ser costos, y además tiene un tiempo de implementación más largo que la plataforma Template que ofrecemos, pero son una opción viable e incluso pueden ofrecer fuerza de desarrollo con la experiencia sobre la tecnología elegida.
Y otra potencial amenaza son las soluciones SAS que a pesar de ser costosas generalmente su implementación es muy rápida, pero la empresa que contrata el servicio genera una dependencia muy grande con el oferente, y esto suele ser un problema cuando se quiere cambiar de producto o ir a un desarrollo propio.

2.5 Cliente potencial

El desarrollo está orientado a una software factory, área de sistemas en gobierno o cualquier empresa que disponga de un equipo de desarrollo para sus productos.
Ofrecemos un template que se adapta casi a cualquier requerimiento del mercado y además de la implementación disponemos de un equipo de profesionales que podrá capacitar al equipo interno del cliente.
Argentina posee algo más de 600 empresas que desarrollan software a medida registradas en la Cámara de la Industria Argentina de Software.
Con un crecimiento constante, el sector de IT requiere constantemente nuevos talentos, y lo que ofrecemos es un “acelerador” de proyectos, para que nuevos equipos puedan pasar la curva de aprendizaje los más rápido posible, y puedan tener un producto funcionando con las últimas tecnologías en tiempo record.

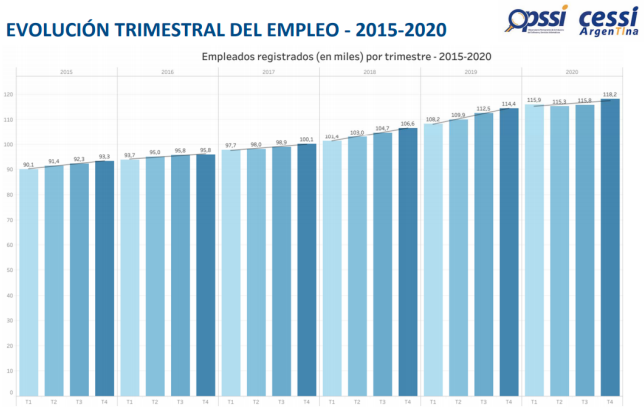


Figure 1: Fuente: Boletín de Empleo Registrado - Observatorio de Empleo y Dinámica Empresarial - para el area de IT 2021

3 Objetivo

El proyecto consta de un objetivo principal que es el de proveer una marco de trabajo que permita acortar considerablemente el tiempo de StartUP de un proyecto de desarrollo de software.
La idea consiste en proveer toda las herramientas necesarias para que un equipo de desarrolladores tanto FrontEnd como Backend, dispongan de una arquitectura funcional sobre una plataforma Cloud elegida.

3.1 Soportar Eventos

Se orienta la solución de arquitectura a soportar cualquier cantidad de datos de entrada utilizando un proceso de producción y encolado de mensajes.
Los mismos permiten encapsular las distintas etapas del proceso en arquitecturas separadas, ya que el proceso de ingesta de datos y generación de mensajes es sincrónico, pero el resto del flujo de trabajo se puede procesar de forma asincrónica.

3.2 Escalar cuando sea necesario

Se busca que toda la arquitectura sea escalable, desde el ingreso de datos por FrontEnd hasta el procesamiento de los mismos BackEnd.
La arquitectura propuesta tiene como objetivo poder escalar en todas sus partes y especialmente poder reducir su tamaño a la mínima expresión cuando la misma se encuentra en reposo.

3.3 Calidad y Seguridad

Como todo proceso de desarrollo este debe ser auditado, tanto para asegurar su calidad como la seguridad que ofrece ante amenazas.
Como objetivo del proyecto se encuentra el chequeo constante del código a desplegar dentro de la arquitectura propuesta.
El mismo deberá pasar por una serie de controles que aseguren su calidad, seguridad y uso de buenas prácticas, y que en su defecto informen y detengan el proceso hasta solucionar los errores.

