Universidad Autónoma de Ciudad Juárez

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

Maestría en Cómputo Aplicado

“Desarrollo de marco técnico de referencia y metodología para el desarrollo de software como servicio (SaaS) usando contenedores”

Trabajo recepcional para obtener el grado de
Maestro en Cómputo Aplicado

Ing. Samuel Martín Martínez Magdaleno

“Becado por el Consejo Nacional de Ciencia y Tecnología”

Bajo la dirección del
Dr. Ramón Parra Loera
y la codirección del
Mtro. Luis Felipe Fernández Martínez

Ciudad Juárez, Chihuahua, agosto de 2021

Índice

Introducción 9
Capítulo I: Planteamiento general del trabajo recepcional 10
1.1 Antecedentes 10
1.2 Descripción del problema 11
1.3 Objetivo general 12
1.4 Objetivos específicos 12
1.5 Justificación 12
1.6 Metodología 13
1.7 Impacto social, económico y tecnológico. 13
1.8 Alcances y limitaciones 14
Capitulo II: Marco teórico 15
2.1 Nube 15
2.2 Características claves de la nube 16
2.3 Promotores del crecimiento del uso de la nube 17
2.4 Modelos de servicio en la nube 19
2.4.1 Infraestructura como servicio 20
2.4.2 Plataforma como servicio 22
2.4.3 Software como servicio 24
2.5 Modelos de despliegue 26
2.5.1 Nube pública 26
2.5.2 Nubes privadas 28
2.5.3 Nubes híbridas 30
2.6 Pronóstico y crecimiento de la nube 32
2.7 Contenedores 36
2.7.1 Contenedores de aplicación 36
2.7.2 Funcionamiento de contenedores 37
2.7.3 Mecanismos de seguridad en contenedores 38
2.7.4 Contenedores Docker 40
2.8 Orquestación de contenedores 43
2.8.1 Kubernetes 44
2.9 Modelos y paradigmas de desarrollo de software 45
2.9.1 Modelo cascada 45

2.9.2 Modelo ágil 46
2.9.3 DevOps 47
2.10 Patrones de arquitectura de software 51
2.10.1 Patrón de arquitectura en capas 52
2.10.2 Patrón de arquitectura microservicios 53
2.11 Servicios de Amazon Web Services 54
Capítulo III: Propuesta de marco técnico de referencia y metodología 56
3.1 Marco técnico de referencia en el dominio de SaaS usando contenedores 57
3.2 Metodología en el dominio de SaaS usando contenedores 59
3.2.1 Conceptualización de la solución (UML, requisitos, diagramas de flujo) 60
3.2.2 Arquitectura de la solución (Patrón de arquitectura de software) 61
3.2.3 Selección de un modelo de desarrollo (Modelo de desarrollo clásico o ágil) 62
3.2.4 Implementación de la solución (Modelo de servicio IaaS, PaaS y contenedores) 63
3.2.5 Despliegue de la solución (Modelo de despliegue en la nube) 64
3.2.6 Operación (Monitoreo y mantenimiento) 65
3.3 Caso de aplicación para marco técnico de referencia y metodología 66
3.4 Aplicación del marco técnico de referencia y metodología al caso de aplicación 69
3.5 Patrones de arquitectura de software para la solución 71
3.6 Arquitectura de software de la solución 73
Capítulo IV: Implementación del marco técnico de referencia y metodología 74
4.1 Implementación de la solución para el caso de aplicación 74
4.2 Arquitectura de servicios en la nube para el caso de aplicación 77
4.3 Implementación de servicios para el caso de aplicación 79
Capítulo V Resultados y conclusiones 87
Anexos 88
Anexo I: Configuración de IAM 88
Anexo II: Configuración de usuario 92
Anexo III: Configuración de Cognito 94
Anexo IV: Configuración de DynamoDB 104
Anexo V: Configuración de EC2 106
Anexo VI: Configuración de S3 para Code Deploy 113
Anexo VII: Configuración de Code Deploy para EC2 114
Anexo VIII: Configuración de SNS para Lambda 127

Anexo IX: Configuración de ElasticSearch para Lambda 128
Anexo X: Configuración de rol para Lambmda 131
Anexo XI: Configuración de Lambda 133
Anexo XII: Pruebas de SNS a Lambda 137
Anexo XIII: Configuración de Gateway API 142
Anexo XIV: Creación de imagen Docker 147
Anexo XV: Configuración de ECS 151
Anexo XVI: Configuración de Fargate 156
Referencias 164

Índice de figuras

Figura 1: Modelo de servicios en la nube 19
Figura 2: Modelo de servicio IaaS 21
Figura 3: Modelo de servicio PaaS 23
Figura 4: Modelo de servicio SaaS 25
Figura 5: Ganancias de la nube de Amazon [16]. 32
Figura 6: Uso de la nube en el año 2018 y planes de migración a la nube [16]. 33
Figura 7: Uso de SaaS empresarial en el segundo trimestre 2018 [16]. 34
Figura 8: Pronóstico de las ganancias e ingresos de la nube en último trimestre 2018 [16]. 35
Figura 9: Diagrama de arquitectura Docker 42
Figura 10: Modelo en cascada (desarrollo de software) 45
Figura 11: Modelo ágil (desarrollo de software) 46
Figura 12: Desafíos actuales en las empresas 47
Figura 13: Ciclo de vida DevOps 49
Figura 14: Ventajas de implementar DevOps para el desarrollo 50
Figura 15: Arquitectura de aplicación monolítica 52
Figura 16: Arquitectura de aplicación basada en microservicios 53
Figura 17: Modelación del marco técnico de referencia general 57
Figura 18: Modelación del marco técnico de referencia específico 58
Figura 19: Metodología para el desarrollo de SaaS usando contenedores 59
Figura 20: Diagrama de flujo del caso de aplicación 68
Figura 21: Marco técnico de referencia aplicado al caso de aplicación 69
Figura 22: Metodología aplicada al caso de aplicación 70
Figura 23: Modelación patrones de arquitectura general 71
Figura 24: Modelación patrones de arquitectura especifica 72
Figura 25: Diagrama de arquitectura de software de la solución 73
Figura 26: Implementación usada para el caso de aplicación 74
Figura 27: Implementación con contenedores 75
Figura 28: Implementación con servicios de nube y contenedor 75
Figura 29: Implementación con contenedores y servicio de nube 76

Figura 30: Arquitectura de servicios AWS para el caso de aplicación 77
Figura 31: Servicios usados para ejecución del microservicio contenerizado 78
Figura 32: Configuración de IAM 79
Figura 33: Configuración de usuario AWS en el entorno 79
Figura 34: Configuración de Cognito 80
Figura 35: Configuración de DynamoDB 80
Figura 36: Configuración de EC2 80
Figura 37: Configuración de S3 para Code Deploy 81
Figura 38: Configuración de Code Deploy para EC2 81
Figura 39: Configuración de SNS 82
Figura 40: Configuración de ElasticSearch 82
Figura 41: Configuración de rol para Lambda 83
Figura 42: Configuración de Lambda 83
Figura 43: Pruebas de SNS a Lambda 84
Figura 44: Configuración de Gateway API 84
Figura 45: Creación de imagen Docker 85
Figura 46: Configuración de ECR 85
Figura 47: Configuración de Fargate 86
Figura 48: Página de inicio IAM 88
Figura 49: Página para añadir usuarios en IAM 89
Figura 50: Página para añadir grupos y políticas en IAM 90
Figura 51: Página para añadir etiquetas en IAM 90
Figura 52: Página para revisar el usuario en IAM 91
Figura 53: Página de notificación en IAM 91
Figura 54: Carpeta AWS en el directorio home del usuario 92
Figura 55: Archivo de credenciales guardado en la carpeta AWS 92
Figura 56: Contenido del archivo “credentials” 93
Figura 57: Página de inicio Cognito 94
Figura 58: Página grupo de usuario en Cognito 94
Figura 59: Página nombre del grupo de usuarios en Cognito 95
Figura 60: Página nombre del grupo de usuario en Cognito 95
Figura 61: Página para editar valores de grupos en Cognito 96
Figura 62: Página para personalizar email en Cognito 97
Figura 63: Página para personalizar mensajes en Cognito 98
Figura 64: Página para añadir etiquetas en Cognito 99
Figura 65: Página para recordar dispositivos en Cognito 99
Figura 66: Página para configurar clientes de aplicación en Cognito 100
Figura 67: Página para configurar desencadenadores en Cognito 100
Figura 68: Página de resumen de configuración en Cognito 101
Figura 69: Datos de id grupo y ARN en Cognito 102
Figura 70: Página de configuración del grupo de usuario en Cognito 102
Figura 71: Página para añadir clientes de aplicación en Cognito 102
Figura 72: Página para añadir cliente de aplicación en Cognito 103
Figura 73: Configuración del cliente de aplicación y clave secreta en Cognito 103

Figura 74: Página de inicio DynamoDB 104
Figura 75: Página para crear tablas en DynamoDB 105
Figura 76: Página de inicio EC2 106
Figura 77: Página para crear instancias en EC2 106
Figura 78: Página para seleccionar imagen de máquina en EC2 107
Figura 79: Página para seleccionar tipo de instancia en EC2 107
Figura 80: Página para configurar detalles de instancia en EC2 108
Figura 81: Página para añadir almacenamiento en la instancia EC2 108
Figura 82: Página para añadir etiquetas en EC2 109
Figura 83: Página para configurar grupo de seguridad en EC2 109
Figura 84: Página de resumen de configuración en EC2 110
Figura 85: Llave cifrada para obtener contraseña en EC2 110
Figura 86: Página de estatus de instancia en EC2 111
Figura 87: Página de instancias disponibles en EC2 111
Figura 88: Desencriptando la contraseña por defecto en EC2 112
Figura 89: Notificación de desencriptación de contraseña en EC2 112
Figura 90: Página para crear cubetas en S3 113
Figura 91: Página que muestra las cubetas en S3. 114
Figura 92: Página para crear aplicaciones en CodeDeploy 114
Figura 93: Página para crear aplicaciones en CodeDeloy 115
Figura 94: Página para crear grupos de implementación en CodeDeploy 115
Figura 95: Página para crear grupo de implementación en CodeDeploy 116
Figura 96: Pagina para configurar el tipo de implementación en CodeDeploy 116
Figura 97: Página para configurar el entorno en CodeDeploy 117
Figura 98: Pagina para configurar la implementación en CodeDeploy 118
Figura 99: Página que notifica la creación del grupo de implementación 119
Figura 100: Compilando el microservicio “AdvertApi” desde la terminal 119
Figura 101: Publicando el microservicio en un directorio especifico 120
Figura 102: Archivo de configuración utilizado por CodeDeploy 120
Figura 103: Directorio con microservicio y archivo de configuración comprimido 120
Figura 104: Página de cubetas en S3 121
Figura 105: Página para cargar archivos en cubeta S3. 121
Figura 106: Página con archivo cargado en cubeta S3 122
Figura 107: Permisos de usuarios en los objetos de la cubeta S3. 122
Figura 108: Página para configurar tipo de almacenamiento en S3 123
Figura 109: Página con resumen de la configuración en S3 123
Figura 110: Página para ver los archivos existentes en la cubeta S3 124
Figura 111: Página para crear implementaciones en CodeDeploy 124
Figura 112: Página para crear una implementación en CodeDeploy 125
Figura 113: Página que notifica la creación de la implementación en CodeDeploy 126
Figura 114: Verificando la implementación del microservicio en EC2 126
Figura 115: Página para crear temas en SNS 127
Figura 116: Página para editar temas en SNS 127
Figura 117: Página de inicio Elasticsearch 128

Figura 118: Página para seleccionar tipo de implementación en Elasticsearch 128
Figura 119: Página para configurar tipo de implementación en Elasticsearch 129
Figura 120: Página para configurar tipo de implementación en Elasticsearch 129
Figura 121: Pagina para configurar acceso y seguridad en Elasticsearch 130
Figura 122: Página que notifica la creación del dominio en Elasticsearch 130
Figura 123: Página de inicio roles 131
Figura 124: Paso 1 para crear roles en AWS 131
Figura 125: Paso 2 para crear roles en AWS 132
Figura 126: Paso 3 para crear roles en AWS 132
Figura 128: Página de funciones en Lambda 133
Figura 129: Página para crear funciones en Lambda 133
Figura 130: Página para crear funciones en Lambda 134
Figura 131: Página para editar funciones en Lambda 134
Figura 132: Página para añadir desencadenadores en Lambda 135
Figura 133: Directorio en donde se encuentra la función desarrollada. 135
Figura 134: Empaquetando la función para subirla a Lambda 136
Figura 135: Página para editar funciones en Lambda 136
Figura 136: Página de temas en SNS 137
Figura 137: Página tema AdvertApi en SNS 137
Figura 138: Página para publicar mensajes en SNS 138
Figura 139: Página para publicar mensajes en SNS 139
Figura 140: Página de notificaciones en SNS 140
Figura 141: Página para editar funciones en Lambda 140
Figura 142: Pestaña de monitorización en Lambda 141
Figura 143: Página de inicio en CloudWatch 141
Figura 144: Página de inicio en API Gateway 142
Figura 145: Página para crear una API en API Gateway 143
Figura 146: Página para editar API en API Gateway 143
Figura 147: Página para nuevos recursos en API Gateway 144
Figura 148: Página para editar recursos en API Gateway 144
Figura 149: Página para editar API en API Gateway 145
Figura 150: Página para implementar API en API Gateway 145
Figura 151: Página para editar etapas en API Gateway 146
Figura 152: Navegador Web con URL del Proxy en API Gateway 146
Figura 153: Soporte para Docker en Visual Studio 2019 147
Figura 154: Archivo DockerFile 148
Figura 155: Comando para acceder a la ruta del archivo DockerFile 149
Figura 156: Comando para ver los archivos en la ruta seleccionada 149
Figura 157: Comando para crear imágenes Docker 150
Figura 158: Ejecución del comando Docker build 150
Figura 159: Página de inicio ECS 151
Figura 160: Página repositorios ECR 151
Figura 161: Página para crear repositorios ECR 152
Figura 162: Página para crear repositorios ECR 152

Figura 163: Página para ver y editar repositorios ECR 153
Figura 164: Página para editar repositorios ECR 153
Figura 165: Comandos para subir imagen Docker a ECR 154
Figura 166: Comandos para subir imagen Docker a ECR 154
Figura 167: Comando para obtener token de autentificación 155
Figura 168: Comando para construir imagen Docker 155
Figura 169: Comando para etiquetar imagen Docker 155
Figura 170: Comando para subir imagen Docker a ECR 155
Figura 171: Página de Clusters en ECS 156
Figura 172: Página para crear clusters en ECS 156
Figura 173: Página para crear clusters en ECS 157
Figura 174: Notificación de creación de cluster en ECS 157
Figura 175: Definición de tareas en ECS 158
Figura 176: Página para crear tareas en ECS 158
Figura 177: Página para crear tareas en ECS 159
Figura 178: Página para crear tareas en ECS 159
Figura 179: Página para crear tareas en ECS 160
Figura 180: Página para crear tareas en ECS 160
Figura 181: Página para crear tareas en ECS 161
Figura 182: Notificación de tarea creada en ECS 161
Figura 183: Página para ejecutar tareas en ECS 162
Figura 184: Página para editar tarea en ejecución en ECS 162
Figura 185: Página para editar tarea en ejecución en ECS 163
Figura 186: Página para ver y editar clusters en ECS 163

Introducción

El desarrollo del presente trabajo recepcional inicia con el planteamiento general que comprende antecedentes, descripción del problema, objetivos, justificación, metodología, su impacto social y sus alcances y limitaciones. Posteriormente en el capítulo II se explica los conceptos claves del cómputo en la nube y contenedores que se utilizaron para el desarrollo del marco técnico de referencia y metodología. Se comienza por la definición de nube, cuáles son sus características claves, promotores, modelos de servicios y modelos de despliegue. Seguido de estas definiciones, se entra en detalle con la definición de los contenedores, su funcionamiento, tipos de contenedores, orquestadores de contenedores y se concluye con algunas de las diferentes metodologías y paradigmas existentes del desarrollo de software. Todas estas definiciones conceptuales se utilizaron para el desarrollo de las secciones posteriores. En el capítulo III se presenta la propuesta del marco técnico de referencia y la metodología desde el dominio de SaaS usando contenedores. Para terminar con la sección III se muestra el desarrollo del caso de aplicación, patrones de arquitectura de software y la arquitectura de software que permitieron validar lo propuesto.

En el capítulo IV se muestra la implementación del marco técnico de referencia y metodología. En ella se describe como se implementó la solución para el caso de aplicación, cuales servicios de nube se utilizaron para su desarrollo mediante un diagrama de arquitectura que explica cómo se relacionan e interactúan cada uno de estos componentes y la implementación de cada uno de estos servicios mediante diagramas de flujo que explican la secuencia de pasos necesarios para poner en ejecución lo propuesto en el caso de aplicación. En esta última sección se describen brevemente los pasos a seguir, no obstante, se incluye un anexo en donde se muestra de manera detallada todo el proceso de configuración a seguir en el proveedor de nube pública Amazon. En el capítulo V se presentan los resultados y conclusiones.

Finalmente, en este trabajo recepcional convergen la investigación referencial de distintas fuentes y la experiencia profesional dentro del ámbito del desarrollo de software en la nube usando contenedores, esto proporcionó una guía estable para el desarrollo de la propuesta del marco técnico de referencia y metodología. En este sentido, el trabajo directo en una empresa de telecomunicaciones (Transtelco) permitió enriquecer y confirmar lo propuesto.

Capítulo I: Planteamiento general del trabajo recepcional

1.1 Antecedentes

La rápida industrialización y los avances constantes en las tecnologías de la información y comunicación (TICS), han dado surgimiento a nuevas tecnologías emergentes que brindan la posibilidad a las empresas de ofrecer servicios modernos y disruptivos. En gran medida este proceso de evolución y adopción de nuevas tecnologías se puede atribuir a la industria 4.0. La cual surge como una propuesta del gobierno alemán en el año 2011 en la feria de Hannover y se anunció de manera oficial en el año 2013 como una iniciativa que pretende ser un elemento clave en las industrias que se encuentran revolucionando el sector manufacturero. La importancia de la industria 4.0 se ve presente en países como Alemania, en donde es considerada como un proyecto clave que permitirá al país convertirse en uno de los líderes del sector industrial. De la misma manera la relevancia de la industria 4.0 se mira presente en países asiáticos en donde contemplan la industria 4.0 como una herramienta para convertir sus países de talleres manufactureros a potencias de manufactura mundial. Uno de estos países es China, la cual tiene contemplada la industria 4.0 en un plan denominado “Made-in-China 2025” [1].

Estos ejemplos de planes e iniciativas requieren una combinación de tecnologías emergentes recientes tales como lo son el cómputo en la nube, internet de las cosas, cómputo cognitivo, realidad aumentada, robots autónomos, entre otras. Las cuales son los pilares tecnológicos sobre los que se sustenta la industria 4.0. Dentro de estas tecnologías emergentes una de las que está teniendo un gran crecimiento en los últimos años en cuanto a su uso y popularidad es el cómputo en la nube. En particular ha tenido un crecimiento exponencial en el año 2020 dada la situación sanitaria actual en la que muchos negocios y empresas se han visto en la necesidad de migrar su operación de trabajo a una modalidad remota [2].

Esto se puede apreciar con la gran demanda que han tenido los servicios basados en la nube. Por ejemplo, Microsoft Teams vio un incremento del 775% y Zoom reporto un incremento del 354% en comparación del año anterior [3]. Además de estos incrementos los proveedores de nube pública reportan de igual manera un incremento en la demanda de sus servicios. Esto sin duda son indicadores de los beneficios que aporta el uso del cómputo en la nube. Sin embargo, existen empresas que se cuestionan la conveniencia de la adopción del cómputo en la nube. Esto las coloca en una desventaja de competitividad, ya que de acuerdo con [4], están un 90% atrasadas con respecto a otras empresas que sí han adoptado este paradigma. No obstante, a pesar de ser un gran paradigma innovador para el sector público y privado, existen algunos problemas para su aplicación en la industria e investigación académica los cuales se puntualizan en la sección 1.2.

1.2 Descripción del problema

El rápido crecimiento del paradigma de cómputo en la nube y el tránsito hacia un modelo de software como servicio (SaaS), ha ocasionado que las metodologías y marcos técnicos de referencia para el desarrollo de aplicaciones se queden obsoletos. De acuerdo con varios estudios [4], [5] es necesario el desarrollo de marcos de trabajo y metodologías que se validen de manera empírica y sirvan para guiar a la ingeniería de software en este nuevo dominio. En este sentido, la falta de metodologías y marcos técnicos de referencia genera una seria problemática, tal como la falta de trabajadores calificados que tengan el conocimiento necesario para desarrollar aplicaciones en este nuevo paradigma. Este problema junto con la falta de visión de empresas que no han decidido invertir en nuevas tecnologías se ven desplazadas por aquellas que han iniciado con los nuevos paradigmas de la era digital. Un ejemplo de ello es AMD una empresa dedicada a la producción de semiconductores la cual se encontraba derrochando dinero, sumida en deudas y en dirección a una posible quiebra. Hasta que en el año 2014 esta situación cambia con el nombramiento de la nueva CEO, la Dra. Lisa Su quien logró rescatar a AMD de un valor mínimo histórico en el precio de sus acciones hasta más de un 1300% en el año 2020 [6].

Este gran logro se da gracias a que la compañía ha centrado sus esfuerzos en la construcción de aplicaciones de alto rendimiento que utilizan tecnologías de nueva generación como cómputo en la nube, inteligencia artificial y video juegos [6]. En este sentido, cómputo en la nube forma parte de estas tecnologías de nueva generación, ya que ofrece servicios mediante una suscripción lo cual permite eliminar la inversión inicial de capital TI y pagar solo por los servicios que se utilizan. Es decir, permite pasar de una economía basada en activos (Capex) a una economía basada en servicio (Opex) con lo cual las empresas pueden ser más competitivas ya que puede usar ese mismo capital en liberar aplicaciones de forma más frecuente [7].

Sin embargo, en adición a la obsolescencia de los marcos técnicos de referencia y metodologías también es necesario considerar que el desarrollo de software como servicio (SaaS) se realiza tradicionalmente mediante el uso de una plataforma como servicio (PaaS). Esta forma de desarrollar SaaS tiene grandes beneficios como el desarrollo rápido, flexible y de menor costo operativo. No obstante, aun con estos beneficios es importante mencionar que tiene el inconveniente de la dificultad para la migración de aplicación hacia otra plataforma, ya que implica un uso considerable en tiempo y costo para adaptar la aplicación a una nueva plataforma. Por esta razón, se hace necesario la creación de un esquema para el desarrollo de software en la nube que sea independiente de la plataforma para permitir una migración fácil a otras plataformas [8].

1.3 Objetivo general

Desarrollar un marco técnico de referencia y metodología para el desarrollo de software como servicio (SaaS) usando contenedores.

1.4 Objetivos específicos

- Englobar en un marco técnico de referencia los conceptos de SaaS y contenedores.
- Desarrollar una metodología para desarrollar SaaS usando contenedores.
- Determinar caso de aplicación de la metodología.

1.5 Justificación

En la actualidad existen diversos estudios que muestran el potencial del cómputo en la nube en los sectores público y privado [8], [9], [10]. Sin embargo, una de las principales barreras para la adopción de este paradigma es la falta de personal con el conocimiento necesario para el desarrollo de sistemas que utilicen estas nuevas tecnologías que les otorga una ventaja competitiva a las empresas en esta nueva era digital [4]. En este sentido, el desarrollo de un marco técnico de referencia que ayude a proporcionar la base conceptual para el desarrollo de sistemas que utilicen estas nuevas tecnologías, así como la propuesta de una metodología que sirva como guía para la implementación de este paradigma resultan de gran utilidad.

No obstante, es importante mencionar que el modelo SaaS basado en PaaS tiene el problema de la dependencia del proveedor. Esta característica inherente dificulta las migraciones hacia otros proveedores de nube, con lo cual se impide o desaprovecha el tomar ventaja de la gran competencia que existe y seguirá existiendo entre los distintos proveedores de nube. Una alternativa que conserva las ventajas del modelo PaaS y elimina la dependencia del proveedor de servicios de cómputo en la nube es el uso de contenedores.

El SaaS basado en contenedores permite que las aplicaciones sean portables, interoperables y de fácil migración ya que se empaqueta la aplicación con todas las librerías que requiere para su funcionamiento, con lo cual se evita la dependencia de un proveedor de nube y se puede migrar fácilmente a otros entornos sin que sea necesario reprogramar la aplicación.

En adición a esta portabilidad que otorgan los contenedores, también se evita el problema de en caso de que un proveedor de nube llegue a presentar fallas simplemente se toman el o los contenedores y se instalan en el nuevo proveedor alternativo [11]. Por todos estos motivos, se propone la creación de un esquema de desarrollo de software en la nube que conserve las ventajas de este paradigma y permita la migración sencilla a otras plataformas de nube.

1.6 Metodología

Para la generación del marco técnico de referencia, se utilizó investigación referencial que proporcionó las bases teóricas para construir la base conceptual del trabajo recepcional. En cuanto a la metodología se realizó investigación referencial sobre metodologías de desarrollo de software en la nube existentes y se realizó experimentación con diversos lenguajes y proveedores de nube, para con ello determinar cuál era el más adecuado para el desarrollo del caso de aplicación.

Para terminar, la validación de la metodología implicó realizar cursos y experiencia profesional en el desarrollo de software en la nube usando contenedores, para con ello diseñar un caso de aplicación en donde se recolectaron datos detallados de la validación de lo propuesto en el marco técnico de referencia y metodología [12].

1.7 Impacto social, económico y tecnológico.

La aplicación del marco técnico de referencia y metodología dentro de las empresas tiene un impacto social y económico, ya que cambia la forma tradicional que existe en el desarrollo de una aplicación o sistema. Al existir este cambio se pueden observar los beneficios que aporta el uso de cómputo en la nube y los contenedores, para con ello lograr un ahorro en costos operativos y ser más competitivos frente a la competencia. En cuanto al impacto tecnológico, el desarrollo de la metodología generó un nuevo esquema tecnológico que sirve como un promotor del uso de cómputo en la nube y contenedores [7].

1.8 Alcances y limitaciones

Desarrollar un marco técnico de referencia en donde se engloben los conceptos de PaaS, SaaS y contenedores, los cuales servirán como base conceptual para el desarrollo de una metodología. La metodología proporcionará a los desarrolladores las pautas necesarias para crear software como servicios (SaaS) usando contenedores.

Una limitante del marco técnico de referencia y metodología es que se probará su funcionamiento con un solo caso de estudio. Aún y cuando en la metodología se utilicen diferentes adaptaciones y/o modificaciones.

Capitulo II: Marco teórico

2.1 Nube

El NIST define el cómputo en la nube de la siguiente manera. "Es un modelo para permitir el acceso ubicuo, conveniente y bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar e implementar rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios" [13].. Está compuesto de cinco características claves, tres modelos de servicio y cuatro modelos de implementación.

Sin embargo, se han generado debates acerca de la definición correcta de qué es la nube. Algunos autores la definen como un conjunto de tecnologías, lo cual hasta cierta medida es cierto, ya que un entorno de nube está conformado por diversas tecnologías, pero no es lo único que se requiere para que exista una nube. Una definición un poco más acertada sería "es un servicio o grupos de servicios que son accedidos mediante internet" [14]. Pero como sucede con la mayoría de los servicios, la nube y sus servicios que ofrece han cambiado con el paso del tiempo. Por lo general, estos servicios avanzan a un paso rápido para adaptarse a las diferentes necesidades de los clientes.

Si nos detenemos a preguntarnos qué servicios tecnológicos utilizamos que no hayan sufrido algún cambio. La respuesta es que están en constante cambio, de igual modo sucede con la nube. Esto es lo que precisamente da entrada a los debates acerca de la definición correcta de nube. Sin embargo, muchos autores aceptan la definición de nube por el Instituto Nacional de Estándares y Tecnologías (NIST), por sus siglas en inglés, como el estándar cuando se quiere hablar sobre que es la nube.

2.2 Características claves de la nube

Existe un gran número de empresas y proveedores que buscan integrarse al ofrecimiento de servicios en la nube debido a la gran popularidad que está teniendo [14]. Muchos de estos proveedores afirman que ofrecen servicios o aplicaciones basados en la nube, cuando en realidad no es cierto. El hecho de comercializar una aplicación web no significa que se esté ofreciendo una aplicación basada en la nube. Para que una aplicación o sistema se considere basado en la nube el NIST describe cinco características claves que deben cumplirse para que una aplicación o servicio se pueda considerar un producto basado en la nube. Estas características se enlistan a continuación [13]:

Servicio bajo demanda: Característica en la cual los clientes deben ser capaces de solicitar y recibir acceso a los servicios sin que exista un intermediario que sea responsable de realizar dicha solicitud de manera manual. La asignación de los recursos informáticos debe ocurrir de manera automática, lo cual representa una ventaja para el cliente y el proveedor del servicio.

Acceso amplio y estandarizado vía Internet: Los clientes deben ser capaces de acceder a los servicios del proveedor sin la necesidad de instalar algún software adicional. Por lo tanto, los servicios deben tener la capacidad de ser accesibles desde cualquier dispositivo: celulares, relojes, tabletas, laptops, computadoras de escritorio, televisiones, entre otros.

Agrupación de recursos: Esta se basa en la premisa de que los clientes no tendrán necesidad de utilizar todos los recursos informáticos disponibles. Cuando los recursos asignados a un cliente no están en uso, en lugar de permanecer inactivos se asignan a otro cliente. Esto resulta de gran beneficio para el proveedor ya que permite flexibilidad de atender a un mayor número de clientes de los que realmente puede, en comparación con que cada cliente utilizará recursos dedicados, esta flexibilidad que se logra implica un ahorro de costos para el proveedor.

Elasticidad y escalabilidad rápida: Esta permite que la nube crezca de manera ágil, para satisfacer los requerimientos de los clientes. Un buen diseño arquitectónico de nube debe permitir expandir los recursos de una manera sencilla. Por ejemplo, el añadir más recursos computacionales, espacio de almacenamiento, entre otros. El componente principal es que aun cuando se tengan los recursos disponibles, no se utilizan hasta que son requeridos. Lo cual permite que los proveedores ahorren costos de refrigeración y energía eléctrica.

Servicio medido: Los servicios de nube deben disponer la capacidad de medir su uso. El cual se puede determinar utilizando diferentes métricas, como el ancho de banda o tiempo utilizados. Esta característica es lo que posibilita el pago por uso del cómputo en la nube. Cuando el proveedor ha identificado la métrica que utilizará se determina la tarifa con la cual se cobrará al cliente. Así se factura en función a los niveles de consumo.

2.3 Promotores del crecimiento del uso de la nube

En la manera tradicional de desarrollar e implementar aplicaciones, las empresas invierten un gran capital en la instalación de sistemas y capacitación de personal, ya que tienen que presupuestar la compra de infraestructura, su mantenimiento, operación y gasto de compras de licencias de software, ya sea mediante medios digítales o físicos. El uso de cómputo en la nube favorece una reducción drástica de estos costos y permite que los usuarios se cambien de forma más sencilla a otra aplicación sin tener que estar sujetos exclusivamente a una aplicación. Para lograr esto la nube tiene cinco promotores claves que han hecho que gane bastante popularidad [14]:

Escalabilidad y elasticidad: Este primer punto se refiere a la capacidad de la nube para escalar de forma automática y con ello atender las necesidades de sus usuarios. La nube debe permitir agregar recursos computacionales de manera dinámica, lo que le permite desempeñarse en incrementos de uso. Esto resulta de gran utilidad ya que se pueden añadir estos recursos bajo demanda, lo cual significa que los recursos no tienen que estar siempre disponibles en espera, lo cual implica un ahorro en la utilización de recursos como energía y enfriamiento.

Rendimiento: El rendimiento de la nube es monitoreado y medido constantemente por el proveedor de servicio. Si ocurre una situación en la que el rendimiento de los servicios sufra una baja por cierto nivel establecido, los sistemas deben ajustarse automáticamente para cumplir con el acuerdo. Si se incumple con el acuerdo, el proveedor deberá buscar una manera de restituir a sus clientes.

Facilidad de mantenimiento: Es una de las cualidades que hace atractivo el cómputo en la nube. Esto se logra debido a que el proveedor es el que administra la infraestructura y los sistemas para proporcionar el servicio, lo que significa que el proveedor es el encargado de dar mantenimiento. Esto implica que los clientes no se ocupen en mantener la infraestructura y los sistemas con las últimas actualizaciones de hardware y software.

Agilidad: La agilidad que ofrecen los entornos de nube se debe a que permiten reconfigurar los recursos cuando se requiere. Esta cualidad de la nube permite que agreguemos recursos a los sistemas que lo requieren y quitemos recursos de sistemas que se encuentren en reposo. También se pueden agregar nuevos sistemas que faciliten expandir los recursos computacionales.

Confiabilidad: Muchos proveedores ofrecen la opción de utilizar sus servicios en múltiples ubicaciones geográficas, lo cual permite la redundancia y una mayor confiabilidad de los servicios que nos brindan. Por lo general, tienen un costo menor en comparación a un entorno tradicional, en donde se debería tener varios sistemas o ubicaciones de centros de datos. Toda esta redundancia en un entorno tradicional también implica tener un plan de recuperación de desastres y planificación para la continuidad del servicio.

2.4 Modelos de servicio en la nube

Conforme a la definición de nube proporcionada por el Instituto Nacional de Estándares y Tecnología (NIST), hay tres modelos de servicios en la nube: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS). Estos son los tres modelos originales de servicios en la nube. No obstante, cabe mencionar que debido a que se tratan de servicios prácticamente todo es negociable.

A medida que el uso de la nube creció, los servicios existentes se modificaron y se agregaron nuevos servicios para satisfacer la demanda de los clientes. Esto llevó al surgimiento del paradigma de todo como servicio (XaaS), el cual lejos de remplazar los tres modelos originales busca integrar de manera total o parcial estos modelos [15].

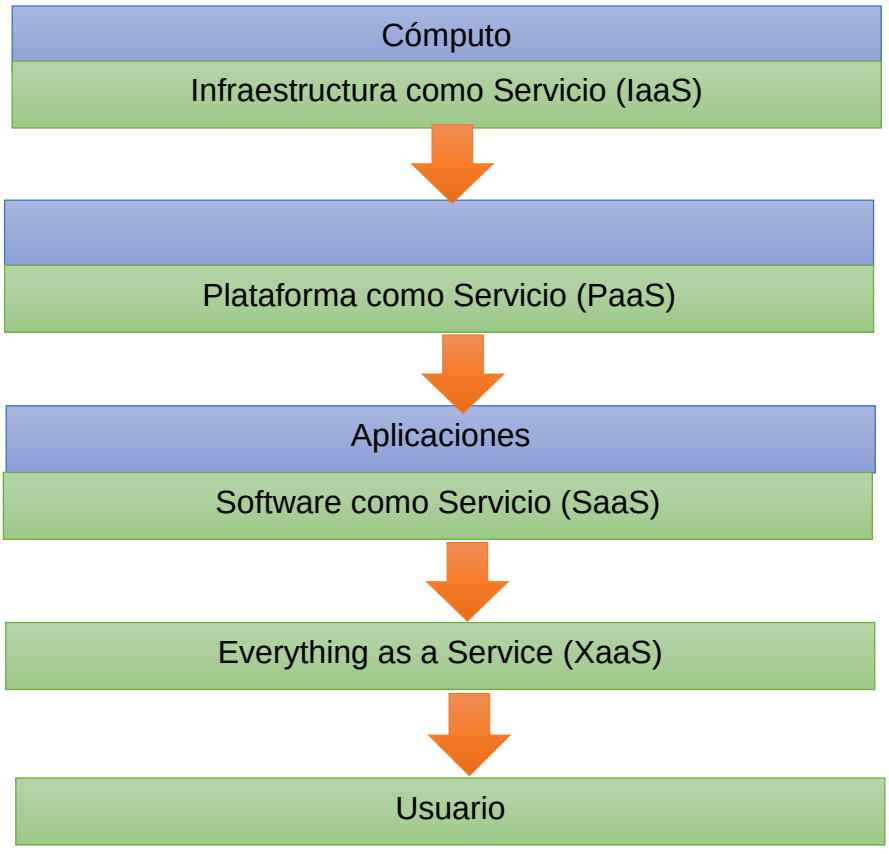

Figura 1: Modelo de servicios en la nube

2.4.1 Infraestructura como servicio

El primer modelo de servicio en la nube es IaaS. En este modelo el proveedor proporciona recursos de cómputo, almacenamiento, redes y sistemas. Los servidores que proporciona el proveedor se hospedan en su centro de datos a los cuales accede el cliente. El cliente es el responsable de construir su entorno y puede instalar lo que requiera en los servidores. Esta forma de proveer el servicio de IaaS es más costosa, debido a que el proveedor no puede aprovechar la infraestructura para hospedar a varios clientes, lo cual resulta en una factura más costosa para el cliente [14].

Otra forma en la que los proveedores pueden proporcionar IaaS es por medio de máquinas virtuales, en las cuales los usuarios pueden instalar el software que requieran. Las máquinas virtuales pueden ejecutar cualquier sistema operativo, Windows y Linux son los más comunes. Dado que, los sistemas están virtualizados el proveedor puede aprovechar los multi inquilinos. Esto quiere decir que se pueden hospedar a varios clientes en el mismo conjunto de hardware, lo cual resulta en ahorros que se reflejan en la factura del cliente [15].

Responsabilidades: En este modelo el cliente es responsable de mantener la mayoría del entorno. El cliente se encarga del mantenimiento del sistema operativo y las aplicaciones. No obstante, existen algunos elementos no tan evidentes, tal como el sistema de seguridad o protección contra virus. El cliente se debe asegurar que los sistemas que está utilizando tengan el software adecuado para protegerlo contra amenazas como virus, spyware, troyanos, entre otros.

El proveedor es responsable de mantener la capa del hipervisor y la infraestructura que está formada generalmente por hardware físico, almacenamiento y redes que se encuentran dentro del centro de datos del proveedor, al cual los clientes tienen acceso.

Promotores: Uno de los principales promotores del modelo IaaS, son las empresas que buscan ampliar su capacidad de cómputo sin tener que desembolsar capital en la expansión de un centro de datos, o la construcción de uno nuevo, este modelo les permite a las empresas rentar los sistemas proveídos por el proveedor. Otro de los promotores son las empresas que buscan disponer de una mayor capacidad, pero solo en ciertas ocasiones. Lo que les permite no gastar capital en soluciones costosas y permanentes. Con este modelo los clientes pueden pagar por el incremento de capacidad cuando es requerido.

Desafíos: Existen diversos desafíos para lograr la adopción de IaaS en las empresas, algunos de ellos son el costo. Generalmente los entornos IaaS se cobran por el uso de recursos computacionales como el procesador y memoria. Si el cliente no supervisa cuidadosamente el uso que le está dando puede llevarse una sorpresa al momento de que llegue la factura.

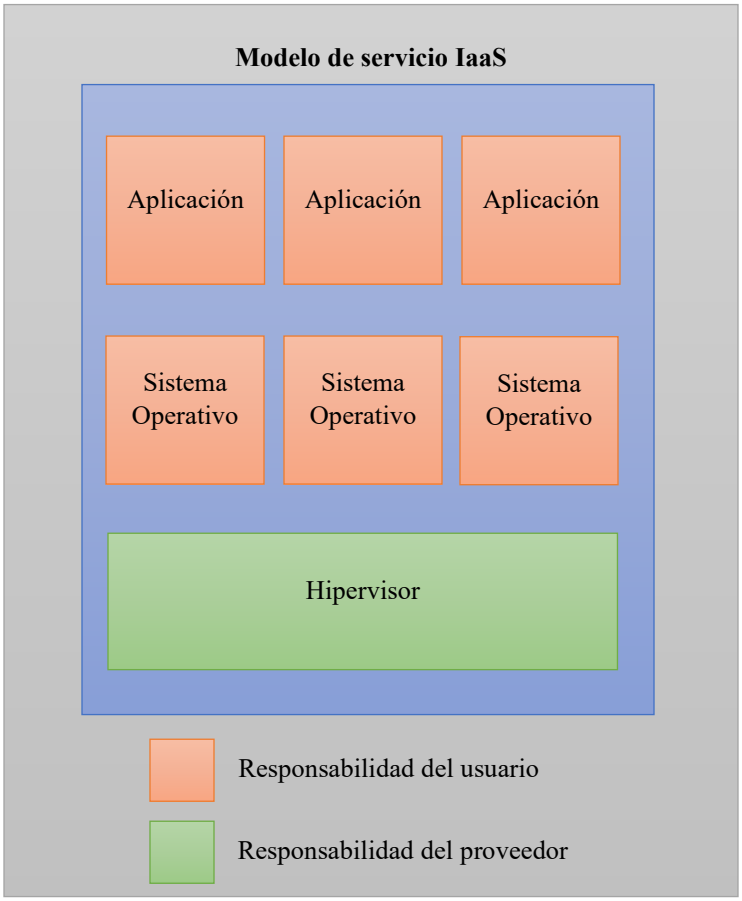

Figura 2: Modelo de servicio IaaS

2.4.2 Plataforma como servicio

El segundo modelo de servicio en la nube es PaaS. En este modelo el proveedor proporciona una plataforma a los clientes para sus necesidades de cómputo. Dependiendo del proveedor del servicio que el cliente seleccionó, la plataforma puede componerse de únicamente el sistema operativo o una plataforma de desarrollo completa que brinde un servidor web y librerías para el desarrollo.

Este modelo permite a las empresas desarrollar e implementar aplicaciones web sin tener que lidiar con la administración de la infraestructura. Las ofertas de PaaS por lo general facilitan la tarea del desarrollo mediante la integración y el testeo. Sin embargo, el cliente es responsable de supervisar el desarrollo de la aplicación [15].

Personalización: El modelo de servicio PaaS permite tener completo control sobre la aplicación o servicio, por lo que el cliente es libre de personalizar la aplicación a sus necesidades. No obstante, no se pueden realizar cambios en la plataforma de desarrollo ya que es administrada por el proveedor de servicio. Probablemente la plataforma cuente con algunas opciones para configuración que el cliente puede cambiar, pero es limitado a comparación de un modelo IaaS.

Integración: En el modelo de servicios PaaS los datos son almacenados en los centros de datos del proveedor, a los cuales los clientes tienen acceso. Los clientes pueden acceder los datos para realizar tareas de inteligencia empresarial o reportes sin problema. Sin embargo, pueden existir problema en el consumo del ancho de banda ya que probablemente se estarán moviendo grandes cantidades de datos en el entorno interno de la empresa y el entorno del proveedor, lo que puede generar ciertos problemas de rendimiento.

Promotores de PaaS: Existen numerosos promotores que han influido en el crecimiento del modelo PaaS en el mercado. Tal como, el hecho de que cada vez más empresas buscan avanzar hacia un modelo de nube pública, pero algunas veces no encuentran proveedores que ofrezcan servicios SaaS que se adapten a sus necesidades. El modelo de PaaS les facilita mover la infraestructura y plataforma fuera de sus centros de datos internos, y realizar el desarrollo de aplicaciones en los centros de datos del proveedor.

Responsabilidades de PaaS: El proveedor es responsable de mantener toda la plataforma de desarrollo e infraestructura. El proveedor tiene las responsabilidades de un modelo de servicios IaaS y aparte es responsable de mantener el sistema operativo con las ultimas actualizaciones y parches de seguridad. Sin embargo, el cliente es responsable de mantener las aplicaciones que se encuentren en ejecución dentro de la plataforma.

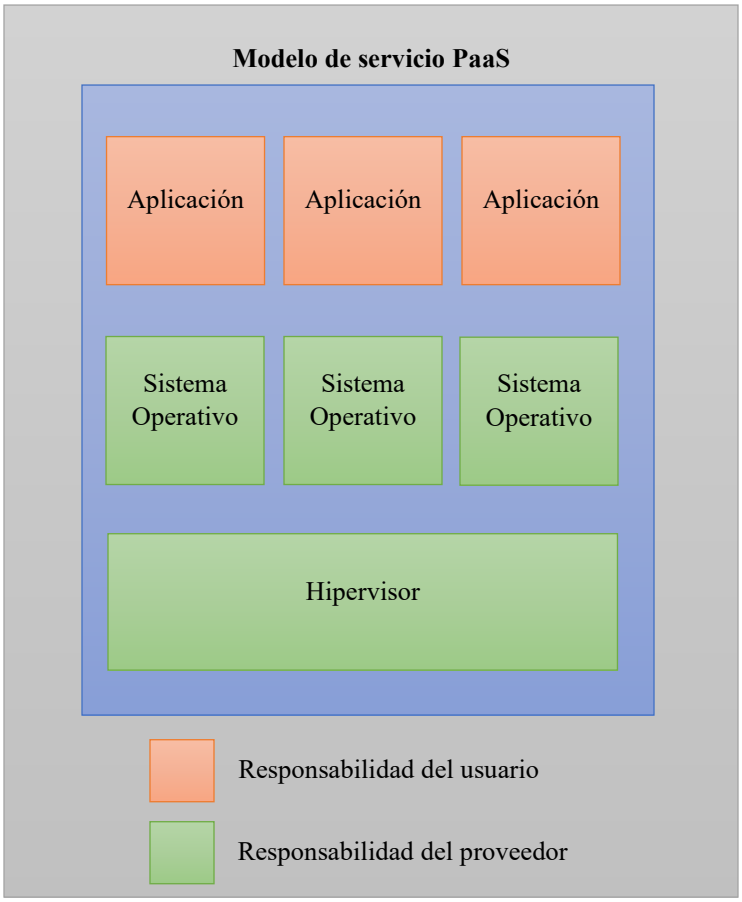

Figura 3: Modelo de servicio PaaS

2.4.3 Software como servicio

El último modelo de servicio es SaaS, el cual es considerado por muchos como el modelo de servicio en la nube original. Este modelo de servicio es similar al modelo de servicio de aplicaciones (ASP, por sus siglas en inglés, Application Service Provider). Pero existen algunas diferencias importantes. En el modelo ASP las aplicaciones son generalmente cliente-servidor que requieren algún tipo de infraestructura y cliente para acceder a sus aplicaciones. En comparación con el modelo SaaS las aplicaciones están basadas en web y no requieren de clientes para ser accedidas. Otra diferencia significativa es que en el modelo ASP los clientes generalmente accedían a diferentes instancias de una aplicación. En el modelo SaaS los clientes acceden a la misma aplicación con diferentes vistas.

En las siguientes secciones se describen algunas características, promotores y responsables que tiene el modelo SaaS [15].

Personalización: El modelo SaaS es controlado por el proveedor de servicios, lo cual limita cualquier personalización que los clientes quieran hacer. No obstante, los clientes pueden solicitar que la interfaz de usuario o apariencia de la aplicación se modifique ligeramente. Estos cambios no pueden ser de índole mayor y son administrados por el proveedor de servicios.

Integración: En este modelo, los datos son almacenados en el centro de datos del proveedor e implica que en la mayoría de los casos los clientes no pueden acceder a ellos. Lo cual puede ser problemático cuando el cliente requiere utilizar reportes e inteligencia empresarial. También puede ser problemático si se requiere realizar una reparación manual de los datos o cargar datos de forma masiva. Algunos proveedores permiten a los clientes mover los datos entre la instancia SaaS y los sistemas internos de una empresa. Sin embargo, el cliente debe ser cuidadoso con el uso de ancho de banda durante estas operaciones.

Promotores de SaaS: Existen numerosos factores que han influenciado el crecimiento del modelo SaaS en el mercado, como el incremento en la creación y consumo de aplicaciones web. Lo cual ha ocasionado que los usuarios se acostumbren a este tipo de aplicaciones y por lo tanto la aceptación del modelo SaaS. Estos promotores han generado que se mejore el aspecto de las aplicaciones, calidad y la facilidad con la que se pueden desarrollar.

Responsabilidades: En el modelo SaaS casi todas las responsabilidades recaen en el proveedor, lo cual ha generado en gran medida la popularidad de este modelo. Prácticamente el proveedor es responsable de todo. El proveedor se hará cargo de mantener la aplicación, sistema y la infraestructura actualizada y con últimos parches de seguridad. Además, debe monitorear los sistemas para medir el rendimiento y hacer los ajustes que se requieran. Esto permite que las empresas utilicen sus recursos internos en otras actividades en lugar de utilizarlos en la administración del sistema.

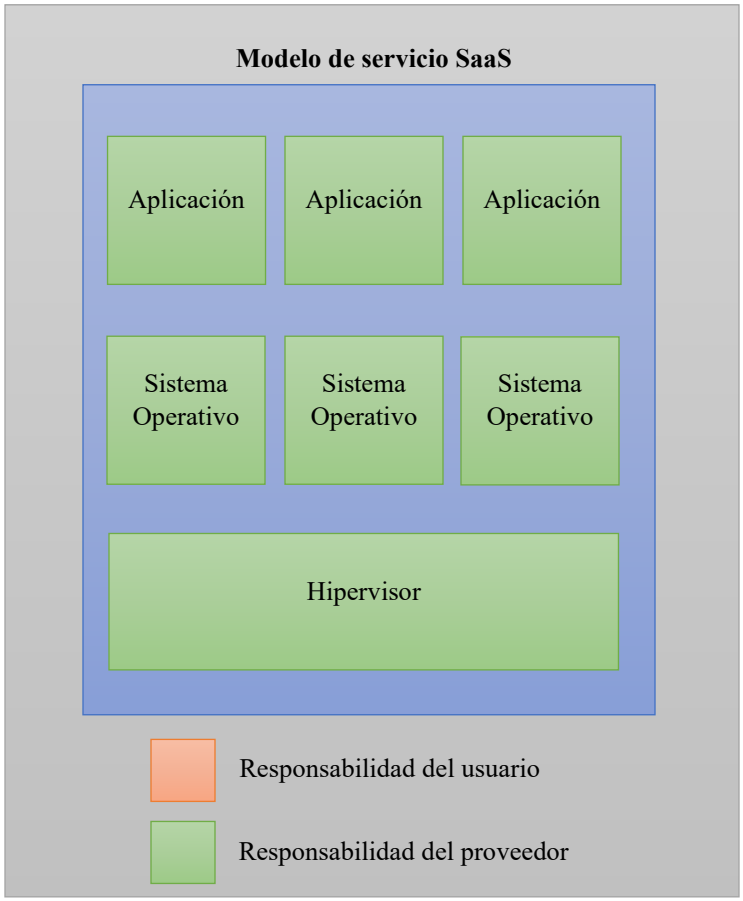

Figura 4: Modelo de servicio SaaS

2.5 Modelos de despliegue

El Instituto Nacional de Estándares y Tecnología (NIST) define cuatro modelos de despliegue: nube pública, nube privada, nube comunitaria, nube híbrida. En el presente trabajo recepcional solo se definen los primeros tres modelos. Cada modelo de despliegue solventa diferentes necesidades del usuario, por lo que es indispensable seleccionar un modelo que satisfaga las necesidades con los criterios en donde se esté implementando [13].