3.4 Mantener bajo los costos de operación

No menos importante que los objetivos anteriores, el lado económico debe estar presente en todas las etapas del proyecto, y por eso se busca que la arquitectura pueda funcionar ,en su mayoría dentro de la capa “Free Tier” (Capa gratuita).
Teniendo en cuenta que al menos sin estresar la solución, la misma debería poder funcionar con el mínimo de recursos provistos por la plataforma Cloud, que sería el equivalente a lo necesario para tener un ambiente de “Desarrollo”.
Para luego poder crecer tanto en nodos de la arquitectura como en costos al momento de realizar alguna prueba ya en un ambiente de Testing.

4 EPA

EPA **(Event Pipelines Architecture)** nace como un solución de arquitectura de desarrollo orientada a tratar eventos de forma asincrónica y escalable.
Pero para que esta solución se pueda implementar lo más rápido y económicamente posible lo que se propone es automatizar la creación y despliegue de la misma.
Para esto se eligió la plataforma Cloud número uno en el mundo , Amazon Web Services.
Dentro de los servicios que ofrece la plataforma se eligieron los que permiten alta disponibilidad, escalabilidad automática y automatización mediante scripts.
La idea que se plantea es tener una serie de repositorios, scripts y piezas de desarrollo que desplieguen un ejemplo funcionando de esta arquitectura.
Como ejemplo funcional elegimos un proceso que la mayoría de los países del mundo tuvo que proporcionar a sus ciudadanos: El formulario de inscripción para acceder a las vacunas del Covid-19.

4.1 Flujo Funcional

4.1.1 Frontend

Formulario frontend generico donde se cargan los datos del ciudadano con validaciones que se realizan “browser Side”. El mismo genera el contenido en formato Json y es enviado mediante un POST al primer servicio lambda que interviene en el proceso (Producer).

4.1.2 Producer

Es el encargado de dispararse al recibir el POST del contenido en formato Json , para luego “producir” un mensaje que es encolado en el broker SQS.

4.1.3 Broker SQS

Este es el servicio que se encarga de romper la sincronía en nuestro proceso, ya que los mensajes se encolan, devolviendo un ACK al servicio Lambda sin importar si los mismos fueron o serán procesados.

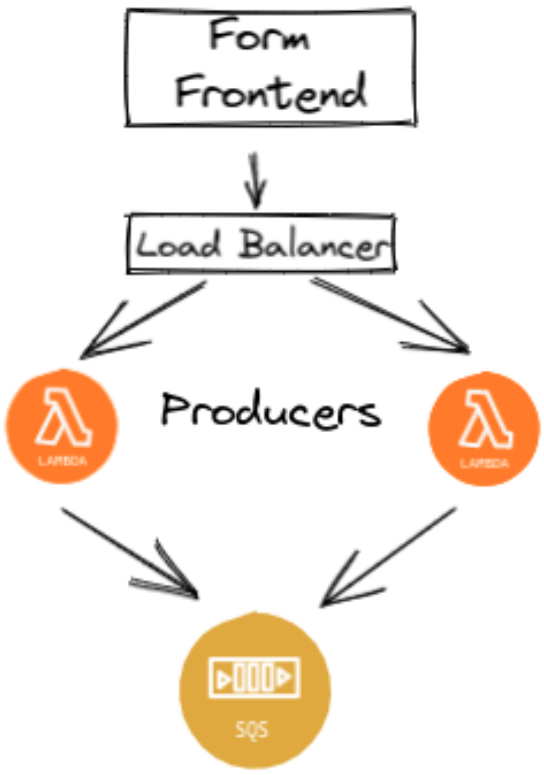


Figure 2: Etapa de producción de mensajes en SQS.

4.1.4 Consumer

Servicio que se encarga de ”vigilar” la cola de mensajes del servicio SQS, y al ver que existe un mensaje desencolarlo. Esta acción dispara un proceso de validación que para el ejemplo sería consultar a una API “externa” que tipo de vacuna le toca al paciente.

4.1.5 External API

Será un MOCK, osea un servicio falso que devuelva de forma aleatoria una respuesta que nos sirva para chequear el proceso, y que en la realidad debiera ser reemplazado.

4.1.6 Data Managment

Como casi cualquier aplicación, tenemos un almacenamiento de datos estructurado que nos permitirá posteriormente explotar estos datos.

4.1.7 Notification

Y por último mediante el servicio SNS se realizará la notificación por correo de la finalización del proceso y vacuna disponible.