La selección de un modelo de despliegue es una de las decisiones más críticas a la hora de implementar un entorno de nube. El modelo de despliegue está definido en gran medida en donde se encuentra la infraestructura para su despliegue y quien tiene control sobre ella. Es importante resaltar que cada modelo de despliegue aporta un valor de propuesta diferente y tiene costos asociados distintos, para algunos clientes la elección del modelo lo determina el costo del servicio [15].

2.5.1 Nube pública

El primer modelo de despliegue es la nube pública, en este modelo los entornos se encuentran administrados y atendidos en su totalidad por un proveedor de servicios. Cuando las personas piensan en computo en la nube por lo general están suponiendo que son nubes públicas. Estos en gran medida se debe al hecho de que el primer entorno que surgió fue el de nube pública. La idea de la existentica de otro tipo de implementación tomo un poco de tiempo. Hoy en día las nubes publicas siguen siendo los entornos más populares e implementados [15].

Beneficios: Las implementaciones en donde se utiliza un modelo de despliegue de nube pública siguen en aumento a un ritmo rápido, en virtud de los numerosos beneficios que ofrece este modelo de despliegue. La propuesta de valor de este modelo es muy fuerte, aunque tiene algunas desventajas como se detallarán más adelante.

Disponibilidad: El modelo de despliegue de nube pública brinda una mayor disponibilidad en comparación a lo que se puede lograr de manera local. Esto se consigue debido a que en las empresas existe cierto nivel de disponibilidad real y un nivel que les gustaría alcanzar. Pero existe la problemática de que una implementación local con alta disponibilidad tiene costos asociados, entre los que se encuentran el costo de hardware, software y personal.

Este modelo ya tiene los recursos de hardware, software y personal necesarios para garantizar que sus soluciones sean de alta disponibilidad. Probablemente el proveedor cobre un cargo adicional por proporcionar este servicio, pero no se compara al costo de hacerlo localmente. Sin embargo, no se debe asumir una alta disponibilidad o tolerancia a fallos solo por el hecho de que seleccionemos un proveedor de nube pública.

Se debe realizar un análisis de lo que ofrece el proveedor con el servicio. Por ejemplo, si la alta disponibilidad es parte del acuerdo a nivel de servicio (SLA) o si el aumento de alta disponibilidad implica un aumento de costo. También, se debe tener en consideración que la alta disponibilidad depende del modelo de servicios que el usuario contrate. En un modelo de servicio SaaS la aplicación debe estar disponible, pero si estamos utilizando un modelo de servicios IaaS o PaaS puede que la infraestructura y plataforma este disponibles pero la aplicación dependerá completamente de cliente.

Escalabilidad: El modelo de despliegue de nube pública brinda una arquitectura altamente escalable, al igual que los otros modelos de despliegue. La principal diferencia que ofrecen los despliegues de nube pública es la capacidad de escalar la capacidad de una empresa sin preocuparse por construir una infraestructura interna.

Este modelo tiene la virtud de ofrecer un incremento de cómputo temporal o permanente, de acuerdo con las necesidades de los clientes. Si la empresa está utilizando un modelo SaaS puede agregar usuarios a la aplicación sin preocuparse por la infraestructura necesaria. En cambio, si la empresa opta por un modelo IaaS o PaaS se tendrá que preocupar de que la aplicación que se desarrolló pueda manejar los incrementos de carga.

Accesibilidad: Una de las características más importantes de este modelo de despliegue es la accesibilidad. Para que los proveedores amplíen su cartelera de clientes al máximo, deben cerciorarse de que pueden atender a diferentes tipos de clientes. Esto quiere decir que los clientes deben poder acceder a sus servicios desde cualquier dispositivo conectado a internet, sin la necesidad de algún software adicional.

Actualmente existe una gran cantidad de dispositivos que acceden a internet, tal como, teléfonos inteligentes, relojes inteligentes, computadoras de escritorio, laptops, tabletas, televisiones y un gran número de dispositivos IoT por lo que los proveedores deben ofrecer soluciones que se puedan adaptar a toda esta variedad de dispositivos.

Inconvenientes: El modelo de despliegue de nube pública tiene algunas desventajas. Muchas de las cuales son debido a que la infraestructura es controlada y administrada por un tercero. Lo cual hace que uno de sus promotores más importantes, también sea una de sus desventajas.

Limitaciones de integración: En los modelos SaaS públicos la infraestructura y software son externos a la empresa que los esté usando, lo cual quiere decir que también los datos que se generen lo son. Tener los datos almacenados de esta manera puede ocasionar problemas cuando se realizan informes o se intenta migrar a sistemas locales. Por lo general, si el cliente requiere realizar un análisis de inteligencia empresarial (BI) o reportes con los datos, estos se terminan trasmitiendo por internet. Lo cual suscita problemas de rendimiento y seguridad. Los reportes y análisis de inteligencia empresarial se procesan más rápido cuando se realizan en la misma ubicación que los datos.

Otra situación que puede resultar algo problemática es la integración de las aplicaciones en los modelos SaaS públicos. Esto se genera debido a que el proveedor de la aplicación debe exponer las API o los servicios web, para que una aplicación sea capaz de llamar a la funcionalidad de una aplicación con la finalidad de no repetir funcionalidades en dos aplicaciones diferentes.

Flexibilidad reducida: Cuando el cliente prefiere utilizar la oferta de un proveedor de nube pública está sujeto al calendario de actualización del proveedor. En muy pocas situaciones tendrá autoridad sobre cuando se realizan las actualizaciones. Por lo cual, los usuarios necesitarán tomar capacitaciones sobre las modificaciones en los sistemas, lo cual puede generar un impacto negativo en la productividad.

2.5.2 Nubes privadas

El modelo de despliegue de nube privada se hospeda en un centro de datos que la empresa es responsable de administrar y mantener. Lo cual implica que la empresa se debe hacer cargo de la compra, el mantenimiento y soporte. Muchas personas piensan que las nubes privadas no son nubes lo cual es completamente incorrecto, ya que si nos detenemos a realizar un análisis vemos que las características anteriormente expuestas por el NIST se cumplen. Aunque la propuesta de valor si sufre cambios. La propuesta de valor de la nube cambia cuando se habla de nubes privadas en lugar de nubes públicas; pero la propuesta de valor no determina si es una nube o no [15].

Las nubes privadas son completamente administradas y mantenidas por su organización. En general, toda la infraestructura para el entorno se hospedará en un centro de datos que el usuario controla. Por lo tanto, el usuario es responsable de la compra, el mantenimiento y el soporte.

Beneficios: Existen cuantiosos beneficios de optar por un modelo de despliegue de nube privada entre los que podemos encontrar la capacidad de control y monitoreo del entorno de nube.

Soporte y solución de problemas: Los modelos de despliegue de nube privada permiten acceder directamente a los sistemas para ejecutar rastreos de red, depuración o cualquier otra tarea que permita solucionar un problema. En contraste a un entorno de nube pública en donde se dificulta algo el acceso y depende del proveedor dar una solución al problema.

En teoría si la empresa realiza el soporte y la solución de problemas de manera local, tiene la capacidad de proporcionar tiempos de respuesta más rápidos, lo que le ayuda a mantener una mejor satisfacción del cliente y como bien sabemos esto es fundamental para mantener el éxito de una empresa.

Mantenimiento: Con este modelo de despliegue la empresa tiene el control total del ciclo de actualización. No se ve forzada a actualizar cuando el proveedor lo decida, puede decidir si la versión más reciente tiene alguna característica o funcionalidad que se sea de provecho en la empresa. También tiene control sobre cuando se realizan las actualizaciones mediante ventanas de mantenimiento programadas, lo cual ayuda a reducir el tiempo de interrupción de los sistemas.

Algunas empresas tienen entornos en donde se requieren ejecutar múltiples versiones de una aplicación por motivos de compatibilidad. Si la empresa no tuviera el control de los sistemas como sucede en este modelo no tendría la posibilidad de tener múltiples versiones de una aplicación. Esta flexibilidad alcanzada por el modelo de nube privada les brinda a las empresas una mayor capacidad para atender necesidades específicas de clientes.

Monitoreo: Debido al acceso total que brinda el modelo de nube privada a los sistemas, la empresa puede realizar cualquier monitoreo que requiera, puede monitorear el hardware del sistema, software y aplicaciones lo que proporciona una gran ventaja ya que la empresa puede tomar medidas preventivas para evitar las interrupciones, lo que se traduce en ser más proactivos con sus clientes.

Inconvenientes: El modelo de despliegue de nube privada aporta muchos beneficios a sus usuarios. Sin embargo, se debe tener en consideración que tiene algunos inconvenientes, los cuales se describen en esta sección. Al momento de decidir si este es el mejor modelo de despliegue para nuestra empresa se debe realizar un análisis de los beneficios y desventajas para determinar si es la solución adecuada.

Costo: La implementación de un modelo de nube privada requiere una inversión de capital sustancial. Se debe tomar en consideración un presupuesto en donde se cubra las necesidades actuales y necesidades futuras. También, se debe considerar la inversión en una infraestructura que pueda soportar las horas pico o los aumentos de carga repentinos. Una ventaja de este modelo es que permite tener los recursos

necesarios para soportar estas subidas de carga sin que los sistemas estén siempre en ejecución, siempre y cuando se tenga una manera de iniciarlos de manera automática.

Compatibilidad de hardware y software: El usuario que implemente este modelo de despliegue debe comprobar que el software que desea implementar sea compatible con su hardware. También debe cerciorarse que el software que quiere implementar es compatible con los clientes de su entorno. Pueden surgir situación en las que se requiera hardware especializado (almacenamiento, gráficos, entre otros.) para implementar una aplicación particular.

Experiencia necesaria: Una importante desventaja de este modelo es que se requiere experiencia en el sistema y las aplicaciones que se deseen implementar. Lo cual puede llevar a una capacitación y educación costosas. Esto debido a que el usuario de este modelo es responsable de la instalación, mantenimiento y soporte técnico, en consecuencia, se debe tener el conocimiento interno para ejecutar estas tareas o bien la capacidad de contratar personal o consultores externos.

La implementación de un entorno de nube privada requiere personal con habilidades especificas en almacenamiento, redes, seguridad y virtualización. Este tipo de personal puede ser difícil de encontrar.

Responsabilidades: En un modelo de despliegue de nube privada la responsabilidad es completa de la empresa. La empresa se debe encargar de mantener la infraestructura, el sistema y las aplicaciones de punto a punto. Debe de garantizar el funcionamiento óptimo de las aplicaciones que utilizan los clientes.

2.5.3 Nubes híbridas

A medida que el cómputo en la nube evoluciona, el modelo despliegue de nube híbrido tiende a ser el más usado. Muchas personas tienen la idea errónea de lo que es una nube híbrida y piensa que es un entorno en donde algunos componentes son privados y otros son públicos. Esto es incorrecto, una nube híbrida es aquella en la que varios entornos de nubes independientes se interconectan entre sí.

El modelo de nube híbrida tiene la virtud de ofrecer lo mejor de ambos mundos, así como sus desventajas. Este modelo ofrece la libertad de implementar lo que se requieran para satisfacer las necesidades de una empresa, sin embargo, pueden resultar costosas y complejas de implementar [15].

Beneficios: El modelo de nube híbrida brinda una mayor flexibilidad para situaciones en donde una empresa quiera migrar sus sistemas a un proveedor de nube público, mediante este modelo la empresa lo puede hacer por partes sin necesidad de migrar todas las aplicaciones por completo, hasta que el momento

sea adecuado. Esto es rentable cuando hay ofertas de aplicaciones en nubes públicas que resulten muy costosas. Al utilizar este modelo de nube híbrida, se pueden tener las aplicaciones internas, hasta que el precio se reduzca.

Otro de los motivos para utilizar este modelo es que permite dejar ciertos datos de manera interna, hasta que la empresa se consolide de que es seguro almacenar datos en un entorno de nube pública. Finalmente, algunas empresas utilizan este modelo para obtener mayor tolerancia a fallos y alta disponibilidad, ya que se pueden tener ciertas aplicaciones en distintos entornos. De esta manera, si algún entrono llega a fallar, la aplicación sigue disponible en algún entorno alterno.

Inconvenientes: El modelo de despliegue híbrido, es uno de los más complejos de implementar. Se tienen que tomar en consideración diferentes reglas dependiendo del tipo de entorno que se busque implementar. No hay reglas o procedimientos que apliquen a todos los entornos. Requiere el desarrollo de reglas y procedimientos para cada entorno.

Integración: Existen entornos en los que algunas aplicaciones están en una nube pública y otras en una privada, lo cual puede crear la necesidad de que estas aplicaciones accedan y utilicen los mismos datos. Hay dos posibles opciones, la primera es mover datos según se requiera. Pero esto genera el problema de preocuparse por el ancho de banda. La segunda opción es duplicar copias de datos, lo cual requiere configurar un mecanismo de replicación para sincronizar.

Consideraciones de Seguridad: Finalmente, cuando se implementa este modelo de despliegue hay que tener en consideración que no solo debemos preocuparnos por los problemas de seguridad en los entornos individuales, también hay que considerar los problemas que puedan surgir al conectar dos o más entornos.

2.6 Pronóstico y crecimiento de la nube

En las secciones anteriores se definió que es la nube, sus características claves, diferentes modelos de servicios y modelos de despliegue. En esta sección se muestran los pronósticos y estimaciones del crecimiento, uso y ganancias, lo cual permite visualizar el posible crecimiento de este paradigma, así como el impacto que está teniendo en los mercados y las ganancias que obtienen los proveedores de nube que ofrecen dichos servicios.

El primer informe que se presenta en la figura 5, muestra que un 55% de las ganancias operativas de Amazon del segundo trimestre del año 2018 provinieron de AWS a pesar de que solamente contribuyó con el 12% de sus ventas netas. En contraste al primer trimestre del año 2018, el cual representó un 40% de sus ganancias y aun de forma más notoria tres años atrás, en donde solamente representaba un 26% de sus ganancias [16].

Figura 5: Ganancias de la nube de Amazon *[16]*.

.

El segundo informe que se presenta en la figura 6, muestra el porcentaje de empresas que se encuentran probando o experimentando con diferentes proveedores de nube. En este informe se puede apreciar que AWS es la plataforma preferida por las empresas, con un 80% ejecutando o experimentando con estos servicios. En segundo lugar, se encuentra Microsoft Azure con un 67% de empresas, ejecutando o experimentando con dichos servicios. Por último, en tercer lugar, se encuentra Google Cloud con un 41% [16].

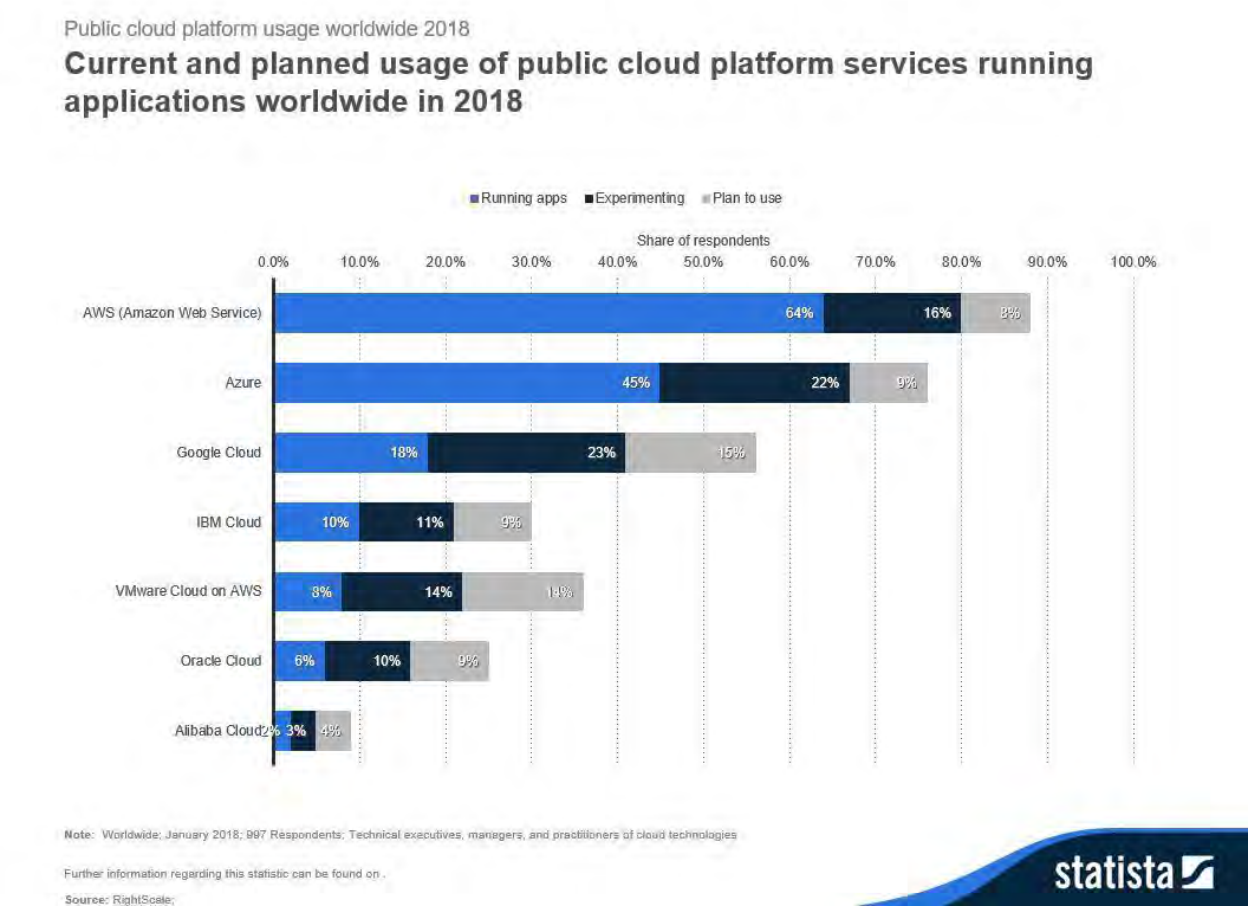

Figura 6: Uso de la nube en el año 2018 y planes de migración a la nube *[16]*.

El tercer informe que se muestra en la figura 7, indica que en el segundo trimestre del 2018 se estaban generando 20,000 millones de dólares en ingresos trimestrales para los proveedores de SaaS empresarial, y tienen un crecimiento anual del 32%. En este informe, se puede ver como Microsoft está lidereando este sector, con una cuota del 17% de uso en el segundo trimestre del 2018 [16].

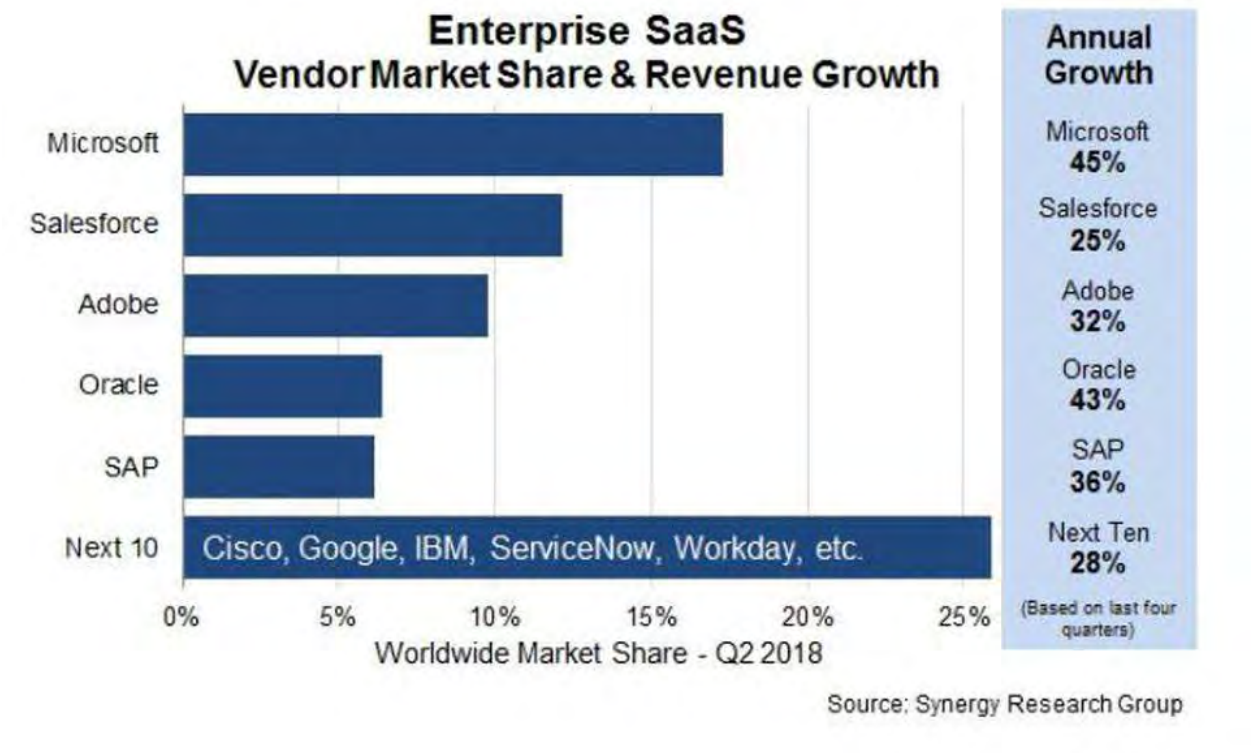

Figura 7: Uso de SaaS empresarial en el segundo trimestre 2018 *[16]*.

Para concluir, el cuarto informe que se muestra en la figura 8, indica que se espera un crecimiento a nivel mundial de los servicios de nube pública en un 17.3% para el año 2019, lo cual representa 206,200 millones dólares frente a los 175,800 millones de dólares del año 2018. Para el 2018, se espera un crecimiento del 21% comparado con los 145.3 mil millones de dólares del 2017. De acuerdo con esta firma de investigación y asesoría para el 2022, se espera que el 90% de las empresas que compren servicios de IaaS en lo nube pública, lo hagan a partir de una IaaS y PaaS integrada [16].

Table 1. Worldwide Public Cloud Service Revenue Forecast (Billions of U.S. Dollars)

	2017	2018	2019	2020	2021
Cloud Business Process Services (BPaaS)	42.2	46.6	50.3	54.1	58.1
Cloud Application Infrastructure Services (PaaS)	11.9	15.2	18.8	23.0	27.7
Cloud Application Services (SaaS)	58.8	72.2	85.1	98.9	113.1
Cloud Management and Security Services	8.7	10.7	12.5	14.4	16.3
Cloud System Infrastructure Services (IaaS)	23.6	31.0	39.5	49.9	63.0
Total Market	**145.3**	**175.8**	**206.2**	**240.3**	**278.3**

BPaaS = business process as a service; IaaS = infrastructure as a service; PaaS = platform as a service; SaaS = software as a service
Note: Totals may not add up due to rounding.

Source: Gartner (September 2018)

Figura 8: Pronóstico de las ganancias e ingresos de la nube en último trimestre 2018 *[16]*.

2.7 Contenedores

Los contenedores, son una tecnología de virtualización ligera o virtualización a nivel de sistema operativo. Ellos son una respuesta a las problemáticas que presenta las soluciones basadas en tecnologías de máquinas virtuales, tales como falta de portabilidad, lentitud para el aprovisionamiento de recursos y densidad de las máquinas virtuales. En el caso de la tecnología de máquinas virtuales, estas utilizan un hipervisor que se encarga de proporcionar aislamiento de los recursos, para cada máquina virtual a nivel de hardware, incluyendo cada una de estas máquinas un sistema operativo completo, con aplicaciones y datos en adición al sistema operativo anfitrión [17]. En contraste, en la tecnología de contenedores, estos comparten el núcleo y los recursos del sistema operativo anfitrión, lo que les permite ocupar menos recursos e iniciar más rápido que una máquina virtual, ya que son específicos a una familia del sistema operativo en particular [18]. Actualmente se clasifican en contenedores de sistema operativo y contenedores de aplicación. Para fines de este trabajo recepcional sólo se consideran los contenedores de aplicación [19].

2.7.1 Contenedores de aplicación

Los contenedores de aplicación, también conocidos como virtualización de aplicación, tienen como objetivo la ejecución de una o más aplicaciones por cada instancia virtual. En la actualidad se utilizan para empaquetar una aplicación o servicios con sus dependencias, archivos de configuración y librerías que son requeridas para su funcionamiento [17]. En contraste con las arquitecturas de aplicaciones tradicionales, las cuales se dividen en capas y cada capa tiene un servidor o máquina virtual, las arquitecturas de contenedores por lo general, tienen una aplicación que se divide en componentes, cada uno de estos componentes tiene una función en específico y se ejecutan en un contenedor separado [20]. Al grupo de contenedores, que trabajan en conjunto para proveer la funcionalidad de la aplicación, se les conoce como microservicios [19].

2.7.2 Funcionamiento de contenedores

Creación de imágenes

Durante la primera fase del desarrollo de la contenerización de la aplicación, los componentes de la aplicación se crean y se ponen en una o más imágenes. Las imágenes son paquetes en los que se incluyen todos los archivos necesarios para la ejecución de un contenedor (bibliotecas, binarios y archivos de configuración, entre otros) [17]. En ellas se emplea el concepto de inmutabilidad, es decir, funcionan como entidades en las que no se efectúan modificaciones. Una vez que se dispone de una imagen actualizada, se realiza una nueva implementación en el contenedor [20].

Pruebas y acreditación de imágenes

La segunda fase consiste en el proceso de pruebas y acreditación de imágenes. Esta fase generalmente es administrada por los desarrolladores que se encargaron de empaquetar la aplicación en la primera fase. Inmediatamente después de realizar el proceso del ensamble de imagen, se utilizan herramientas de automatización de pruebas, dichas herramientas realizan pruebas para la validación del funcionamiento de la aplicación final. Finalmente, el equipo de seguridad se encarga de realizar la acreditación de las imágenes [21].

Almacenamiento y recuperación de imágenes

Una vez que se ha creado la imagen y realizado el proceso de acreditación y pruebas, se requiere de algún lugar en donde almacenar y recuperar las imágenes. Es aquí en donde la tercera fase del ciclo nos proporciona servicios que se encargan de esta tarea [19]. Los registros son servicios que proporcionan a los desarrolladores, una manera de almacenar imágenes de forma sencilla. Estos servicios se pueden utilizar en sitio o consumidos desde un proveedor como Amazon o Docker [22].

Gestión de administradores de contenedores

En la última fase se realiza la extracción de las imágenes de los registros, lo cual se puede llevar a cabo por un administrador de sistema o por un disparador que se encuentre en un proceso de automatización [22]. Posterior a que se extrajo la imagen, el orquestador se encarga de implementar y gestionar los contenedores en los administradores de contenedores. Esta última fase es la responsable de entregar una aplicación utilizable, lista para responder a las solicitudes de los usuarios [20].

Implementación de contenedores

Existen cuatro maneras principales de implementar los contenedores. La primera opción es una implementación local. La segunda es una implementación local utilizando un hipervisor tipo 1, este se ejecuta directamente sobre el hardware. La tercera es utilizar un hipervisor tipo 2, el cual se ejecuta sobre un sistema operativo anfitrión. Finalmente, la última opción es utilizar un proveedor de nube en donde montamos al contenedor [18].

2.7.3 Mecanismos de seguridad en contenedores

Existe dos mecanismos de seguridad principales para la protección de los contenedores. El primero es utilizar software, el cual generalmente se encuentra en las características y módulos de los sistemas operativos Linux. El segundo es utilizar hardware, el cual generalmente está en la máquina anfitrión en la que se ejecuta el contenedor.

Soluciones basadas en software

Las soluciones basadas en software implementan mecanismos de seguridad en forma de características y módulos para la protección de contenedores. Estos mecanismos se clasifican en características de seguridad Linux (LSF), para la protección de contenedores que se encuentran por defecto en el núcleo de Linux y están formadas de cuatro componentes principales: (Espacios de nombres, CGroups, Capacidades y SecComp) [20]. En cuanto a los módulos de seguridad Linux (LSM), son módulos adicionales que los usuarios pueden cargar o configurar para una protección adicional a los contenedores, siendo SELinux y AppArmor los más utilizados en la literatura [19].

Espacios de nombre (Namespaces)

Los espacios de nombres se encargan de aislar y virtualizar los recursos con los que un contenedor puede interactuar [22]. Esto incluye los sistemas de archivos, interfaces de red, nombre del anfitrión, información de usuario y procesos [20]. Es decir, garantiza que los contenedores solo vean su lista de recursos y no puedan ver los procesos de ejecución de otros contenedores [19]. Por lo tanto, se evita que un contenedor no consiga accesos privilegiados a los sockets o interfaces de otro contenedor.

CGroups

Esta característica de Linux establece cuántos recursos de procesamiento (CPU), memoria, entrada/salida (E/S) y ancho de banda es asignado a cada proceso. Se encargan de limitar cuantos recursos puede utilizar un contenedor, a diferencia de los nombres de espacio que se encargan de delimitar que recursos pueden ver los contenedores [22]. Esto garantiza que un contenedor no consuma por completo los recursos de un sistema y deje sin recursos a otros contenedores o procesos [20]. Por ejemplo, un sistema operativo anfitrión con 16 GB de memoria total y 12 contenedores gestionados por el administrador de contenedores, se puede asignar 1 GB a cada contenedor, lo que evita que interfieran con las operaciones de otros contenedores [19].

Capacidades (Capabilities)

Las capacidades de Linux desglosan los permisos por capas o niveles sencillos de utilizar, lo cual evita que se asignen permisos innecesarios. Por ejemplo, en un servidor web dentro de un contenedor, el cual requiere vincularse a un puerto en específico, generalmente el puerto TCP 80, esto requiere de la asignación de permisos de super usuario. En el ejemplo anterior, se puede asignar la capacidad de CAP_NET_BIND_SERVICE, lo que permite al contenedor realizar la acción de vincularse al puerto en específico, sin asignarle permisos de super usuario a todo el contenedor. Al tener esta capacidad habilitada, se limitarán los ataques exclusivamente a operaciones de puertos de enlace [20].

SecComp

Los perfiles de Secure Computing (SecComp), son una característica del núcleo Linux para filtrar las llamadas del sistema, lo cual reduce las posibles amenazas ya que la mayoría de los ataques se aprovechan de las vulnerabilidades que ocurren en las llamadas al sistema [20]. Docker incluye perfiles predeterminados, que eliminan llamadas inseguras e innecesarias en el sistema, asimismo se pueden crear perfiles personalizados y ponerlos en ejecución para limitar aún más las capacidades del contenedor [19].

Control de Acceso Obligatorio (MAC)

MAC determina el acceso a los recursos mediante políticas de acceso, las cuales son controladas por un administrador de sistema y son utilizados por tecnologías como SELinux y AppArmor, lo que permite que MAC brinde aislamiento [17]. Por ejemplo, se puede utilizar una de estas tecnologías para delimitar el acceso que tienen los contenedores a directorios de archivos, procesos y sockets de red [19]. En consecuencia, se limita la capacidad que tiene un contendor para afectar a otros contenedores o máquina anfitrión.

2.7.4 Contenedores Docker

Los contenedores Docker son una alternativa de virtualización más ligera a la tradicional. Esto debido a que las imágenes que se creen utilizando esta tecnología comparten el mismo núcleo que el sistema operativo huésped. En contraste las imágenes que se crean usando máquinas virtuales tradicionales, las cuales ejecutan un sistema operativo completo e independiente.

Los contenedores son más ligeros y utilizan menos recursos de sistema. Eliminan el consumo de recursos de un hipervisor tradicional, lo que permite lograr un rendimiento muy similar al que se logra al implementar una aplicación en un sistema operativo base. El tiempo en que tarde en iniciar una aplicación contenerizada usando Docker es más rápida a comparación de iniciar una máquina virtual, esto debido a la poca sobrecarga que generan los contenedores [22].

Además de esto, los contenedores permiten la implementación de cientos de contenedores en pocos segundos, lo cual reduce el tiempo requerido para poner en marcha una aplicación. Esto resulta en una gran ventaja si los utilizamos en una arquitectura de microservicios. La cual nos permite tener pequeños fragmentos de software que son fáciles de mantener, reutilizar e implementar.

En resumen, un contenedor Docker es un paquete que tiene lo necesario para ejecución de una aplicación de forma independiente. Esto se logra almacenando la aplicación y sus dependencias (archivos binarios, librerías, archivos de configuración, scripts) en un solo paquete. Se pueden tener varios contenedores Docker en una sola máquina y están completamente aislados uno de otros, así como de la máquina que ejecuta el administrador. En la siguiente sección se discute la arquitectura de Docker [22].

Arquitectura Docker

Una vez que una aplicación ha sido contenerizada se puede caer en la confusión de que es utilizable en cualquier entorno. Si bien esto se cumple en gran medida, se debe tener en consideración cuales proveedores de nube soportan la tecnología Docker o bien contemplar su instalación en los servidores locales de la compañía. No obstante, debido al gran crecimiento que ha tenido Docker es soportando en un gran número de proveedores de nube. Sin importar si realizamos una implementación con un proveedor de nube o en servidores locales, Docker utiliza una serie de componentes para su funcionamiento [23]:

Imágenes Docker: Son una colección de todos los archivos que conforman a una aplicación de software (archivos binarios, librerías, scripts, etc.). Las imágenes Docker se originan a partir de una imagen base y cada cambio que se realice a la imagen original se almacena en una capa separada. Cada vez que se guardan cambios en una imagen de Docker se crea una nueva capa. En otras palabras, la imagen original y cada capa preexistente permanecen sin cambios y son típicamente de solo lectura.

Contenedor Docker: El término contenedor tiene una analogía similar al envío de contenedores. Esto quiere decir que debe ser posible enviar contenedores desde un entorno de desarrollo a un entorno de implementación y las aplicaciones que estén en los contenedores deben comportarse de la misma manera. El contenedor Docker comparte similitudes a una instancia de una máquina virtual tradicional con la diferencia significativa de que para la ejecución de sus procesos comparte el mismo núcleo del sistema operativo anfitrión.

Capa Docker: Para entender que es una capa Docker, primero es importante resaltar que los contenedores se crean a través de la confirmación de cambios en una imagen Docker. Cada vez que se inicia un contenedor se está haciendo referencia a un ID de imagen único y es aquí en donde las capas entran en función. Las capas Docker representa las imágenes que pueden ser de solo lectura o imágenes que son de escritura-lectura.

Proceso Docker: Se encarga de ejecutar las solicitudes de la API Docker y administración de objetos Docker (imágenes, contenedores, redes y volúmenes). Los procesos Docker se pueden comunicar con otros procesos y a diferencia del cliente utiliza privilegios de administrador o super usuario.

Cliente Docker: Se encarga de la interacción con el proceso Docker utilizando una API la cual es responsable de inicializar o administrar los contenedores. Por ejemplo, cuando un usuario ingresa el comando “Docker Run” el cliente se encarga de enviar el comando al proceso de Docker, el cual se encarga de la ejecución.

Repositorio Docker: Su funcionamiento es similar a un sistema de control de versiones con la diferencia que en este sistema se guardan imágenes Docker con nombre y sus diferentes versiones. Igual que en los sistemas de control de versiones los usuarios pueden optar por utilizar diferentes sistemas para llevar un control y almacenamiento de las imágenes Docker [23].

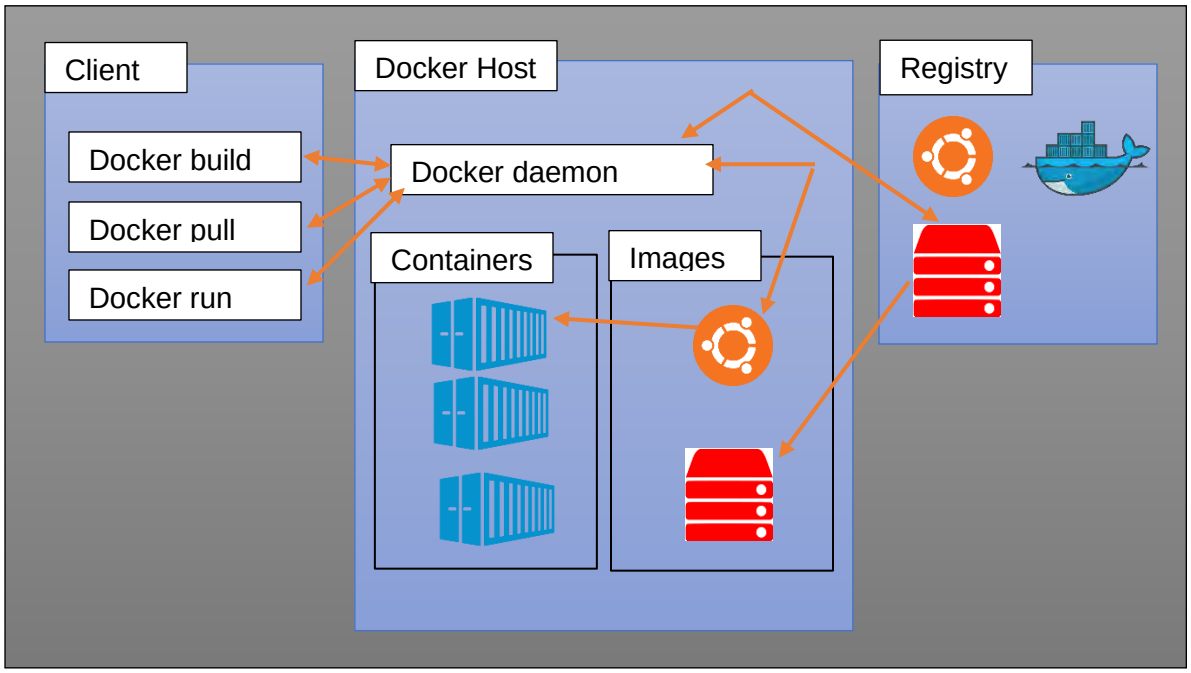

Figura 9: Diagrama de arquitectura Docker

2.8 Orquestación de contenedores

La orquestación de contenedores no solamente se encarga de encender o apagar contenedores y moverlos entre servidores. La orquestación se puede definir como la construcción o la administración de clústeres de contenedores en donde el usuario selecciona, implementa monitorea y controla de manera dinámica los contenedores que se pueden encontrar en entornos locales, virtuales o en la nube. En otras palabras, los orquestadores se encargan de administrar el despliegue, la ejecución y mantenimiento de los contenedores. Para lo cual los orquestadores ofrecen una serie de características que ayudan los usuarios en dichas tareas [17].

La primera característica que ofrece el orquestador es el límite de recursos. Esta característica permite reservar una cantidad especifica de cómputo (CPU y RAM) para los contenedores, lo que evita la interferencia de un contenedor con otros contenedores en ejecución. La segunda característica es la programación la cual se encarga de poner los contenedores que necesitemos en los nodos que necesite una aplicación en un tiempo específico. La tercera característica es el balanceador de carga el cual se encarga de distribuir la carga de trabajo entre las múltiples instancias de contenedores. La cuarta característica es el auto escalamiento que permite al orquestador agregar o remover instancias de contenedores de manera automática [18].

Como se puede ver los orquestadores son de vital importancia al momento de realizar desarrollos grandes en donde se tiene la necesidad de usar múltiples instancias de contenedores. Por tal motivo, existen soluciones para utilizarlos en entornos locales, virtualizados o en la nube. Entre algunos de los orquestadores más populares se encuentra Docker Swarm y Kubernetes [24].

2.8.1 Kubernetes

Kurbenetes es un proyecto de código libre cuya responsabilidad es la orquestación de contenedores. Es decir, se encarga de administrar y orquestar aplicaciones contenerizadas en clústeres. Tiene una interacción directa con Docker y se encarga de actualizar, ejecutar, mover, escalar y eliminar contenedores Docker de los clústeres. Adicionalmente, Kubernetes debe monitorear los contenedores en ejecución y garantizar que solo los contenedores responsivos estén en ejecución. Está conformado por una serie de entorno virtuales que tiene un nivel de abstracción diferente. Para comprender como funciona se explicarán algunos de sus conceptos y como es que interactúan entre ellos [24].

Cluster: Es una colección que Kubernetes usa para la ejecución de las diferentes cargas de trabajo de sus sistemas y está conformado por recursos de red y almacenamiento de máquinas anfitriones. Se puede tener sistemas en donde se utilicen uno o múltiples clústeres.

Nodo: Consiste en una maquina anfitrión que puede ser física o virtual. El trabajo que desempeña el nodo es la ejecución de los pods y cada uno de los nodos puede ejecutar diferentes componentes de Kuberntes como lo son Kubelet o un proxy Kube.

Master: Es el panel de control de Kurbenetes y está conformado por componentes como el servidor API, programador y el administrador de controladores. Desempeña el rol de la programación global a nivel de del clúster de los pods y el manejo de eventos.

Pod: Son las unidades de trabajo más pequeñas en Kubernetes. Los pods contienen uno o más contenedores los cuales se ejecutan en la misma máquina. Todos los contendores que pertenezcan al mismo pod tienen la misma dirección IP y puerto, por lo cual se puede comunicar usando localhost.

Etiqueta: Son pares de clave-valor que son usados para agrupar conjuntos de objetos, por lo general pods. Las etiquetas tienen una relación NxN entre objetos y etiquetas, esto quiere decir que cada objeto puede tener varias etiquetas y cada etiqueta puede aplicar a diferentes objetos.

2.9 Modelos y paradigmas de desarrollo de software

Los modelos de desarrollo de software permiten la administración en el proceso de desarrollo de software desde su fase de conceptualización hasta su implementación dentro de un presupuesto y tiempo acordados. Ahora bien, dentro de los modelos más utilizados por las empresas se encuentra los modelos ágil y cascada [25].

2.9.1 Modelo cascada

El modelo en cascada el cual se muestra en la figura 10 es un proceso secuencial que ha sido utilizado durante mucho tiempo para el desarrollo de software. Tiene como ventajas una comprensión sencilla y facilidad para su administración ya que es un proceso secuencial que se sigue en orden. No obstante, es un modelo que solamente es funcional en donde se encuentran requisitos predefinidos y fijos, ya que es un modelo dirigido con procesos en secuencia y no permite regresar a una fase anterior para realizar cambios. Asimismo, no permite las revisiones modulares, la integración de comentarios, la adaptación en proyectos grandes o la entrega de pequeños fragmentos funcionales de un proyecto [25].

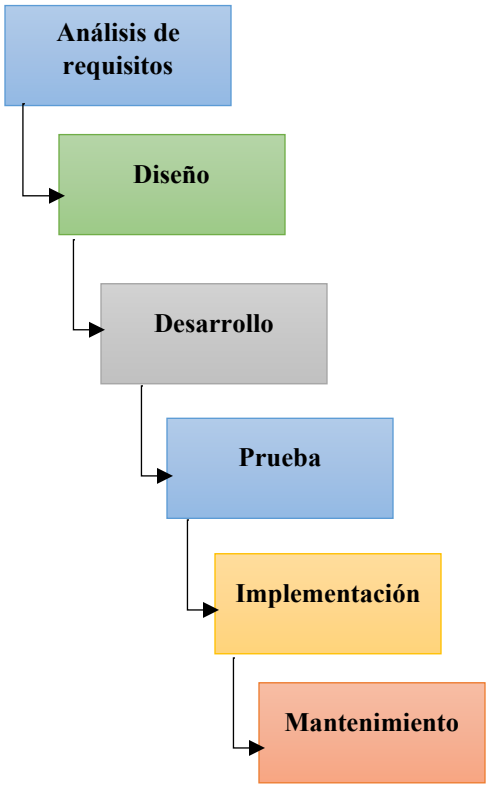

Figura 10: Modelo en cascada (desarrollo de software)

2.9.2 Modelo ágil

El modelo ágil nace como una respuesta a los problemas que se generan por el uso del modelo en cascada tal como los tiempos largos de comercialización, la estimación ineficiente de requisitos. En contraste el modelo ágil el cual se presenta en la figura 11, es un método que fomenta las interacciones, la colaboración con el cliente mediante la retroalimentación continua para mejorar pasos anteriores y con ello tener una respuesta a cambios de manera eficiente. Es decir, busca la satisfacción del cliente mediante la entrega continua en pequeñas iteraciones funcionales en plazos de tiempo cortos.

Después de esta breve introducción de los modelos de software y su papel en el desarrollo de software en la siguiente sección se explica que es DevOps, su ciclo de vida y la similitud que comparte con el modelo de desarrollo ágil [25].

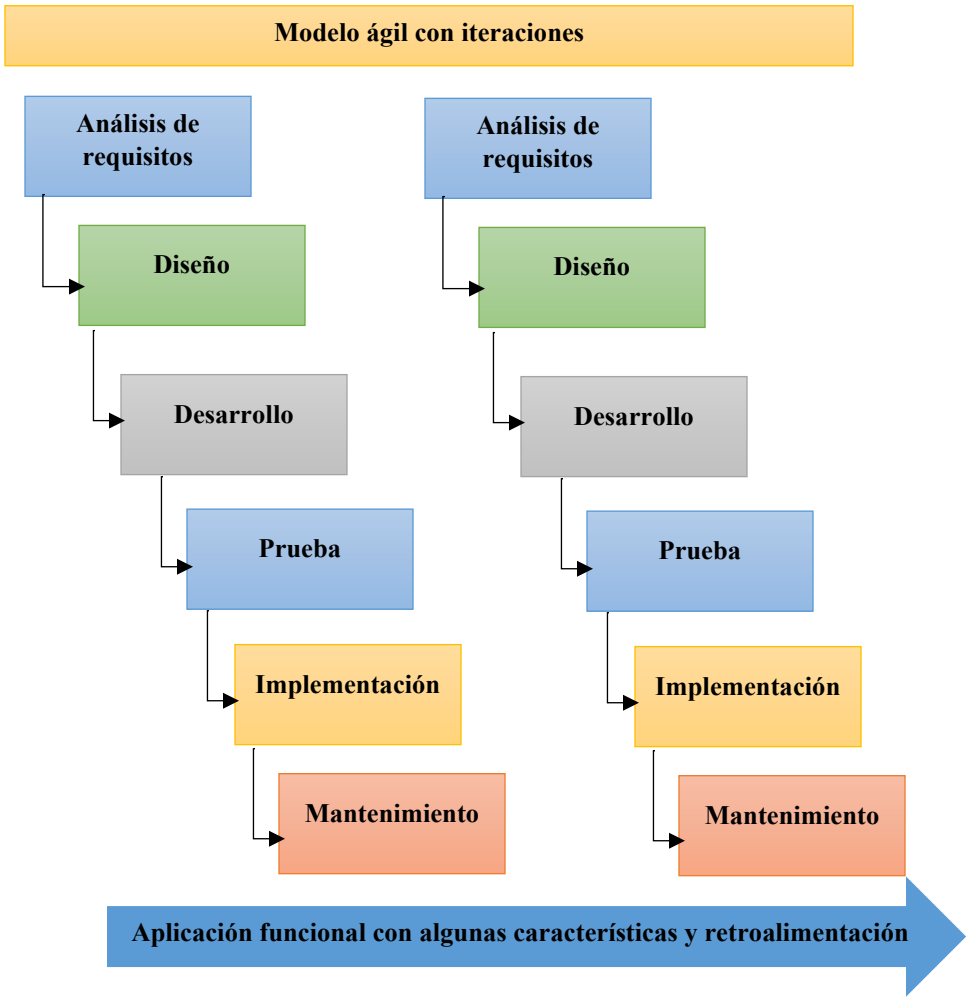

Figura 11: Modelo ágil (desarrollo de software)

2.9.3 DevOps

Existen diferentes opiniones acerca de la definición de DevOps, algunos autores lo definen como un concepto, cultura, desarrollo, filosofía y movimiento. En la industria lo definen como la combinación de filosofías culturales, prácticas y herramientas con la finalidad de incrementar la rapidez con la que una organización entrega servicios y aplicaciones a los usuarios finales. Por lo tanto, se puede inferir que DevOps es un paradigma que hace énfasis en la comunicación, colaboración y la interacción entre los equipos de desarrollo y operaciones.

Este paradigma está formado por las palabras “Development” y “Operations” y surge como una respuesta para resolver los desafíos de las empresas mediante un cambio cultural en el que se promueve el trabajo coordinado entre los equipos de desarrollo y operaciones en las organizaciones. En la figura 12 se presentan algunos de los desafíos a los que se enfrentan las organizaciones para el desarrollo de productos de software. Estos desafíos se generan en gran medida por que el desarrollo de software es un área que se encuentra en constante evolución y cambio, así como por la demanda de clientes que buscan una respuesta rápida a sus necesidades tecnológicas [26].

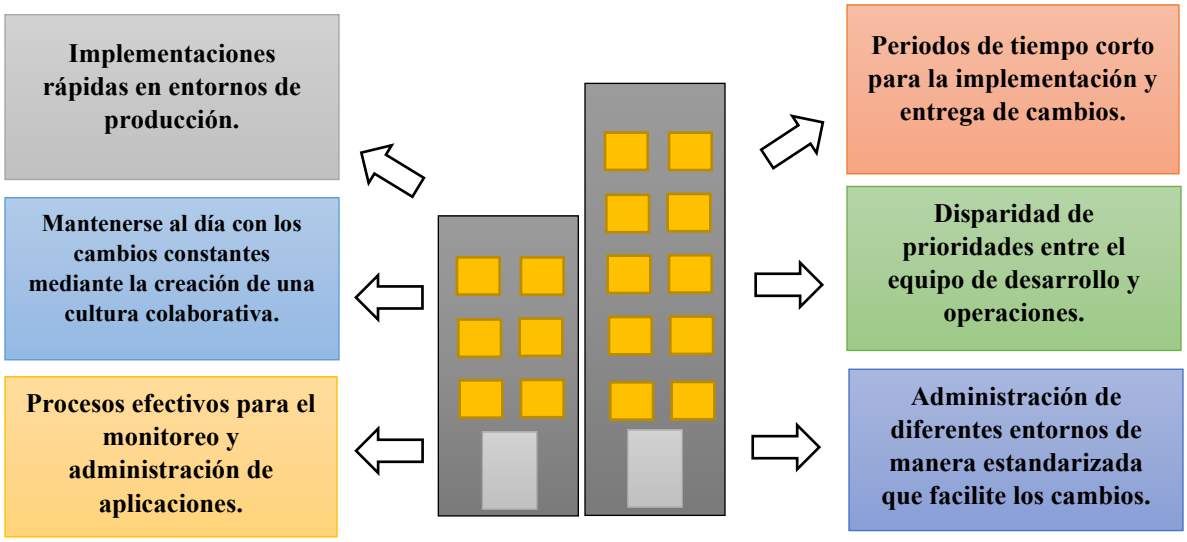

Figura 12: Desafíos actuales en las empresas

Ahora bien, en adición a los desafíos a los que se enfrentan las empresas es necesario mencionar que se generan una serie de problemas derivados del uso de metodologías de desarrollo obsoletos o tradicionales las cuales no están preparadas para los cambios constantes que demandan los clientes en la actualidad. Para comprender como surgen estos problemas se muestra el flujo de trabajo dentro de una empresa que usa metodologías o paradigmas tradicionales u obsoletos para el desarrollo de software.

El primer evento que ocurre es la asignación de un tarea o corrección al equipo de desarrollo. Después de esta asignación el equipo de desarrollo escribe el código de dicha tarea el cual usualmente se implementa y prueba en un entorno de desarrollo. El segundo evento es la implementación del código a una rama o repositorio de control de calidad donde el equipo de pruebas verifica el código. El tercer evento es proporcionar el código al equipo de operaciones para su implementación en el entorno de producción. Finalmente, el cuarto paso es el mantenimiento y operación del código el cual generalmente es llevado a cabo por el equipo de operaciones [26].

Como se puede ver esta forma de trabajo propicia que los diferentes equipos de una empresa trabajen de forma separada, derivando en las siguientes limitantes:

- La transición de la aplicación más reciente del entorno de desarrollo puede demorar semanas o meses en ser puesto a producción.
- Las prioridades entre los equipos de las empresas son diferentes. No es la misma prioridad que tienen el equipo de desarrollo y el equipo de operaciones.
- El equipo de desarrollo siempre se enfoca en la última versión del producto de software. En cambio, el equipo de operaciones se preocupa por la estabilidad.
- Se genera una separación entre los equipos de desarrollo y operaciones. Por lo cual, no conocen el trabajo y cultura laboral de los otros equipos.

DevOps busca dar respuesta y solución a todas estas limitantes mencionadas mediante la automatización en las diferentes fases del ciclo de vida del desarrollo de software. Esta automatización que forma parte del ciclo de vida DevOps permite que las empresas entreguen productos de software de forma rápida [27].

El ciclo de vida DevOps:

El ciclo de vida de DevOps está conformado por diferentes etapas las cuales se muestran en la figura 13. Para lograr acelerar y automatizar estas etapas DevOps tiene pilares importantes que permiten automatizar dichos procesos. Entre los pilares que permiten esta automatización se encuentra la integración continua (CI), las entregas continuas (CD) y las pruebas continuas (CT). Ahora bien, (CI) se ocupa de la automatización del proceso de compilación y empaquetado de la aplicación, (CT) se ocupa de automatizar las pruebas unitarias a la aplicación y finalmente (CD) se encarga de entregar la aplicación en los distintos entornos de la empresa [27].

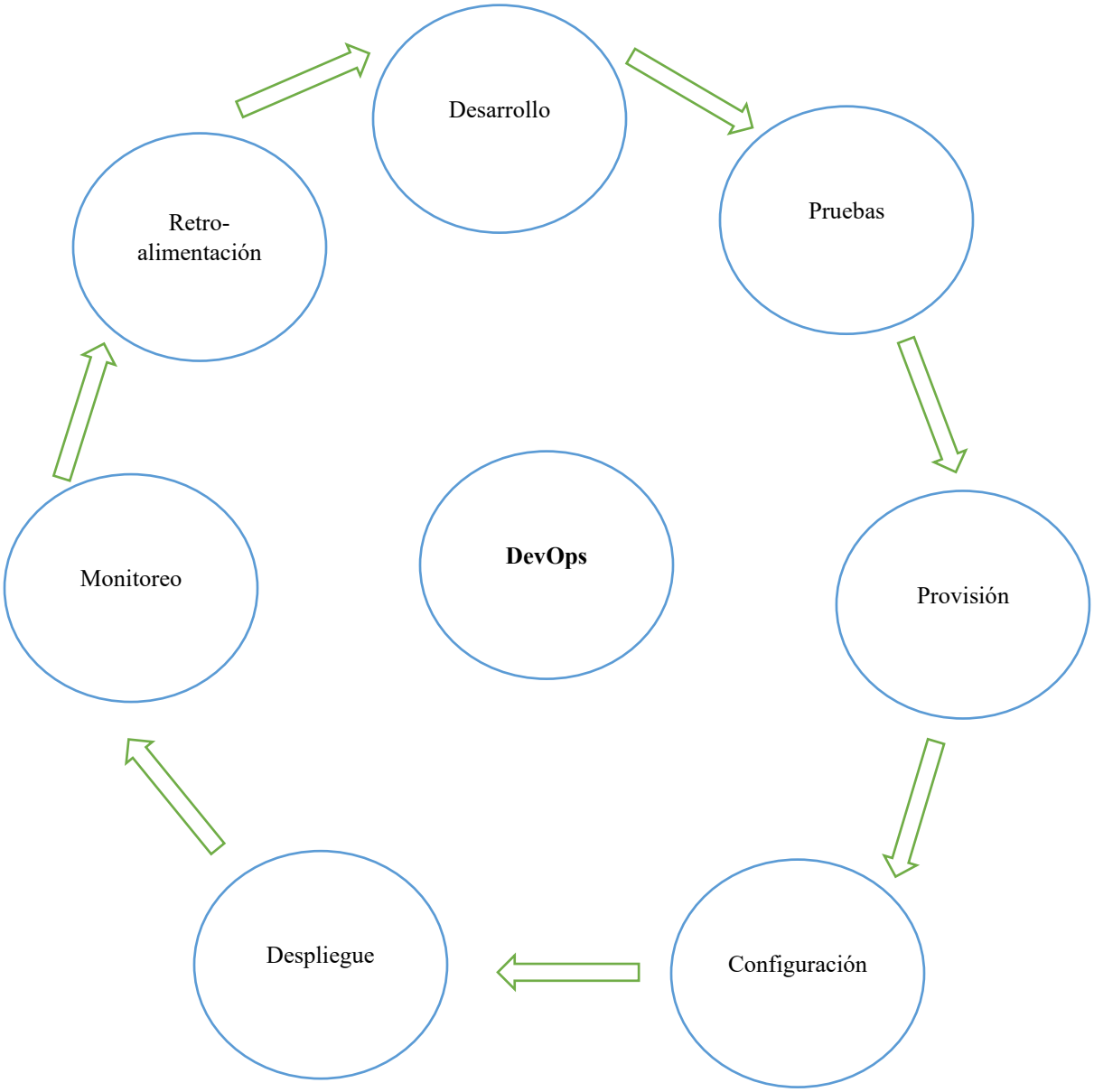

Figura 13: Ciclo de vida DevOps

Ventajas de DevOps:

El paradigma DevOps aporta un gran número de ventajas a las empresas que deciden utilizarlo ya que busca impulsar la comunicación y colaboración entre los distintos equipos de una empresa, siendo más concreto busca eliminar las barreras que se generan entre los equipos de desarrollo y operaciones dentro de una empresa que trabaja usando paradigmas o metodologías tradicionales. Esto se logra debido a que el empleo de este paradigma promueve el uso de nuevas metodologías, herramientas de automatización, recursos agiles de proveedores de servicios en la nube y muchas otras invocaciones, prácticas y tecnologías [27].

El uso del paradigma DevOps se acopla perfectamente con la utilización de un modelo d desarrollo ágil ya que permite a las empresas las entregas continuas de software en plazos de tiempo cortos. Lo cual resulta prioritario dentro de las empresas que buscan satisfacer las demandas actuales de los clientes que cada vez exigen cambios o solicitudes de forma más constante. Para terminar, en la figura 14 se muestran algunas ventajas inherentes al uso del paradigma DevOps [27].

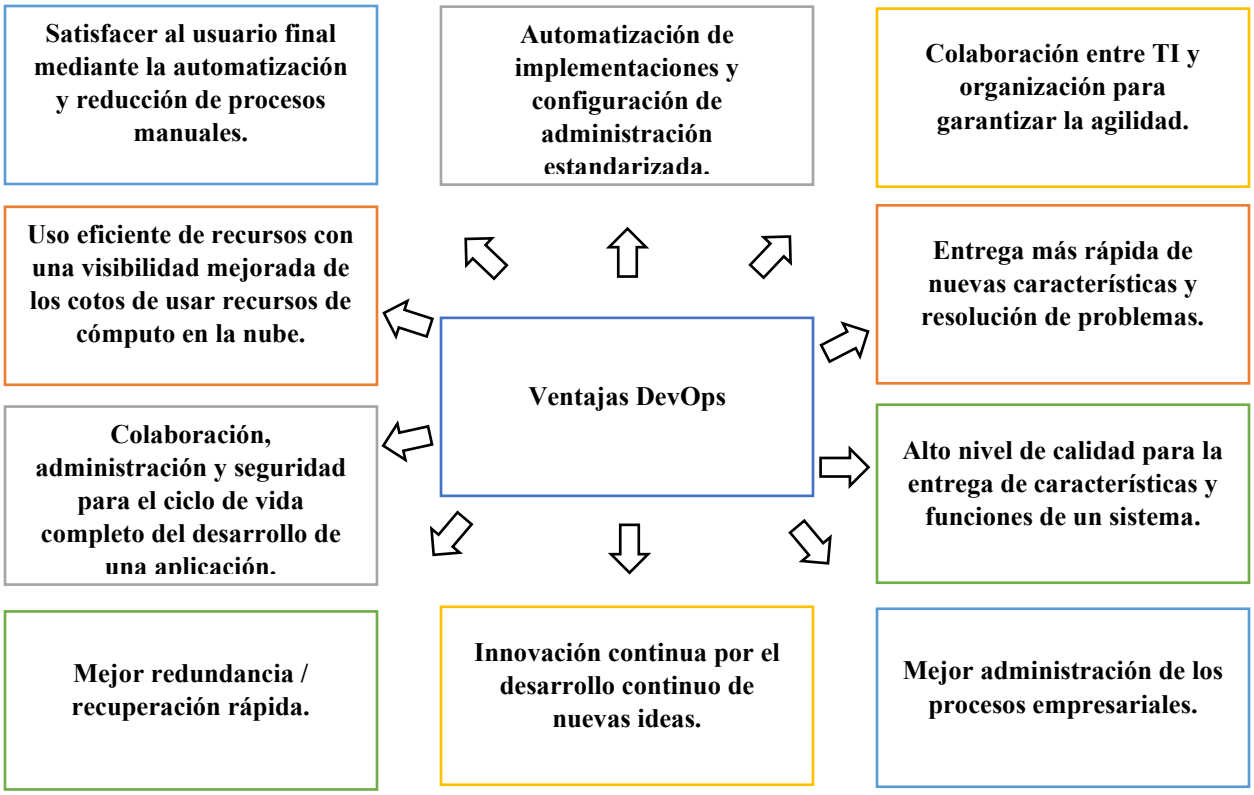

Figura 14: Ventajas de implementar DevOps para el desarrollo

2.10 Patrones de arquitectura de software

Una manera sencilla de comprender que son los patrones de arquitectura es mediante la analógica de la construcción de un edificio. Para la construcción del edificio fue necesario diseñar planos que ayudaran con la tarea de la construcción del edificio. De igual manera ocurre en el contexto del desarrollo de software, se podría decir que los patrones de arquitectura son los planos que determinan las características básicas y de comportamiento de una aplicación, es decir ayudan al ingeniero de software a desarrollar software de manera correcta y eficiente. Sin embargo, dentro del área de desarrollo de software hay una gran cantidad de patrones de arquitectura, cada uno de los cuales tienes sus características, ventajas y desventajas que los hacen ideales en diferentes escenarios de uso.

Dada esta gran cantidad de patrones de arquitectura existentes, la tarea de la selección de un patrón de arquitectura adecuado se puede volver una tarea compleja para el arquitecto de software. Sin embargo, existen diferentes libros [28], [29] y estudios [30], [31] que explican cada uno de estos patrones y factores que son necesarios tomar en cuenta al momento de su selección. Por tal motivo, en el presente trabajo no se evaluará cual patrón es más eficiente y solamente se tomará en cuenta el patrón de arquitectura basado en microservicios ya que los estudios anteriormente mencionados lo ponen como uno de los más novedosos y que aporta un gran número de ventajas en comparación de los patrones monolíticos.

En adición a esto, el artículo titulado “diseño eficiente de arquitectura para software como servicio (SaaS) en entornos de nube” resalta que muchas empresas en la actualidad necesitan vender e integrar servicios de manera continua lo cual ha ocasionado que las empresas busquen migrar patrones de arquitectura monolíticos a patrones de arquitecturas más flexibles como lo es el patrón de arquitectura de microservicios. En las siguientes secciones se discutirán únicamente el patrón de arquitectura en capas, los orientados en servicios y los microservicios ya que estos patrones representan los más utilizados y relevantes dentro de la industria del desarrollo de software [32].

2.10.1 Patrón de arquitectura en capas

El patrón de arquitectura basado en capas separa los componentes de un sistema en capas horizontales en la que cada capa desempeña una función dentro de una aplicación. Este patrón tiene la particularidad de que no establece un mínimo o máximo en la cantidad de capas que una aplicación puede tener. Por ejemplo, puede existir aplicaciones empresariales grandes en las que se utilicen cuatro, cinco o incluso más capas o por el contrario pueden existir aplicaciones pequeñas en las que se utilicen tres capas. No obstante, el uso de tres capas es de los más predominantes dentro del área de desarrollo de software.

En la figura 15 se presenta un diagrama de una aplicación en la que se está haciendo uso de un total de tres capas. La primera capa es la de presentación la cual se encarga de administrar la interfaz de usuario. La segunda capa es la de lógica de negocio es la encargada de ejecutar las reglas de un flujo de trabajo y finalmente la capa de la base de datos es utilizada por la capa de lógica de negocio para almacenar los datos de forma persistente.

Cada una de estas capas dentro de la arquitectura forma una abstracción alrededor de la aplicación para satisfacer una solicitud. Es decir, la capa de presentación no necesita preocuparse de cómo obtener datos de cliente, simplemente se encarga de mostrar la información al usuario. De esta misma manera, la capa de lógica de negocio no necesita preocuparse por como mostrar los datos. Solamente se encarga de obtener los datos de la capa de persistencia y realizar operaciones de la lógica del negocio [32].

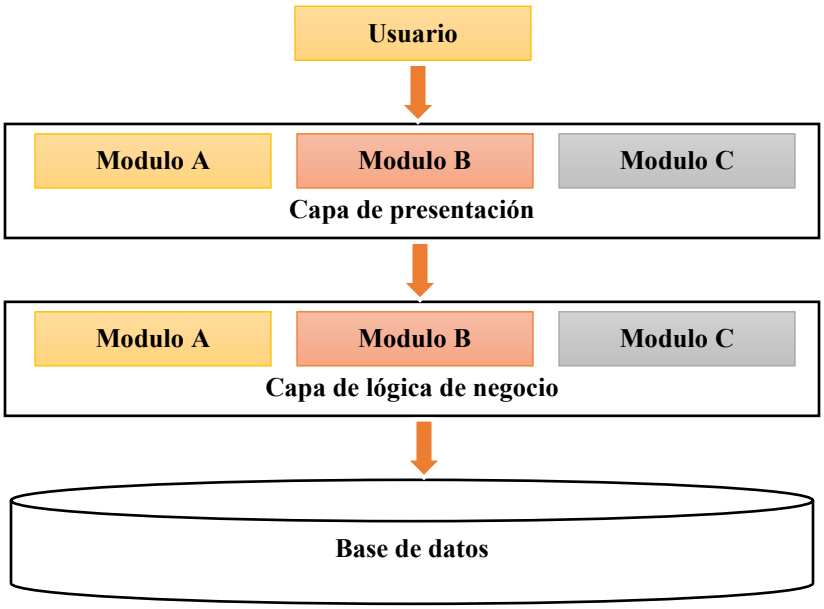

Figura 15: Arquitectura de aplicación monolítica

2.10.2 Patrón de arquitectura microservicios

El patrón de arquitectura de microservicios está ganando una gran aceptación dentro de la industria ya que es una alternativa a los patrones de arquitectura monolíticos y orientados a servicios. Dado que este patrón de arquitectura se encuentra en evolución, existe confusión o discrepancia acerca de cómo se implementa o de que trata. Sin embargo, existen dos conceptos claves que son independientes de la implementación o topología que se decía utilizar.

El primer concepto clave es entender que son las unidades implementadas, básicamente es separar cada componente de la arquitectura en una sola unidad independiente funcional, lo cual facilita las entregas eficaces y optimizadas, una mayor escalabilidad y un alto grado de modularidad dentro de los componentes de la aplicación. El segundo concepto clave es que es una arquitectura distribuida, esto quiere decir que cada uno de los componentes están desacoplados entre ellos y se comunican a través de algún protocolo de acceso (REST, SOAP, RMI, entre otros). Esta característica es lo que les permite a los microservicios tener una escalabilidad superior a otros patrones de arquitectura.

En la figura 16 se muestra como el patrón de microservicios aborda los problemas que se presentan en arquitectura monolítica separando la aplicación en múltiples unidades desplegables (componentes de servicio) que se pueden desarrollar, probar e implementar de forma individual [32].

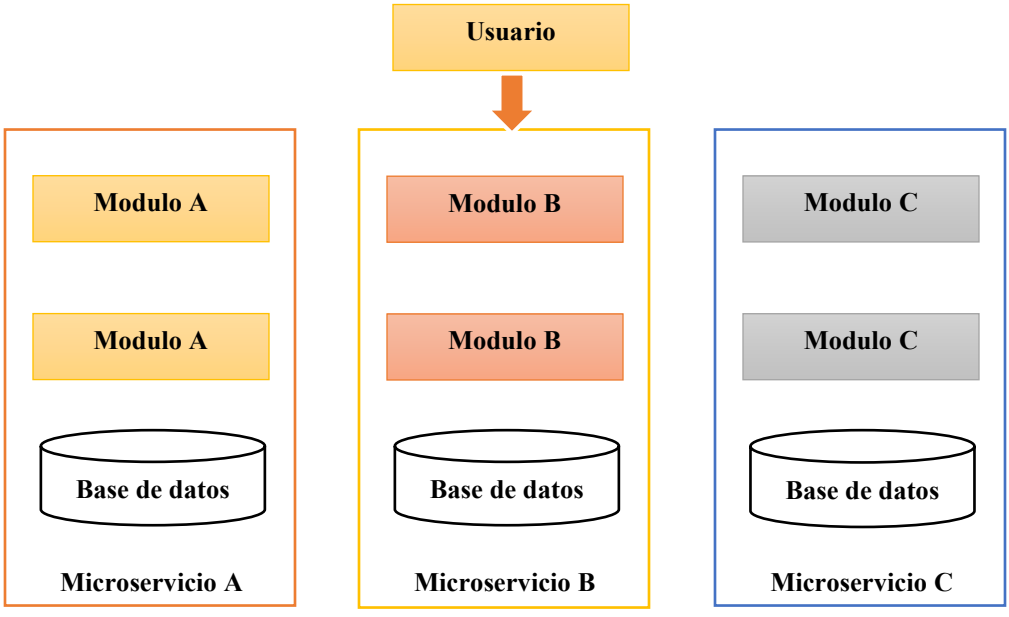

Figura 16: Arquitectura de aplicación basada en microservicios

2.11 Servicios de Amazon Web Services

Amazon EC2

El servicio de Amazon EC2 está dentro del modelo IaaS, ya que brinda a los usuarios capacidad de cómputo en la nube pública de Amazon. El servicio dispone de diferentes tipos instancias como propósito general, computación acelerada y computación optimizada. Por ejemplo, un usuario puede seleccionar una instancia de cómputo acelerado que permite el procesamiento basado en GPU. Esta instancia es de gran utilidad para la ejecución de aplicaciones de aprendizaje profundo las cuales se benefician de la aceleración proporcionada por el GPU. Aparte de los diferentes tipos de instancia, Amazon ofrece distintos distritos de Linux y versiones de Windows al momento de configurar la instancia EC2. En cuanto a los paquetes de software se tiene a disposición administradores de bases de datos o contendores.

Después de realizar la configuración del sistema operativo, paquete de software y tipo de instancia. Amazon EC2 permite configurar la memoria RAM, CPU y almacenamiento de la instancia EC2. Todas estas configuraciones se pueden agrupar en una máquina de Amazon (AMI) la cual se puede utilizar posteriormente para aprovisionar o desmontar múltiples instancias virtualizadas mediante llamadas al servicio web. Además, las instancias EC2 permiten reajustar la instancia o un grupo de instancias aumentando o reduciendo el hardware previamente configurado, lo cual permite que se ajusten a los requerimientos y necesidades de los usuarios [33].

Servicio de contenedor Amazon EC2

Este servicio es utilizado para la administración de configuración y clústeres para aplicaciones que utilizan contenedores. Permite iniciar y detener aplicaciones mediante llamadas API.

Amazon S3

Es un servicio de almacenamiento de datos con una disponibilidad y distribución alta. Permite almacenar y recuperar grandes cantidades de datos en forma de objetos en cubetas o contenedores. Estos objetos se pueden acceder desde la web mediante protocolo HTTP. Además, es una opción económica para almacenar grandes cantidades de datos que suelen utilizar aplicaciones de análisis o IA. Todos estos objetos y datos se pueden proteger mediante políticas de seguridad que admite Amazon S3 [33].

AWS Lambda

Este servicio permite la ejecución de código a eventos que ocurren dentro de una aplicación. Por ejemplo, lecturas inconsistentes en un sensor, clics en una página web, la carga de archivos, actualizaciones en los campos de una tabla o formulario, detección de anomalías en el funcionamiento de una aplicación o errores detectados en un log de eventos, entre otros. Además, permite la integración con el servicio SNS para dar respuesta o mandar mensajes de los eventos que ocurren [34].

Amazon DynamoDB

Este servicio ofrecido por Amazon proporciona base de datos NoSQL el cual soporta el modelo basado en documento o valores clave. Permite la integración con otros servicios de Amazon, como Data Pipeline el cual facilita mover los datos dentro y fuera de DynamoDB. Dado que se puede utilizar el servicio para aplicación Big Data y BI, se utiliza un particionamiento automático con SSD. Esto permite que el servicio de Amazon tenga un alto rendimiento y baja latencia cuando se necesite escalar la base de datos. Es un servicio totalmente administrado, lo cual significa que el usuario no emplea tiempo en realizar tareas de un DBA. Por ejemplo, Amazon se encarga de replicar de manera automática el servicio en tres instalaciones dentro de una región [34].

Amazon CloudWatch

Este servicio proporciona monitoreo para los otros servicios o recursos que se estén utilizando en AWS. Permite establecer alarmas en cuanto al consume de recursos, así como para manejo de errores. Además, permite obtener datos de monitoreo para solucionar problemas que se presenten en el entorno de nube.

Amazon Gestión de Identidad y Acceso (IAM)

Este servicio permite crear usuarios y grupos con credenciales de seguridad únicas lo cual permite controlar el acceso a los servicios y recursos de AWS. Además, se puede administrar permisos y roles para cada uno de estos usuarios. Estos roles permiten realizar llamadas API desde microservicios o aplicaciones de forma segura sin tener que crear y distribuir credenciales de AWS [34].

Capítulo III: Propuesta de marco técnico de referencia y metodología

En el presente capítulo se desarrolla una propuesta de marco técnico de referencia que permita establecer cuáles conceptos son necesarios para el desarrollo de software como servicio (SaaS) usando contenedores. Para posteriormente presentar la metodología, la cual toma los conceptos del marco técnico de referencia y los ordena con el propósito de exponer de manera sistematizada cómo se puede desarrollar software como servicio (SaaS) usando contenedores. Para terminar el capítulo III se muestra un caso de aplicación que permita validar lo propuesto de manera conceptual en el marco técnico de referencia y metodología.

Es importante mencionar que el marco técnico de referencia y la metodología presentados en este capítulo tienen como fundamento una serie de artículos y libros tales como: un modelo de proceso en el ciclo de vida de software en la nube [35], arquitecturas de software de la convergencia de la computación en la nube y el internet de las cosas [36], directrices de diseño para el desarrollo de SaaS [37], Aprendiendo AWS: diseño, construcción e implementación de aplicaciones responsivas en AWS [33], entre otros y la experiencia profesional de aplicar las tecnologías y conceptos presentados en este documento.

3.1 Marco técnico de referencia en el dominio de SaaS usando contenedores

El marco técnico de referencia reúne una serie de conceptos relacionados para explicar un evento determinado y con ello dar una compresión más precisa de un problema de investigación o un fenómeno de interés [38]. En este sentido, el marco técnico de referencia contiene únicamente la parte conceptual de cómo desarrollar software como servicio (SaaS) usando contenedores. Ahora bien, falta englobar dichos conceptos para explicar la manera de cómo se desarrolla software como servicio (SaaS) usando contenedores. Para ello se realiza una modelación en la que parte de los conceptos más generales a los más específicos, tal y como se presenta en la figura 17 en la que se utilizan una serie de colores para mostrar los diferentes niveles de profundidad del marco técnico de referencia [35].

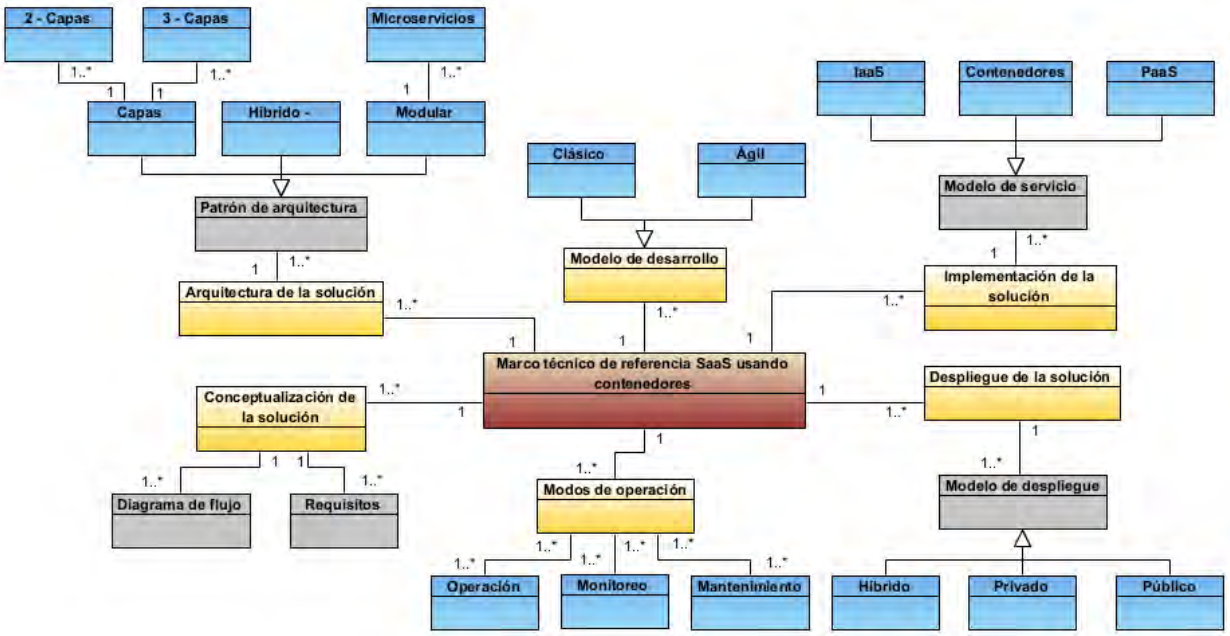

Figura 17: Modelación del marco técnico de referencia general

Como se puede observar en la modelación de la figura número 17, se muestra de manera general las diferentes opciones que tiene un usuario para cumplir con el objetivo desarrollar software como servicios (SaaS) usando contenedores. Sin embargo, es necesario diseñar una modelación específica que delimite el conjunto de conceptos que se utilizarán para el desarrollo del caso de aplicación, tal y como se muestra en la figura número 18. No obstante, el usuario que utilice el marco técnico de referencia puede adaptarlo a diferentes modelos de servicio, de despliegue y arquitecturas de solución [35].

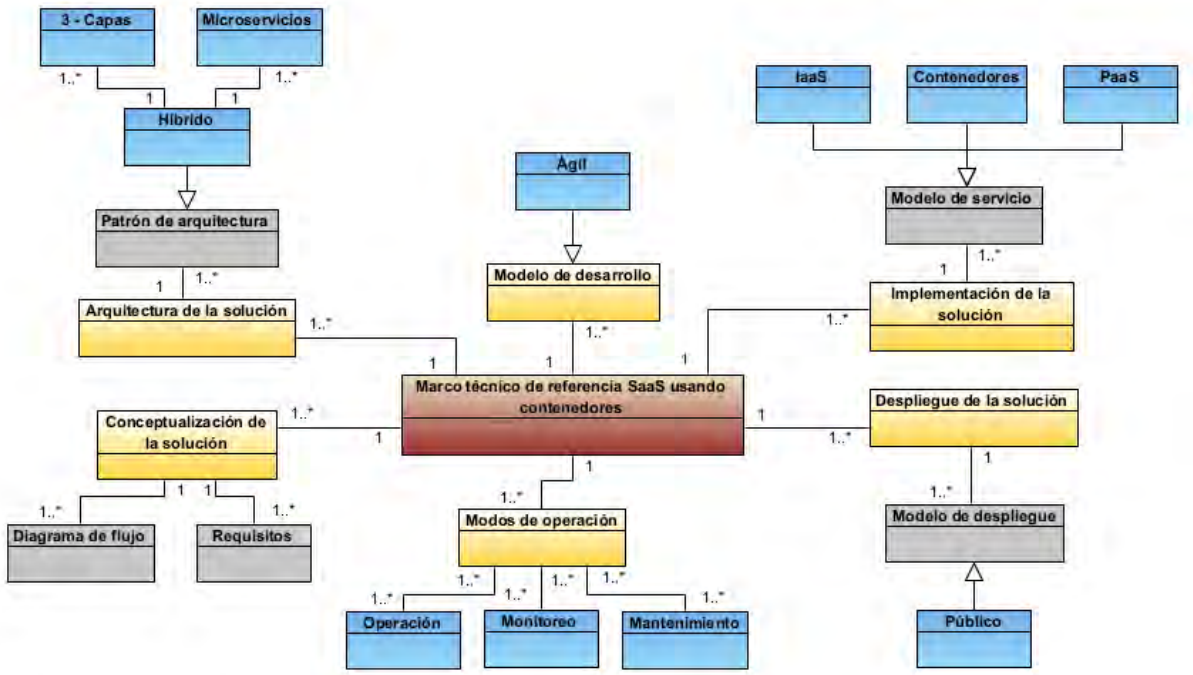

Figura 18: Modelación del marco técnico de referencia específico

3.2 Metodología en el dominio de SaaS usando contenedores

De acuerdo con el diccionario de Oxford una metodología es “un sistema de métodos usados en un área particular de estudio o actividad”. En cambio, el diccionario de Cambridge define que una metodología es “un sistema de formas de hacer, enseñar o estudiar algo”. Para fines del presente trabajo la segunda definición es más acorde ya que la metodología propuesta buscar enseñar de manera sistematizada como desarrollar software como servicio (SaaS) usando contenedores [36]. Para lograr esto, la metodología toma los conceptos del marco técnico de referencia y los ordena de la siguiente manera [39]:

1. Conceptualización de la solución (UML, Requisitos, Diagramas de flujo).
2. Arquitectura de la solución (Patrón de arquitectura de software).
3. Selección de un modelo de desarrollo (Modelo de desarrollo clásico o ágil).
4. Implementación de la solución (Modelo de servicio IaaS, PaaS y Contenedores).
5. Despliegue de la solución (Modelo de despliegue en la nube).
6. Operación (Monitoreo, mantenimiento).

Esto permite la ejecución de cada uno de los conceptos de manera secuencial. Tal y como se aprecia en el diagrama de flujo de la figura 19 señalando de manera general el flujo de las acciones que debe realizar un usuario para desarrollar software como servicio (SaaS) usando contenedores [40]. De la misma manera que ocurrió en el marco técnico de referencia es importante resaltar que estos seis pasos que se muestran en la figura 19 son los más generales, dado que tienen diferentes etapas de decisión, que son particulares al problema de dominio en donde se estén aplicando.

Figura 19: Metodología para el desarrollo de SaaS usando contenedores

Sin embargo, más allá del esquema grafico de la figura 19 es necesario desarrollar cada uno de estos seis pasos, en lo relacionado a su contenido para la metodología propuesta en el trabajo recepcional.

3.2.1 Conceptualización de la solución (UML, requisitos, diagramas de flujo)

El primer aspecto por tener en consideración en la aplicación de la metodología propuesta en el trabajo recepcional es la conceptualización de la solución. En el campo de la conceptualización de la solución existen una gran variedad de artículos y libros [41], [42], [43], [44] que discuten en profundidad sobre el tema. Sin embargo, algo que se puede observar al igual que en cualquier otra área de la disciplina de ingeniería de software es que está en constante evolución, de igual manera que el cómputo en la nube y los contenedores. En el artículo [43] menciona que "los modelos conceptuales son componentes críticos en la identificación de un problema, porque apoyan la comunicación entre las partes interesadas de un proyecto para detectar posibles interpretaciones inadecuadas, o falta de información antes de continuar con la construcción de un sistema o solución". En este sentido el libro [44] es una muy buena base conceptual y practica sobre cómo se pueden diseñar sistemas de información mediante la modelación conceptual.

Ahora bien, en los últimos años han surgido diferentes paradigmas y enfoques sobre cómo realizar la conceptualización de la solución [42]. Estos nuevos enfoques consideran a los modelos no únicamente como parte de la documentación, si no como un pieza central y clave en el campo de la ingeniería de software. Estos lenguajes de modelación se han definido dentro de un enfoque metodológico, como es el caso del paradigma orientado a objetos o bien metodologías o procesos unificados orientados a facilitar y compartir una visión común y coherente del estudio de un sistema, por consecuencia la comunicación entre las partes interesadas de un proyecto de software. Entre los lenguajes que define "Object Management Group (OMG)", el más conocido y utilizado es el lenguaje UML [44].

El lenguaje de modelo UML es el estándar más utilizado para documentar cualquier sistema de forma precisa, ya que representa de forma gráfica las especificaciones, construcciones y documentos que modelan a un sistema. Por todos estos motivos, fue el leguaje de modelado que se eligió para desarrollar la metodología, marco técnico de referencia y los requerimientos del caso de aplicación del presente trabajo recepcional. Sin embargo, es importante resaltar que en el trabajo no se evalúa cual paradigma o enfoque es mejor, ya que existe bastante literatura que presenta los avances, beneficios y desventajas de cada uno de estos enfoques para el modelado de sistemas [44].

3.2.2 Arquitectura de la solución (Patrón de arquitectura de software)

El segundo aspecto por tener en consideración es que el desarrollo de software como servicio (SaaS), al igual que el desarrollo de software tradicional, tiene un ciclo de vida de desarrollo. Dentro de este ciclo de vida se encuentra la fase de arquitectura de la solución [32]. Esta fase está dentro de las fases iniciales del ciclo de vida del desarrollo y tienen un papel importante en el éxito o fracaso de cualquier sistema de software, dado que es responsable de la estructura base de los subsistemas y la interacción entre los subsistemas [28]. Por tal motivo, se posicionó en el segundo elemento que a tomarse en consideración. Ahora bien, la arquitectura de software se pude ver como un proceso en el que se toma una decisión, en donde la selección de un patrón arquitectónico de software forma parte de este proceso de decisión [29].

En este sentido, los patrones arquitectónicos son soluciones universales y reutilizables, que tienen como objetivo satisfacer distintos requerimientos funcionales y de calidad. Por esta razón, el arquitecto de software es el responsable de tomar en consideración, diferentes soluciones y alternativas que resuelvan su problema específico [29]. Dicho de otra forma, no existe un patrón de arquitectura que sea mejor que otro, simplemente cada patrón se adapta a distintas necesidades. Por ejemplo, si se está desarrollando un sistema distribuido en el que se requiera tener procesamiento y datos distribuidos en múltiples servidores, la arquitectura basada en microservicios resulta una buena alternativa a considerar [31].

Evidentemente no siempre resulta conveniente utilizar el patrón de arquitectura basado en microservicios. Por esta razón, en el caso de aplicación del presente trabajo recepcional se utilizó el patrón de arquitectura basado en microservicios, en conjunto con el patrón de arquitectura Modelo-Vista-Controlador (MVC). De esta manera, se aprovechan las ventajas que aporta cada uno de los modelos [29]. Por ejemplo, en el caso del patrón de arquitectura basado en microservicios, se aprovecha las ventajas de la modularidad, reutilización del microservicio en otros sistemas, entregas continuas, entre otros. En contraste, el patrón MVC no tiene ese nivel de modularidad, pero es ideal en entornos donde no hay una buena red o existe mucha latencia entre los servicios [29].

3.2.3 Selección de un modelo de desarrollo (Modelo de desarrollo clásico o ágil)

El tercer aspecto por tener en consideración es que el modelo de desarrollo de software es parte del ciclo de vida del desarrollo. Por lo tanto, también es parte de los criterios a tener en consideración al momento de desarrollar software como servicio (SaaS) usando contenedores [32]. La importancia en la selección de un modelo adecuado radica, en que establece distintas etapas y estados por los que un producto de software vive, desde la fase de su conceptualización hasta la fase de implementación. Ahora bien, los modelos se clasifican principalmente en dos grupos: los tradicionales y los ágiles [25]. En estos dos grupos existen una gran variedad de distintos modelos de desarrollo que un arquitecto puede seleccionar de acuerdo con las necesidades del proyecto. Por ejemplo, en el modelo tradicional se puede seleccionar el modelo cascada, incremental, prototipo, entre otros. En cambio, en los modelos ágiles se puede optar por SCRUM, Extreme Programming (XP), Crystal, DSD, entre otros [25].

Cada uno de estos modelos tiene sus ventajas y desventajas inherentes. Por consiguiente, la decisión de cual utilizar depende del arquitecto de software o administrador de proyectos. No obstante, los modelos ágiles tienen una serie de ventajas en comparación con los modelos tradicionales. Por ejemplo, permiten los cambios en la fase de desarrollo, involucran al cliente durante el proceso del desarrollo de la aplicación y la adopción rápida de nuevos paradigmas y patrones de arquitectura de software [25]. Sin embargo, el utilizar modelos ágiles ocasionan por lo general una barrera entre los equipos de desarrollo y operaciones, si no se tienen en consideración la cultura, herramientas y tecnologías disponibles al momento de implementarlo. Para evitar esto, se recomienda el uso del paradigma DevOps [26].

Por tal motivo, para validar los beneficios que aporta el uso del paradigma DevOps en la metodología propuesta, se crearon mecanismos de despliegues automáticos para uno de los microservicios propuestos en la arquitectura de la solución del caso de aplicación. Esto permitió que se integraran pequeños fragmentos de código funcionales en el microservicio, que se integraron e implementaron de manera continua en el entorno de AWS. Todas estas ventajas son parte del uso del modelo ágiles en combinación con el paradigma DevOps [26].

3.2.4 Implementación de la solución (Modelo de servicio IaaS, PaaS y contenedores)

A partir de esta sección es donde hay un cambio con respecto al desarrollo de un sistema de software tradicional. En esta parte al igual que en las anteriores, el arquitecto de la solución es el encargado de seleccionar cual de todos los modelos es el que más se adecua a las necesidades de su proyecto. En este sentido, es importante mencionar que existen dos maneras principales en las que se puede desarrollar software como servicio (SaaS) [37]. La primera es mediante el uso de Infraestructura como Servicio (IaaS), en la cual el usuario deberá ser responsable de gestionar el sistema, actualizaciones al sistema y la plataforma sobre la que va a realizar el desarrollo del software [32]. La segunda manera es mediante el uso de una Plataforma como Servicio (PaaS), en la cual el usuario solo será responsable de mantener el desarrollo de la aplicación [32]. Tal y como se describe en la sección del marco teórico, este último modelo de servicio proporciona ventajas como el desarrollo de aplicaciones, sin la necesidad de tener que lidiar con la administración de la infraestructura y/o plataforma, lo cual facilita la tarea del desarrollo, reducción del esfuerzo en tiempo de desarrollo mediante un integración y testeo rápido.

Sin embargo, esta forma tradicional de desarrollar Software como Servicio (SaaS) tiene el problema de depender de un proveedor o plataforma [37]. Para resolver este problema, se propone el uso de contenedores con lo cual se puede empaquetar la aplicación SaaS, con todas las dependencias necesarias y tener la posibilidad de realizar migraciones a otros proveedores de nube o entornos de manera ágil. Pero al igual que ocurre con la mayoría de las tecnologías o patrones, los contenedores tienen la limitante de depender de la tecnología del contenedor, con la que se haya creado la imagen, así como el sistema operativo para el cual se creó [45].

En consecuencia, el arquitecto que este aplicando la metodología, necesitará evaluar cual tecnología de contenedores cumple con los requerimientos de su entorno. Además de esto, es necesario evaluar si es necesario el uso de un orquestador de contenedores [20]. No obstante, aun con estas desventajas, los beneficios de los contenedores son mayores, y esos dos problemas se puede resolver de la siguiente manera. La primera es creando imágenes de contenedores para distintos sistemas operativos, y la segunda es seleccionando una tecnología que tenga un buen número de adeptos [19]. En este sentido Docker es una de las tecnologías emergentes de contenedores que tiene un gran número de adeptos, y por consiguiente bastante soporte por parte de proveedores de nube como Amazon, Google y Microsoft, entre otros. Por tal razón, se recomienda el uso de esta tecnología en la metodología y en el caso de aplicación [22].

3.2.5 Despliegue de la solución (Modelo de despliegue en la nube)

El quinto aspecto que se tiene que considerar en la metodología es el despliegue de la solución. Es decir, en donde se estará ejecutando o implementado la aplicación. Para ello, se tiene que recordar que el NIST establece cinco características claves con las que un servicio o aplicación debe cumplir para ser considerado un servicio de nube [13]. Para cumplir con estas cinco características claves es necesario el uso de algún modelo de servicio como IaaS o PaaS en conjunto con algún modelo de despliegue como lo puede ser un modelo de despliegue público, privado o híbrido [32]. Por tal motivo, en esta sección de la metodología nos enfocamos en la selección de un modelo de despliegue, que al igual que ocurre con las otras secciones dependerá de las necesidades específicas de un proyecto, organización o empresa. Para este punto el arquitecto de la solución deberá tomar en cuenta aspectos como el presupuesto, tamaño y tipo de proyecto, entre otros factores [37].

En este sentido, para el desarrollo del caso de aplicación y para demostrar la metodología se recomendó el uso de un modelo de despliegue público, ya que se tiene la ventaja de que es administrado por un proveedor de servicio [18]. Por lo cual, se puede lograr una disponibilidad, escalabilidad y accesibilidad de manera más rápida y sencilla en comparación de un entorno local [15]. Además de estas ventajas que otorga este modelo de servicio, se recomienda su uso, ya que permite pesar de una economía basada en activos a una economía basada en servicios, con lo cual la empresa se puede enfocar en su visión y misión, evitando preocuparse por el mantenimiento de la infraestructura y plataforma que soporte su negocio [15].

No obstante, es importante mencionar que cada modelo de despliegue de la nube conlleva una serie de ventajas/desventajas, y dependerá del arquitecto de la solución determinar cuál es el modelo que mejor se adapta a sus necesidades [15]. Por ejemplo, en un escenario en donde se tiene infraestructura existente y se quiere aprovechar las ventajas que ofrece la nube, un modelo de despliegue híbrido pudiera resultar adecuado, ya que permite a las empresas u organizaciones realizar una migración de sistema de manera paulatina [9]. Sin embargo, se puede tener el problema de que sean complejas y costosas de implementar. Por tal motivo, al igual que la mayoría de los puntos de la metodología se recomienda que la elección y configuración del modelo de despliegue la realice un arquitecto de solución [5].

3.2.6 Operación (Monitoreo y mantenimiento)

El sexto y último aspecto de la metodología, es la operación (monitoreo y mantenimiento) del sistema o solución. Para esta sección el arquitecto de la solución en conjunto con los ingenieros DevOps deberán ser los encargados de seleccionar e implementar las soluciones de monitoreo y mantenimiento [26]. En relación con el monitoreo de sistemas, existen un gran número de herramientas de distintos proveedores (AWS, Google, Azure), o incluso la empresa puede optar por implementar un sistema de monitoreo propietario [34]. Para fines del trabajo recepcional se optó por utilizar AWS CloudWatch, ya que es una herramienta completa que permite llevar un control de logs, métricas y eventos del caso de aplicación. Asimismo, en lo referente al mantenimiento de la aplicación se utilizó AWS Code Deploy para automatizar los despliegues de nuevas versiones a la instancia de ECS [34].

En este aspecto, de igual manera que en las secciones anteriores, no se evaluó cual es la mejor herramienta de logs o cual es el mejor enfoque para la operación y mantenimiento de la solución, ya que existen un gran número de factores que se necesitan tomar en consideración [33]. Por ejemplo, una universidad grande que este ubicada en México y tenga convenios con Microsoft conviene que mire alternativas y precios que ofrece la nube Azure. En contraste, una empresa pequeña que no tiene mucho presupuesto para poner en marcha su sistema es preferible que opte por mirar las ofertas y precios que ofrece la nube de Amazon y Google [33].

Para concluir, lo referente al monitoreo y mantenimiento, es fundamental su mención en la metodología propuesta, dado que con el monitoreo del sistema se pueden obtener métricas de uso, rendimiento o incluso detectar errores que resultan esenciales al momento de optimizar un sistema, para con ello buscar ahorrar recursos [33]. En cuanto al mantenimiento de la solución los ingenieros DevOps pueden crear automatizaciones para que el código o las imágenes de contenedores, que el equipo de desarrollo libera se publiquen de manera automatizada a los entornos de nube [34]. Esta forma de automatización es parte de la filosofía DevOps y se le conoce como integración y despliegue continuos (CI/CD), lo cual permite que las empresas dispongan de nuevas versiones del sistema de forma rápida y constante, logrando con ello ser más competentes [34].

3.3 Caso de aplicación para marco técnico de referencia y metodología

Una vez concluida la parte del marco técnico de referencia y metodología, es necesario la creación de un caso de aplicación que permita validar lo que se propone de manera conceptual. Para ello se ha diseñado un sistema orientado a la publicidad y mercadotecnia el cual cuenta con su objetivo, alcance, tipos de usuarios, requerimientos y diagramas de flujo que describen su funcionamiento [12].

Descripción de los elementos que componen al sistema propuesto para el caso de aplicación

Objetivo de proyecto

Desarrollar un sistema que permita a los usuarios registrados crear publicidad de sus bienes y servicios en internet y a los usuarios públicos navegar a través de la publicidad o anuncios por palabras clave.

Alcance

Para fines del desarrollo del caso de aplicación el sistema no incluirá pagos y se asumirá que la publicidad del producto es gratuita.

Tipos de usuario

- **Usuario registrado:** Es un usuario que ha creado una cuenta en el sistema web y esta autentificado en el sistema mediante el uso de su usuario y contraseña.
- **Usuario público:** Todos los usuarios en el sistema, incluyendo aquellos que no se encuentran registrados.

Tecnologías y entorno

- La arquitectura del sistema será basada en la arquitectura de microservicios.
- El sistema será desarrollado usando .Net Core y ASP.NET Core.
- El sistema será programado usando C#.
- El sistema será implementado en AWS.

Requerimientos

1) Los usuarios se registran en el sistema y crean una cuenta
    a) La dirección de correo electrónico debe ser usada como nombre de usuario.
    b) La contraseña debe ser de al menos 6 caracteres.
2) Solamente los usuarios registrados pueden crear publicidad.
3) La publicidad tendrá los siguientes atributos
    a) Título.
    b) Descripción.
    c) Precio.
    d) Una imagen
4) Los usuarios públicos deben tener acceso para navegar a través de la publicidad.
5) Los usuarios públicos deben tener acceso para ver los detalles de la publicidad.
6) Los usuarios públicos deben tener acceso para buscar publicidad por palabras clave.

Diagrama de flujo

En esta sección se describe el funcionamiento de la aplicación de publicidad y mercadotécnica propuesta en el caso de aplicación mediante un diagrama de flujo. Es importante resaltar que en este diagrama solamente se muestra la lógica de negocio de la aplicación propuesta y no se están especificando que tipo de servicios se van a utilizar para su desarrollo. Es hasta la sección 3.3 donde se describen cuales tecnologías se usarán para el desarrollo.