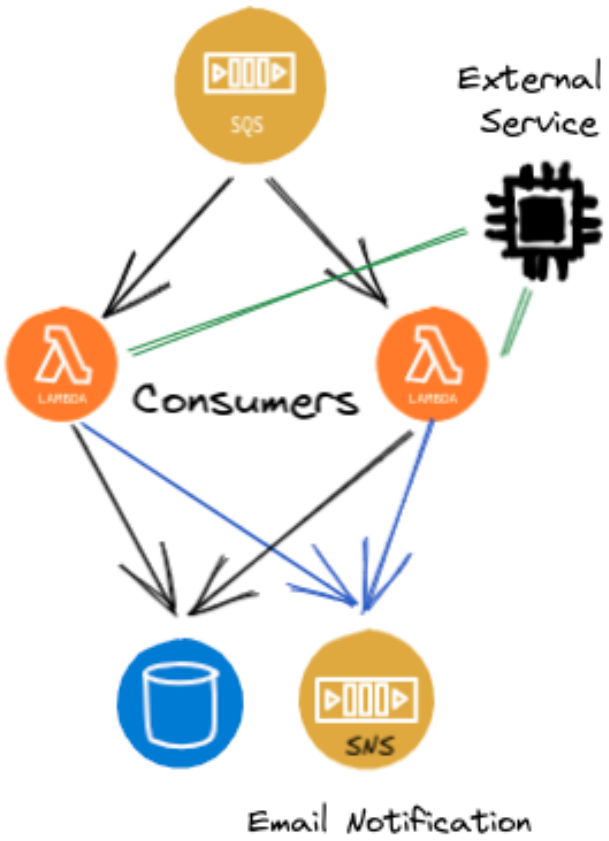


Figure 3: Procesado asíncrono de información, consumo de servicio externo y posterior almacenado y notificación vía email

4.2 Servicios AWS Implementados

Lista de serivicios implementados dentro de la arquitectura propuestam con sus ventajas y funcionalidad y SLA correspondiente

4.2.1 Lambda

El servicio Lambda se utiliza para ejecutar nuestro código escrito en la mayoría de los lenguajes actuales, y disparado a partir de un evento determinado.
La principal ventaja de este servicio es que se paga solo cuando se utiliza, ya que al no disponer de una instancia, máquina virtual, o web server "prendido" solo nos importa el momento y duración y consumo de recursos que hace nuestro código al ejecutarse.
El costo de tener este servicio disponible es cero, y además se encuentra dentro de la capa gratuita de AWS que ofrece 1 millón de solicitudes mensuales a costo cero.
Este servicio deberá interactuar tanto con el Frontend como con los broker de mensajes que encolen las peticiones o disparen los eventos.
Y por último elegimos el servicio S3 para la automatización del despliegue del código en el servicio lambda.

¿Porque lo elegimos?

No requerimos administración de infraestructura o sistema operativo.
Pagamos solo cuando se usa.
AWS se encarga de escalar la infraestructura que corre nuestro código en tiempo de ejecución ofreciendo un SLA del 99.95El SDK permite desarrollar en Java, GO, PowerShell, Node Js, C, Python y Ruby, permitiendo que casi cualquier desarrollador pueda trabajar sin problemas.

4.2.2 S3

Amazon Simple Storage Service, es uno de los servicios más utilizados en la plataforma, proporciona una api tanto para el upload, download, o modificación de los archivos almacenados o a almacenar.
Es otro servicio SAS , osea que no debemos administrar la infraestructura que soporta este storage, y nos permite generar ciclos de vida para ir moviendo nuestros archivos a distintos tipos de almacenamiento para mejorar performance o reducir costos.
Utilizamos este servicio para al menos 3 funciones principales: Almacenar el contenido estático del Frontend, proporcionar el código fuente que luego se convertirá en una función lambda y por último almacenar el estado actual de nuestra infraestructura (Tema que ampliaremos en la sección Terraform de este documento. Nos asegura una infraestructura en alta disponibilidad, multi región, securizada.
Dice ofrecer una durabilidad de los datos del 99,99999999999 (11 nueves).
Permite elegir cuánto queremos pagar por nuestro almacenamiento (tipos de hardware).
Posee versionado sobre los archivos cualquiera sea su tipo.