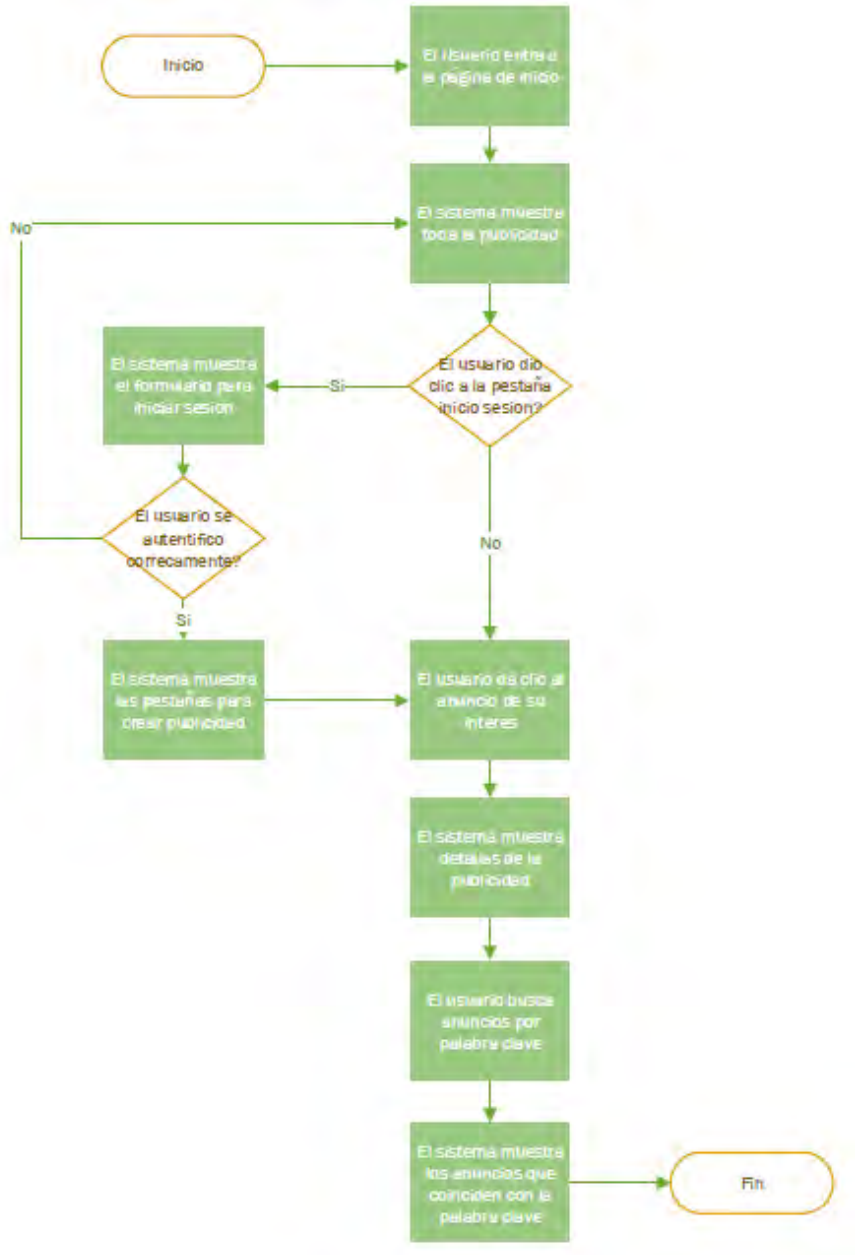

Figura 20: Diagrama de flujo del caso de aplicación

3.4 Aplicación del marco técnico de referencia y metodología al caso de aplicación

Para la sección de aplicación del marco técnico de referencia y metodología se realizó una modelación en la cual se determinaron el conjunto de tecnologías a utilizar para aplicar el caso de aplicación al marco técnico de referencia y a la metodología. Por consiguiente, se inició con la asignación del conjunto de tecnologías al marco técnico de referencia y posteriormente a la metodología [35]. En la figura 21 se muestra al conjunto de tecnologías que se utilizaron en el marco técnico de referencia, comenzando por la arquitectura de la solución. En ella se optó por utilizar un patrón de arquitectura híbrido en el cual se conserva las ventajas del modelo-vista-controlador y la arquitectura basada en microservicios [46], en las que se usa la tecnología ASP.NET CORE. En cuanto al modelo de desarrollo se utilizó un modelo ágil que facilitó el desarrollo del caso de aplicación en pequeñas partes funcionales [47]. Con respecto a la implementación, despliegue de la solución y modos de operación se eligió usar las tecnologías ofertadas por la nube pública AWS.

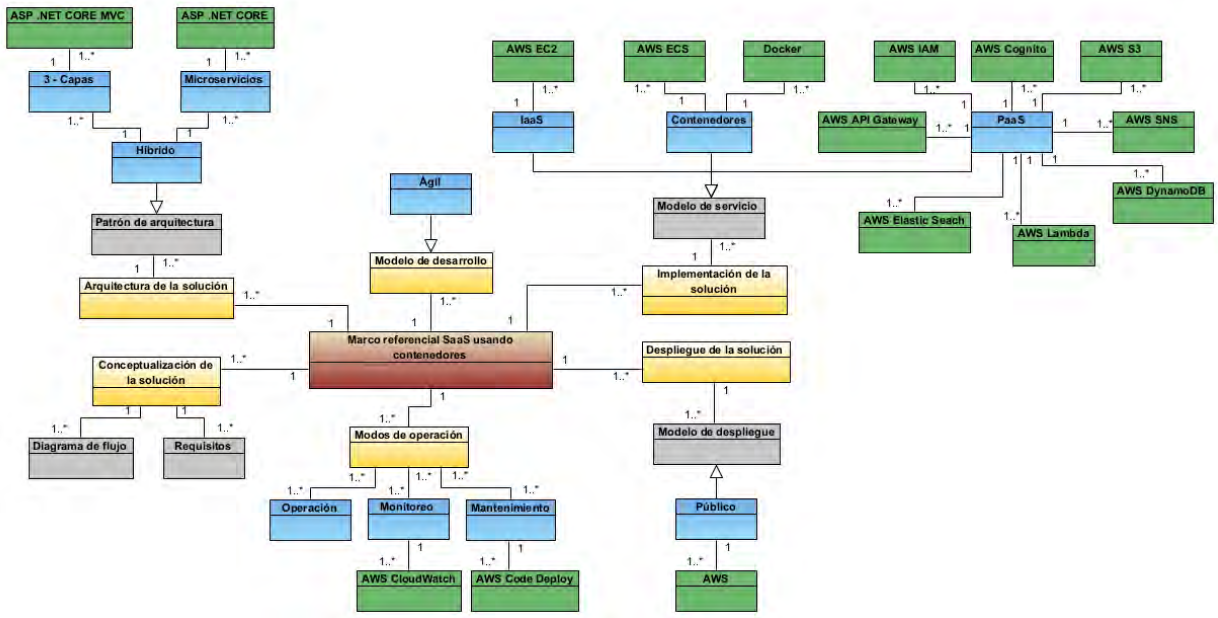

Figura 21: Marco técnico de referencia aplicado al caso de aplicación

Finalmente, para cerrar la aplicación del marco técnico de referencia y metodología al caso de aplicación, se realizó la asignación de tecnologías que se requieren para la metodología. Para ello se presenta un diagrama de flujo en la figura 22 en el cual se muestra de manera específica su aplicación en el dominio de mercadotecnia y publicidad mediante el uso de tecnologías de la nube pública de AWS [33].

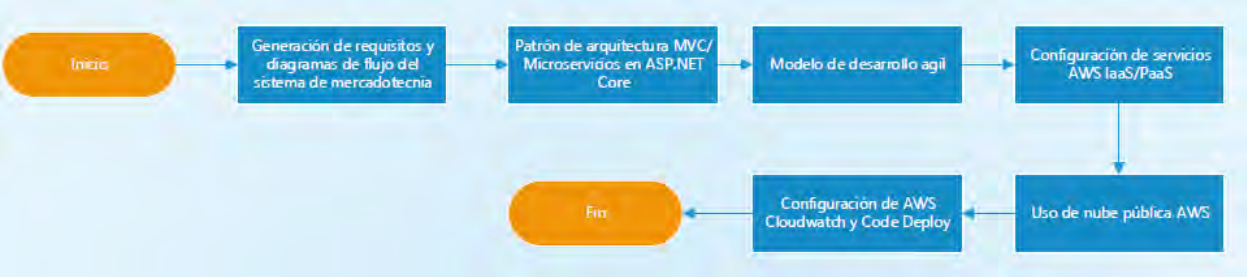

Figura 22: Metodología aplicada al caso de aplicación

3.5 Patrones de arquitectura de software para la solución

Para explicar la arquitectura de software en la solución que se utilizó para el desarrollo del caso de aplicación se presentan dos modelaciones. La primera modelación muestra algunos de los diferentes patrones de arquitectura que existen y se pueden utilizar para el desarrollo de software como servicio (SaaS) usando contenedores. Este primer diagrama de modelación que se muestra en la figura 23, al igual que las modelaciones realizadas para el marco técnico de referencia, usa el método deductivo, pero con la diferencia de que las modelaciones están orientadas a mostrar únicamente los diferentes patrones de arquitectura de software y las tecnologías que se pueden usar para cumplir con el objetivo del trabajo recepcional.

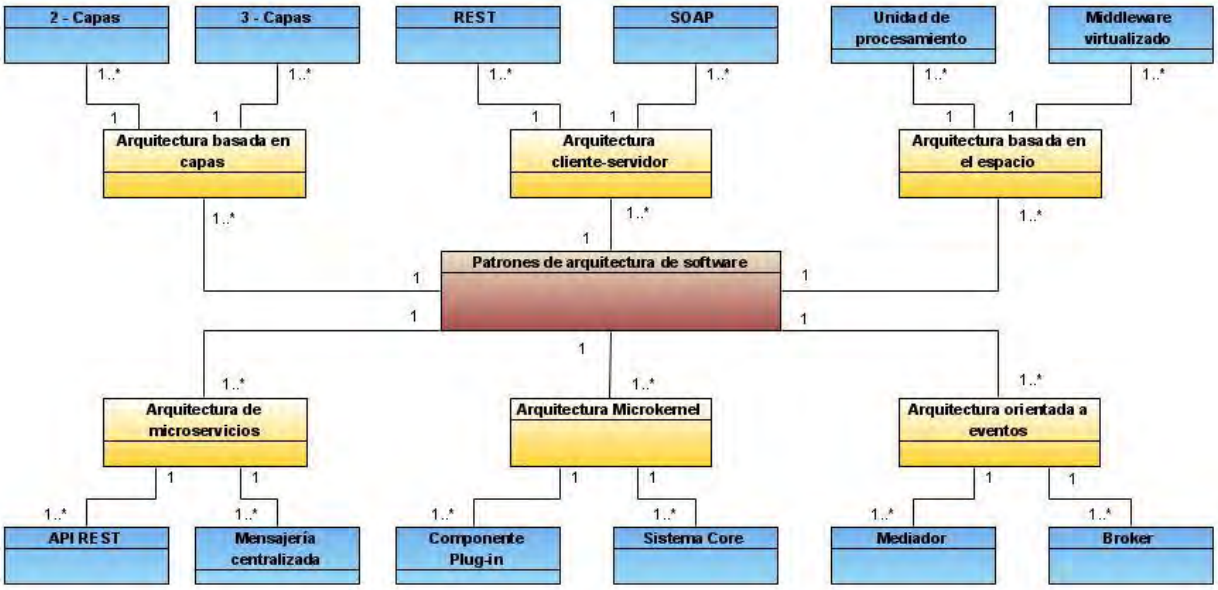

Figura 23: Modelación patrones de arquitectura general

El segundo diagrama de la modelación que se muestra en la figura 24, se pueden ver cuáles son los patrones de arquitectura que se utilizaron para el desarrollo de la solución, así como las tecnologías usadas de manera específica. Sin embargo, es importante mencionar que en el presente trabajo no se realiza una evaluación de cual patrón de arquitectura es mejor, o cual tecnología conviene usar para el desarrollo de los patrones de arquitectura, ya que esto está fuera del alcance de este trabajo recepcional. Para ver cuales patrones de arquitectura resultan convenientes emplear existen diversos libros [25], [29] y estudios [30], [36] que realizan análisis y comparativas de los diferentes patrones de arquitectura de software, lo cual puede servir como una guía para el arquitecto de solución al momento de tomar una decisión en cuanto a cuál patrón o tecnología usar para la implementación de una solución.

Figura 24: Modelación patrones de arquitectura especifica

3.6 Arquitectura de software de la solución

En esta sección se realizó un diagrama de arquitectura de software de la solución. Este diagrama describe la funcionalidad del caso de aplicación que se utilizó para validar la metodología. La figura 25 describe el funcionamiento del caso de aplicación y se encuentra dividido en dos secciones. La primera sección muestra que existen distintos tipos de usuarios que pueden acceder a la aplicación web mediante algún dispositivo móvil o navegador web. La segunda sección describe el funcionamiento de la aplicación web de publicidad y anuncios. Esta sección está conformada por dos partes, la primera parte es una aplicación web que atiende las solicitudes de los usuarios y muestra el contenido del sitio web y la segunda parte muestra la interacción que hay entre la aplicación web y los distintos microservicios que se encargan de brindarle funcionalidad a la aplicación web. Por ejemplo, un microservicio se encarga de la parte de autenticación de usuarios, otro se encarga de guardar los anuncios y otro se puede encargar de realizar las búsquedas de los anuncios, en la figura 25 se puede ver en detalle toda la interacción completa.

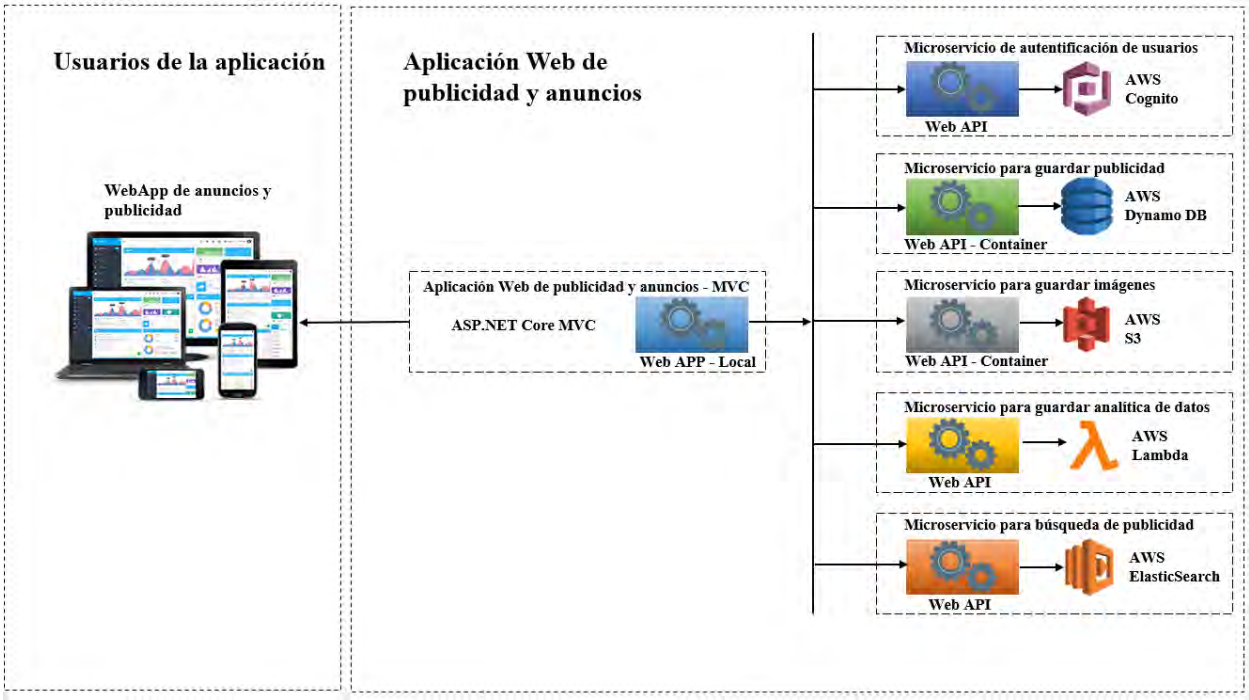

Figura 25: Diagrama de arquitectura de software de la solución.

Capítulo IV: Implementación del marco técnico de referencia y metodología

En el capítulo anterior se realizó la propuesta del marco técnico de referencia, la metodología y el desarrollo de un caso de aplicación que permitiera validar lo propuesto de forma conceptual y un diagrama de arquitectura de software que describe el funcionamiento del caso de aplicación, y en esta sección se explica cómo se implementaron los distintos microservicios, que tecnologías de nube se utilizaron y como implementar cada uno de ellos en el proveedor de nube AWS. Para tal propósito, se presentan las siguientes tres secciones que explican en detalle dichos puntos.

4.1 Implementación de la solución para el caso de aplicación

En esta primera sección del capítulo cuatro se presenta cómo se encuentra implementada la solución del caso de aplicación. Para ello se diseñó un diagrama con los diferentes servicios que conforman al caso de aplicación y como están implementados. En la figura 26 se presenta dicho diagrama en cual se puede apreciar que la aplicación principal esta implementada de manera local, dos de sus microservicios en la nube y un tercero en un contenedor Docker el cual se ejecuta en la nube. Se optó por este esquema de implementación ya que es ideal para el desarrollo de nuevas aplicaciones o bien para migrar aplicaciones existentes de manera gradual. No obstante, se presentarán otros diagramas en donde se pueden ver las diferentes maneras en las que se puede realizar implementaciones [45].

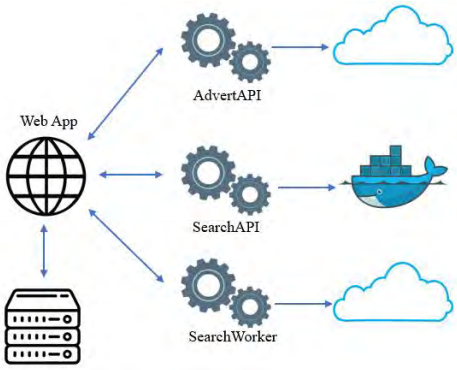

Figura 26: Implementación usada para el caso de aplicación

La segunda manera en la que se puede implementar una aplicación SaaS que usa contenedores se muestra en la figura 27 y consiste en utilizar contenedores para cada uno de los microservicios que se consumen en la aplicación web. Este nivel es el que más portabilidad otorga ya que permite ejecutar la aplicación web y sus microservicios en entornos locales, virtualizados o bien en despliegues de nube pública, privada o hibrida. Sin embargo, este método puede no resultar el más apropiado si se requiere que la aplicación tenga velocidades de ejecución "bare-metal" [45].

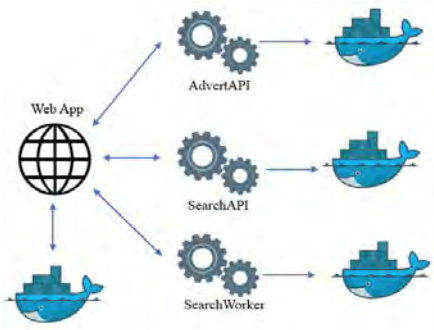

Figura 27: Implementación con contenedores

La tercera forma en la que se puede implementar una aplicación SaaS que usa contenedores se muestra en la figura 28 y es utilizar servicios de nube para cada uno de los microservicios que se consumen en la aplicación web y usar un contenedor para la web App. Esto permite que la aplicación principal tenga portabilidad. No obstante, se tiene la desventaja de que todos los microservicios que se consumen son dependientes de un proveedor de servicios en la nube [45].

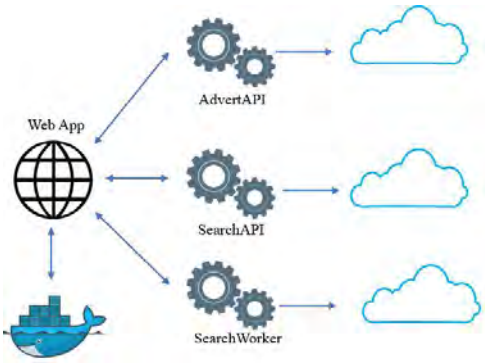

Figura 28: Implementación con servicios de nube y contenedor

El cuarto modo en el que se puede implementar una aplicación SaaS que usa contenedores se muestra en la figura 29 y es utilizar contenedores para cada uno de los microservicios que se consumen en la aplicación web y usar un servicio de nube para la web App. Esto permite que los microservicios tengan portabilidad. Sin embargo, se tiene la desventaja de la dependencia del proveedor de nube en la web App [45].

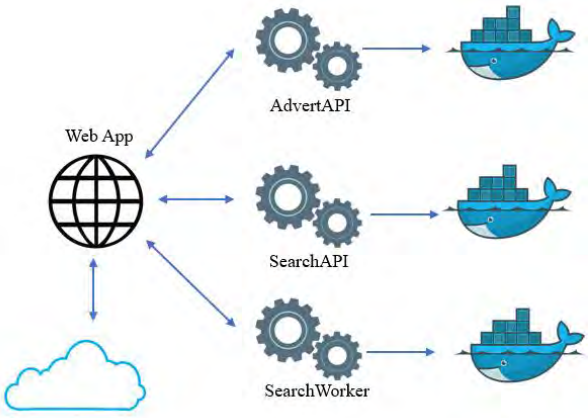

Figura 29: Implementación con contenedores y servicio de nube

4.2 Arquitectura de servicios en la nube para el caso de aplicación

En esta sección se realizó un diagrama de arquitectura para mostrar la interacción de los diferentes servicios de nube usados para la operación y desarrollo del caso de aplicación. En la figura 30 se presenta el diagrama el cual se debe leer de izquierda a derecha comenzando por los usuarios del caso de aplicación [33]. Tal como se puede observar existen dos tipos de usuarios. Los usuarios tradicionales los cuales usan un navegador web para acceder a la aplicación y los usuarios móviles los cuales utilizan un dispositivo móvil para acceder a la aplicación [34].

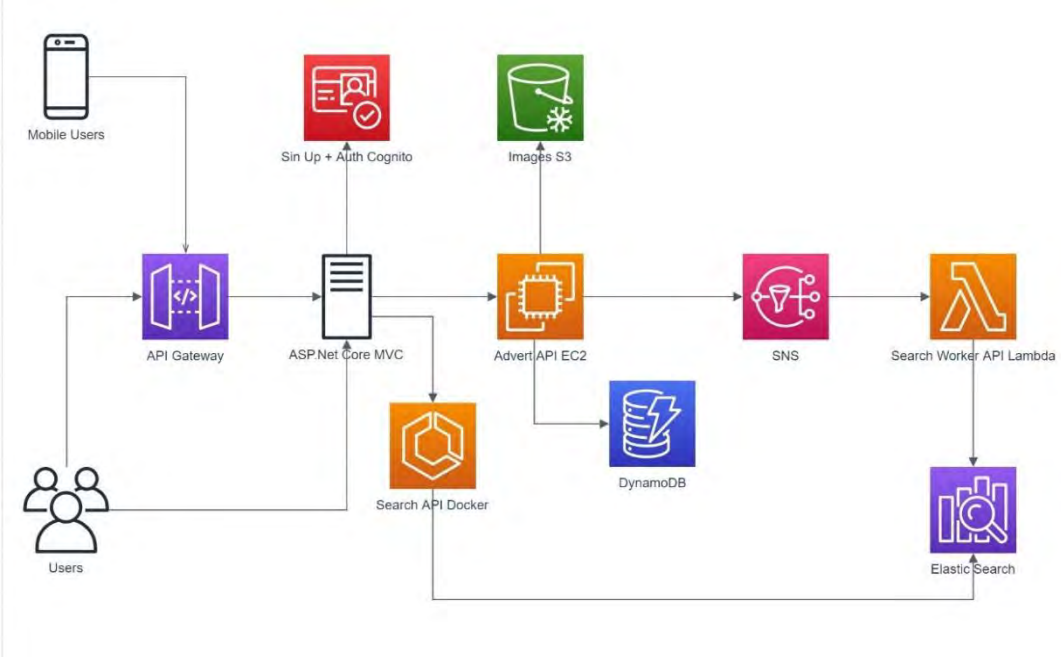

Figura 30: Arquitectura de servicios AWS para el caso de aplicación

La diferencia entre el primer tipo de usuario y el segundo radica en cómo se accede a la aplicación. En el caso de los usuarios tradicionales se puede acceder a la aplicación directamente o por medio del servicio AWS API Gateway. Para el caso de los usuarios móviles es necesario el uso forzoso del AWS API Gateway para encriptar la comunicación. Después de que los usuarios acceden a la aplicación se muestran todos los anuncios publicados en la aplicación. En este punto el usuario puede decidir si quiere autentificarse en la aplicación o crear una nueva cuenta. De ser así los formularios para iniciar sesión y crear cuentas hacen llamadas al servicio de AWS Cognito el cual se encargará de la administración de las cuentas de usuario en la aplicación [33].

Ahora bien, en caso de que el usuario tenga una cuenta y haya iniciado sesión en la aplicación se mostrará un formulario que permite crear anuncios. Este formulario funciona mediante llamadas al microservicio "Advert API" el cual se comunicará con dos servicios de AWS para guardar la información del anuncio en la aplicación. El primer servicio con el cual se comunica es AWS DynamoDB para guardar datos generales como lo son el título, precio y descripción del anuncio. El segundo servicio con él se comunica es AWS S3 en cual se guardan las imágenes de los anuncios. Una vez que se guardó la información e imagen del anuncio el microservicio "Advert API" se comunica con el servicio AWS SNS para enviar un mensaje al servicio de AWS Lamba en cual se tiene una función que guarda los datos del anuncio en el servicio AWS ElasticSearch. Este último servicio es utilizado para realizar búsquedas y analíticas de datos en los anuncios por el microservicio "Search API" [34].

Para terminar con esta sección se presenta el diagrama de la figura 31 en el que se muestra cual microservicio y tecnología de contenedores se utilizó y que servicios se emplearon para su implementación.

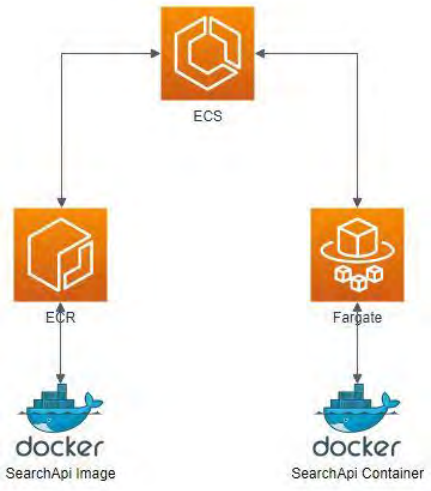

Figura 31: Servicios usados para ejecución del microservicio contenerizado

4.3 Implementación de servicios para el caso de aplicación

Para terminar, después de diseñar un diagrama de arquitectura mostrando la interacción entre las diferentes tecnologías que se usaron en el caso de aplicación, es necesario diseñar los diagramas de flujo que expliquen brevemente como realizar la configuración de dichos servicios. Para ello se inició con el diagrama que se presenta en la figura 32 en el cual se muestra el proceso de configuración del servicio AWS IAM de manera breve. Este es el primer servicio que es necesario configurar ya que es utilizado en la programación de los microservicios al momento de configurar y hacer llamadas a los otros servicios de AWS que se mostraron en la figura 30. Para ver con mayor detalle como configurar este servicio se puede consultar el anexo I [34].

Figura 32: Configuración de IAM

El segundo diagrama que se presenta en la figura 33 muestra el proceso de configuración del usuario AWS en el entorno de desarrollo. En este punto es en donde los servicios, microservicios, APIs, aplicaciones web o componentes que se desarrollen usando cualquiera de los servicios ofrecidos por AWS buscaran un archivo de configuración que contenga la cuenta AWS. Después de que el componente establece una conexión correcta con la cuenta AWS el servicio de IAM se encarga de delimitar a que otros recursos o servicios de AWS puede acceder el componte desarrollado. En el anexo II se puede ver en detalle cómo realizar la configuración del usuario IAM en el entorno de desarrollo [34].

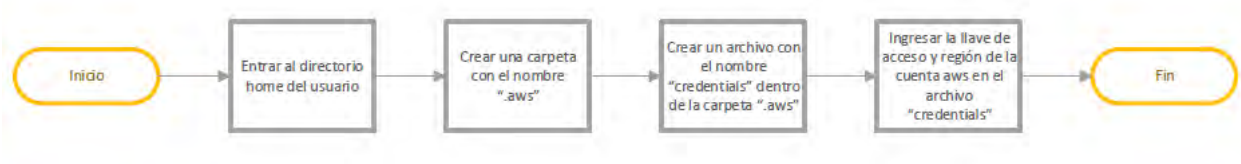

Figura 33: Configuración de usuario AWS en el entorno

El tercer diagrama que se presenta en la figura 34 describe brevemente el proceso de configuración del servicio AWS Cognito. Este servicio permite administrar la creación y autentificación de usuarios dentro de las aplicaciones que se desarrollen. Para el caso de aplicación se utilizará el servicio de Cognito para permitir que los usuarios se registren/autentifiquen y con ello puedan crear anuncios. En el anexo III se puede buscar más ampliamente como configurar el servicio de Cognito para la creación y autentificación de usuarios en la aplicación propuesta [34].

Figura 34: Configuración de Cognito

En el cuarto diagrama que se presenta en la figura 35 se describe brevemente el proceso de configuración del servicio AWS DynamoDB. Este servicio de base de datos NoSQL orientado a documentos permite almacenar los datos que se generen en las aplicaciones desarrolladas. Para el caso de aplicación se empleó para almacenar la información relevante del anuncio como el título, descripción y precio. Para una descripción más detallada de como configurarlo se puede consultar el anexo IV [34].

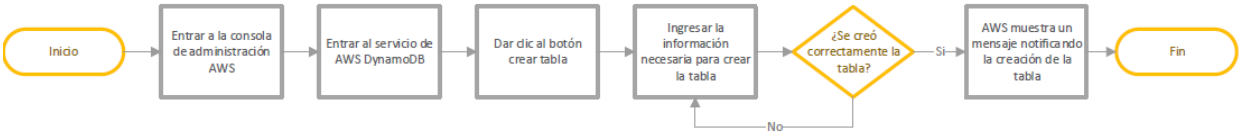

Figura 35: Configuración de DynamoDB

El quinto diagrama que se presenta en la figura 36 describe de manera resumida como configurar el servicio AWS EC2. Este servicio de Amazon sirve para crear infraestructura en la nube y es el primer escalón para migrar hacia SaaS. En el caso de aplicación se utilizó con el microservicio “Advert API” a manera de ejemplificación para empresas que busquen adaptar esta primera forma de migración. Para ver su configuración en mayor detalle se puede revisar el anexo V [34].

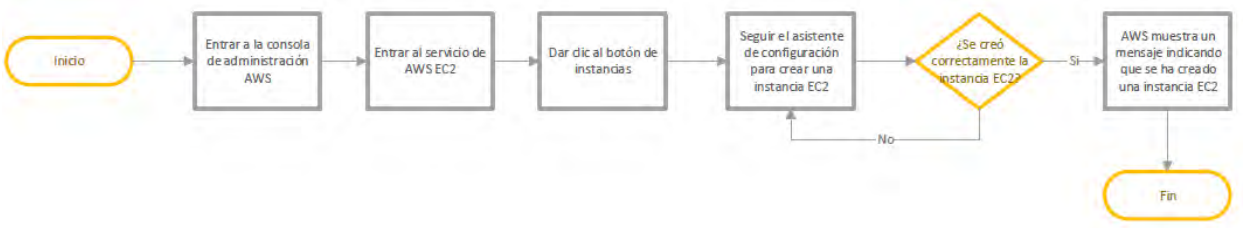

Figura 36: Configuración de EC2

El sexto diagrama que se presenta en la figura 37 describe de manera corta como configurar una cubeta en S3 para usarla en AWS Code Deploy. Este servicio de AWS permite crear cubetas en las que se puede almacenar diferentes tipos de objetos, tal como código de aplicaciones, imágenes, documentos, sitios web, entre otros. En el caso de aplicación se utilizará para almacenar el microservicio "Advert API" el cual posteriormente se desplegará de manera automática a la instancia EC2. Para ver su configuración más detallada se puede revisar el anexo VI [34].

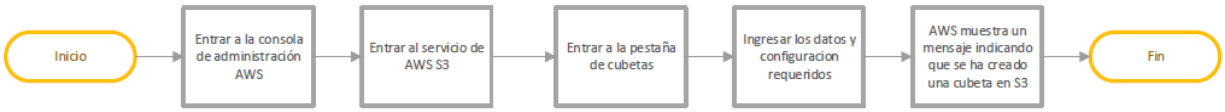

Figura 37: Configuración de S3 para Code Deploy

El séptimo diagrama que se presenta en la figura 38 describe de manera breve como configurar AWS Code Deploy para crear despliegues automáticos a una instancia EC2 una vez que se tiene una aplicación nueva lista para implementar. Este servicio además de permitir crear despliegues a instancias EC2 ofrece la posibilidad de crear despliegues automáticos para AWS Fargate y Lambda para cuando se trabaja con tecnologías de contenedores o sin servidor. Para mirar su configuración en detalle consultar el anexo VII [34].

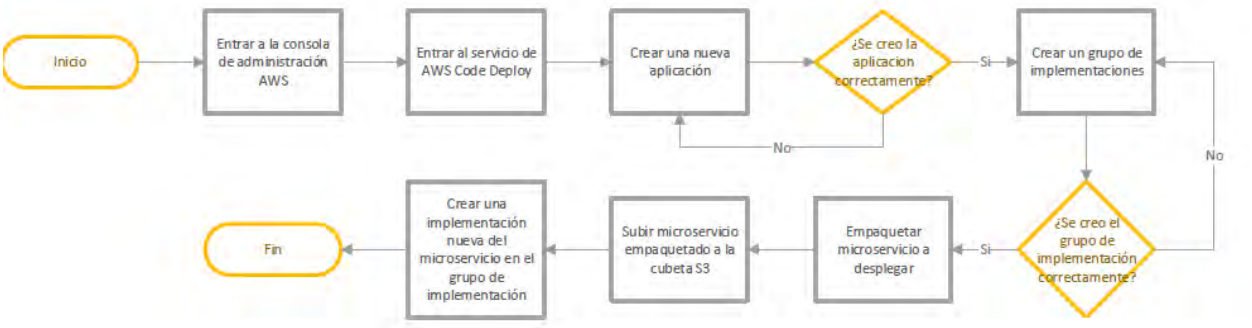

Figura 38: Configuración de Code Deploy para EC2

El octavo diagrama que se presenta en la figura 39 muestra como configurar el servicio de SNS. Este servicio permite enviar mensajes de publicación/suscripción entre los diferentes servicios de AWS. Por ejemplo, en el caso de aplicación se ejecuta posterior a guardar la información del anuncio en AWS DynamoDB para en enviar un mensaje AWS Lambda y guardar esta misma información en el servicio AWS ElasticSearch. La configuración con mayor detalle de este servicio se muestra en el anexo VIII [34].

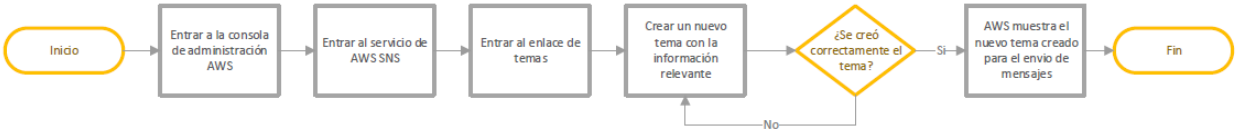

Figura 39: Configuración de SNS

El noveno diagrama que se muestra en la figura 40 describe como configurar el servicio de AWS ElasticSearch. Este servicio permite realizar análisis y monitoreo de los datos. Para el caso de aplicación se utiliza para la búsqueda y análisis de los anuncios que se generan en el caso de aplicación. Este servicio es el que se usa en el microservicio contenerizado "SeachAPI" para la búsqueda de anuncios con filtros de búsqueda al momento de que usuario ingrese una palabra clave. Para mirar su configuración en detalle se puede consultar el anexo IX [34].

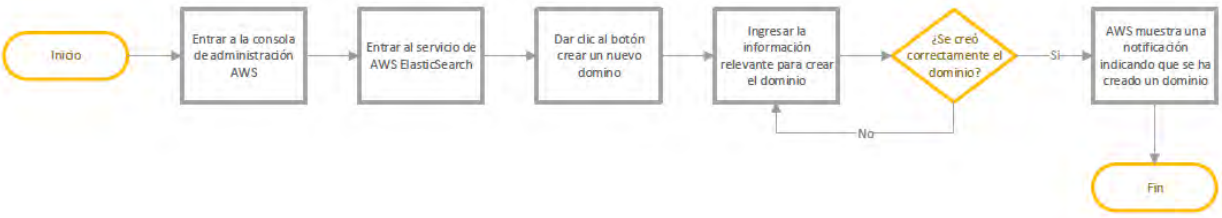

Figura 40: Configuración de ElasticSearch

El décimo diagrama que se presenta en la figura 41 explica como configurar un rol para el servicio de AWS Lambda. Los roles sirven para asignar permisos o funciones a los que un servicio de AWS puede acceder. En este caso se asignó un rol para que Lambda pudiera acceder al servicio CloudWatch y con ello disponer de métricas para verificar el funcionamiento y desempeño del servicio Lambda. Para más detalles de la configuración de este servicio se puede revisar el anexo X [34].

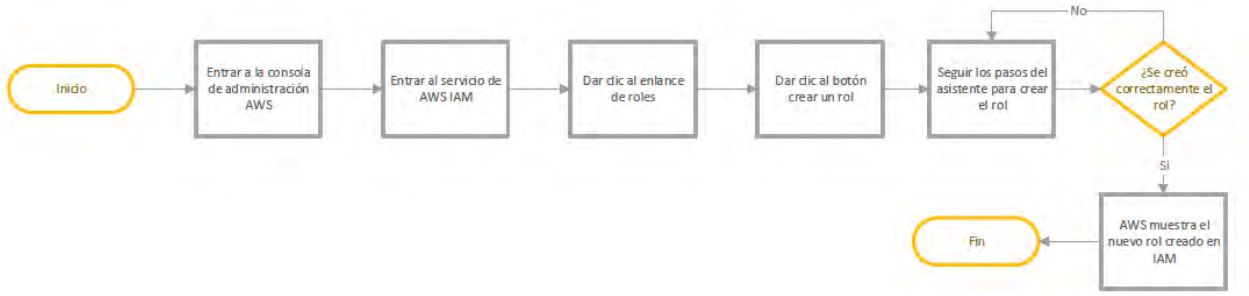

Figura 41: Configuración de rol para Lambda

El décimo primer diagrama que se ve en la figura 42 describe como con figurar el servicio de AWS Lambda. Este servicio sirve para crear funciones o pequeños fragmentos de código los cuales se ejecutan sin servidor. Para el caso de aplicación se utiliza para una función que recibe un mensaje SNS y después guardar la información del anuncio en el servicio ElasticSearch. En el anexo XI se muestra su proceso de configuración en mayor detalle [34].

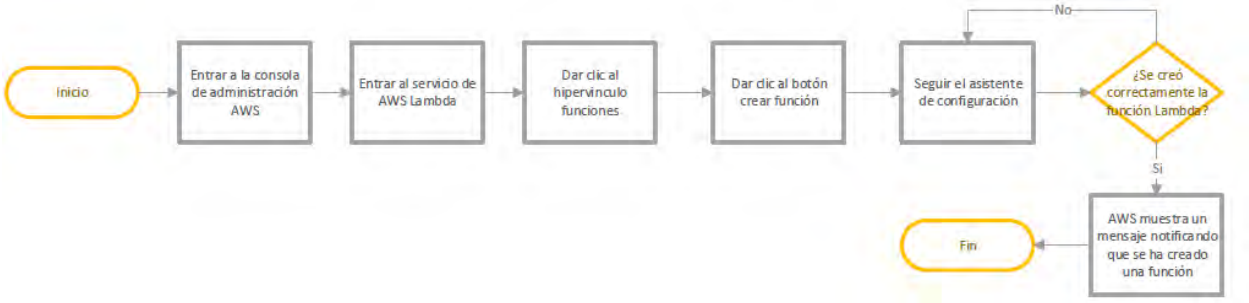

Figura 42: Configuración de Lambda

El décimo segundo diagrama que se muestra en la figura 43 explica cómo realizar pruebas del envío de mensajes SNS a la función Lambda creada en el paso anterior. Esto permite validar que se haya realizado una configuración y programación adecuada del servicio. En el anexo XII se puede observar más en detalle cómo realizar las pruebas de funcionamiento [34].

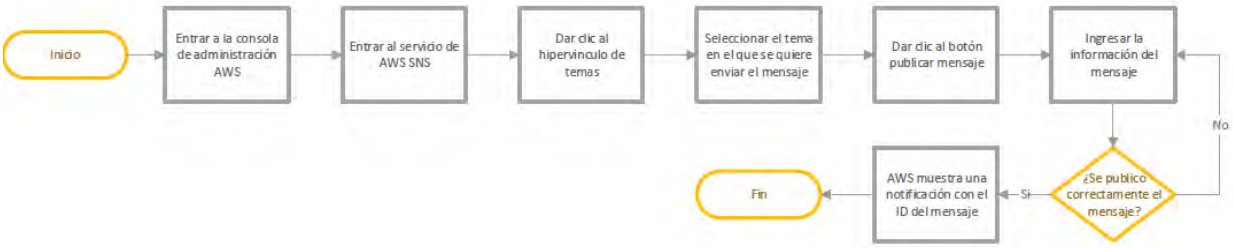

Figura 43: Pruebas de SNS a Lambda

El décimo tercero diagrama de la figura 44 describe como configurar el servicio AWS API Gateway. Este servicio como su nombre lo indica, es una puerta de enlace y sirve para proteger y no exponer la dirección publica de algún servicio o servidor. En el caso de aplicación se usa este servicio como un proxy para acceder al microservicio "AdvertAPI" que se encuentra implementado en la instancia EC2. La configuración más precisa de este servicio se puede revisar en el anexo XIII [34].

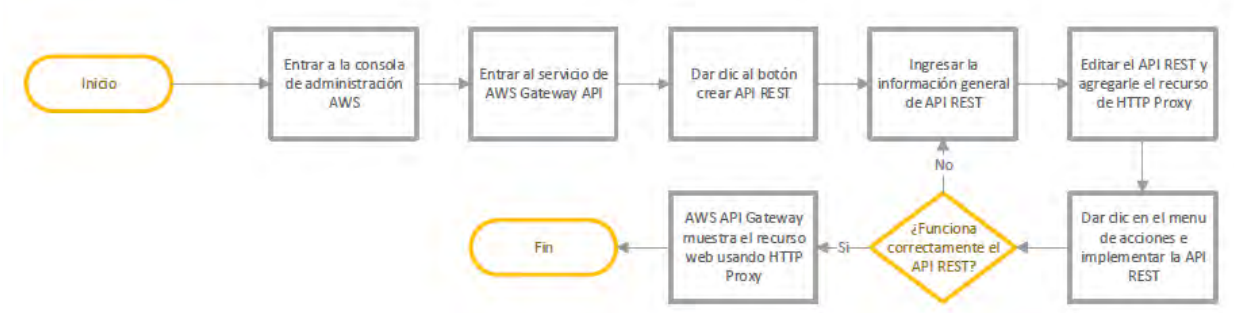

Figura 44: Configuración de Gateway API

El décimo cuarto diagrama de la figura 45 explica cómo crear una imagen Docker. Las imágenes Docker son archivos que contienen una aplicación con todas sus dependencias para su funcionamiento. Para el caso de aplicación se creó una imagen del microservicio "SearchAPI". Es aquí en donde entra las ventajas de usar esta tecnología ya que una vez generada la imagen se puede implementar en un entorno local, virtualizado con hipervisor o bien en la nube. Para ver cómo crear una imagen más en detalle consultar el anexo XIV [34].

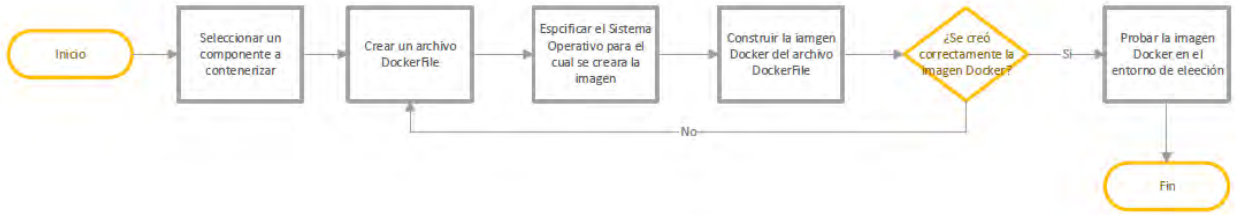

Figura 45: Creación de imagen Docker

El décimo quinto diagrama que se presenta en la figura 46 describe como configurar el servicio AWS ECR. Este servicio sirve para almacenar imágenes de contenedores y es similar a otras ofertas de registros como Docker Hub, Google Container Registry y Azure Container Registry, entre otros. En el caso de aplicación se utilizó para almacenar la imagen Docker del microservicio "Search API" de la cual posteriormente se creará la instancia del contenedor. En el anexo XV se puede ver en mayor detalle como configurarlo [34].

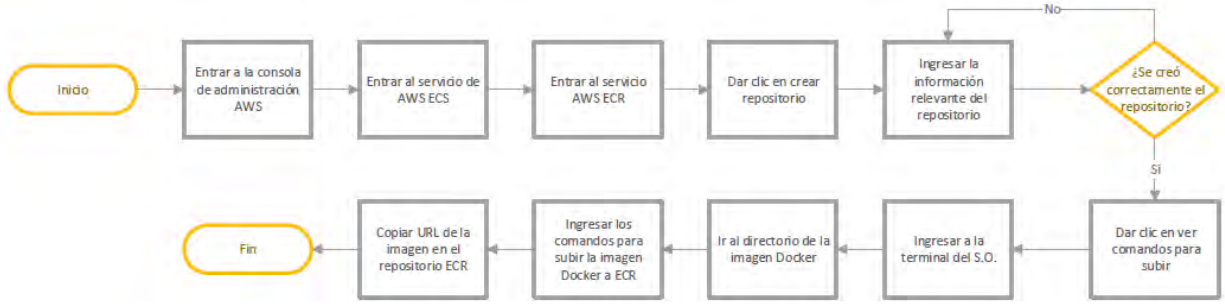

Figura 46: Configuración de ECR

El décimo sexto diagrama que se muestra en la figura 47 explica como configurar el servicio AWS Fargate. Este servicio sirve para crear clústeres de contenedores los cuales pueden tener una o más instancias en ejecución. Para el caso de aplicación se usa este servicio para crear una instancia del microservicio “Search API”. La ventaja de usar este servicio es que se encargará del escalamiento horizontal y vertical de la instancia de manera automatizada. Ahora bien, en el anexo XVI se muestra con mayor detalle como configurarlo [34].

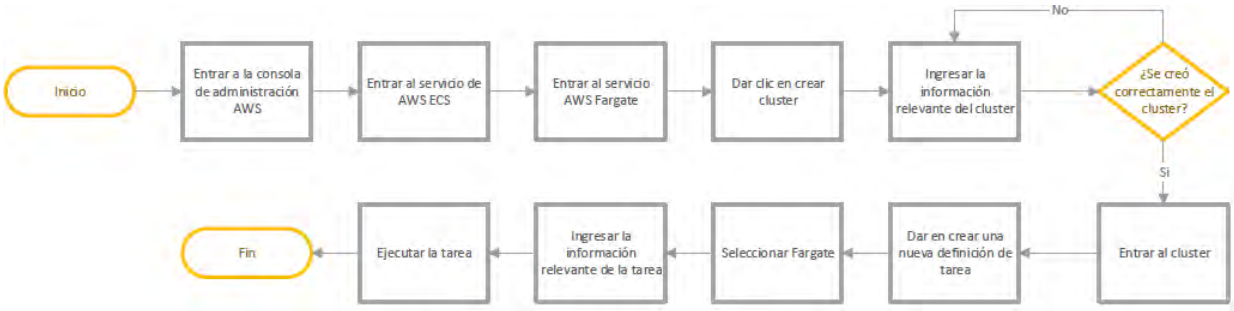

Figura 47: Configuración de Fargate

Capítulo V Resultados y conclusiones

La elaboración del presente trabajo surge como una propuesta que busca contribuir en el dominio del desarrollo de software como servicio (SaaS) usando contenedores, ya que de acuerdo con varios estudios académicos y empíricos se requieren el desarrollo de marcos de trabajo y metodologías que permitan guiar a la ingeniería de software en este nuevo dominio. En adición a ello, la adopción acelerada del cómputo en la nube por parte de las empresas debido a la situación sanitaria actual y la falta de ingenieros con conocimientos en el domino del desarrollo de SaaS usando contenedores han generado que los marcos técnicos de referencia y las metodologías actuales se queden obsoletas. Por tal motivo, el presente trabajo contribuye a este dominio haciendo tres aportaciones. La primera aportación es un marco técnico de referencia en donde se engloben los conceptos necesarios para el desarrollo de software como servicio (SaaS) usando contenedores. La segunda aportación es el desarrollo de una metodología que permita guiar de manera sistematizada a los ingenieros en este dominio y la tercera aportación es un caso de aplicación que permita validar lo propuesto en el marco técnico de referencia y la metodología.

Ahora bien, con respecto al desarrollo del marco técnico de referencia se utilizó el método deductivo partiendo de los conceptos más generales a los más específicos que son necesarios para el desarrollo de software como servicio (SaaS) usando contenedores. En el nivel más general no será necesario actualizar los conceptos ya que son los criterios mínimos establecidos por diferentes artículos, libros y el NIST para lograr cumplir con este objetivo. Sin embargo, para el caso de los niveles más específicos si requerirán actualizaciones conforme los proveedores de nube realicen cambios a sus diferentes servicios o bien dependiendo de las tecnologías particulares que el usuario decida utilizar.

De igual manera la metodología utiliza el modelo deductivo partiendo de la secuencia de pasos más general posible a los más específicos para el desarrollo de software como servicio (SaaS) usando contenedores. La metodología se apoya de los conceptos desarrollados en el marco técnico de referencia y muestra de manera sistematizada y ordenada los pasos necesarios para cumplir con los objetivos del trabajo recepcional. Sin embargo, al igual que ocurre con el marco técnico de referencia los niveles más específicos requerirán actualizaciones de acuerdo con las modificaciones que los proveedores realicen en los servicios que se usaron para su desarrollo o bien de acuerdo con las tecnologías que el usuario haya utilizado.

Para terminar estas dos aportaciones se validaron mediante la propuesta de un caso de aplicación orientada a la mercadotécnica y publicidad en donde se definieron sus requerimientos, diagrama de flujo, diagramas de arquitectura de solución y se explicó de manera detallada como configurar, desarrollar e implementar cada uno de sus servicios en la nube de AWS.

Anexos

Anexo I: Configuración de IAM

La primera sección en la implementación del caso de aplicación trata del proceso a seguir para la configuración del servicio AWS IAM. Este servicio sirve para crear cuentas en AWS, lo que permite acceder a la consola de administración o bien realizar llamadas a otros servicios desde web APIs. Para empezar con la configuración del servicio es necesario entrar a la consola de administración de AWS y buscar el servicio IAM. Después de entrar a IAM se presenta una página de bienvenida. Por lo general, en esta página se muestran varias alertas como se presenta en la figura 48. Esto se debe a que se está utilizando la configuración predeterminada, la cual es poco recomendable por su bajo nivel de seguridad. Sin embargo, una vez que se inicie el proceso de creación de la cuenta, se asigne los roles, permisos y autentificación de doble factor, se eliminarán las advertencias. Aclarado estos puntos, para crear una nueva cuenta en AWS se da clic al hipervínculo de usuarios el cual mostrará un asistente de configuración.

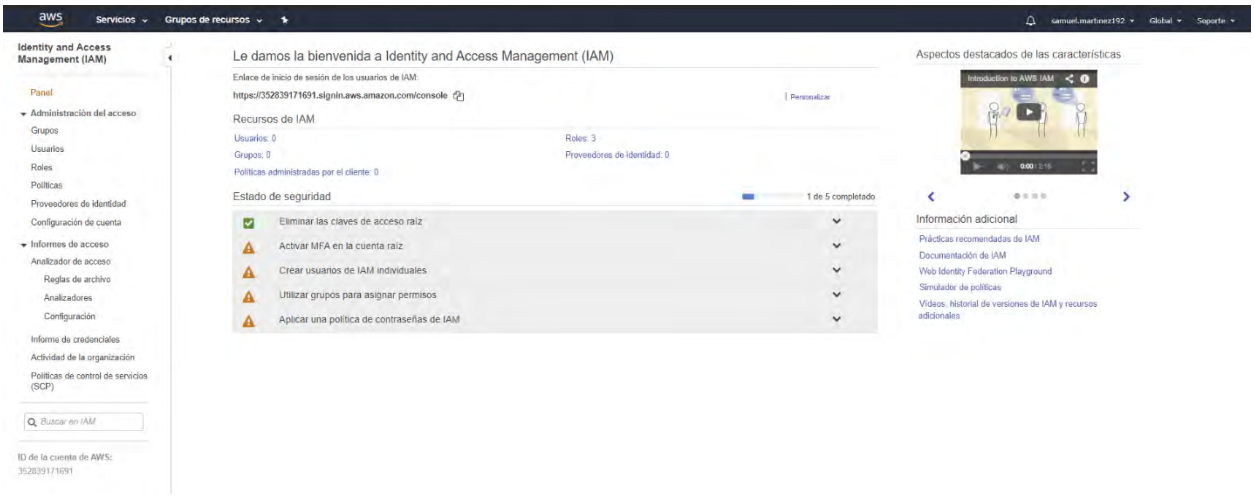

Figura 48: Página de inicio IAM

En el asistente de configuración se deben seguir cinco pasos para crear una nueva cuenta de usuario como se ve en la figura 49. El primer paso del asistente de configuración requiere que se ingrese un nombre de usuario, tipo de acceso y contraseña. Después se determina si es necesario asignar accesos mediante programación y consola de administración. Para el caso de aplicación es indispensable asignar accesos mediante programación ya que esto permitirá utilizar la cuenta de AWS en llamadas Web API. Para concluir, se asignó una contraseña que cumpla con los estándares recomendados de seguridad y se continuó con el siguiente paso del asistente de configuración.

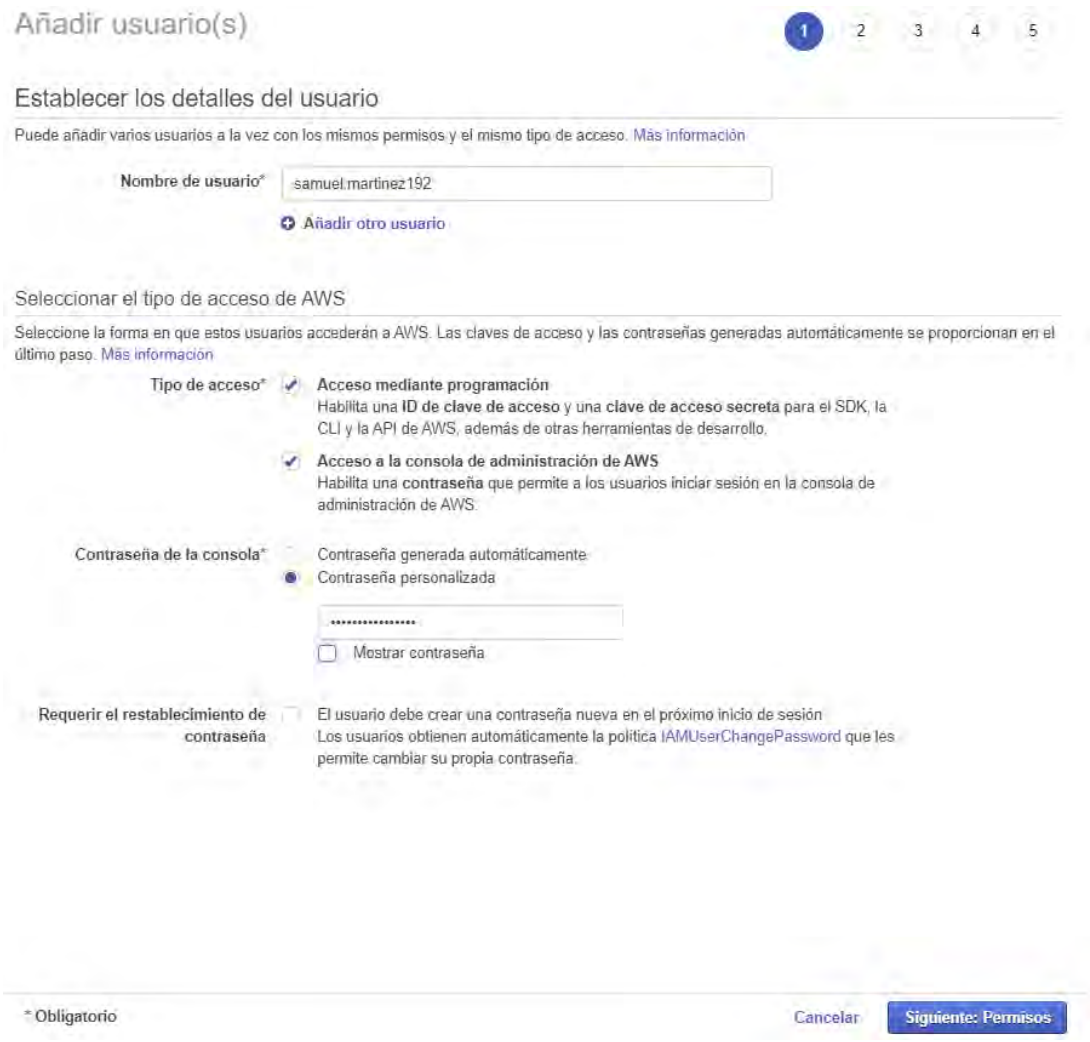

Figura 49: Página para añadir usuarios en IAM

En el segundo paso del asistente de configuración, se asignan los permisos que va a tener la nueva cuenta de usuario. Para el caso de aplicación se decidió crear una sola cuenta con permisos de administrador como se muestra en la figura 50. La decisión de asignar este nivel de permisos es debido a que el desarrollo del caso de aplicación es pequeño y se requiere depurar el software durante el proceso del desarrollo. No obstante, en entornos de producción y proyecto grandes, es recomendado crear cuentas con permisos específicos de las funciones que van a desempeñar.

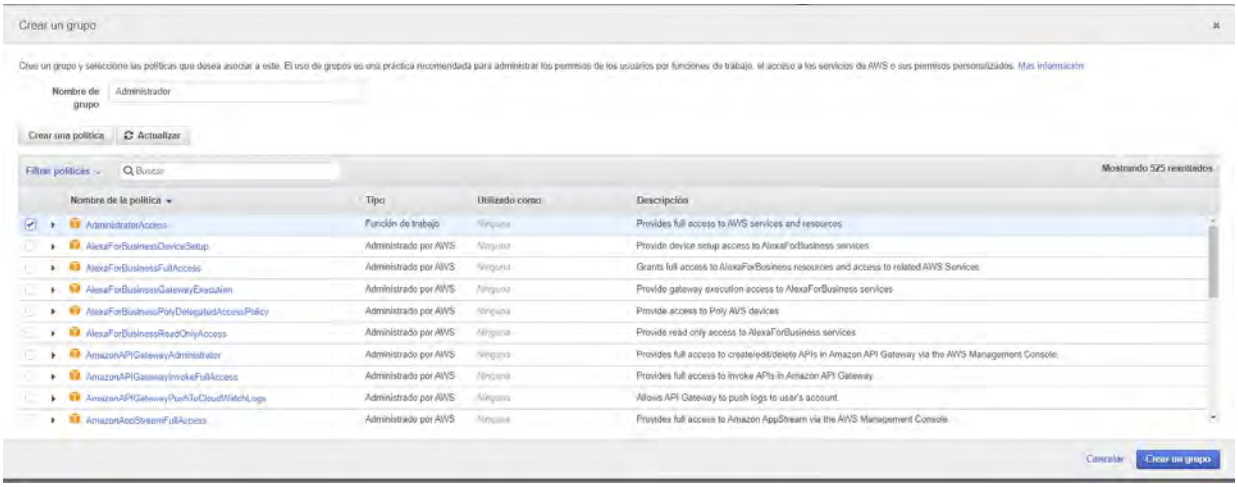

Figura 50: Página para añadir grupos y políticas en IAM

El tercer paso del asistente de configuración permite añadir etiquetas para la identificación de las cuentas de usuario en IAM. Se puede añadir información como correo electrónico, puesto de trabajo, entre otros. Para el caso de aplicación se omitió este paso y se continuó con el siguiente.

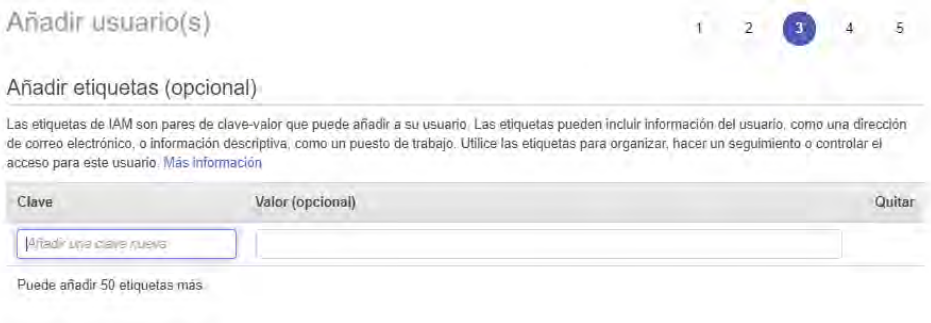

Figura 51: Página para añadir etiquetas en IAM

El cuarto paso del asistente de configuración muestra una página de revisión, en donde se ve un resumen de la configuración realizada en los pasos anteriores. En este apartado se puede realizar modificaciones en el alguno de los pasos anteriores en caso de ser necesario. Dado que no se encontró algún problema en los puntos anteriores se continuó con el último paso del asistente.

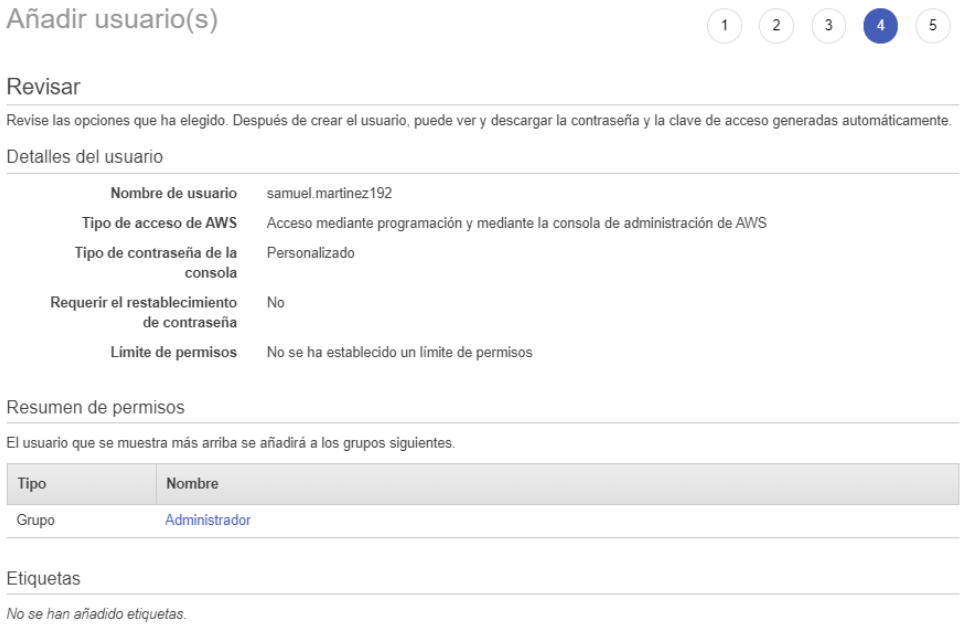

Figura 52: Página para revisar el usuario en IAM

El quinto y último paso del asistente de configuración, muestra un mensaje notificando que la cuenta de usuario se ha creado correctamente. De este paso es necesario descargar el archivo que se muestra en la figura 53, dicho archivo se utiliza en la configuración de credenciales, al momento de que ocurren llamadas a servicios AWS en las Web APIs del caso de aplicación.

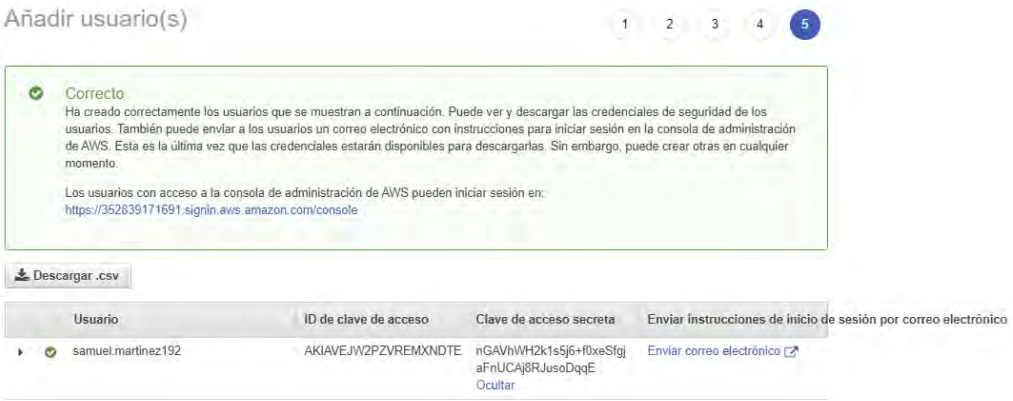

Figura 53: Página de notificación en IAM

Anexo II: Configuración de usuario

La segunda sección para la implementación del caso de aplicación es la configuración del usuario de AWS en el entorno de desarrollo. Esta configuración permitirá realizar llamadas a las librerías y herramientas de AWS. Para iniciar con la configuración se necesita crear una carpeta con el nombre ".aws" en el directorio home del usuario del sistema operativo como se muestra en la figura 54.

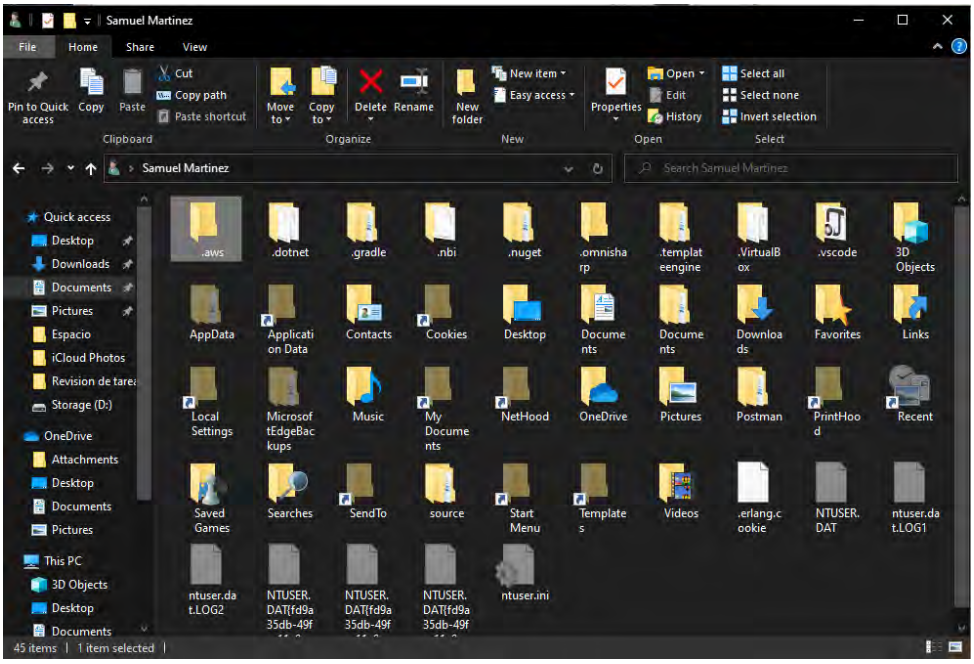

Figura 54: Carpeta AWS en el directorio home del usuario

Después de crear la carpeta ".aws", se precisa crear un archivo con el nombre de "credentials" y sin extensión como se muestra en la figura 55. Este archivo va a contener datos de acceso y región que son utilizados para el funcionamiento de las llamadas a librerías y herramientas de AWS.

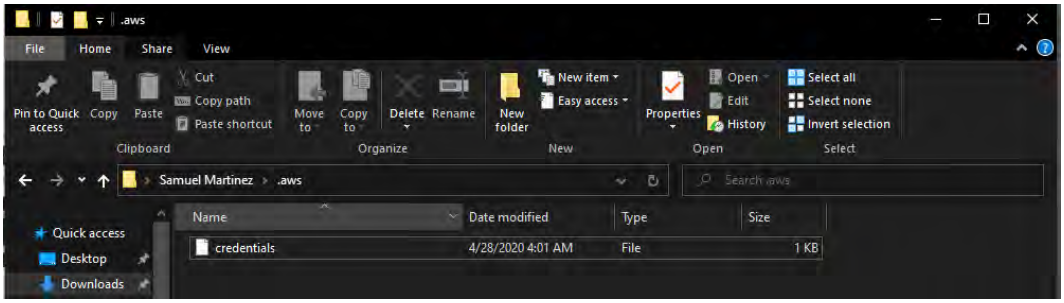

Figura 55: Archivo de credenciales guardado en la carpeta AWS

Con respecto al contenido del archivo “credentials”, deberá tener la información de la llave de acceso, llave de acceso secreta y la región de la cuenta de AWS que se generaron en la creación de la cuenta IAM. El archivo “credentials” es el que utilizan los microservicios y servicios en el caso de aplicación al momento de realizar llamadas a los servicios de AWS. También es utilizado por la consola de comandos cuando se realizan configuraciones a los servicios de nube, o bien cuando se empaqueta una función para subirla a AWS Lambda.

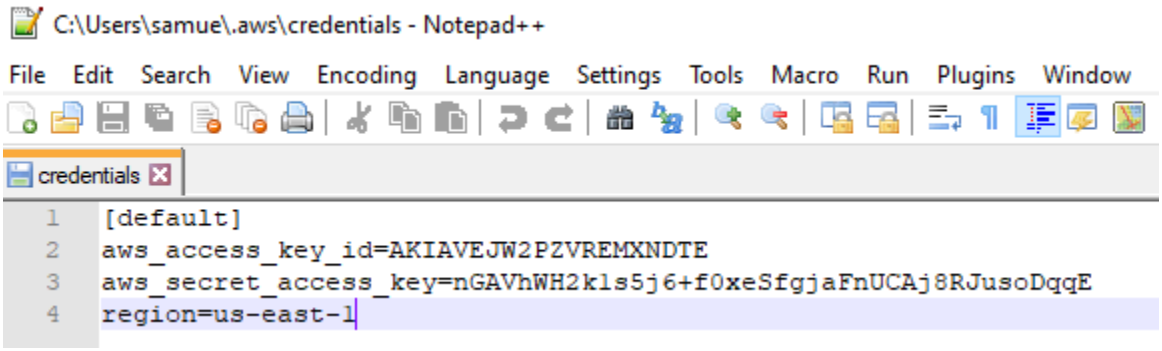

Figura 56: Contenido del archivo “credentials”

Anexo III: Configuración de Cognito

La tercera sección en la implementación del caso de aplicación es la configuración de AWS Cognito. Este servicio se empleará para administrar la autentificación de los usuarios en el caso de aplicación. Para comenzar con la configuración del servicio es necesario entrar a la consola de administración de AWS y buscar el servicio de Cognito. La primera página que se muestra al entrar a este servicio es una página de bienvenida con una breve descripción de Cognito, como se ve en la figura 57. En esta página hay dos botones. El primer botón sirve para crear grupos de usuario y el segundo botón permite administrar grupos de identidades. Para el caso de aplicación se utilizará únicamente el grupo de usuarios, por consiguiente, se da clic al botón administrar grupos de usuarios.

Figura 57: Página de inicio Cognito

Después de dar clic al botón de administrar grupo de usuarios, se presenta la página de la figura 58. En esta página se muestran los grupos existentes, dado que no existe ningún grupo, se creará uno nuevo dando clic al botón crear grupo de usuarios.

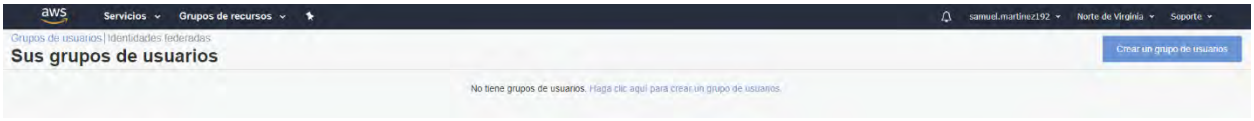

Figura 58: Página grupo de usuario en Cognito

Posterior a dar clic al botón de crear grupo de usuarios se muestra la página de la figura 59. En esta página se tienen múltiples apartados en los que se pedirá información relevante para crear el grupo de usuarios. Por ejemplo, el nombre del grupo de usuarios, atributos, políticas, etiquetas, entre otros. Para dar comienzo con la creación del grupo de usuarios se asignará un nombre relevante al caso de aplicación.

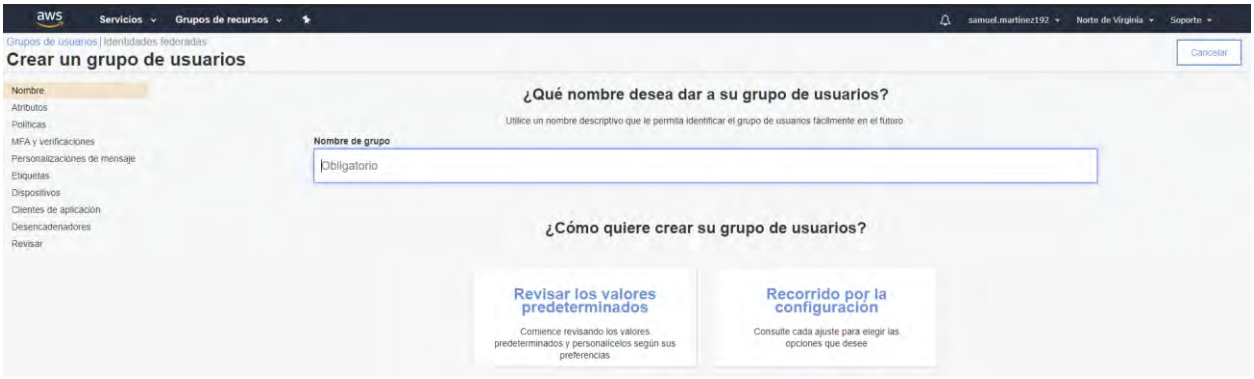

Figura 59: Página nombre del grupo de usuarios en Cognito

En este punto se recomienda asignar un nombre que tenga relación con el proyecto o aplicación con el cual se esté trabajando. Dado que el caso de aplicación está orientado a los anuncios de publicidad y mercadotécnica se optó por asignar el nombre de “WebAdvert” como se muestra en la figura 60.

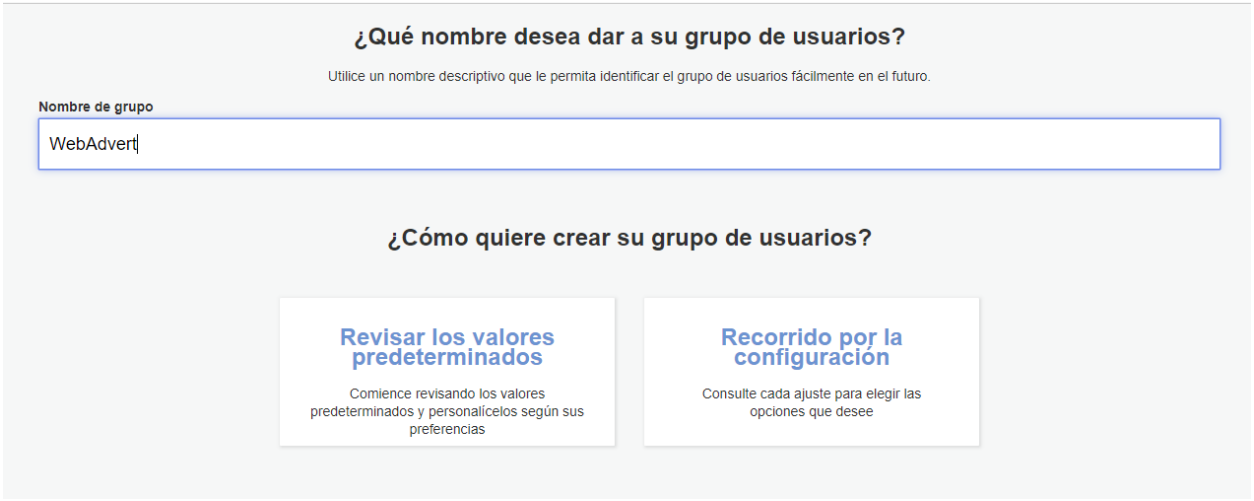

Figura 60: Página nombre del grupo de usuario en Cognito

Después de asignar el nombre al grupo de usuarios, AWS muestra la página de la figura 61. En esta página se editan atributos del grupo de usuarios que se está creando. Por ejemplo, se puede editar la distinción de minúsculas y mayúsculas en las cadenas de texto, la longitud de caracteres mínima para la contraseña, la política de seguridad en la contraseña, la autentificación de doble factor, entre otros. Para el caso de aplicación editaremos algunos atributos dando clic al icono de lápiz que se aprecia en la figura 61.

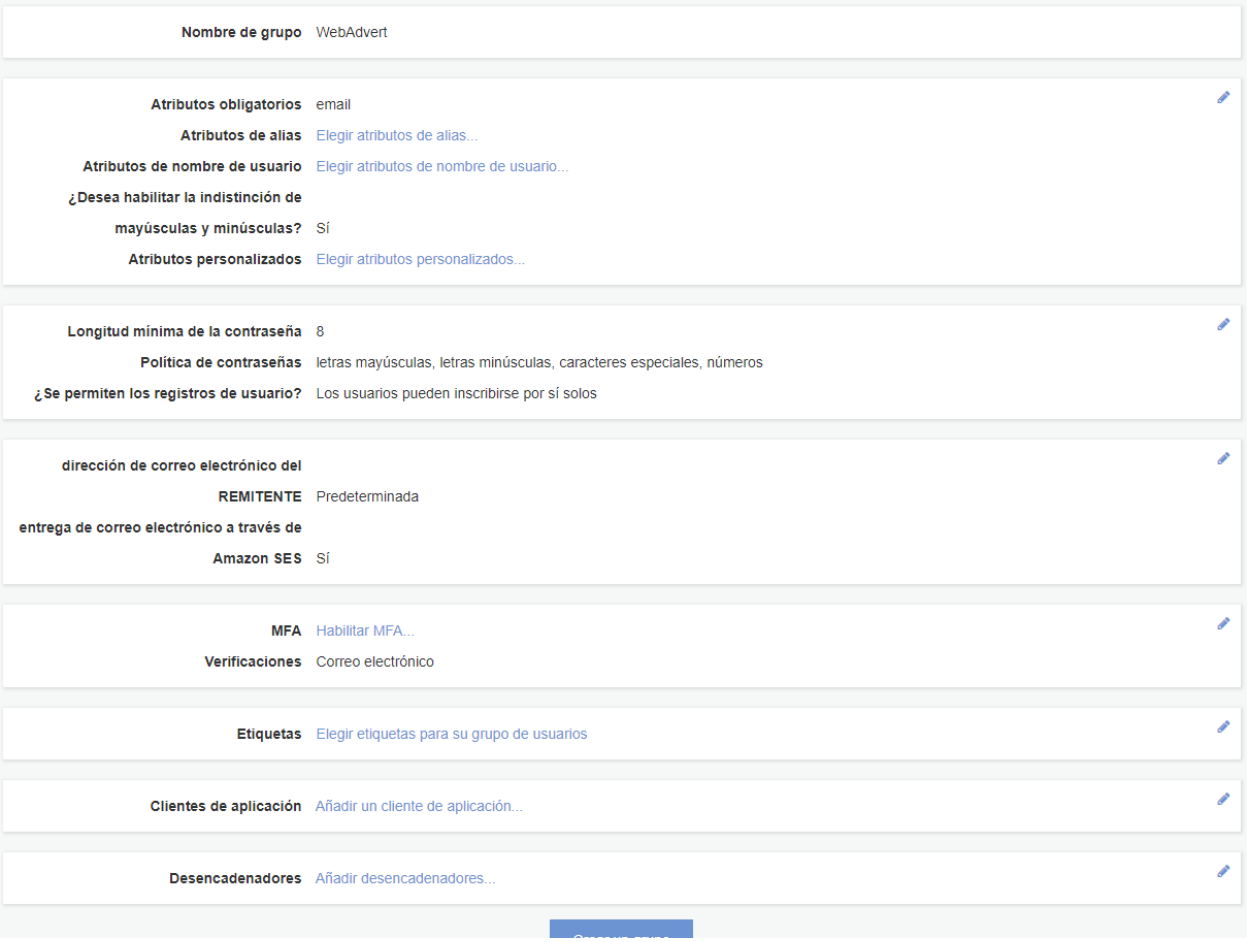

Figura 61: Página para editar valores de grupos en Cognito

El primer atributo que se editó del servicio de Cognito, son los correos electrónicos. En este atributo se verifico que la opción del envío de mensajes por correo electrónico estuviera deshabilitada y en su lugar se utilizará el servicio de Cognito. Esta configuración se muestra en el apartado de configuración Amazon SES en la figura 62. En cuanto las otras opciones de personalización para el correo electrónico no se utilizaron para el caso de aplicación. No obstante, dependiendo del proyecto los usuarios pueden personalizar las opciones de correo remitente, correo destinario y región del SES de acuerdo con sus necesidades.

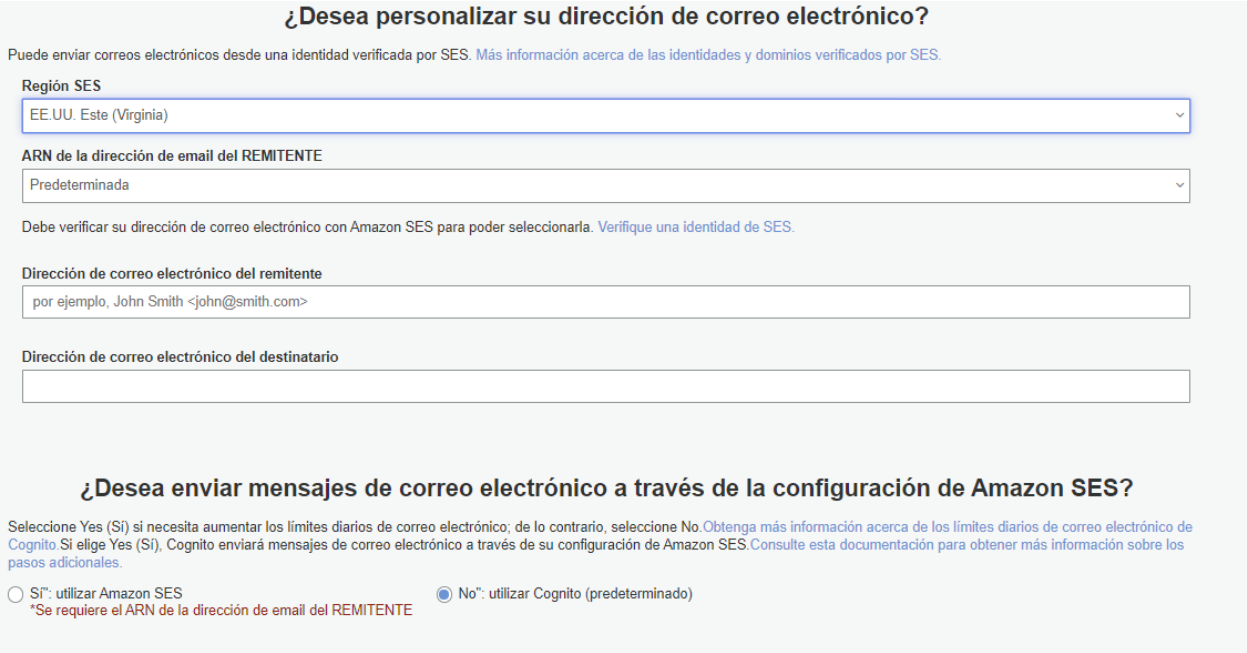

Figura 62: Página para personalizar email en Cognito

El segundo atributo que se editó fue la verificación del correo electrónico. Para fines del caso de aplicación se utilizó la opción predeterminada de verificación por código, como se muestra en la figura 63. Esto quiere decir que una vez que un usuario se registre en el sistema se deberá enviar un código de verificación a su correo electrónico. Es importante resaltar, que cualquier opción que se decida utilizar en este apartado es útil y depende del arquitecto de software determinar cuál opción se adapta mejor a su proyecto.

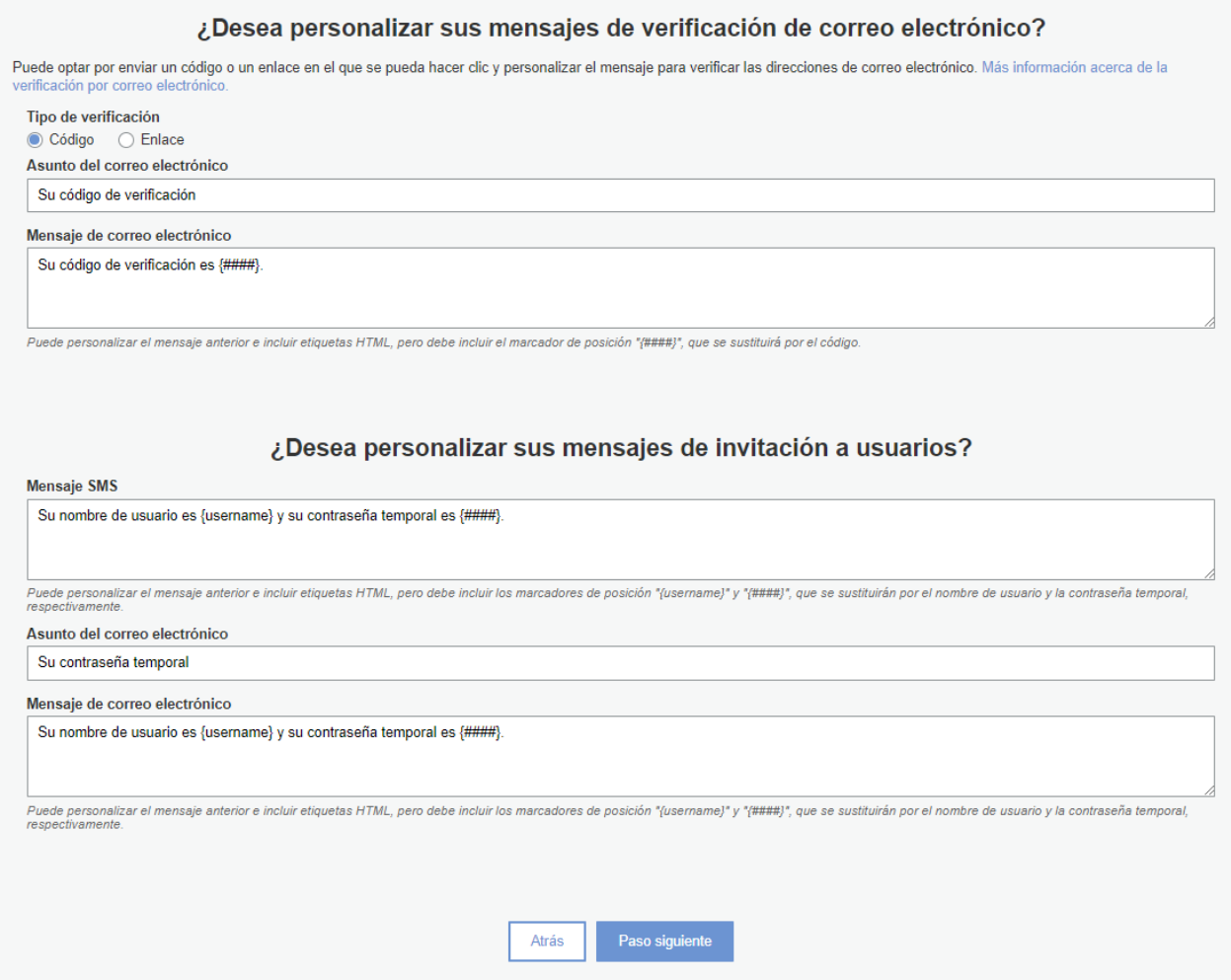

Figura 63: Página para personalizar mensajes en Cognito

El tercer atributo que se editó corresponde a las etiquetas para el grupo de usuarios. Para el caso de aplicación no se realizó alguna edición para las etiquetas. Por lo cual, se continuo con el siguiente punto.

Figura 64: Página para añadir etiquetas en Cognito

El cuarto atributo que se editó sirve para recordar los dispositivos en los que los usuarios del sistema inician sesión. Para el caso de aplicación se decidió no recordar los dispositivos del usuario y pedir la autentificación cada vez que se entre al sistema.

Figura 65: Página para recordar dispositivos en Cognito

El quinto atributo que los usuarios de AWS pueden editar son los clientes de aplicación que tienen acceso al grupo de usuarios. Por el momento se omitirá este atributo y se editará después de que se haya creado el grupo de usuarios.

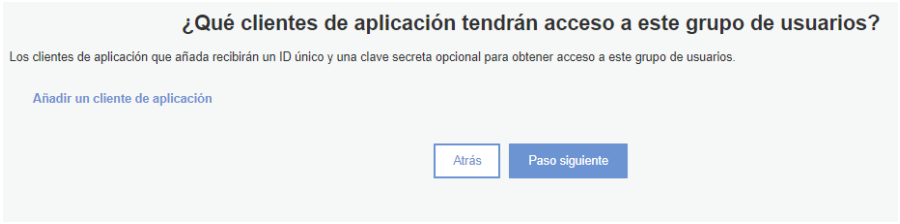

Figura 66: Página para configurar clientes de aplicación en Cognito

El sexto atributo que se editó son los desencadenadores de Cognito, los cuales funcionan mediante llamadas a funciones AWS Lambda. En este apartado AWS pone a disposición una amplia variedad de funciones. Por ejemplo, funciones para la personalización de mensajes, verificación de respuesta de autentificación, desafío de autentificación, entre otras. Sin embargo, no se utilizará desencadenadores en Cognito para el caso de aplicación, ya que es un proyecto pequeño.

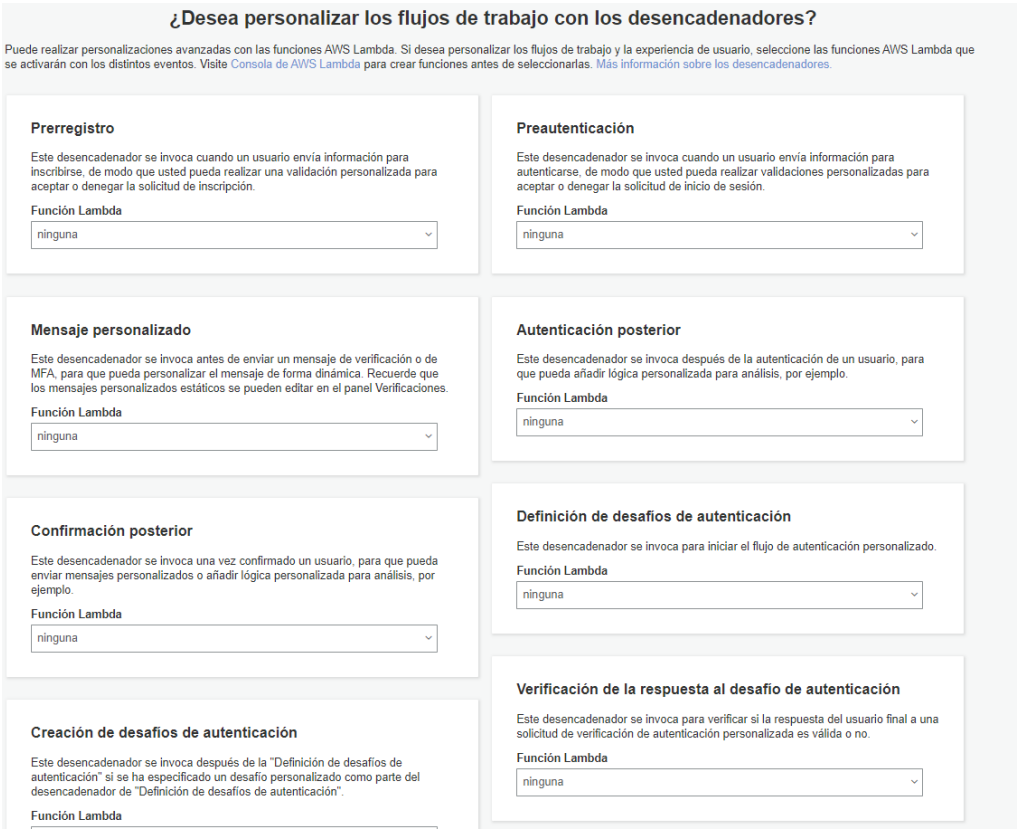

Figura 67: Página para configurar desencadenadores en Cognito

Para terminar con la creación del grupo de usuarios se regresa a la página principal en donde se ve un resumen de todos los atributos que se modificaron o editaron del grupo de usuarios que se va a crear. En la figura 68 se aprecia este resumen en los atributos del nombre de grupo (WebAdvert), el campo de nombre de usuario como atributo obligatorio, la longitud de 6 caracteres para la contraseña, los clientes de aplicación, entre otros. En esta página se da clic al botón de crear un grupo y se espera que el asistente realice la configuración del servicio.

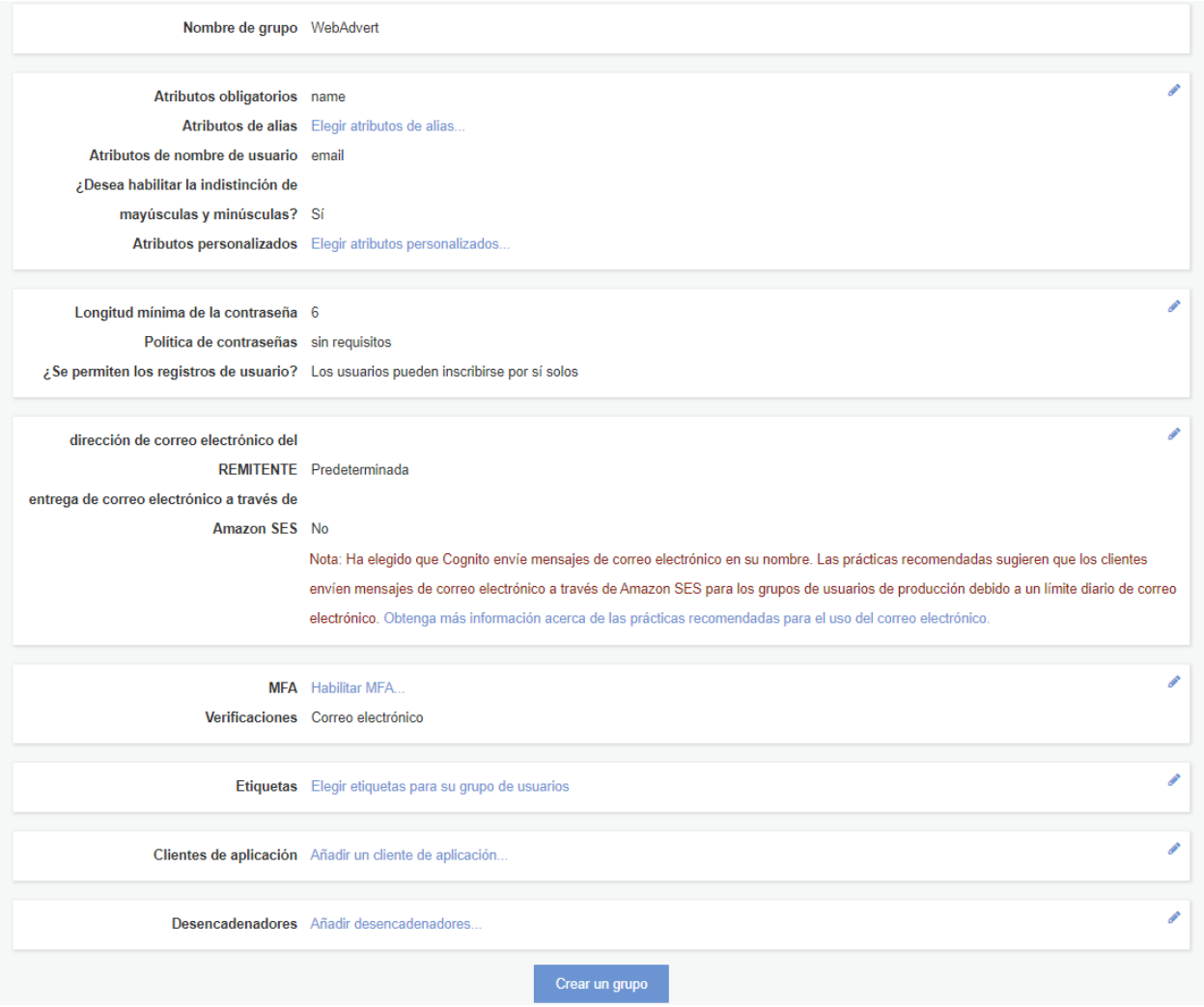

Figura 68: Página de resumen de configuración en Cognito

En la figura 69 se muestra la notificación que indica que se ha creado el grupo de usuarios correctamente. De esta página se necesita guardar los datos de Id de grupo y el ARN ya que se usarán en secciones posteriores del caso de aplicación.

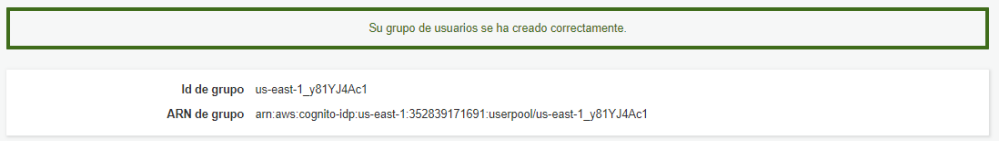

Figura 69: Datos de id grupo y ARN en Cognito

Posterior a crear el grupo de usuarios se necesita crear un cliente de aplicación que permitirá que se acceda a los servicios de autentificación de Cognito desde una aplicación móvil o web. Para ello se entra al grupo de usuarios y en la pestaña de cliente de aplicación se inicia con el proceso.