4.2.3 SQS

Amazon Simple Queue Service, es un servicio de colas de mensajes que se ofrece en modo SAS, el mismo permite integrarse con el servicio Lambda lo que a diferencia de los brokers clásicos como Activemq, este dispara el proceso Lambda y no espera que la función lambda lo esté consultando.
Además el tráfico entre este servicio y sus integraciones es cifrado por AWS.

¿Porque lo elegimos?

No requerimos administrar infraestructura. El servicio escala automáticamente.
Se paga solo por uso, osea que cuando no existen mensajes el servicio no está activo.
Dentro de la capa gratuita tenemos 1 millón de mensajes por mes.

4.2.4 SNS

Amazon Simple Notification Service, viene a reemplazar los servicios tradicionales como Activemq Topics o Rabbit MQ, para proporcionar un servicio SAS de notificación, en diferentes formatos para quien este suscripto al Tópico.
Para nuestra solución solo lo utilizaremos para notificar mediante correo electrónico a los usuarios, pero podria tambien notificar a servicios Lambda, o a Apis o webservices externos.

¿Porque lo elegimos?

No requerimos administrar infraestructura Posee su propio servicio SMTP con reintentos de entrega y cola de mensajes fallidos.
Escala automáticamente.

4.2.5 AURORA MYSQL

Aurora es uno de los servicios más importantes de AWS, ya que es una base de datos que se ofrece como SAS compatible con Mysql y PostgreSQL.
La misma cuenta con nodos de escritura y lectura en alta disponibilidad distribuidos en al menos dos zonas de disponibilidad dentro de la misma región o en distintas regiones.
Cuenta además con su propio sistema de backup.
Dentro de nuestra solución será el encargado de almacenar todos los datos del ciudadano y la selección de vacunas que fue otorgada.

¿Porque lo elegimos?

No requerimos administrar la infraestructura. Geográficamente distribuida, osea que reduce casi a cero el riesgo de pérdida de datos por desastres.
Alta disponibilidad automática.
Escalabilidad controlada.
Nodos dedicados a lectura para explotación de datos que no afecta el nodo master.
Aunque no se encuentra dentro de la capa gratuita de la plataforma, su costo es significativamente menor al servicio administrado RDS.

5 Posicionamiento de tecnologías

Nos centramos en los 3 productos más importantes del mercado para resolver los tres pilares más importantes del proyecto: Proveedor Cloud AWS, Orquestador de automatizaciones Jenkins, y el software para desarrollar infraestructura como código Terraform y por ultimo SonarQube para el análisis de código estático, tanto calidad de código como seguridad.

5.1 AWS Cloud Provider

Dado que proponemos una solución escalable, en alta disponibilidad con la posibilidad de implementarse geográficamente distribuida, elegimos como Cloud Provider al líder actual según el Cuadrante de Gartner.
Y las razones principales son: la adopción en el mercado, osea la gran cantidad de devops, desarrolladores, y dbas que se especializan y lo seguirán haciendo sobre esta plataforma.
La capa gratuita que permite lanzar una demo del producto prácticamente a costo cero.
Y por último la gran cantidad de servicios ofrecidos en modo SAS o sea sin necesidad de administración de infraestructura.

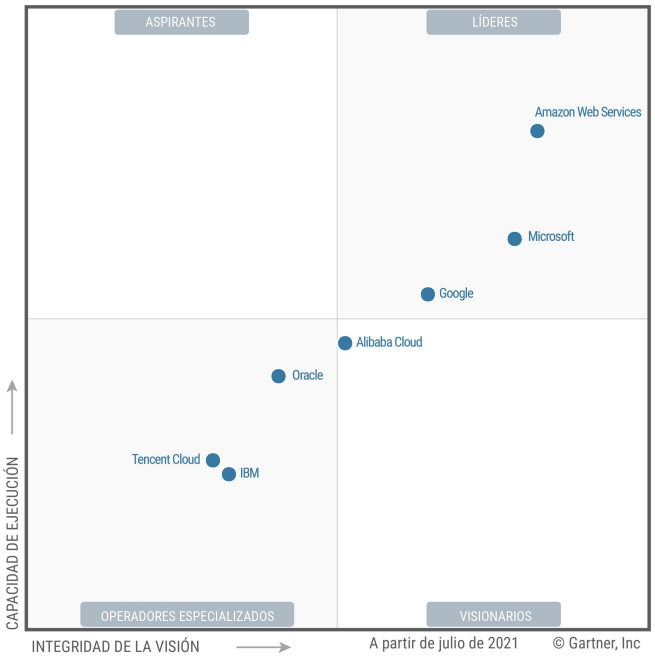


Figure 4: Cuadrante Mágico de Gartner para plataformas Cloud 2021.