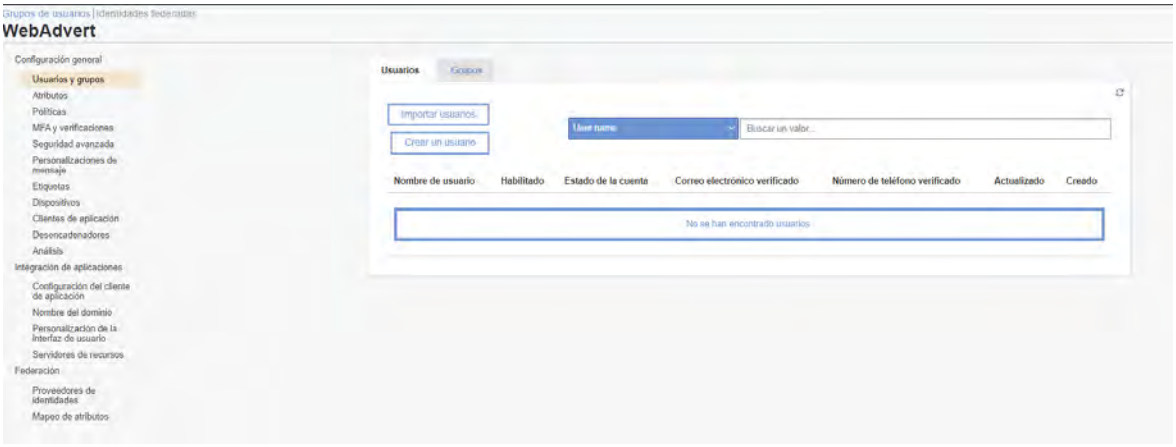

Figura 70: Página de configuración del grupo de usuario en Cognito

Dentro de la pestaña de clientes de aplicación se muestra los clientes existentes. Dado que no existe ninguno se creará el primer cliente de aplicación dando clic al enlace de añadir cliente de aplicación como se muestra en la figura 71.

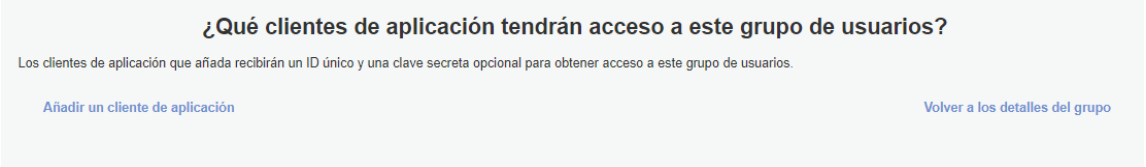

Figura 71: Página para añadir clientes de aplicación en Cognito

El asistente de configuración pedirá ingresar información relevante al cliente de aplicación que tendrá acceso al grupo de usuarios. En este apartado se ingresó el nombre del cliente aplicación, el vencimiento del token, la generación de una clave secreta y los flujos de autentificación del cliente. En la figura 72 se muestra que se habilitó la autenticación basada en el protocolo SRP y la autentificación basada en token de actualización.

Figura 72: Página para añadir cliente de aplicación en Cognito

Después de ingresar los datos relevantes para el cliente de aplicación se da clic al botón de crear cliente de aplicación lo cual generará que se muestre una página emergente con información del ID de cliente de aplicación y la clave secreta. Estos datos son necesarios en el caso de aplicación para las llamadas Web API al servicio de Cognito.

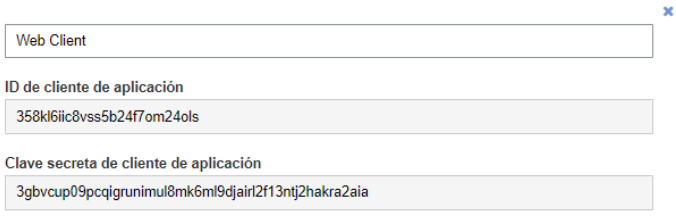

Figura 73: Configuración del cliente de aplicación y clave secreta en Cognito

Anexo IV: Configuración de DynamoDB

La cuarta sección en la implementación del caso de aplicación es con respecto al almacenamiento de anuncios. Para almacenar los anuncios se necesita de un sistema gestos de base de datos para lo cual existen las bases de datos tradicionales SQL y las orientadas a documentos NoSQL. En este punto se decidió utilizar el servicio de DynamoDB el cual es un servicio de base de datos NoSQL. Para empezar con la configuración del servicio se entra a la consola de administración de AWS y se busca el servicio de DynamoDB. Después de entrar al servicio se muestra la página de la figura 74 con una breve descripción del servicio y un botón azul para crear tablas. Para crear una nueva tabla se da clic al botón y se mostrará una página que pedirá más detalles.

Figura 74: Página de inicio DynamoDB

Esta es la única página que se muestra para crear tablas en DynamoDB. En esta página se deben proporcionar datos como el nombre de la tabla y una llave primaria o campo principal. Para el caso de aplicación solo se necesita una tabla para almacenar anuncios. Por lo cual, se ingresó el nombre de "adverts" a la tabla y el valor de "id" como la llave primaria. Después de ingresar esta información se da clic a un botón para crear tablas y listo ya se puede empezar a guardar anuncios en DynamoDB.

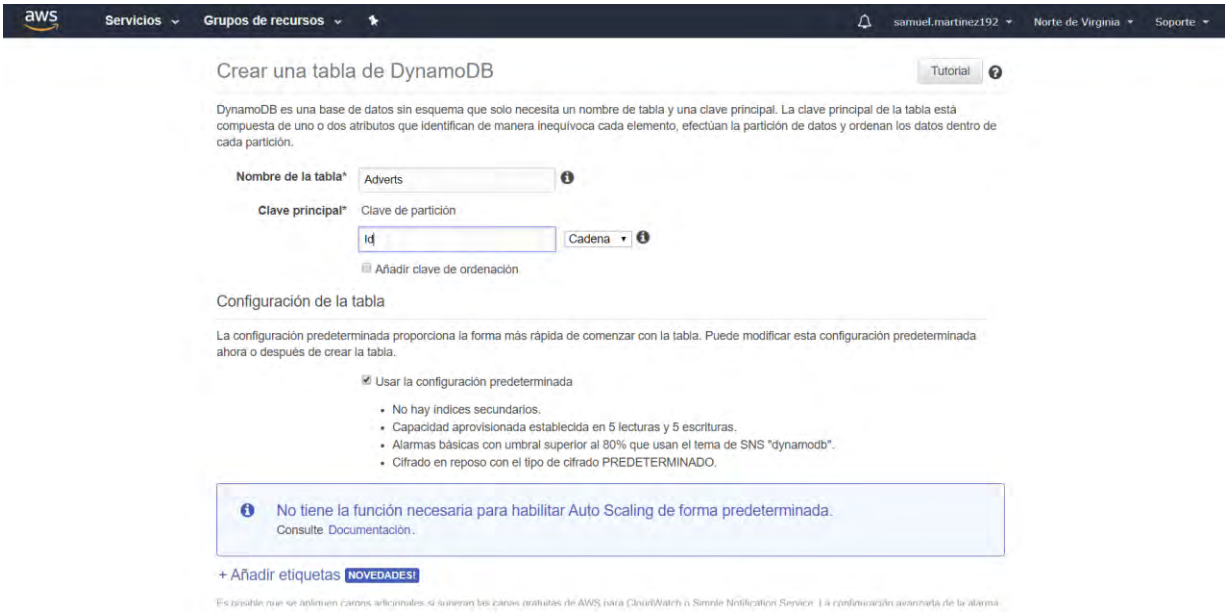

Figura 75: Página para crear tablas en DynamoDB

Anexo V: Configuración de EC2

La quinta sección para la implementación del caso de aplicación es la configuración de una instancia EC2 para el despliegue de uno de los microservicios. Para comenzar con el proceso de configuración de EC2 se entra a la consola de administración de AWS y se busca el servicio EC2. Posterior a esto se muestra la página de inicio de la figura 76. En esta página se da clic al enlace de instancias para crear la instancia EC2 que se usará en el caso de aplicación.

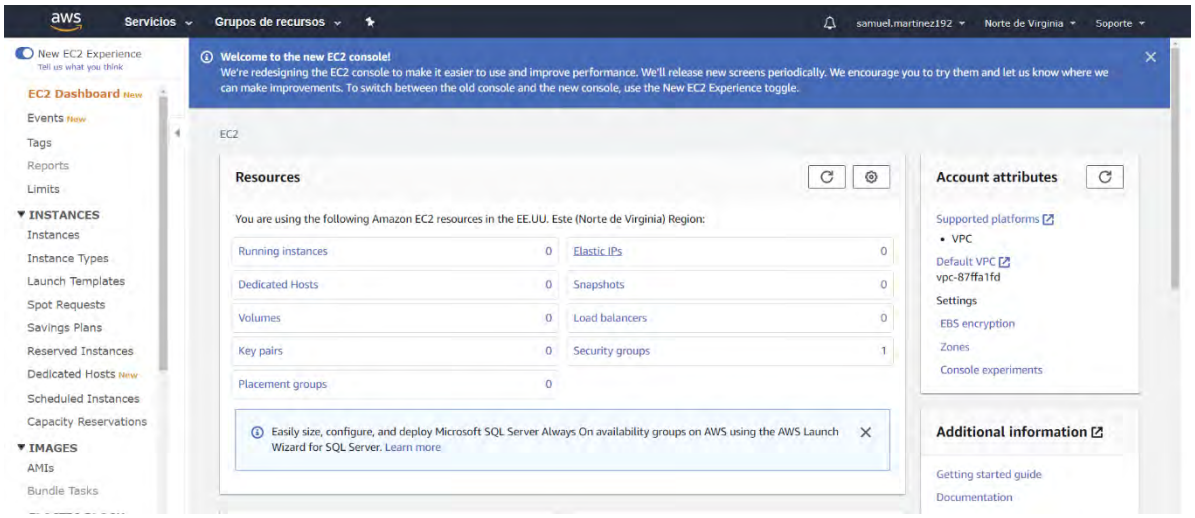

Figura 76: Página de inicio EC2

La primera página que se presenta en el enlace de instancias muestra que instancias existen en la zona regional predeterminada de la cuenta de AWS. En la figura 77 se puede ver que no hay instancias creadas hasta el momento. Por lo cual, se creará la primera instancia dando clic al botón de lanzar instancia lo cual ocasionará que se inicie un asistente de configuración.

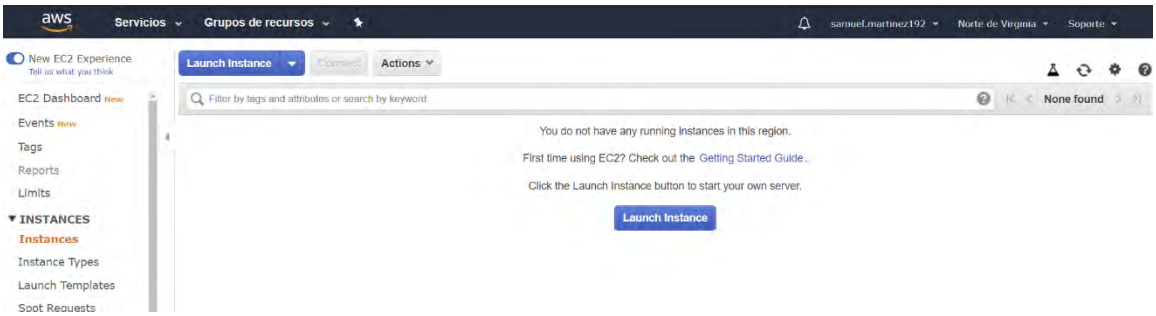

Figura 77: Página para crear instancias en EC2

El primer paso del asistente de configuración para crear instancias en EC2 se ve en la figura 78. En este apartado se debe seleccionar una imagen de máquina Amazon (AMI), en la cual existe una gran variedad de sistemas operativos y distritos Linux. Para el caso de aplicación se utilizará una imagen de Windows Server 2016 por la facilidad de implementación en comparación a un distrito Linux. No obstante, se puede utilizar cualquier distrito de Linux o Mac OS X, ya que el microservicio se desarrolló usado la tecnología .NET Core la cual es independiente del sistema operativo.

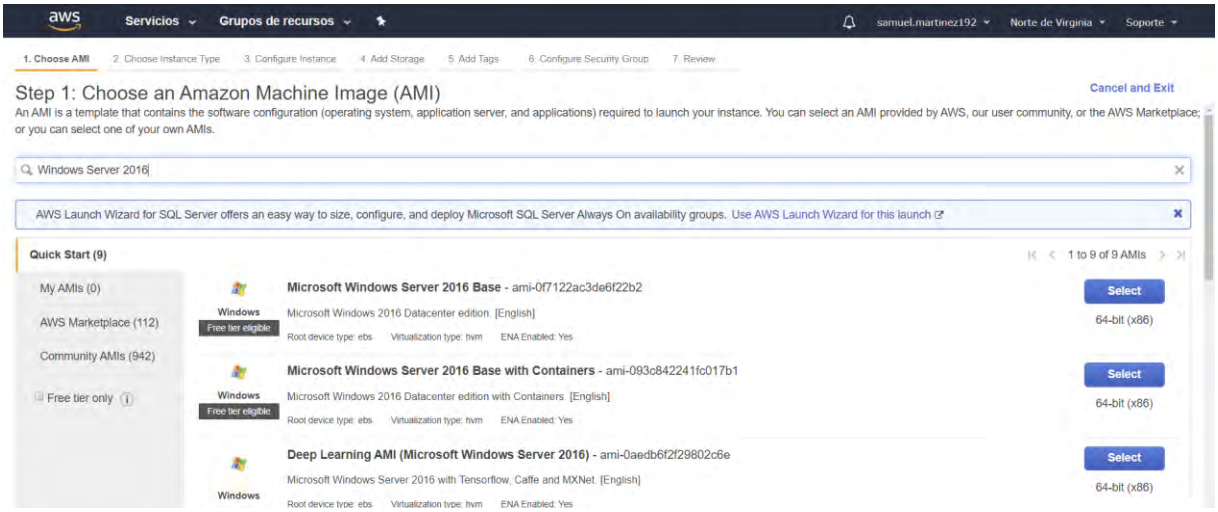

Figura 78: Página para seleccionar imagen de máquina en EC2

En el segundo paso requiere la selección de un tipo de instancia. El tipo de instancia determina cuantos hilos de ejecución, memoria RAM y red tendrá a disposición la instancia EC2 que se está configurando. Dado que el caso de aplicación es una aplicación pequeña en la que se quiere realizar pruebas se optó por el tipo de instancia T2.micro la cual se encuentra en la capa gratuita de AWS.

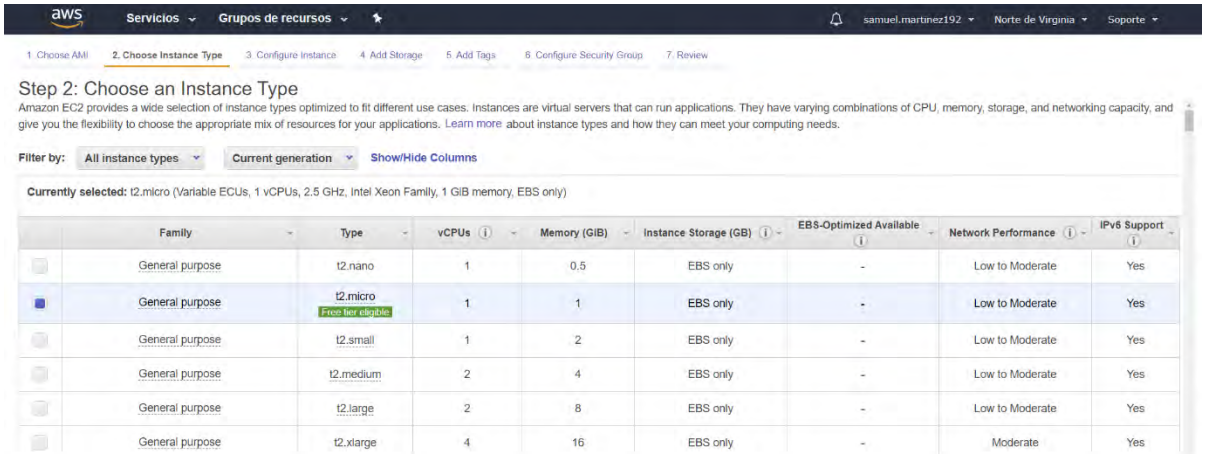

Figura 79: Página para seleccionar tipo de instancia en EC2

El tercer paso requiere ingresar los detalles de la instancia que se va a crear. En este apartado AWS proporciona una gran cantidad de opciones, como la cantidad de instancias que se van a crear, la configuración de VPC privada, la asignación de una IP publica, el comportamiento de las instancias al entrar en hibernación o apagarlas, entre otras. Todas estas opciones se dejaron con sus valores predeterminados como se muestra en la figura 80 y se continua con el siguiente punto.

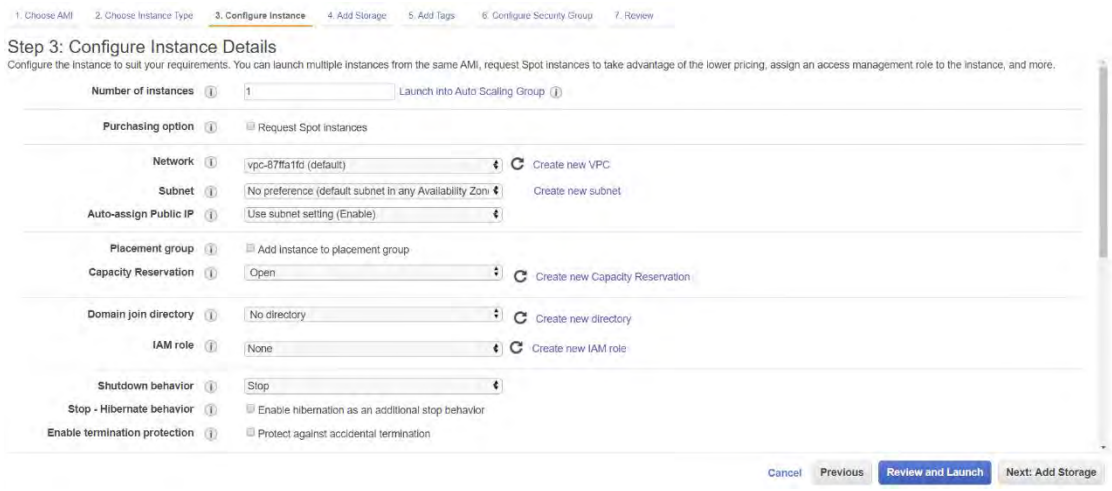

Figura 80: Página para configurar detalles de instancia en EC2

En el cuarto paso el usuario debe especificar el tipo y tamaño de almacenamiento de la instancia EC2. Dado que el microservicio que se implementará en la instancia es pequeño, se seleccionó la opción de almacenamiento SSD de 30GB de propósito general. Sin embargo, AWS tiene múltiples opciones que permite alternar entre almacenamiento mecánico o SSD y diferentes tamaños de memoria.

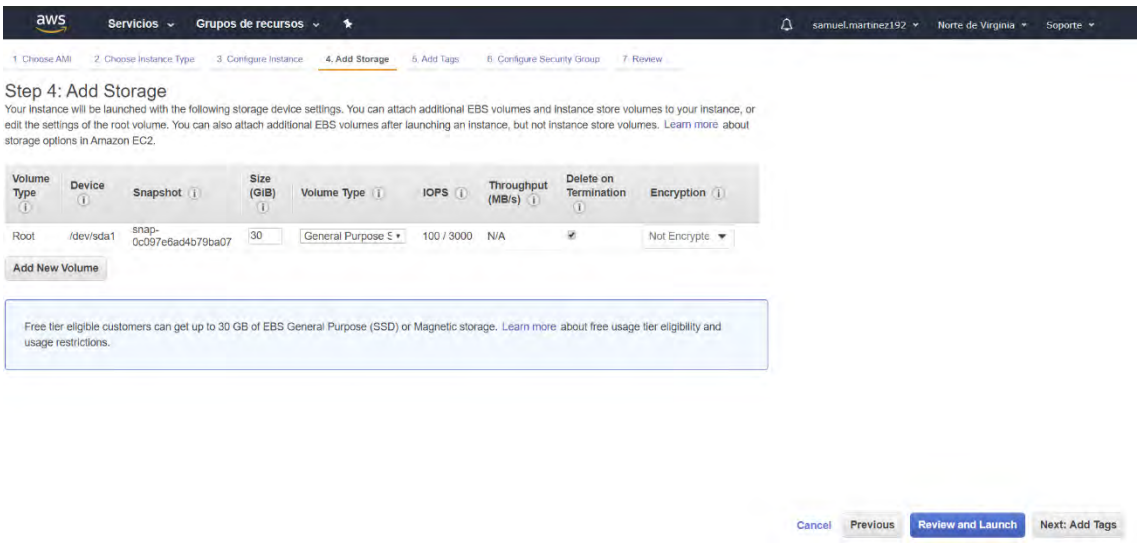

Figura 81: Página para añadir almacenamiento en la instancia EC2

En el quinto paso se definen etiquetas para la instancia EC2. Para este punto solo se definió una etiqueta con la llave de “Nombre” y un valor de “WebAdvertApi” como se muestra en la figura 82. Esta etiqueta se utilizará para identificar la instancia y volúmenes de la instancia. No obstante, el usuario puede añadir más etiquetas que se adapten a las necesidades de su proyecto.

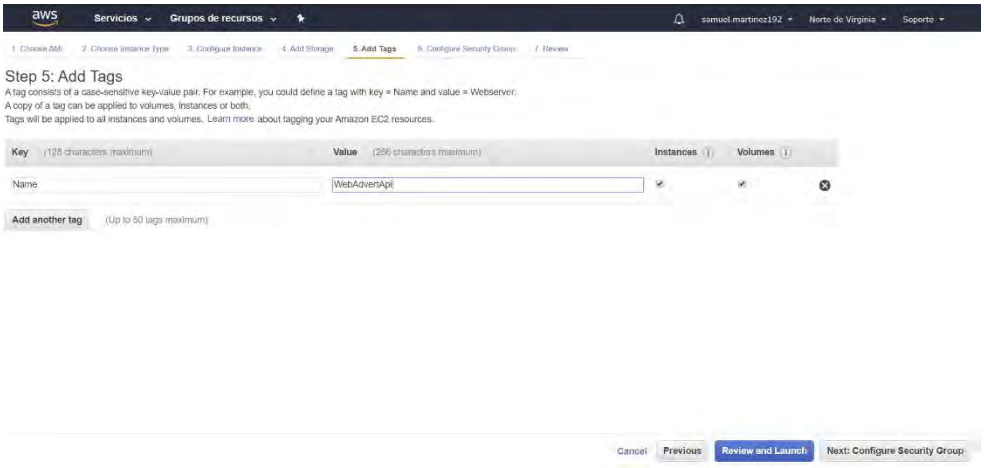

Figura 82: Página para añadir etiquetas en EC2

En el sexto paso se debe definir la seguridad del grupo de la instancia EC2. Este apartado básicamente son las reglas de firewall que controla el tráfico que se dirige hacia la instancia EC2. Por lo cual, se debe definir reglas como el tipo de servicio, protocolo, puertos, origen y descripción.

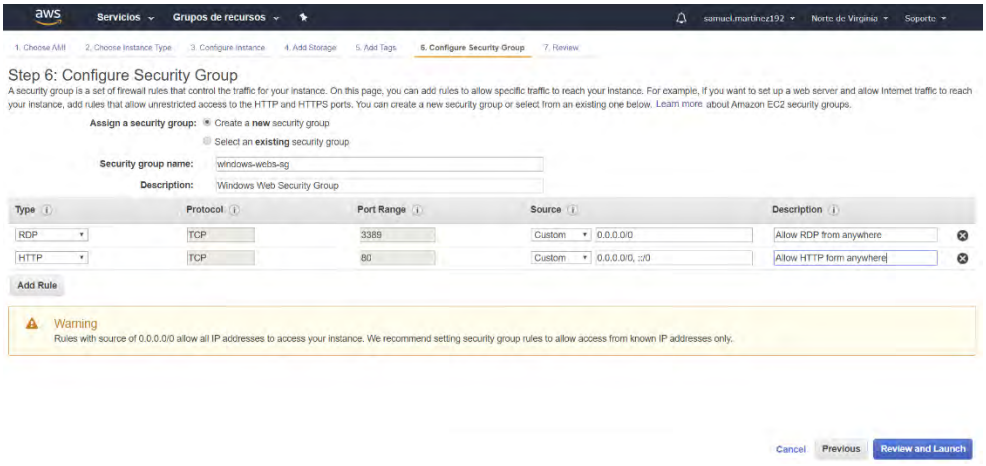

Figura 83: Página para configurar grupo de seguridad en EC2

El séptimo y último paso del asistente es un resumen de todas las configuraciones hechas en los pasos anteriores. En la figura 84 se muestra la descripción de todas las configuraciones realizadas anteriormente, como la máquina Amazon (Windows Server 2016), el tipo de instancia (T2.Micro), los grupos de seguridad, entre otros. En caso de que haya algún parámetro que el usuario necesite cambiar se puede regresar a puntos anteriores. Dado que no es necesario realizar cambios se procede con la creación de la instancia EC2 dando clic al botón de lanzar.

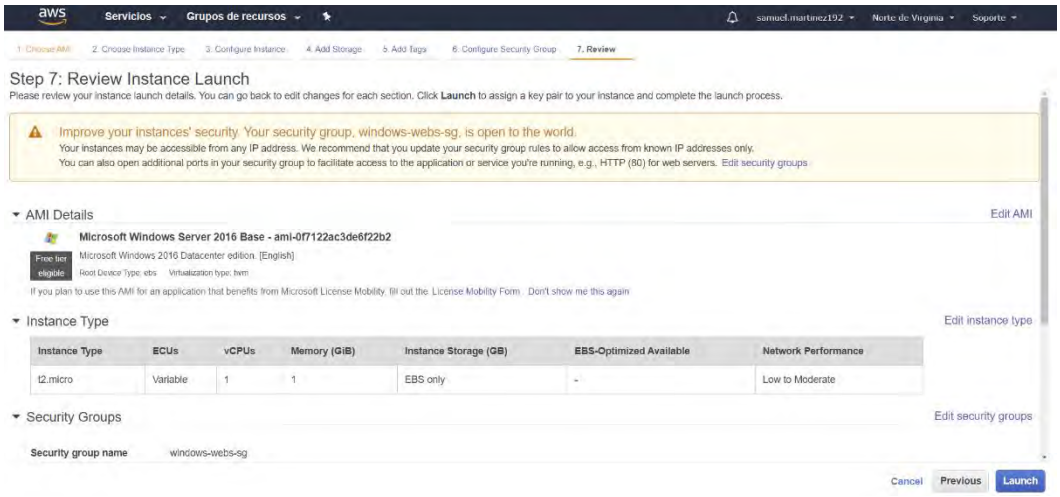

Figura 84: Página de resumen de configuración en EC2

El asistente deberá mostrar la ventana emergente de la figura 85. En esta ventana se notifica que se requiere descargar un archivo de llave que permitirá conectarse de manera segura a la instancia EC2. Este archivo es el que permite desencriptar la contraseña que se crea de manera predeterminada para conectarse a EC2.

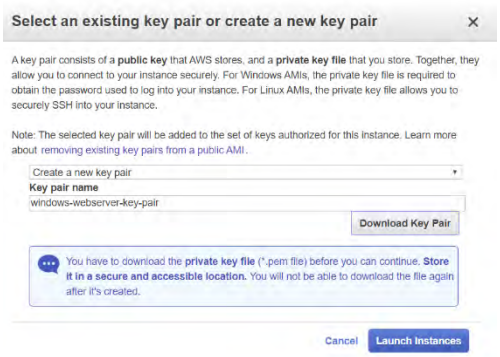

Figura 85: Llave cifrada para obtener contraseña en EC2

Si se realizó correctamente la configuración para crear la instancia EC2 se debe mostrar un mensaje que notifica que la instancia se está inicializando como se ve en la figura 86. Para concluir con la configuración de este servicio, es necesario desencriptar la contraseña para conectarse a la instancia y posteriormente cambiarla.

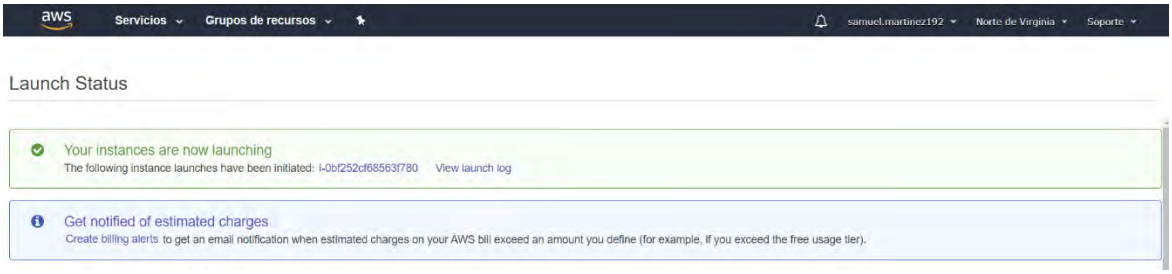

Figura 86: Página de estatus de instancia en EC2

Para desencriptar la contraseña se entra al enlace de instancias y se puede ver que ya existe una instancia con el nombre de “WebAdvertApi” en la figura 87. Después de entrar al enlace de instancias en el apartado de descripción se desencripta la contraseña.

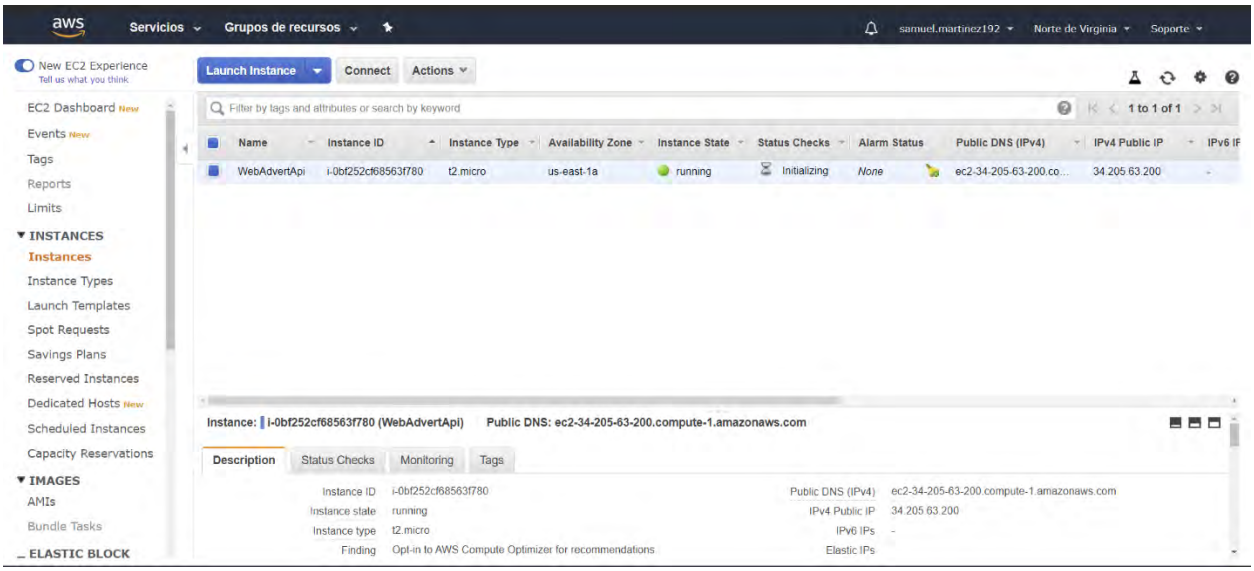

Figura 87: Página de instancias disponibles en EC2

Para desencriptar la contraseña se deberá mostrar la ventana emergente de la figura 88. En esta ventana es necesario utilizar el archivo .key que se descargó en los pasos anteriores. De esta manera se descripta la contraseña predeterminada para conectarse a la instancia y posteriormente se cambia.

Figura 88: Desencriptando la contraseña por defecto en EC2

Después de desencriptar la contraseña se presenta el mensaje de la figura 89. En esta ventana se muestra los datos del nombre de usuario y contraseña en texto plano. Con estos datos se conecta a la instancia EC2 mediante RDP para verificar el estado de la máquina y el sistema operativo. Una vez que se verifica que todo está funcionando adecuadamente se puede cambiar la contraseña.

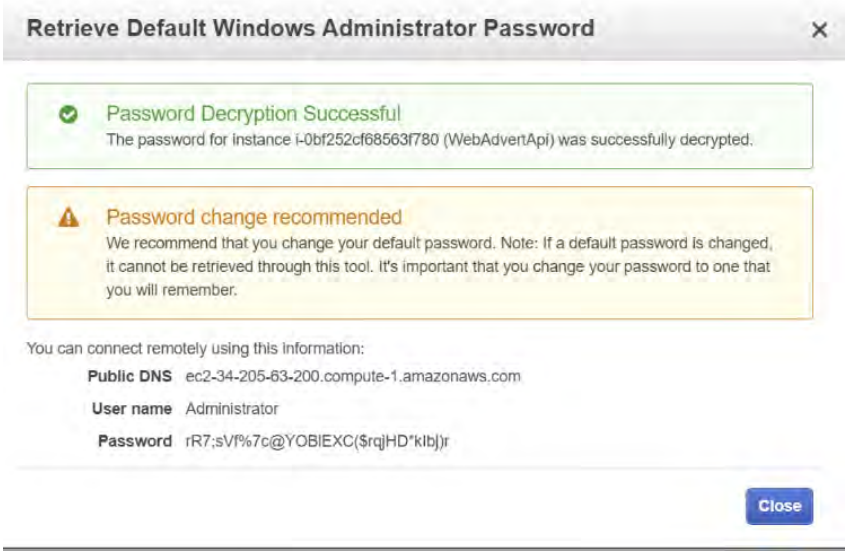

Figura 89: Notificación de desencriptación de contraseña en EC2

Anexo VI: Configuración de S3 para Code Deploy

La sexta sección para la implementación del caso de aplicación es en relación con la configuración de una cubeta en S3 para los despliegues de Code Deploy a la instancia EC2. Para empezar con el proceso de configuración de la cubeta S3 se entra a la consola de administración y se busca el servicio de S3. Después de entrar a la página principal se entra a la pestaña de cubetas como se ve en la figura 90. En esta página se ingresará un nombre de cubeta y se bloqueará el acceso público. La cubeta S3 se utilizará para almacenar el microservicio con sus dependencias en un archivo comprimido en formato zip.

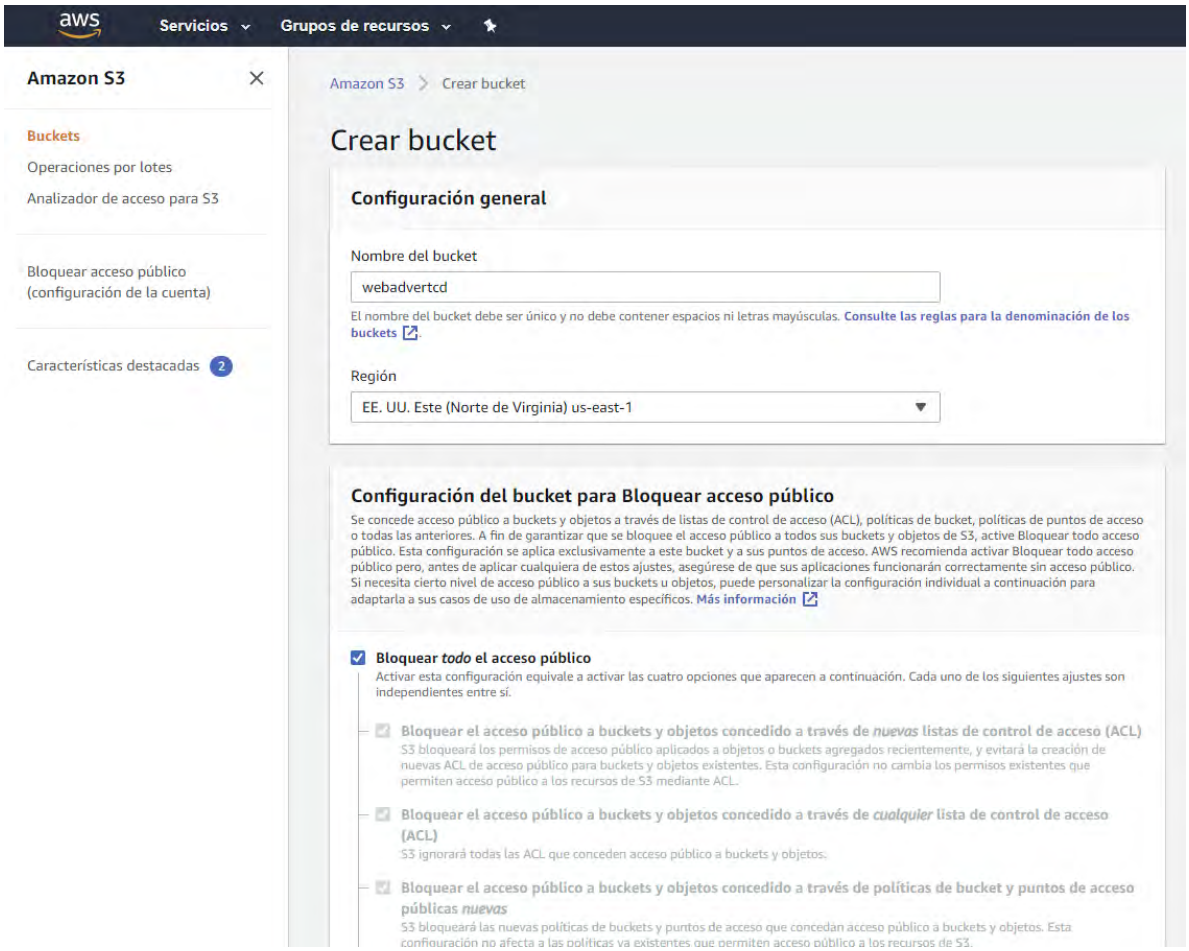

Figura 90: Página para crear cubetas en S3

Para terminar con la creación de la cubeta AWS deberá mostrar un mensaje que se ha creado una cubeta en el servicio S3 como se ve en la figura 91. En la figura 91 se presenta que existe una cubeta llamada "web-advert-code-deploy" en la región norte de Virginia la cual no tiene un acceso público.

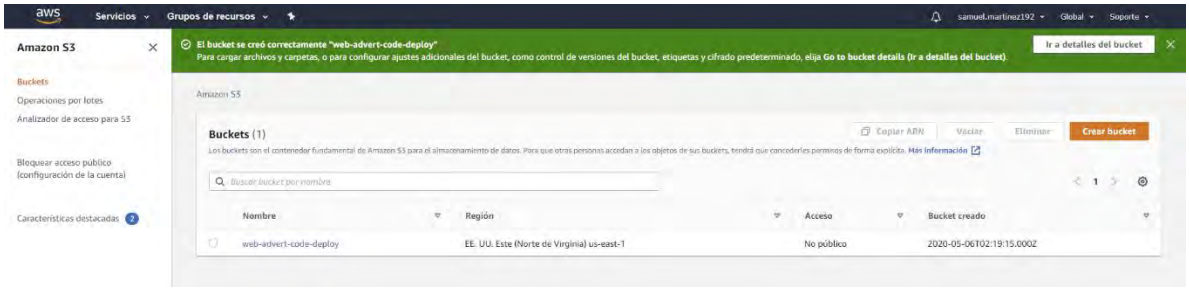

Figura 91: Página que muestra las cubetas en S3.

Anexo VII: Configuración de Code Deploy para EC2

La séptima sección para la implementación del caso de aplicación detalla el proceso a seguir en la configuración de Amazon CodeDeploy para los despliegues automáticos de microservicios a las instancias en EC2. En primer lugar, se entra a la consola de administración y se busca el servicio de CodeDeploy. Después de entrar al servicio se ve la página de la figura 92 con una breve descripción del servicio y las pestañas o enlaces a los que se puede acceder para configurar el servicio. En este apartado se necesita crear una nueva aplicación entrando al enlace de aplicación y dando clic al botón de crear aplicación.

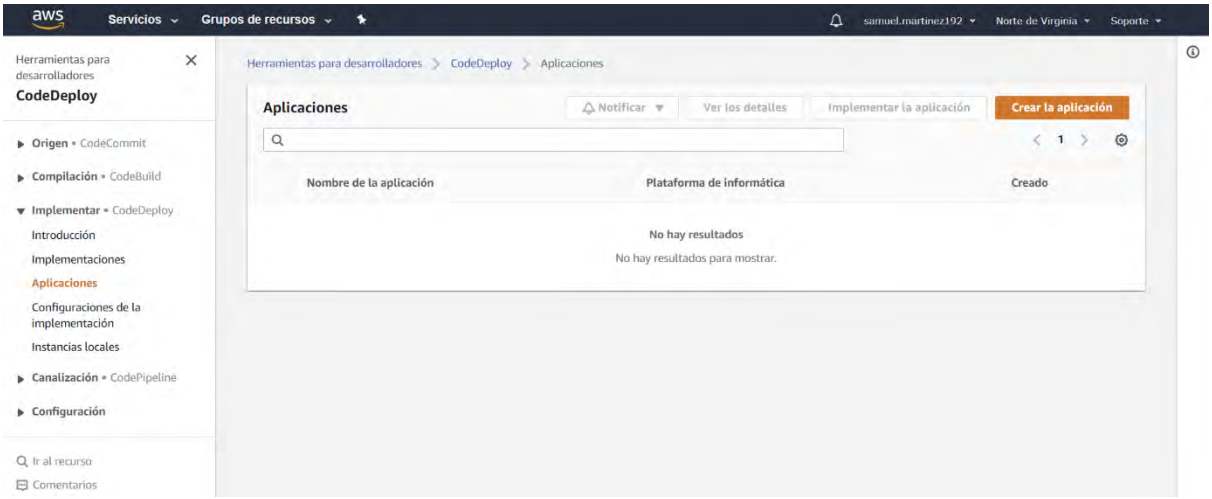

Figura 92: Página para crear aplicaciones en CodeDeploy

La primera página que se muestra para crear una aplicación en Code Deploy se ve en la figura 93. En esta página se debe de asignar un nombre de la aplicación y seleccionar una plataforma informática a la cual se harán los despliegues de código. Dado que el microservicio “AdvertApi” del caso de aplicación se implementará en una instancia EC2, se selecciona esta opción como plataforma informática.

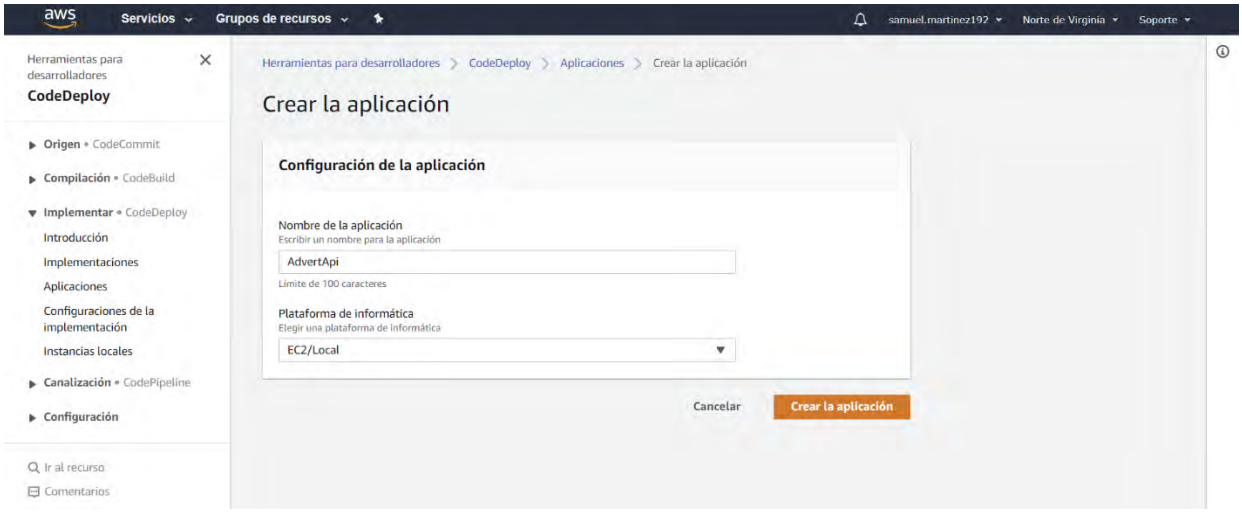

Figura 93: Página para crear aplicaciones en CodeDeloy

Después de ingresar la información para la aplicación, se muestra el mensaje de la figura 94 que indica que la aplicación se creó correctamente. El siguiente paso es crear un grupo de implementación dando clic al botón crear el grupo de implementación.

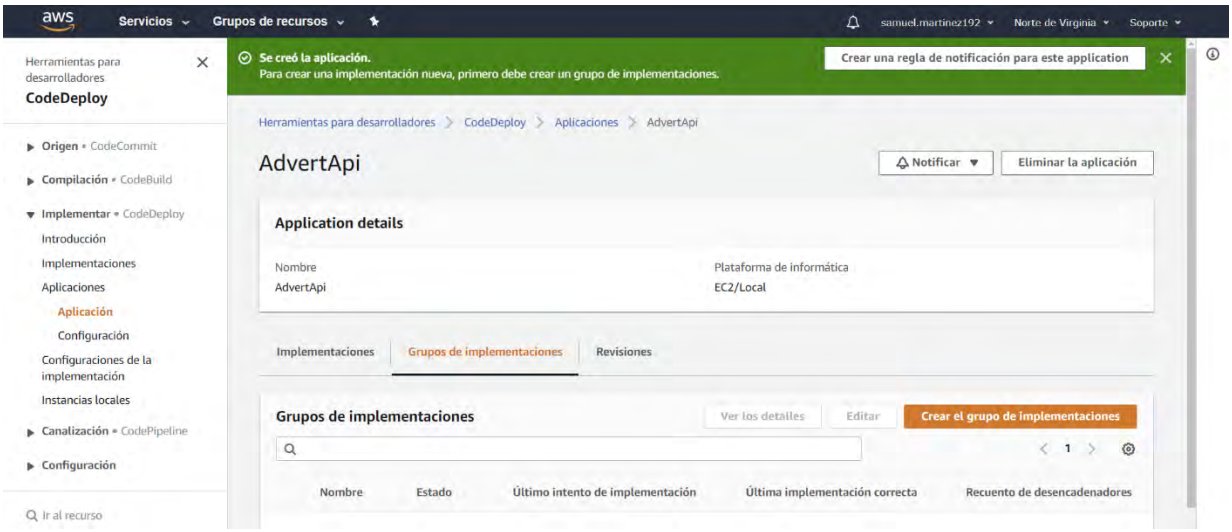

Figura 94: Página para crear grupos de implementación en CodeDeploy

Los grupos de implementación tiene como función guardar un histórico de las implementaciones de código que se hacen a las plataformas informáticas. En la figura 95 se muestra la página para crear un grupo de implementación en la que se necesita ingresar un nombre para el grupo de implementación, así como realizar varias configuraciones en cuanto a implementación, entorno y balanceador de carga.

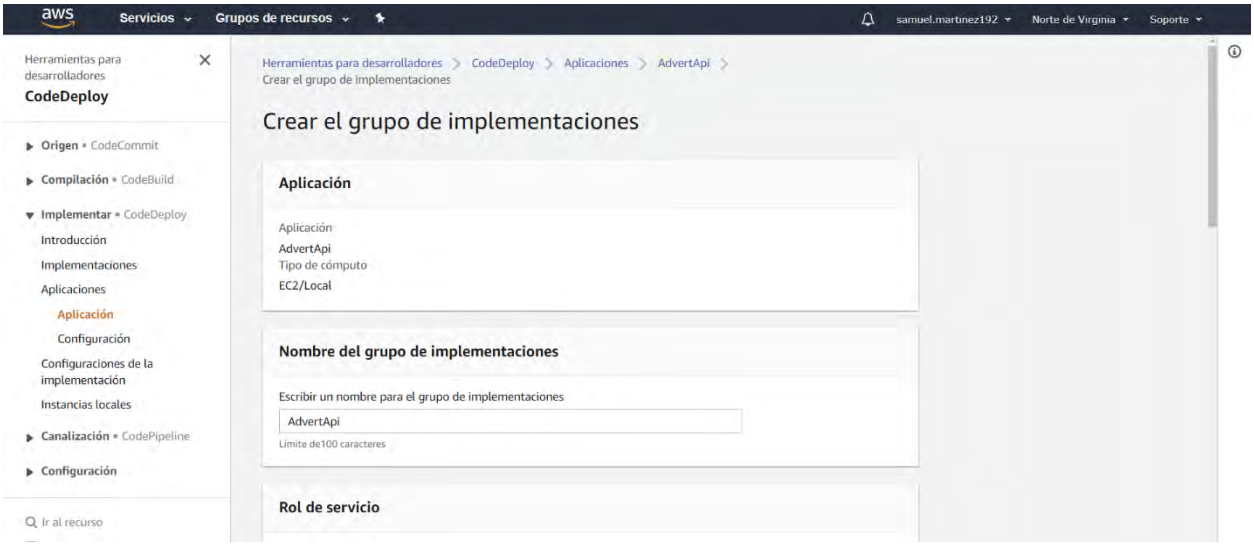

Figura 95: Página para crear grupo de implementación en CodeDeploy

Continuando con la misma página se debe seleccionar un tipo de implementación para la cual se tienen dos opciones. La primera se denomina como “En el lugar” y actualiza todas las instancias existentes con las versiones más recientes de la aplicación. La segunda se conoce como “Azul/Verde” y sustituye las instancias del grupo de implementación con nuevas instancias que contiene la última versión de la aplicación.