5.2 Jenkins

Jenkins se encuentra liderando el mercado junto como TeamCity y Bamboo, pero con la ventaja de ser open-source totalmente gratuito, y además de tener un gran abanico de plugins desarrollados por la comunidad ofrecidos de la misma forma.

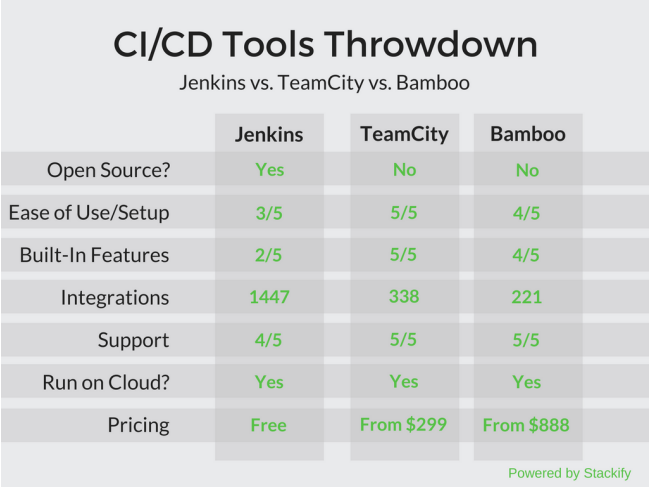


Figure 5: Comparación entre los tres principales software de CI/CD.

5.3 Terraform

Terraform nos permite codear toda la infraestructura que vamos a utilizar en AWS y en los últimos años se volvió el más popular pudiendo implementar mediante distintos "providers" infraestructura en AWS, Google Cloud Platform, Azure entre otros Cloud Providers.
Y no solo automatiza la infraestructura sino que además guarda un versión de la misma en AWS de forma casi automática.
Por todo esto se encuentra primero en los top mundiales de herramientas para automatización de infraestructura, incluso para la automatización en AWS superando ampliamente al sistema nativo AWS CloudFormation.

5.4 SonarQube

SonarQube, es una de las plataformas más conocidas para evaluar calidad de código y bugs de seguridad. Es ampliamente reconocida, open source y fue elegida por tener una integración nativa con Jenkins, lo cual hace muy simple su uso y visualización dentro de la misma plataforma.
Sus rivales son en general pagos con lo cual se posiciona en el mercado como una de las herramientas ideales para empezar a utilizar en nuestro flujo de Integración Continua.

6 Como ofrecemos el Producto

EPA no solo es una "Template" de arquitectura, sino que es un servicio de consultoría adaptado a un cliente que posea un equipo de desarrollo con poca o nula experiencia en tecnologías de microservicios y plataformas cloud.

6.1 Puesta a punto del producto

El producto se entrega funcionando, y para esto se requiere tener un usuario con suficientes privilegios en la cuenta de AWS del cliente.
Teniendo esta información, se realiza el despliegue de una instancia Ec2 que posee la plataforma CI/CD que automatiza el despliegue de la infraestructura y código.
Además se realiza la entrega de todo el código , tanto de infraestructura como de software en repositorios del tipo GIT.
Por último se realiza una POC que incluye el despliegue automático de la infraestructura, el código y una prueba completa del caso de uso ejemplo.

6.2 Capacitación

De forma opcional , se ofrecen 20 horas de capacitación que recorren todos los servicios utilizados e incluyen pruebas de concepto sobre la plataforma instalada.
Aunque es un servicio opcional, creemos que suma mucho conocimiento tanto al equipo de Devops como al de soporte del cliente.

6.3 Soporte y actualizaciones

Como un opcional, poseemos un equipo de Devops que puede dar soporte sobre la plataforma, ante cualquier eventualidad.
Este soporte general puede ser un cargo mensual, o se pueden contratar un mínimo de 4 horas por una eventualidad.