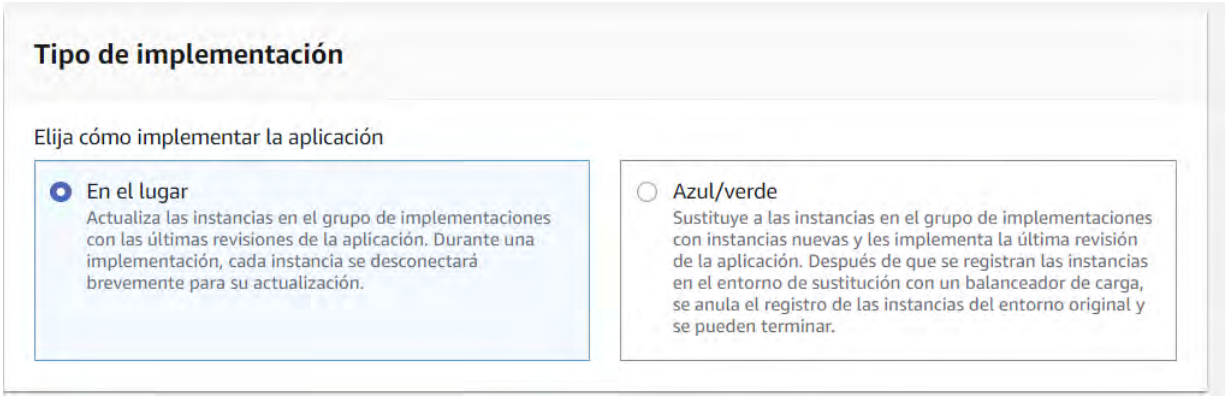

Figura 96: Pagina para configurar el tipo de implementación en CodeDeploy

En cuanto a la sección de configuración del entorno se tiene dos opciones. La primera es hacer despliegues a instancias EC2 que se encuentren en un grupo de auto escalamiento y la segunda opción es hacer despliegues a instancias individuales. Para el caso de aplicación se utiliza la opción de una instancia individual, dado que, solo se implementará un microservicio a una sola instancia EC2. Sin embargo, como se puede apreciar AWS ofrece opciones de configuraciones robustas que son ideales para ayudar a los ingenieros DevOps.

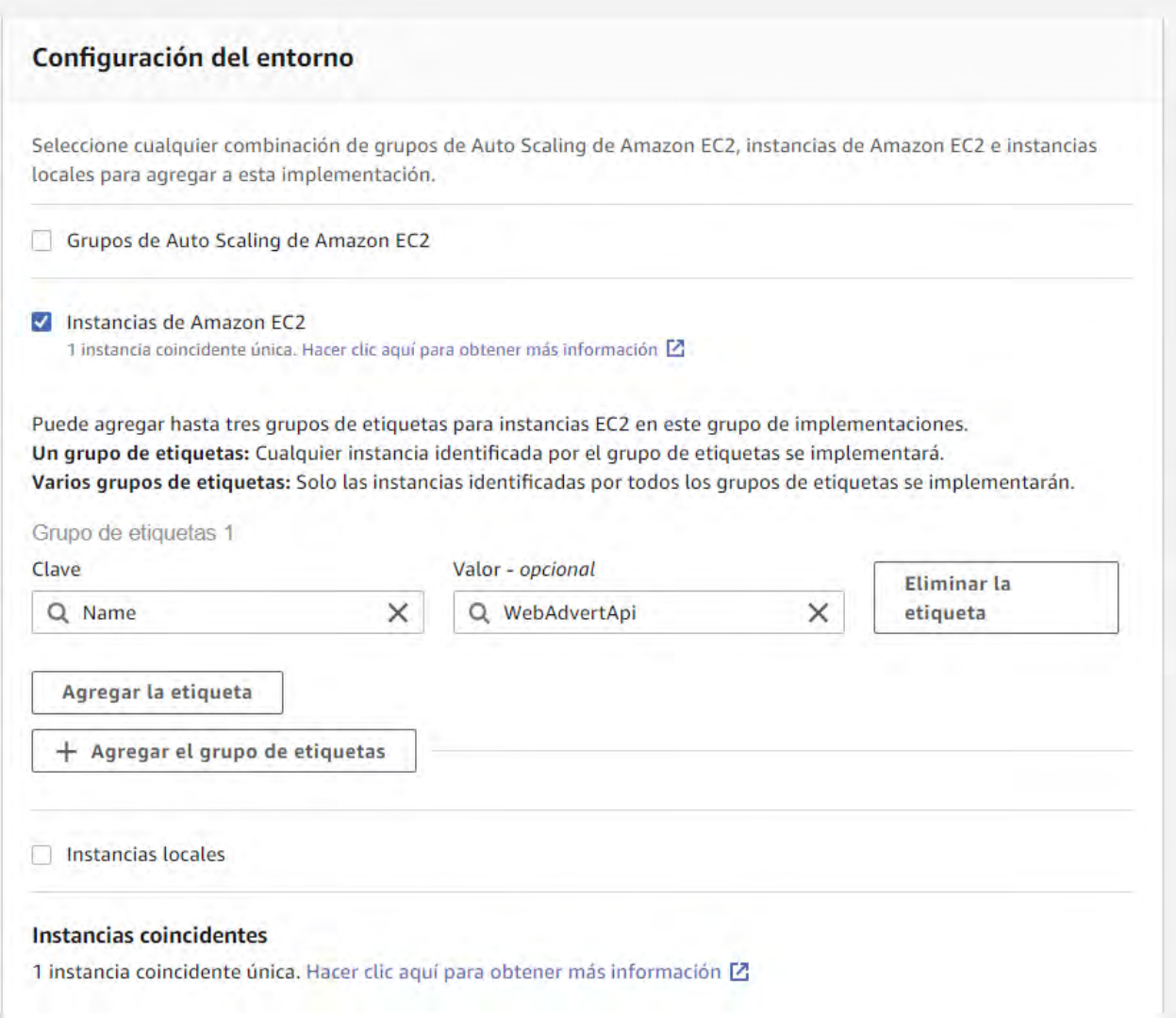

Figura 97: Página para configurar el entorno en CodeDeploy

En cuanto a la sección de configuración de la implementación AWS pone a disposición tres opciones. La primera opción es desplegar solamente una aplicación al mismo tiempo. La segunda opción es desplegar la mitad de las aplicaciones al mismo tiempo. La tercera opción es desplegar todas las aplicaciones al mismo tiempo. Dado que solo se está implementando un microservicio se decidió utilizar la tercera opción. En lo que respecta al balanceador de carga no se habilitó ya que solo usará una instancia EC2 micro en el caso de aplicación.

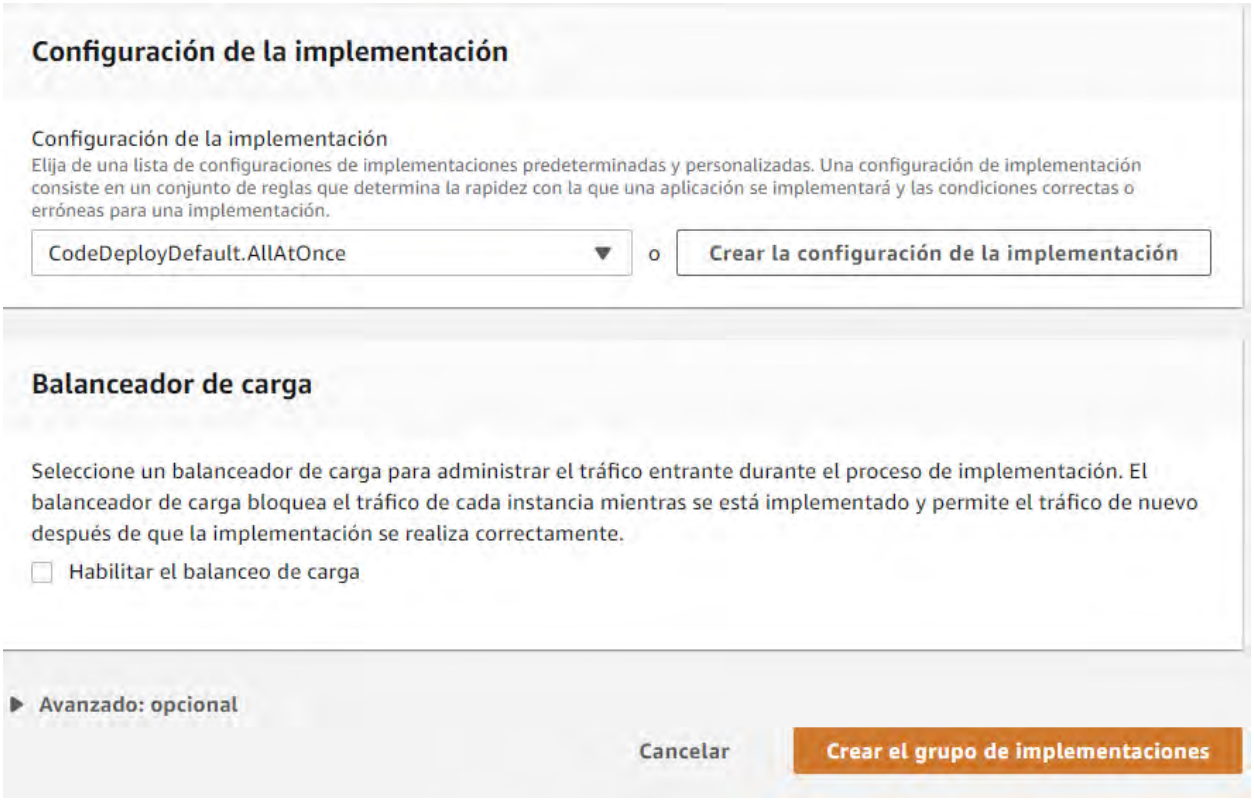

Figura 98: Pagina para configurar la implementación en CodeDeploy

Para concluir con la configuración de Code Deploy AWS muestra el mensaje de la figura 99 indicando que se ha creado el grupo de implementación correctamente. Después de lo cual es necesario empaquetar la aplicación para subirla a la cubeta S3 y así probar el funcionamiento de los despliegues de la aplicación con Code Deploy.

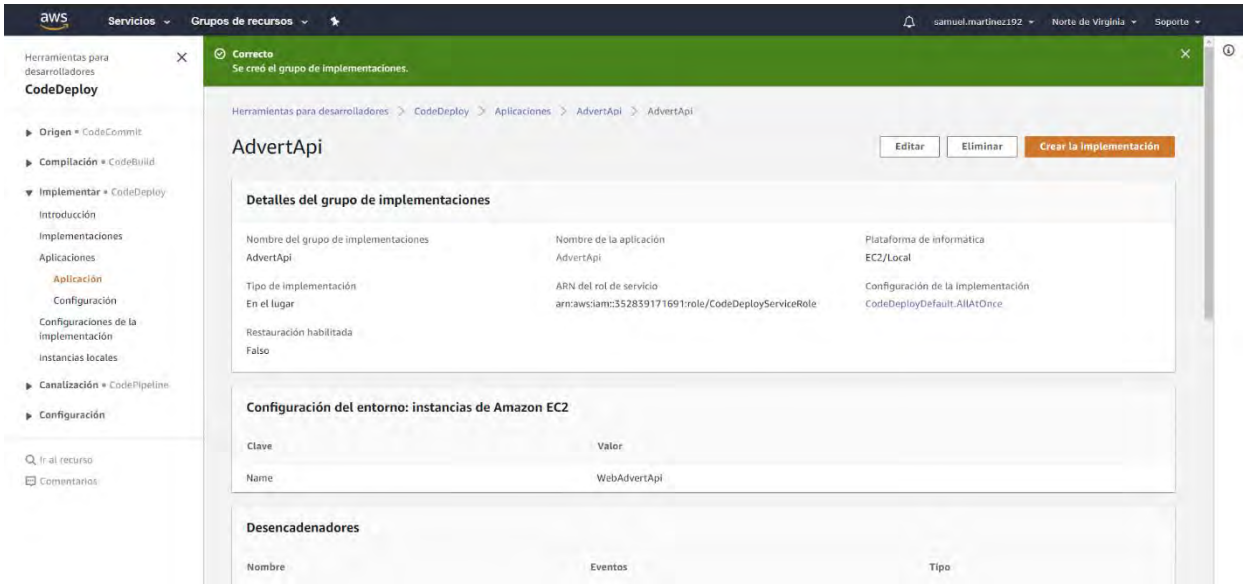

Figura 99: Página que notifica la creación del grupo de implementación

Ahora bien, para empezar el proceso del empaquetado del microservicio es necesario abrir una terminal en donde se encuentra el código. Después de ello se debe comprobar que compile para iniciar con el empaquetado y subirlo a la cubeta S3.

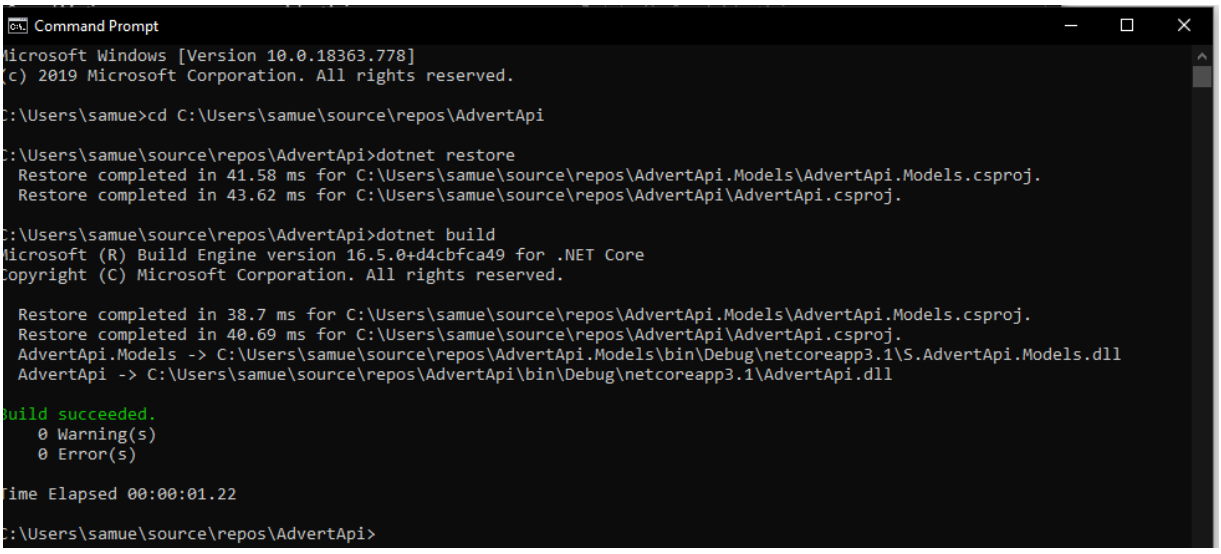

Figura 100: Compilando el microservicio “AdvertApi” desde la terminal

Tras verificar que el microservicio compile correctamente se debe publicar el microservicio en algún directorio que el usuario crea conveniente, tal como se muestra en la figura 101. La publicación del microservicio generará los ensambles necesarios para empaquetarlos y subirlos a la cubeta S3.

```
C:\Users\samue\source\repos\AdvertApi>dotnet publish -o c:\BuildOutPut\AdvertApi
Microsoft (R) Build Engine version 16.5.0+d4cbfca49 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 35.32 ms for C:\Users\samue\source\repos\AdvertApi.Models\AdvertApi.Models.csproj.
  Restore completed in 37.14 ms for C:\Users\samue\source\repos\AdvertApi\AdvertApi.csproj.
  AdvertApi.Models -> C:\Users\samue\source\repos\AdvertApi.Models\bin\Debug\netcoreapp3.1\S.AdvertApi.Models.dll
  AdvertApi.Models -> c:\BuildOutPut\AdvertApi\
  AdvertApi -> C:\Users\samue\source\repos\AdvertApi\bin\Debug\netcoreapp3.1\AdvertApi.dll
  AdvertApi -> c:\BuildOutPut\AdvertApi\

C:\Users\samue\source\repos\AdvertApi>
```

Figura 101: Publicando el microservicio en un directorio especifico

Después de generar los ensambles necesarios se requiere crear un archivo especial con el siguiente nombre y extensión “AppSpec.yml”. Este archivo debe contener el sistema operativo para el cual se realizará la publicación de la aplicación, así como el origen de los ensambles y el destino en donde se van a publicar.

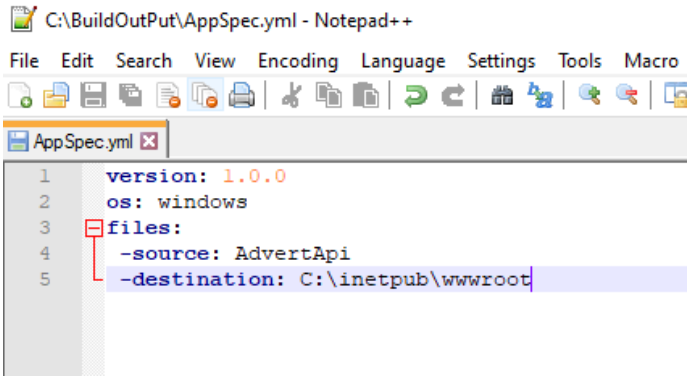

Figura 102: Archivo de configuración utilizado por CodeDeploy

Finalmente, se crea un archivo zip que contiene el archivo AppSpec y la carpeta con los ensambles del microservicio para subirlo a la cubeta S3, tal como se muestra en la figura 103.

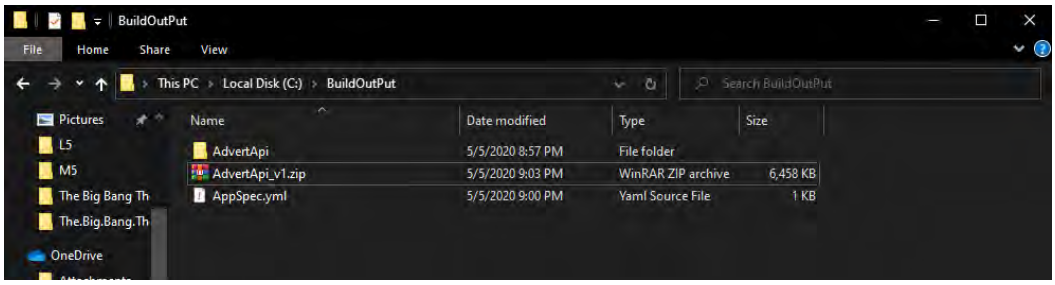

Figura 103: Directorio con microservicio y archivo de configuración comprimido

En este apartado de la configuración de Code Deploy se muestra como subir el microservicio empaquetado a una cubeta S3 para que sea posteriormente leído por Code Deploy y se pueda hacer los despliegues de la aplicación en la instancia EC2. Para comenzar se entra al servicio de S3 desde la consola de administración y al enlace de cubetas como se muestra en la figura 104. Después de ello se entra a la cubeta "web-advert-code-deploy".

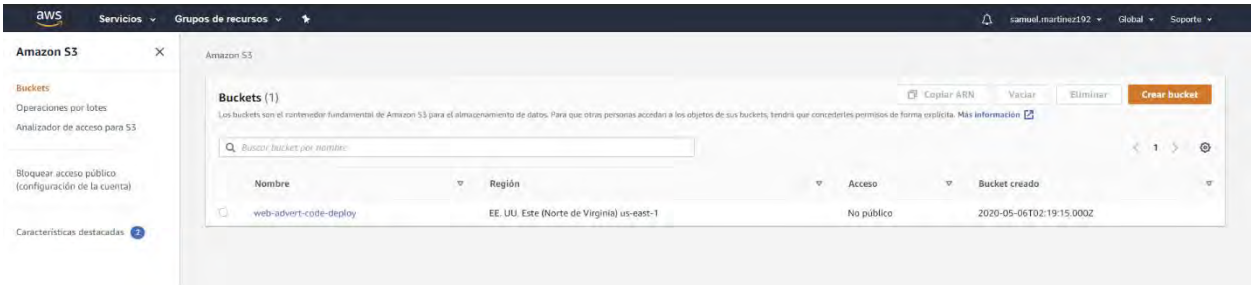

Figura 104: Página de cubetas en S3

Ya dentro de la cubeta "web-advert-code-deploy" se mostrará la página de la figura 105. En esta página se pueden cargar uno o más archivos mediante el botón de añadir archivos. Para el caso de aplicación solo se cargará el microservicio "AdvertApi".

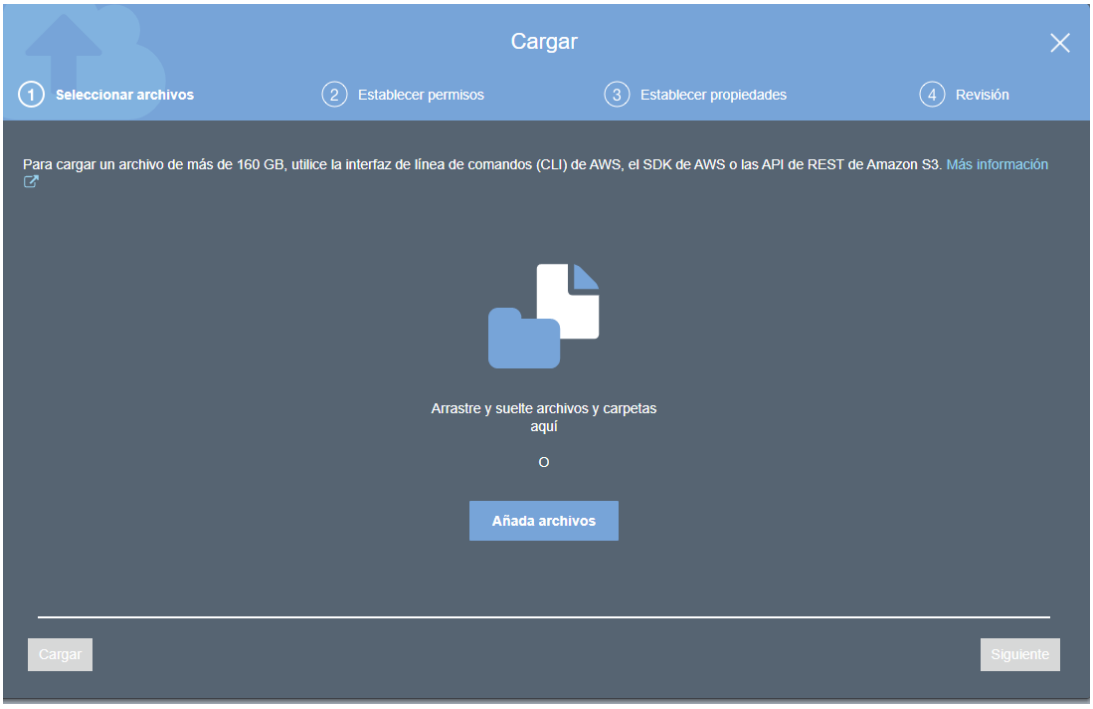

Figura 105: Página para cargar archivos en cubeta S3.

Después de cargar el microservicio a la cubeta S3, se presenta información relevante al archivo como su formato y peso. En la figura 106 se ve el archivo “AdvertApi_v1” con formato zip y un peso de 6.3 MB.

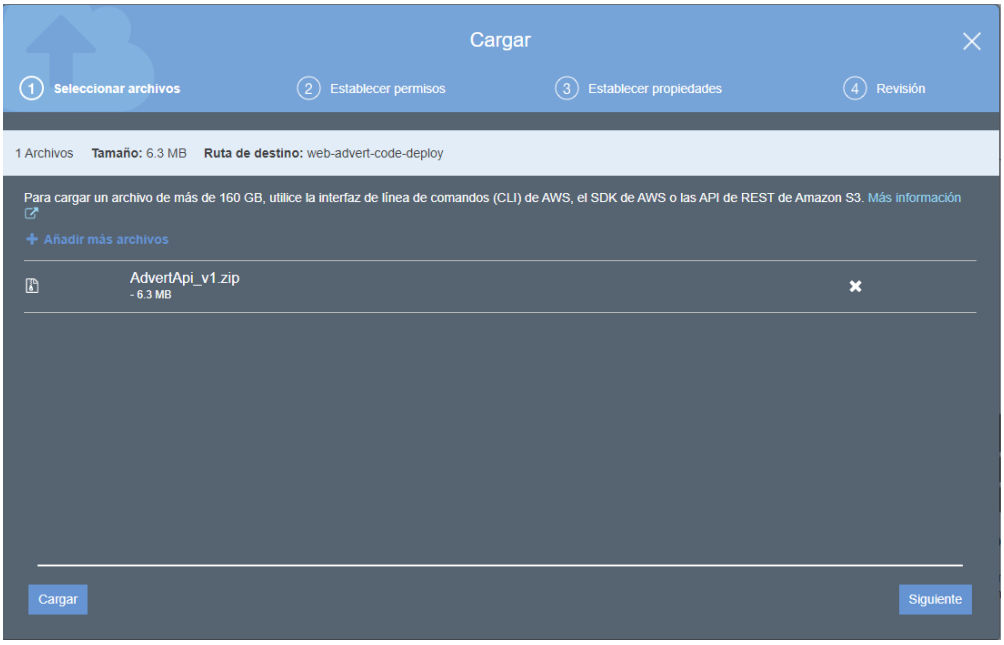

Figura 106: Página con archivo cargado en cubeta S3

En la siguiente página se necesita delimitar el nivel permisos que tienen los usuarios sobre el archivo que se va a subir. En la figura 107 se muestra que se asignaron permisos de escritura y lectura en los objetos de la cubeta S3 a samuel.martinez192.

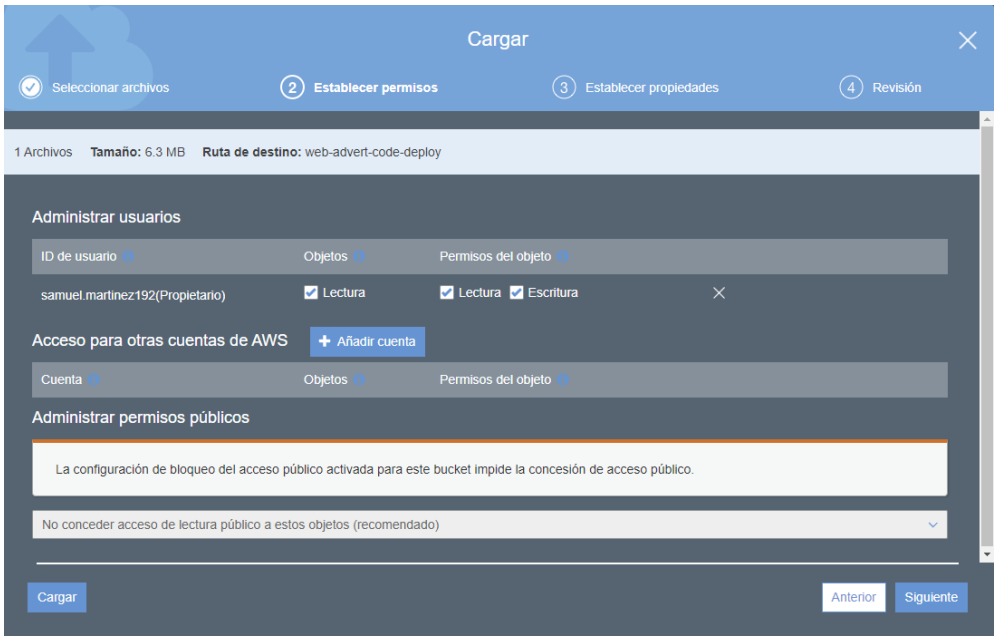

Figura 107: Permisos de usuarios en los objetos de la cubeta S3.

En esta página se configura que tipo de almacenamiento se usará en la cubeta S3. Dado que se tiene una pequeña aplicación se optó por seleccionar la opción estándar como se muestra en la figura 108.

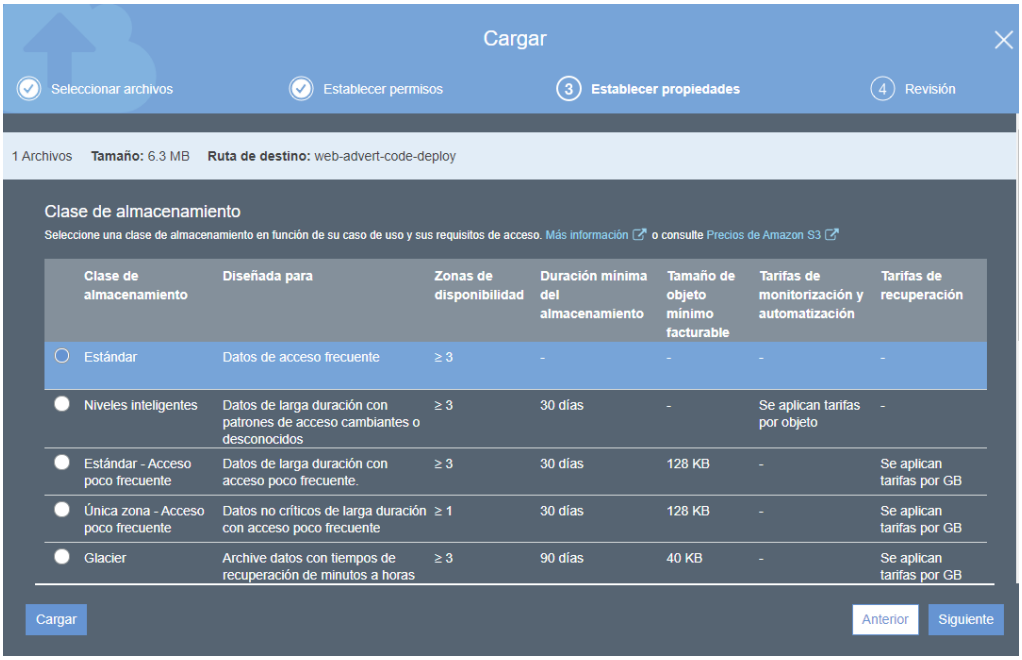

Figura 108: Página para configurar tipo de almacenamiento en S3

En la siguiente página se muestra un resumen de los parámetros de configuración que se seleccionaron en los pasos anteriores. Después de revisar que todo está en orden se sube el archivo a la cubeta S3 dando clic al botón de cargar.

Figura 109: Página con resumen de la configuración en S3

Posterior a que se haya cargado el archivo en la cubeta S3 se puede ver la imagen de la figura 110 en la que se muestra los archivos existentes en la cubeta de S3. Es necesario tener el microservicio empaquetado en la cubeta S3 para probar el funcionamiento de Code Deploy y terminar con la configuración de esta sección.

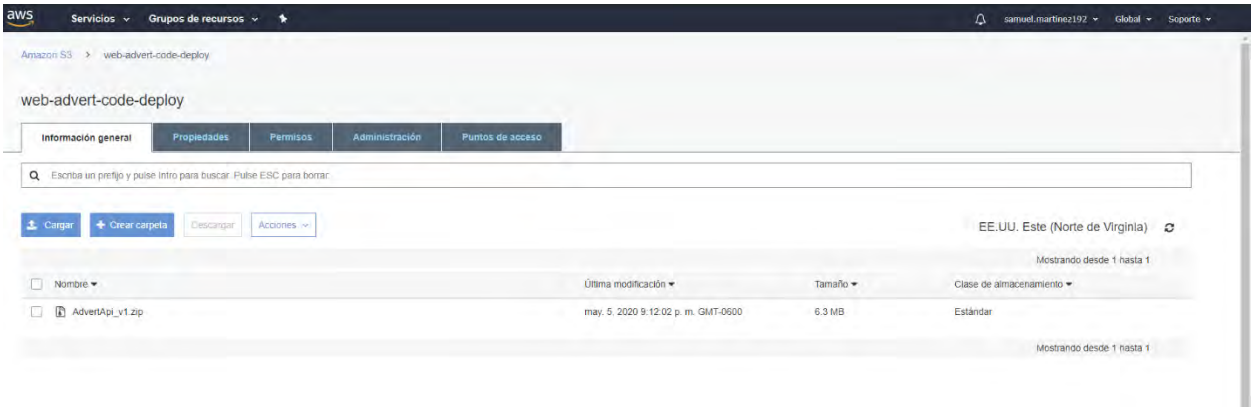

Figura 110: Página para ver los archivos existentes en la cubeta S3

El siguiente punto para concluir la configuración de esta sección es realizar pruebas del funcionamiento de Code Deploy. Ahora bien, para probar que Code Deploy esté funcionando correctamente se entra a la sección de grupo de implementación y se da clic al botón de crear implementación como se ve en la figura 111.

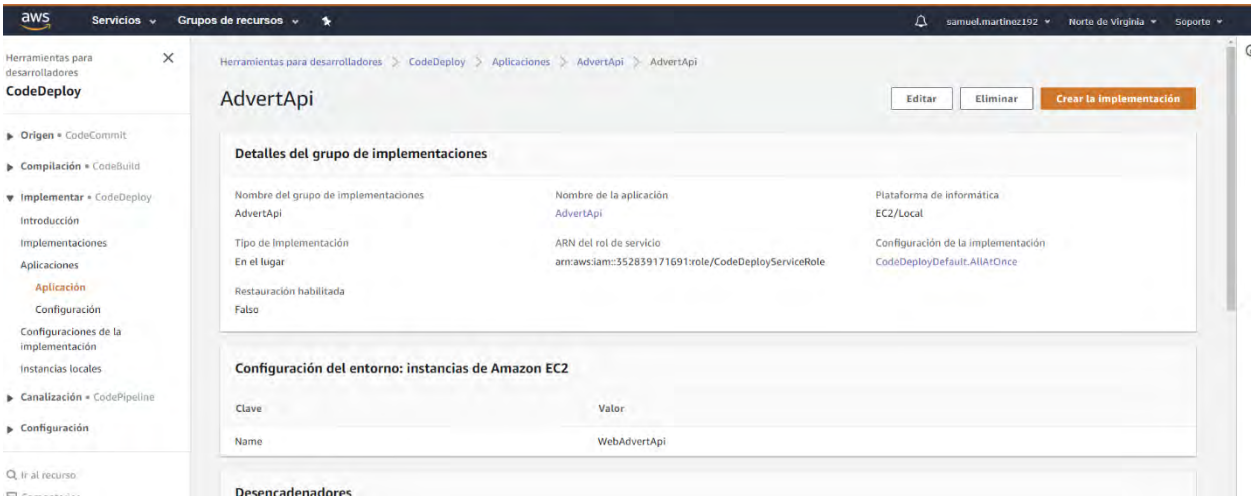

Figura 111: Página para crear implementaciones en CodeDeploy

Ya que se entró al grupo de implementación se muestra información de la aplicación, la plataforma informática y el tipo de implementación como se ve en la figura 112. En esta página se necesita seleccionar el tipo de revisión, que básicamente es indicar a Code Deploy en donde se encuentra almacenada la aplicación que se va a implementar. Dado que en el punto anterior se configuro la cubeta de S3, se seleccionó esta opción para indicar que la aplicación esta almacenada en este servicio. Después se ingresó la URL en donde se encuentra el archivo y se creó el despliegue para la instancia EC2.

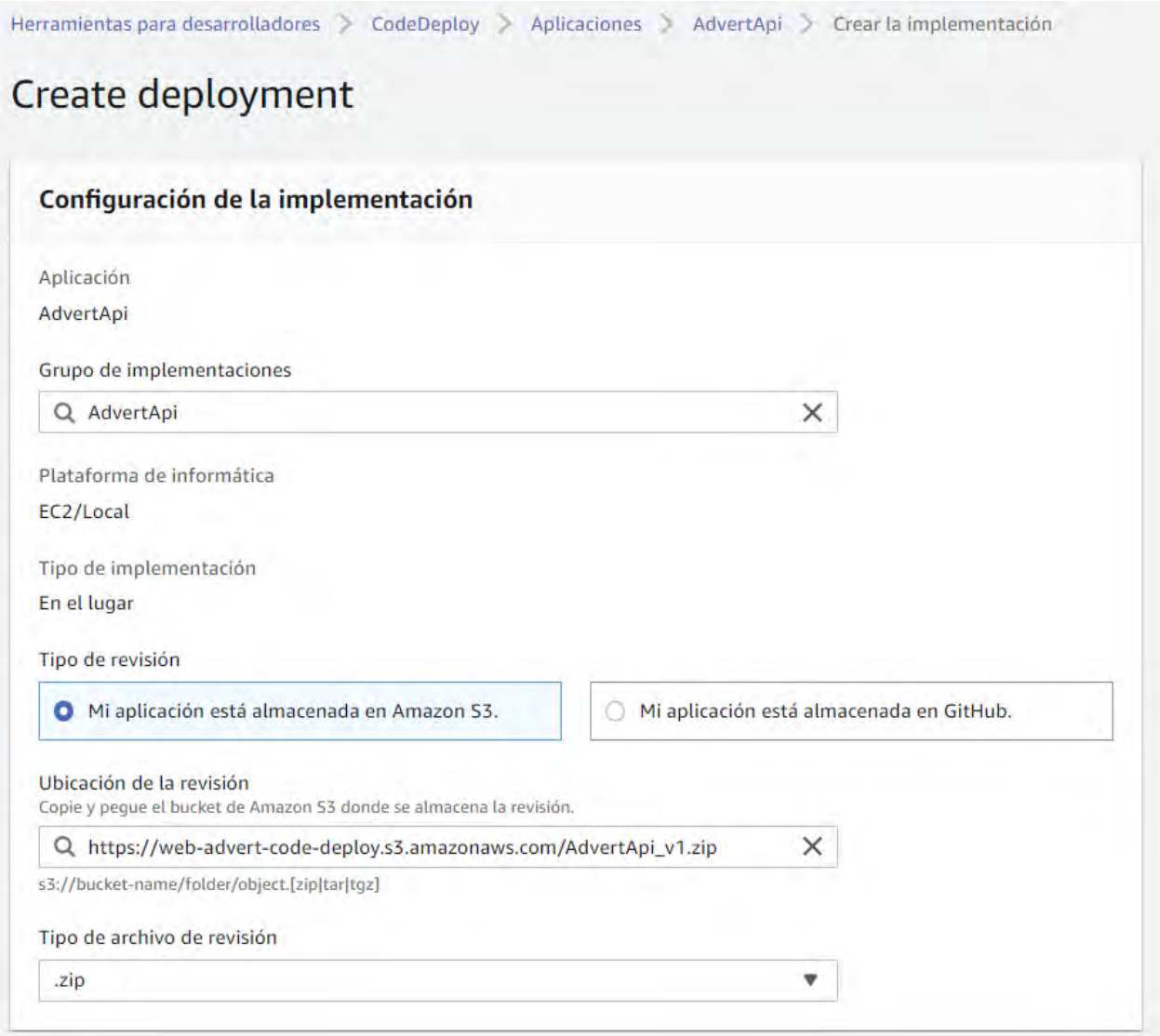

Figura 112: Página para crear una implementación en CodeDeploy

Si todos los pasos y configuraciones se realizaron de manera correcta AWS mostrara un mensaje indicando que se ha creado una nueva implementación en la plataforma informática de destino, tal como se ve en la figura 113.

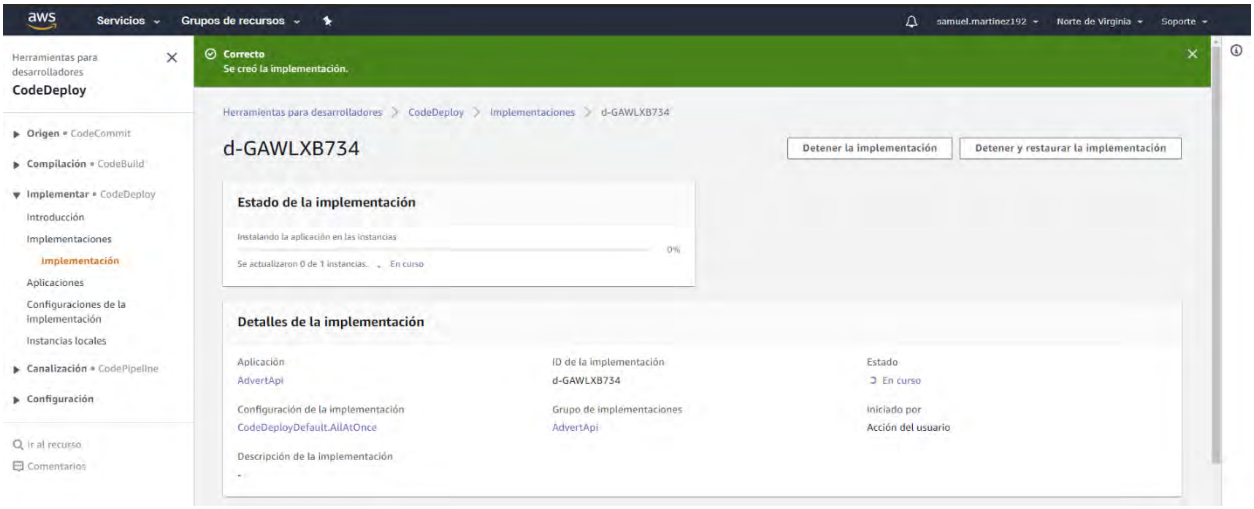

Figura 113: Página que notifica la creación de la implementación en CodeDeploy

Para concluir, se verificará que el microservicio se haya implementado correctamente en la instancia EC2. Para ello se ingresará a un navegador web la URL del DNS público de la instancia y se llamará a una función del microservicio como se presenta en la figura 114. Como se puede ver la llamada a la función para verificar el estado del microservicio regresa un estatus de que se encuentra operativo, por lo cual, se concluye que se realizó una configuración adecuada y se continua con la siguiente sección.

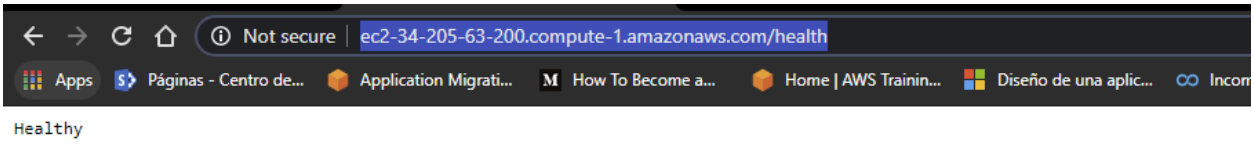

Figura 114: Verificando la implementación del microservicio en EC2

Anexo VIII: Configuración de SNS para Lambda

La octava sección de la implementación del caso de aplicación explica el proceso a seguir para la configuración de Amazon SNS. Este servicio se utilizará para mandar los mensajes de publicación/suscripción entre un microservicio y una aplicación sin servidor (AWS Lambda) del caso de aplicación. Para comenzar con la configuración de este servicio se entra a la consola de administración de AWS y se busca el servicio de SNS. Dentro del servicio SNS se entra a la sección de temas y se crea un nuevo tema con el nombre de AdvertApi como se muestra en la figura 115.

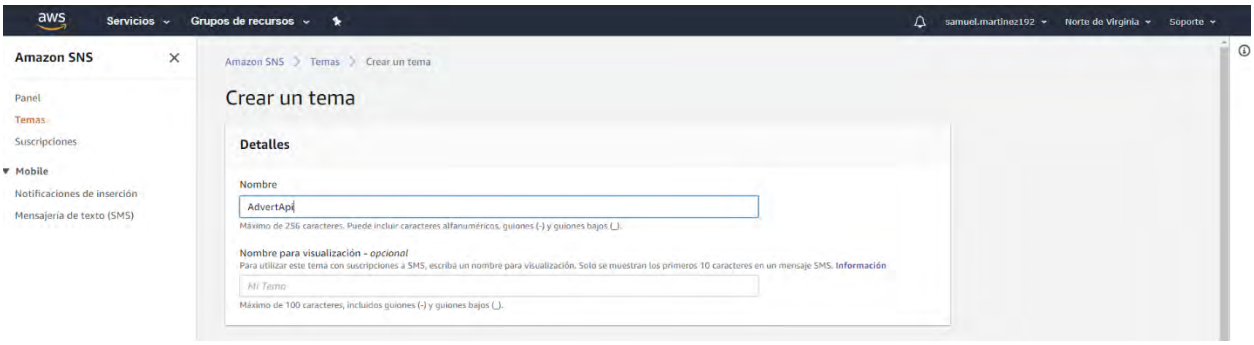

Figura 115: Página para crear temas en SNS

Listo, después de crear el tema ya se puede realizar mensajería de publicación/suscripción entre los diferentes servicios del caso de aplicación. Como último paso es necesario guardar el ARN que se muestra en la página de la figura 116 ya que se utilizará para hacer llamadas al servicio SNS desde las Web APIs.

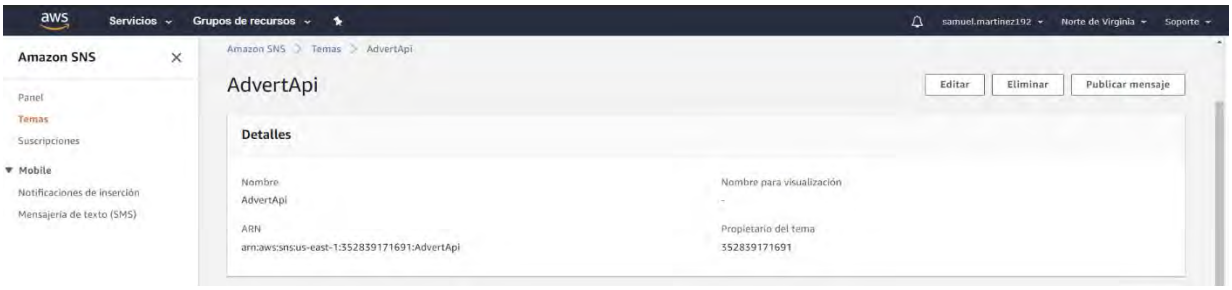

Figura 116: Página para editar temas en SNS

Anexo IX: Configuración de ElasticSearch para Lambda

La novena sección en la implementación del caso de aplicación detalla el proceso a seguir para la configuración de Amazon ElasticSearch Service. Este servicio sirve para realizar análisis de todo tipo de datos. Por ejemplo, datos textuales, números geoespaciales, estructurados, no estructurados. En el caso de aplicación se utilizará para realizar análisis de los anuncios que publiquen los usuarios en el sistema. Para comenzar su configuración se entra a la consola de administración de AWS y se busca el servicio Elasticsearch. Después de entrar al servicio se muestra la página de la figura 117. En esta página se usó la opción de crear un nuevo dominio.

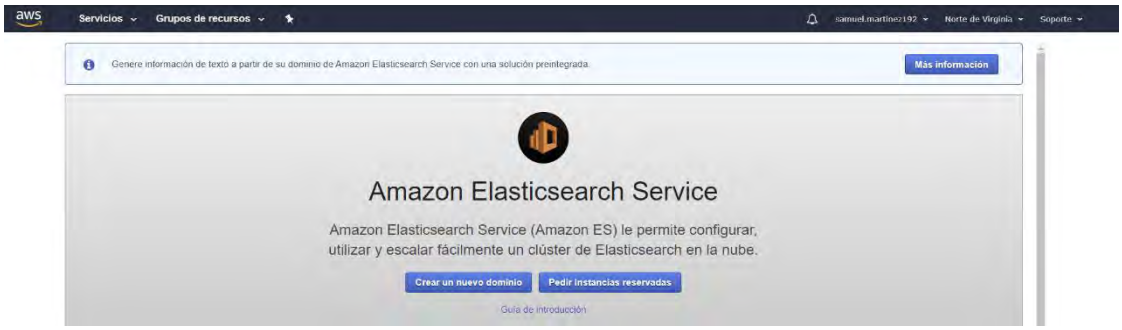

Figura 117: Página de inicio Elasticsearch

La primera página que se muestra para crear un dominio se debe elegir qué tipo de implementación se va a realizar y la versión de ElasticSearch. En esta página se seleccionó una implementación de desarrollo/pruebas y la versión 7.4 como se muestra en la figura 118.

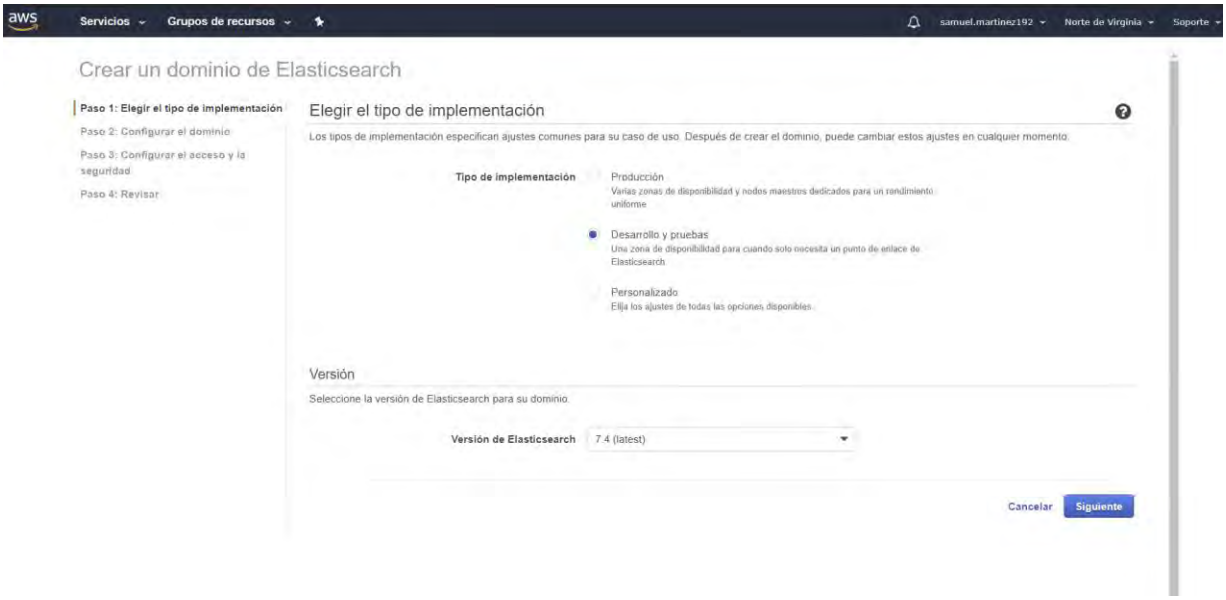

Figura 118: Página para seleccionar tipo de implementación en Elasticsearch

En la segunda página se asignó un nombre de dominio, tipo de instancia y el número de nodos del dominio. En la figura 119 se muestra que se asignó el nombre de advertapi para el dominio y una instancia t2.small que es ideal para entornos de pruebas y desarrollo.