7 Retorno de la inversion

El costo del producto y servicio estaba basado en horas profesionales de tres perfiles diferentes. Arquitecto de soluciones AWS: desarrollo de la arquitectura de solución, elección de los servicios aws, y parametrización de los mismos teniendo en cuenta costos , performance y disponibilidad.
Ingeniero Devops: Desarrollo de la solución CI/CD y automatización de infraestructura como código. Desarrollador Full Stack: Desarrollo del código template para los microservicios Lambda, Frontend de prueba y Mock de servicio externo.
Estimamos que la inversión permite desplegar un producto que llevaría de 3 a 6 meses en solo 2 semanas, con lo cual el gasto inicial equivale al gasto que tendría finalmente la empresa para desarrollar de 0 el producto en

ese tiempo, llegando mucho más rápido al mercado con un producto probado y seguro.

7.1 Opciones de implementación

Los siguientes son costos de implementación y opciones que se pueden sumar al producto según lo requiera el cliente.

7.1.1 Despliegue de arquitectura

Consiste en el despliegue de la solución en la cuenta del cliente, para los ambientes de Desarrollo, Testing y Pre Producción. La modalidad tiene un costo de USD 18000.

7.1.2 Capacitación

La misma consiste en dos semanas de capacitación que equivalen a 20hs abarcando todos los servicios de AWS utilizados como toda la implementación CICD. Tiene un costo de USD 2000.

7.1.3 Soporte

Si el cliente no posee un equipo interno de devops con conocimiento en AWS, se puede optar por un servicio de horas de soporte, las mismas se contratan al momento de necesitarlas y tienen un costo aproximado de USD 50 la hora.

8 Conclusion

Mercado: Para poder competir en el mercado actual se requiere una agilidad en desarrollo que solo se puede lograr con arquitecturas orientadas a microservicios, en donde se tenga la suficiente velocidad y adaptabilidad que permita lanzar nuevas features constantemente.

Cliente: El proyecto propone un startup de arquitectura que permita a un equipo de desarrollo con poco o nulo conocimiento en microservicios Cloud iniciar un proyecto.

Competencia: Aunque existen aceleradores de proyectos en las principales plataformas cloud, son costosos y no ofrecen servicios de consultoría y capacitación.

Ventajas: En conclusión las ventajas que propone el proyecto son: Llegar antes al mercado con una arquitectura utilizada por las principales plataformas digitales. Implementar un circuito de integración y despliegue continuo con análisis de calidad de código y seguridad. Y transferir el know how de esta solución con capacitaciones y un ejemplo concreto de uso.

9 Futuras lineas de desarrollo

La siguiente es una lista de productos en proceso de diseño y desarrollo que estaremos ofreciendo durante el 2022.

Kernel Frontend

Consiste en un Frontend especialmente dedicado al área de desarrollo en donde puedan de forma simple y segura, parametrizar una arquitectura y desplegarla en un entorno AWS.
Así como también realizar despliegues de código en ese entorno, siempre pensando en entornos no productivos.

Otras opciones de arquitectura

Saliendo de la arquitectura orientada a eventos, proponemos nuevas soluciones para problemas existentes, como por ejemplo versionado de la base de datos, carga de archivos en forma asincrónica, Auditoría de procesos centralizada, etc.

10 Links

Listado de sitios de interés y relacionados al proyecto.

https://www.cessi.org.ar Cámara de la industria Argentina de Software.

https://calculator.aws
Calculadora de precios para Amazon Web Services.

https://aws.amazon.com/es/legal/service-level-agreements
Acuerdos de SLA para Amazon Web Services.

https://www.redhat.com/es/products/application-runtimes/features
Redhat Runtimes, competencia en desarrollo de arquitecturas templates.

11 Biografia



Figure 6: Nicolas Alfredo Prosperi

Analista de sistemas del instituto de Tecnología ORT, actualmente cursando la Licenciatura en Tecnologías de la información.

Argentino de 39 años de edad y 16 años trabajando en TI de los cuales los últimos 8 dedicado a funciones de SRE y los últimos 4 específicamente a metodologías Devops.

Actualmente liderando un equipo compuesto por analistas de infraestructura y DevOps Cloud.