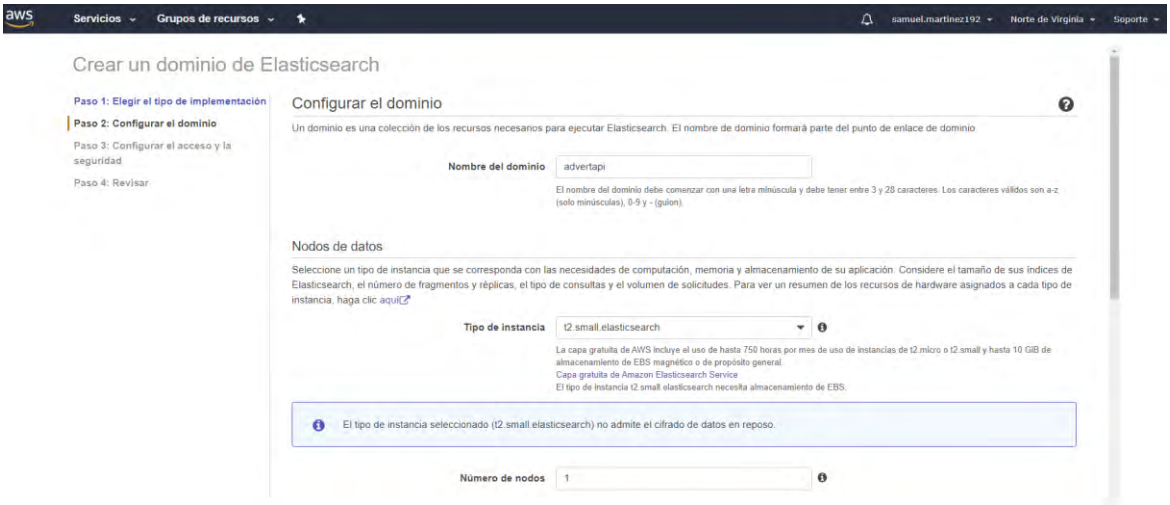

Figura 119: Página para configurar tipo de implementación en Elasticsearch

En la tercera página se seleccionó el tipo de almacenamiento de los nodos, así como los nodos dedicados. Dado que es un dominio para pruebas se decidió seleccionar el almacenamiento sencillo y no habilitar la opción de nodos dedicados.

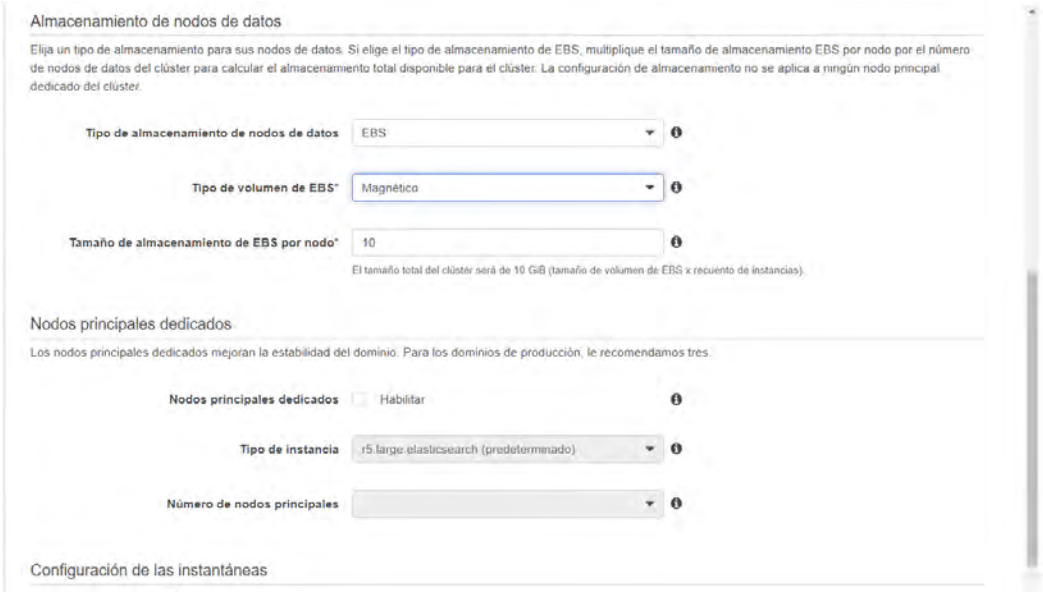

Figura 120: Página para configurar tipo de implementación en Elasticsearch

En la cuarta página se realizan configuraciones con respecto al acceso y seguridad del dominio. Dado que se necesita ingresar a los anuncios almacenados en ElasticSearch desde otras WebAPIs se seleccionó la opción de acceso público como se muestra en la figura 121.

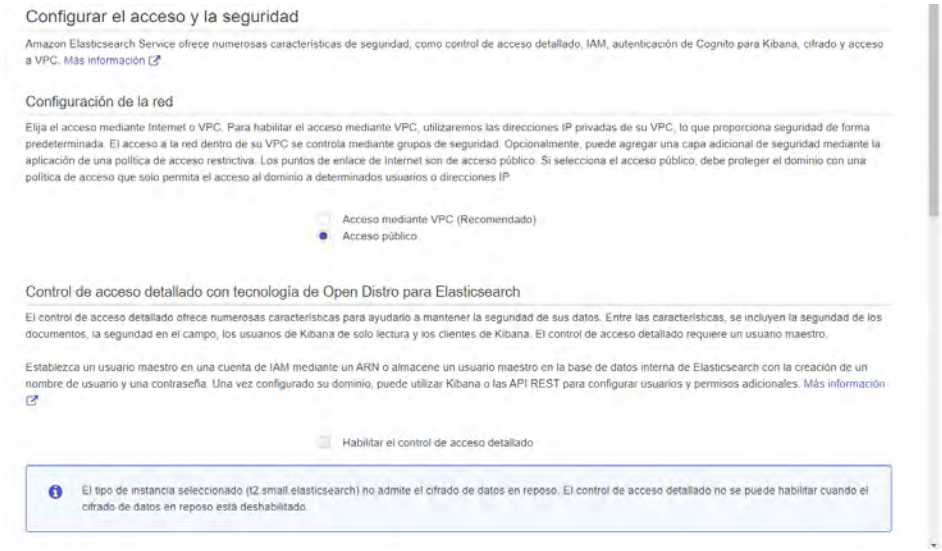

Figura 121: Pagina para configurar acceso y seguridad en Elasticsearch

Para finalizar AWS muestra el mensaje de la figura 122 que indica que se ha creado correctamente un nuevo domino en ElasticSearch, con lo cual ya se puede usar el servicio para guardar datos y realizar analíticas de los datos guardados.

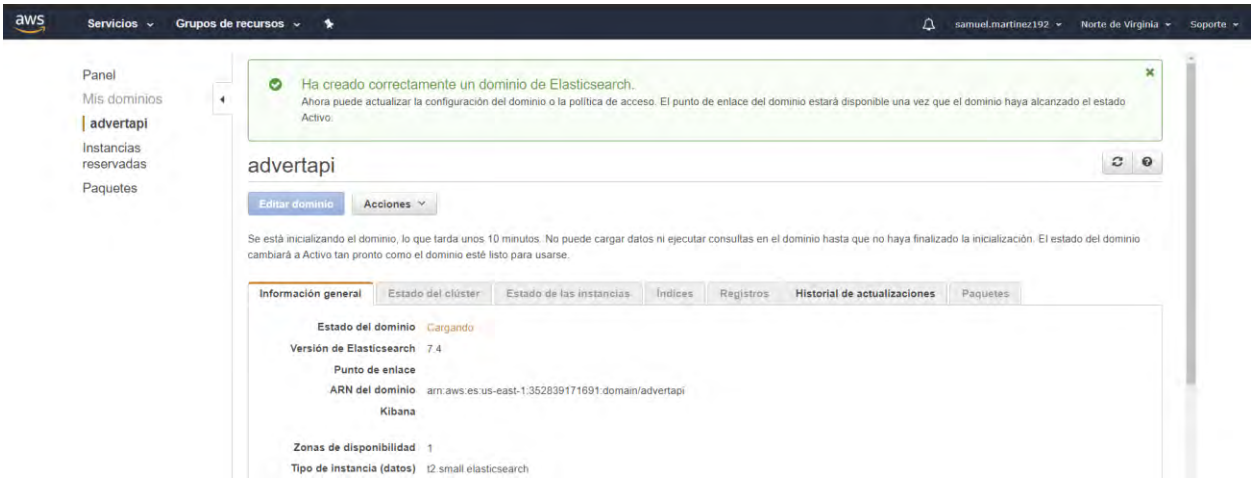

Figura 122: Página que notifica la creación del dominio en Elasticsearch

Anexo X: Configuración de rol para Lambmda

En la décima sección de la implementación del caso de aplicación se describe el proceso de configuración para el rol de cloudwatch para Lambda. No obstante, antes de empezar con la descripción es importante ver que función desempeñan los roles en AWS. En este sentido, se encargan de delimitar a que otros servicios pueden acceder la entidad que se esté configurando. Por ejemplo, permitir que Lambda acceda al servicio de cloudwatch, con la finalidad de monitorear el funcionamiento y operación de las Lambda que estén en funcionamiento. Ahora bien, para iniciar con el proceso de creación del rol se requiere entrar a la consola de administración de AWS y buscar el servicio de IAM, seguidamente, se entra al enlace de roles y se da clic al botón azul de crear un rol.

Figura 123: Página de inicio roles

El siguiente paso en la creación del rol es seleccionar el tipo de entidad y elegir el tipo de caso de uso. Dado que se quiere configurar un rol para AWS Lambda, se debe seleccionar la entidad de servicio de AWS y un caso de uso Lambda como se muestra en la figura 124.

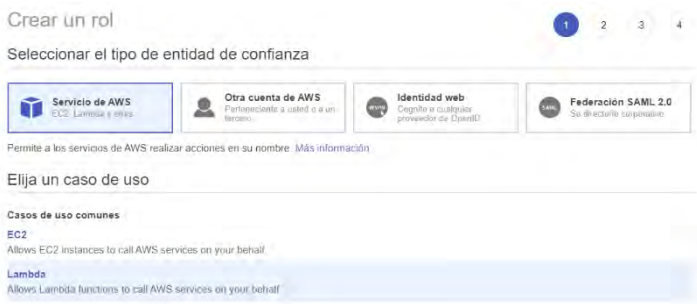

Figura 124: Paso 1 para crear roles en AWS

El segundo paso es seleccionar las políticas de permisos o bien crear una nueva política. Como se mencionó al inicio se requiere que el servicio de Lambda pueda acceder a cloudwatch para su monitoreo y métricas de uso. Por lo tanto, se seleccionó la política de CloudWatchLogsFullAccess como se muestra en la figura.

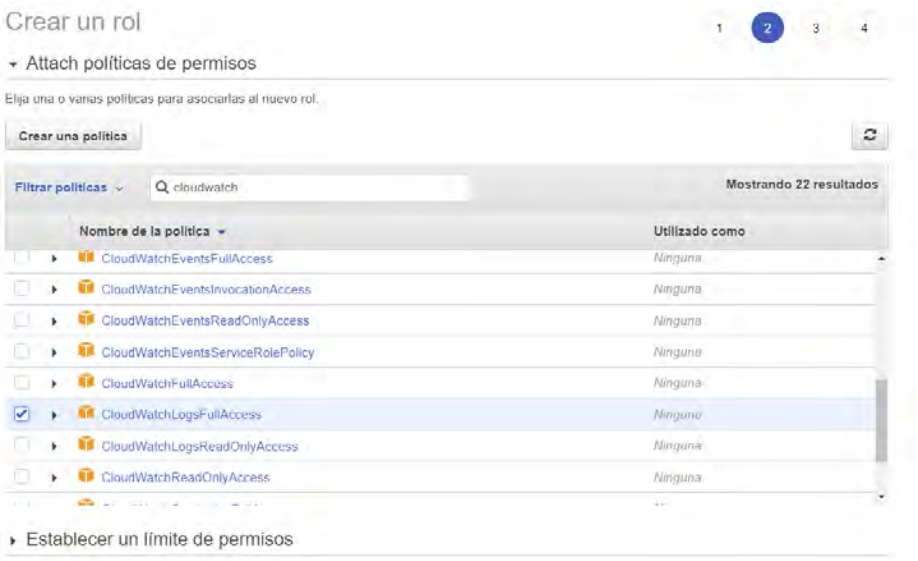

Figura 125: Paso 2 para crear roles en AWS

El tercer paso es opcional y consiste en añadir una etiqueta al rol que se está creando. Se recomienda añadir la etiqueta para facilitar su identificación al momento de utilizarlo en la configuración del servicio Lambda. En este caso se optó por asignar un nombre como valor en la clave y un valor de SearchWorkerRole para su identificación.

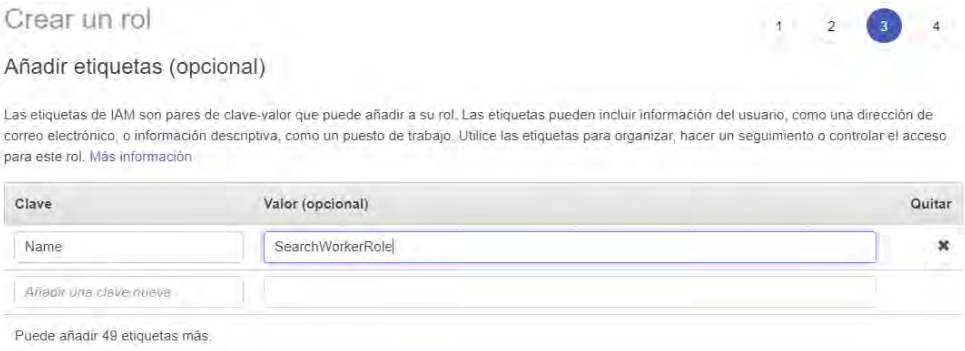

Figura 126: Paso 3 para crear roles en AWS

Anexo XI: Configuración de Lambda

La décima primera sección en la implementación del caso de aplicación explica el proceso de configuración del servicio Lambda. Se utilizará el servicio para ejecutar una función que guarda los anuncios del caso de aplicación en el servicio ElasticSearch para posteriormente realizar consultas y análisis de anuncios. Para comenzar con su configuración se entra a la consola de administración de AWS y se busca el servicio de Lambda. Posteriormente, se entra a la sección de funciones y se da clic al botón de crear función como se ve en la figura 127.

Figura 127: Página de funciones en Lambda

En la primera página del asistente de configuración que se muestra en la figura 128 se necesita asignar un nombre a la función, así como el lenguaje en el que está desarrollada la función. Dado que la función se desarrolló utilizado .Net Core 3.1 se selecciona este lenguaje en el asistente.

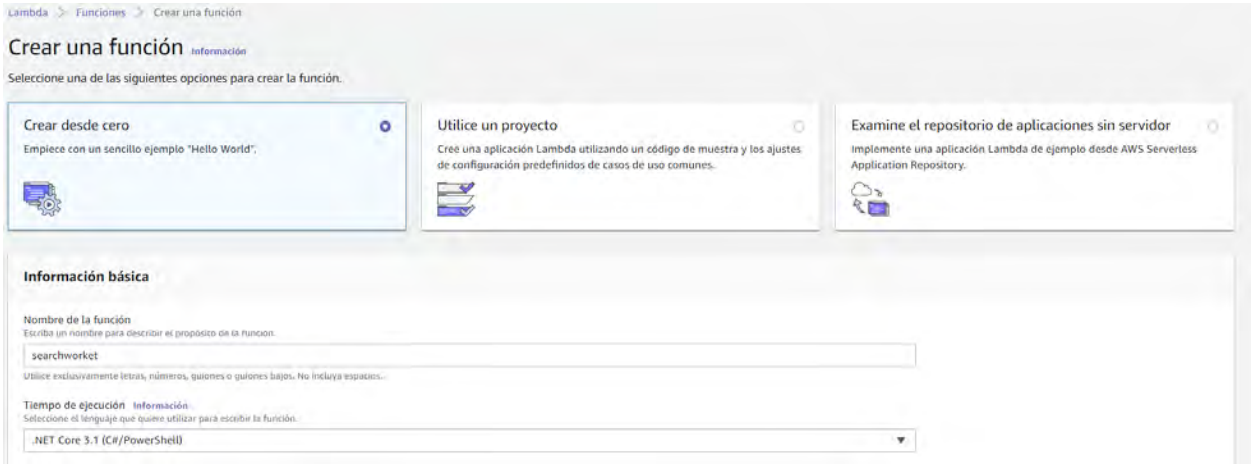

Figura 128: Página para crear funciones en Lambda

El siguiente punto que se necesita configurar en la primera página son la asignación de roles que permitan al servicio de Lambda acceder a otros servicios en caso de ser necesarios. Para este punto se asignó el rol creado con anterioridad el cual permite que Lambda acceda a AWS Cloudwatch para obtener métricas del funcionamiento de Lambda. Después de que se asignó el rol se procede con la creación de la función.

Figura 129: Página para crear funciones en Lambda

Si se realizó una configuración adecuada se mostrará un mensaje indicando que se ha creado correctamente una función como se ve en la figura 130. A partir de este momento se puede editar la función Lambda para añadirle el código de la función y desencadenadores.

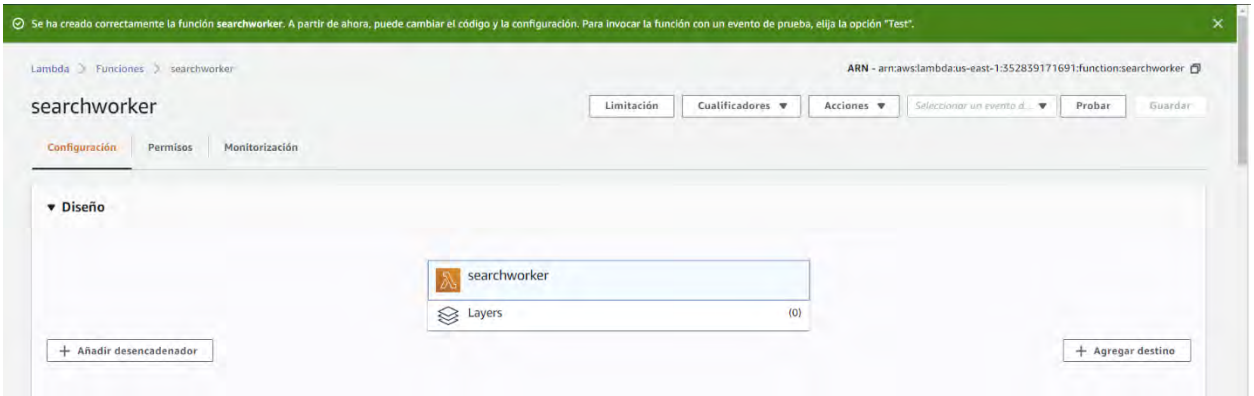

Figura 130: Página para editar funciones en Lambda

El primer atributo que se va a editar de la función Lambda son los desencadenadores como se ve en la figura 131. Los desencadenadores son acciones que ocurren durante la ejecución de la función que esta almacenada en el servicio Lambda. Para el caso de aplicación se configuró un evento del tipo SNS y se seleccionó el tema de SNS que se creó en las secciones anteriores. Para concluir con la configuración de este atributo se marcó la casilla de activar desencadenador y se dio clic al botón de agregar.

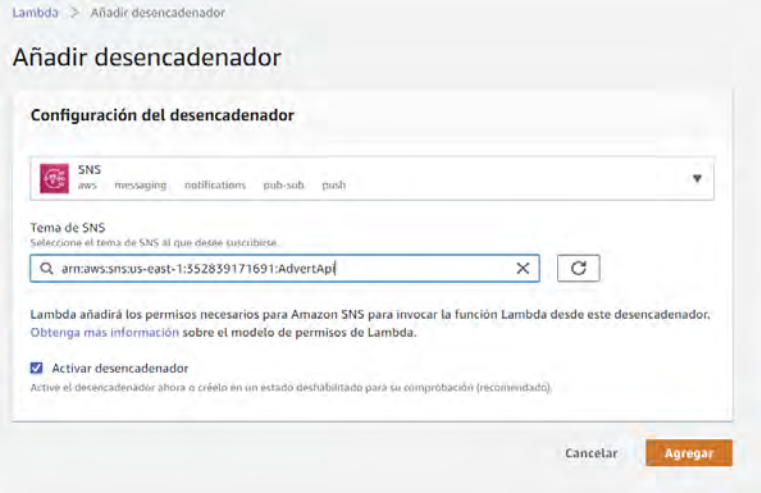

Figura 131: Página para añadir desencadenadores en Lambda

El segundo atributo que se va a editar es el código de la función Lambda. Para comenzar se requiere estar en el directorio en donde se encuentra la función y abrir una terminal en este directorio para empaquetar la función y subirla a Lambda.

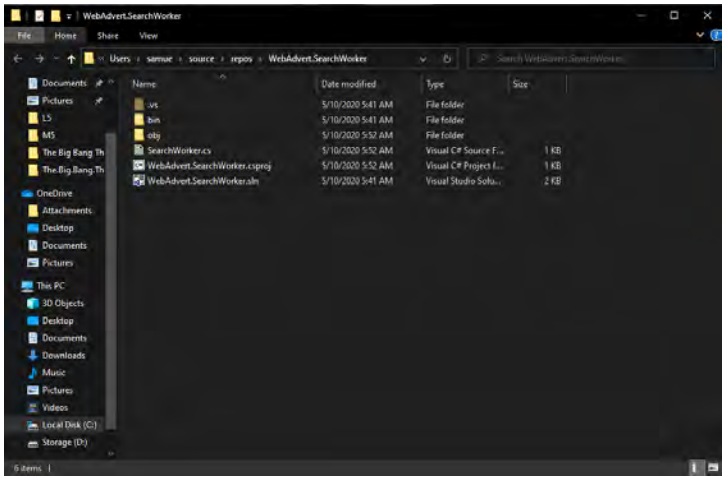

Figura 132: Directorio en donde se encuentra la función desarrollada.

Después de abrir la terminal se requiere ingresar un comando especial que permite empaquetar la función y con ello se pueda ejecutar en Lambda. Este comando recibe los parámetros de salida, nombre de paquete y la versión del framework como se ve en la figura 133. El comando que se ingreso es el siguiente "dotnet lambda package -c release -o SearchWorker.zip –framework netcoreapp3.1".

```
C:\Users\samue\source\repos\WebAdvert.SearchWorker>dotnet lambda package -c release -o SearchWorker.zip --framework netcoreapp3.1
Amazon Lambda Tools for .NET Core applications (4.0.0)
Project Home: https://github.com/aws/aws-extensions-for-dotnet-cli, https://github.com/aws/aws-lambda-dotnet

Executing publish command
... invoking 'dotnet publish', working folder 'C:\Users\samue\source\repos\WebAdvert.SearchWorker\bin\release\netcoreapp3.1\publish'
... publish: Microsoft (R) Build Engine version 16.5.0+d4cbfca49 for .NET Core
... publish: Copyright (C) Microsoft Corporation. All rights reserved.
... publish:   Restore completed in 2.81 sec for C:\Users\samue\source\repos\WebAdvert.SearchWorker\WebAdvert.SearchWorker.csproj.
... publish:   WebAdvert.SearchWorker -> C:\Users\samue\source\repos\WebAdvert.SearchWorker\bin\Release\netcoreapp3.1\linux-x64\WebAdvert.SearchWorker.dll
... publish:   WebAdvert.SearchWorker -> C:\Users\samue\source\repos\WebAdvert.SearchWorker\bin\release\netcoreapp3.1\publish\
Zipping publish folder C:\Users\samue\source\repos\WebAdvert.SearchWorker\bin\release\netcoreapp3.1\publish to C:\Users\samue\source\repos\WebAdvert.SearchWorker\SearchWorker.zip
... zipping: Amazon.Lambda.Core.dll
... zipping: Amazon.Lambda.Serialization.Json.dll
... zipping: Amazon.Lambda.SNSEvents.dll
... zipping: Newtonsoft.Json.dll
... zipping: WebAdvert.SearchWorker.deps.json
... zipping: WebAdvert.SearchWorker.dll
... zipping: WebAdvert.SearchWorker.pdb
... zipping: WebAdvert.SearchWorker.runtimeconfig.json
Created publish archive (C:\Users\samue\source\repos\WebAdvert.SearchWorker\SearchWorker.zip).
Lambda project successfully packaged: C:\Users\samue\source\repos\WebAdvert.SearchWorker\SearchWorker.zip

C:\Users\samue\source\repos\WebAdvert.SearchWorker>
```

Figura 133: Empaquetando la función para subirla a Lambda

El siguiente paso en la configuración del servicio Lambda es subir la aplicación empaquetada, seleccionar el tiempo de ejecución y especificar la ruta del controlador de la función Lambda. Para el tiempo de ejecución se seleccionó el lenguaje .NET Core 3.1 ya que el caso de aplicación se desarrolló usando este lenguaje. En el caso de la ruta del controlador se debe especificar con los siguientes parámetros: "nombre del ensamble\espacio de nombre\nombre de la clase\nombre de la función". Después de asignar estos valores se da clic al botón guardar y se continua con la siguiente sección.

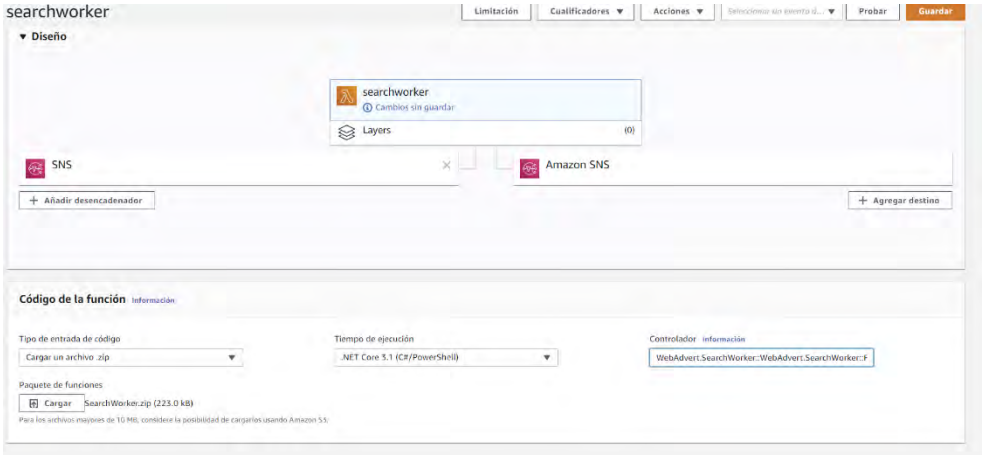

Figura 134: Página para editar funciones en Lambda

Anexo XII: Pruebas de SNS a Lambda

La décima segunda sección en la implementación del caso de aplicación describe la prueba de funcionamiento a la función Lambda. Recapitulando la sección anterior se configuro un desencadenador del tipo SNS por lo cual la prueba que se realiza es en función de este servicio. Para empezar con la prueba se entra a la consola de administración y se busca al servicio SNS. Después de acceder al servicio SNS se entra a la sección de temas como se muestra en la figura 135.

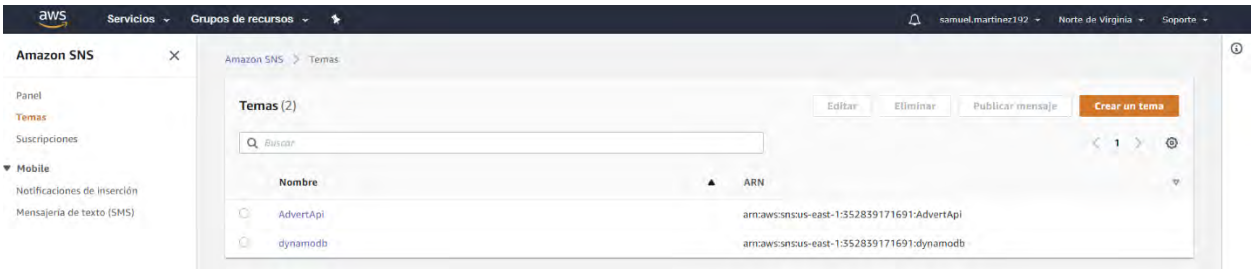

Figura 135: Página de temas en SNS

Dentro de la página de temas se selecciona el tema AdvertApi para realizar la prueba de funcionamiento del envío de mensajes en el servicio Lambda. En esta página se da clic al botón publicar mensaje el cual mostrara la página para publicar mensajes.

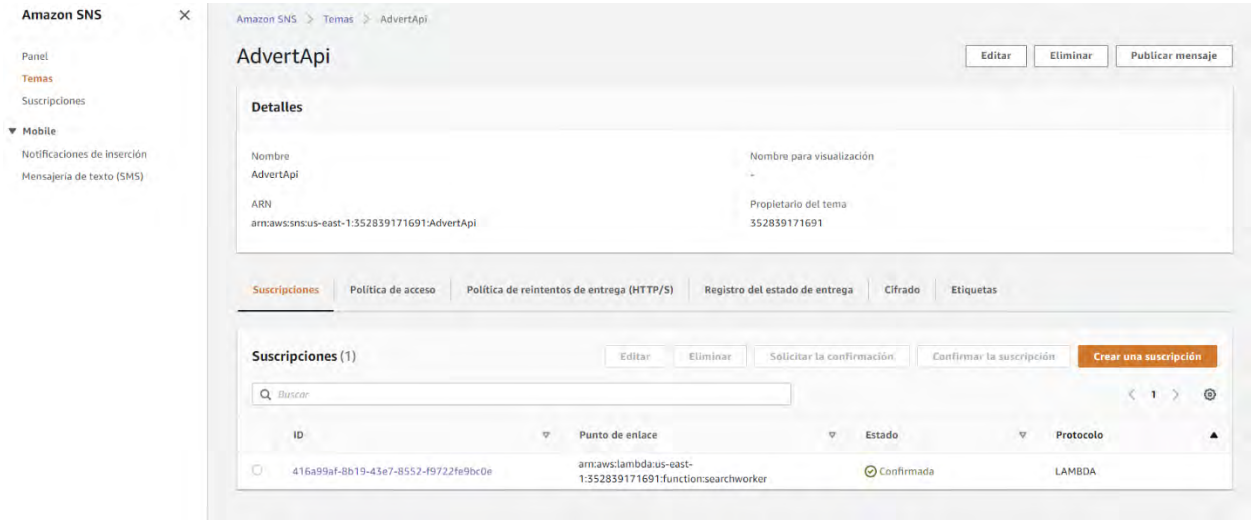

Figura 136: Página tema AdvertApi en SNS

En la página para publicar mensajes en tema se necesita especificar ciertos atributos opcionales y obligatorios para publicar el mensaje. Los atributos opcionales de nombre del asunto y tiempo de vida se dejaron en blanco. En cuanto al primer atributo obligatorio se seleccionó la estructura del mensaje con una misma carga para todos los protocolos de entrega, tal como se ve en la figura 137.

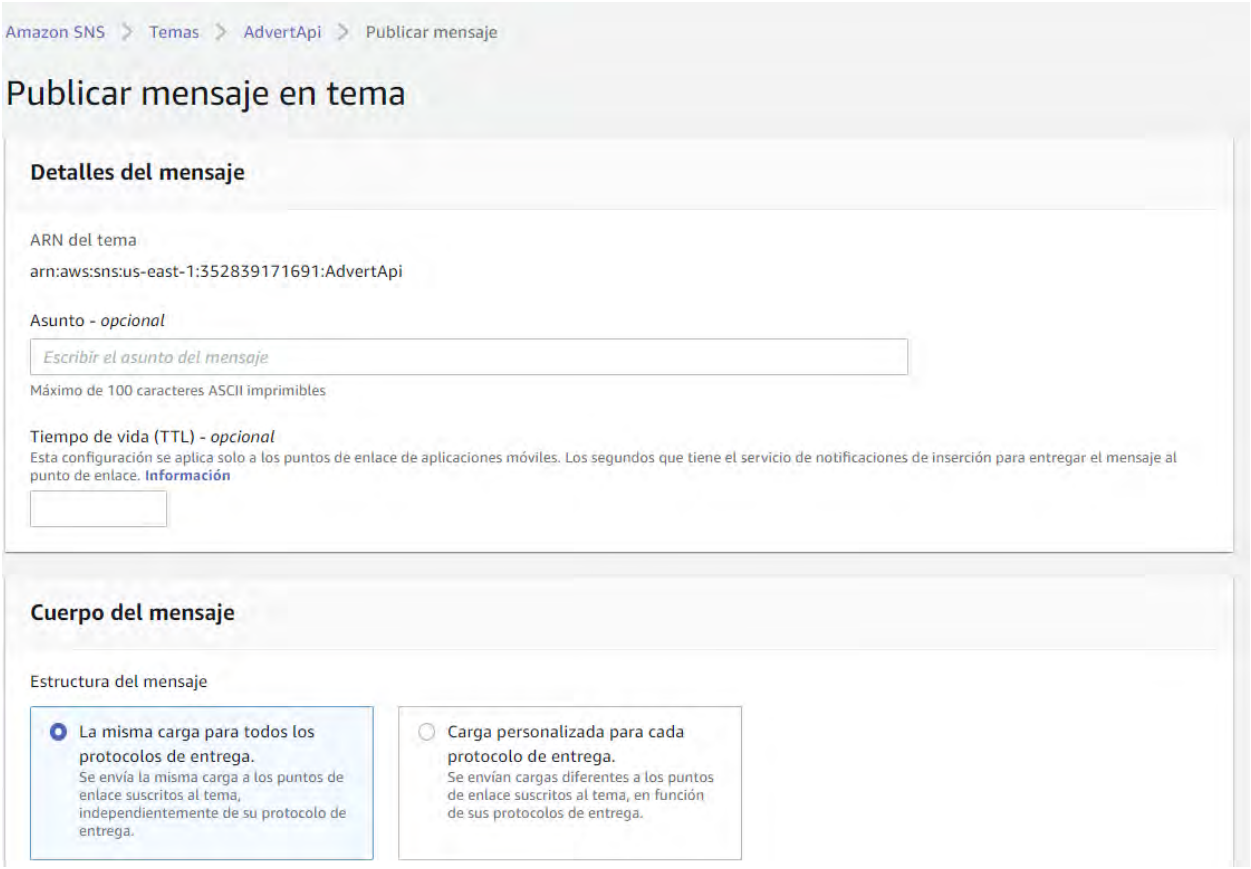

Figura 137: Página para publicar mensajes en SNS

El segundo atributo obligatorio al cual se le asignó un valor fue el cuerpo del mensaje. Para el cuerpo del mensaje se utilizó el formato JSON ya que permite almacenar y transportar datos de una manera ligera. Para usar JSON se requiere asignar una llave y un valor para dicha llave como se muestra en la figura 138. Se asignaron los valores de “Titulo” como llave y una cadena de texto “Mensaje de prueba 3 :)” como el valor de la llave. Después de configurar estos atributos se puede publicar el mensaje.

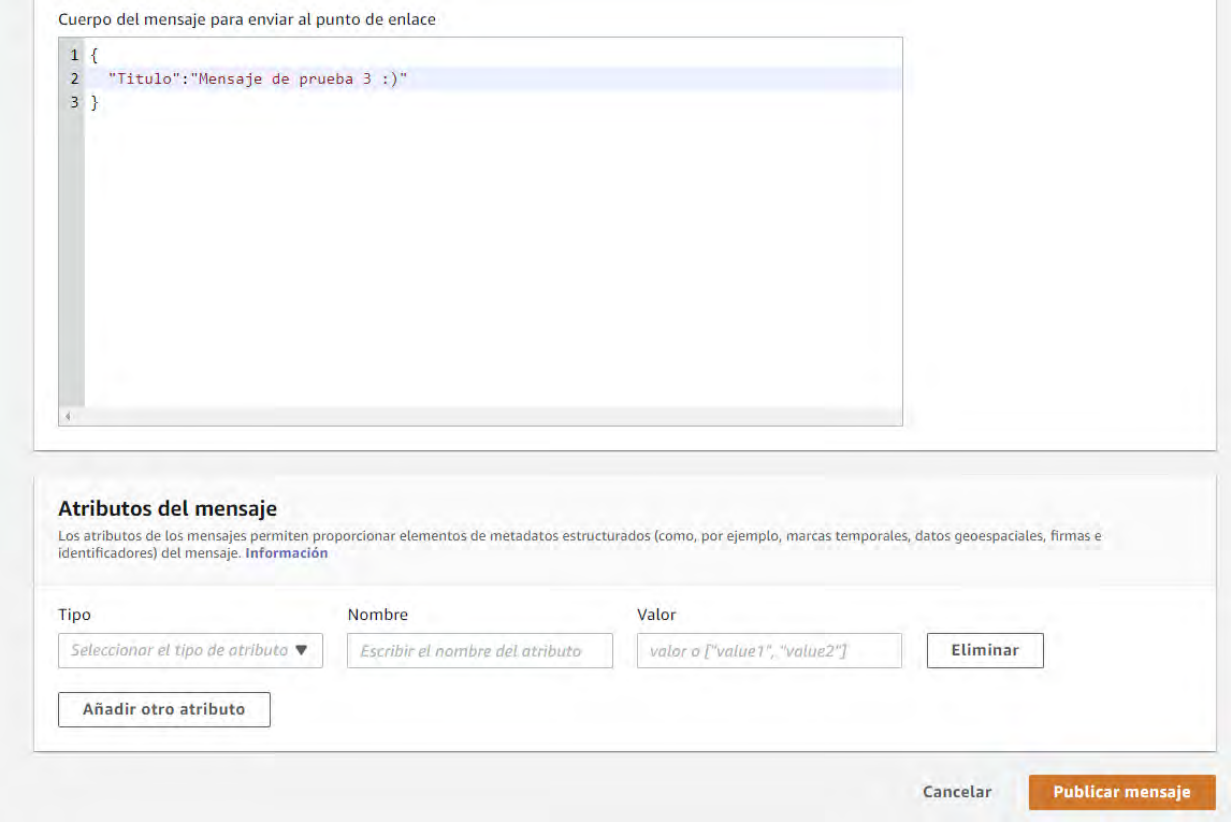

Figura 138: Página para publicar mensajes en SNS

En caso de que se publique correctamente el mensaje en el tema SNS se deberá mostrar una notificación que indique que se ha realizado la publicación correctamente en el tema, como se ve en la figura 139. El siguiente paso es verificar que Lambda haya registrado el evento en Cloudwatch.

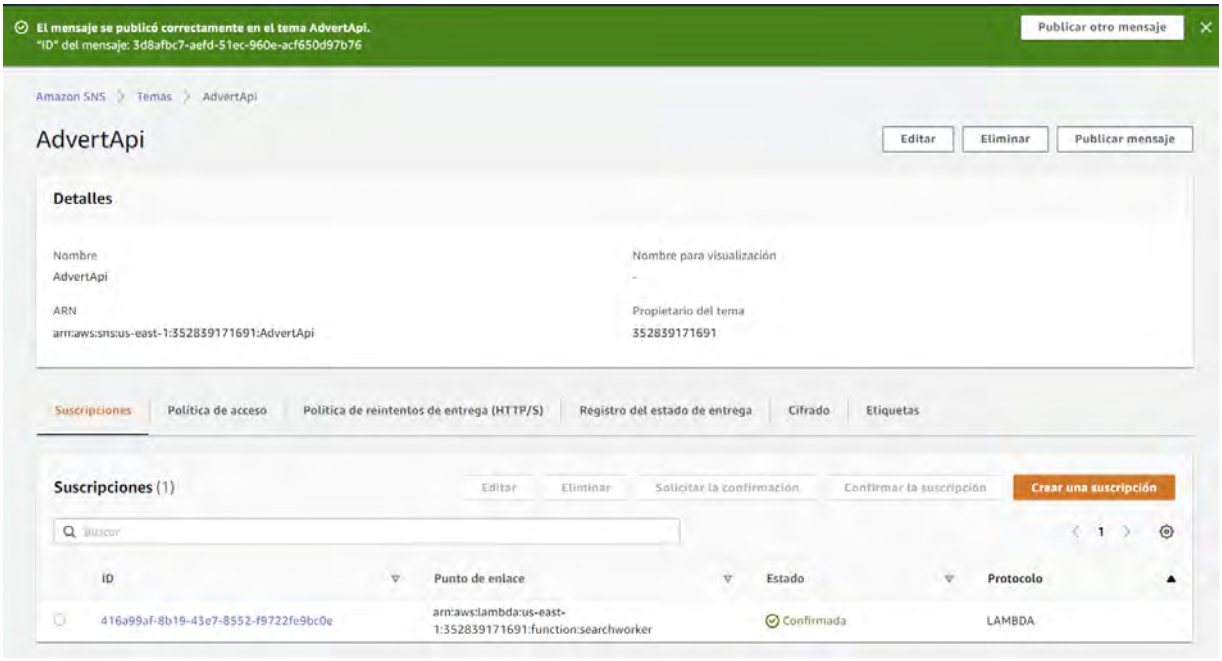

Figura 139: Página de notificaciones en SNS

Para verificar que se haya registrado el evento se debe entrar a editar la función Lambda “searchworker” y en el apartado de monitorización se debe estar usando el servicio de Cloudwatch el cual debió registrar el evento que ocurrió desde SNS en la función Lambda.

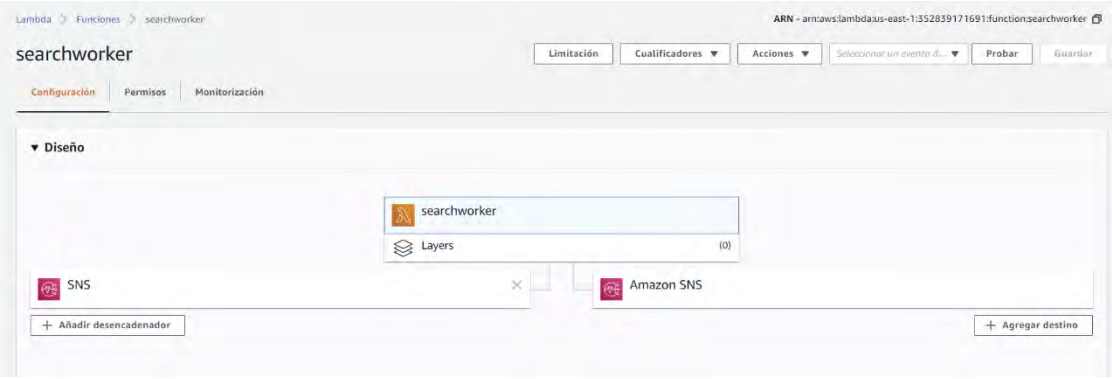

Figura 140: Página para editar funciones en Lambda

Dentro del apartado de monitorización se necesita entrar a la opción de ver los registros de CloudWatch para verificar que haya entrado en funcionamiento el desencadenador que se configuro en la sección anterior.

Figura 141: Pestaña de monitorización en Lambda

Tal como se muestra en la figura 142 se recibió un mensaje de prueba 3. Por lo tanto, se infiere que se realizó una configuración adecuada de Lambda y que está en correcto funcionamiento. Es importante resaltar, que el código que se subió a la función en los pasos anteriores, se programó la función para que ejecutara este evento. En caso de no haber programado el evento no se hubiera generado el registro en CloudWatch.

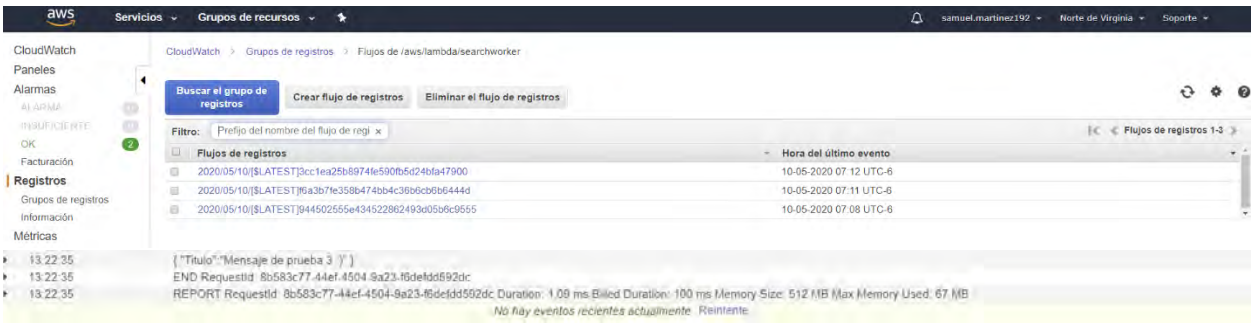

Figura 142: Página de inicio en CloudWatch

Anexo XIII: Configuración de Gateway API

La décima tercera sección en la implementación del caso de aplicación explica el proceso de configuración del servicio Gateway API. Se utilizará este servicio como puerta de enlace para evitar exponer el DNS público de la instancia EC2 y con ello conseguir mejorar la seguridad del caso de aplicación. Para iniciar con el proceso de configuración se entrar la consola de administración de AWS y se busca al servicio Gateway API. Después de entrar a la página principal se muestra la página de la figura 143 y en ella se va a crear una Gateway API del tipo API REST.

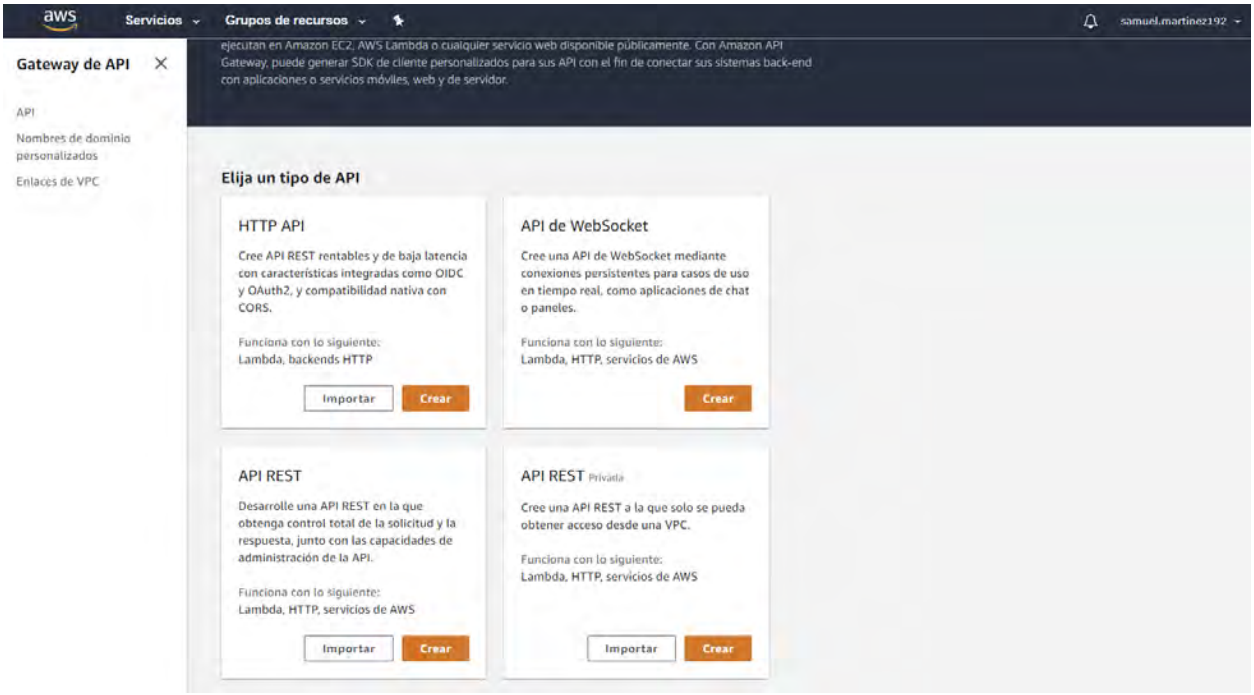

Figura 143: Página de inicio en API Gateway

La primera página que se muestra para crear API REST es la página para crear una API como se muestra en la figura 144. En esta página se seleccionó el protocolo REST, se seleccionó la opción de una API nueva y se asignó el nombre de “Public Web Api Proxy” para el nombre de la API. En cuanto al punto de enlace se eligió regional, ya que la configuración y puesta en marcha de esta opción es más rápida. Además, por el momento los clientes accederán desde la misma zona regional en donde se encuentran los otros servicios de AWS configurados en las secciones anteriores.

Figura 144: Página para crear una API en API Gateway

Posterior a que se haya creado la API Gateway se debe editar ya que se encuentra sin ninguna acción. Para asignarle acciones a la API Gateway se da clic al botón de acciones el cual muestra el menú de la figura 145. En este menú se entra a la opción de crear un recurso.

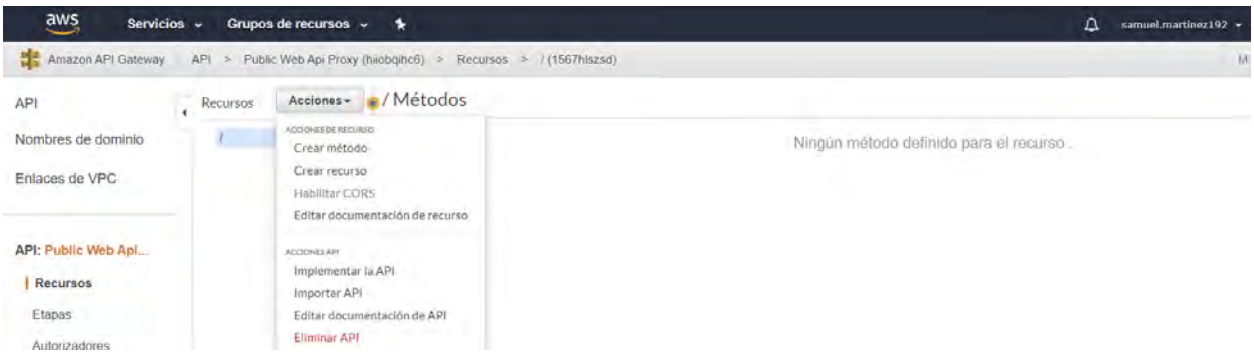

Figura 145: Página para editar API en API Gateway

La página para crear un nuevo recurso es la que se usará para crear el recurso que actuará como proxy para acceder a la instancia EC2 en donde esta implementado el microservicio AdvertApi. En esta página se debe marcar la opción de configurar como recurso proxy y habilitar API Gateway, tal como se muestra en la figura 146.

Figura 146: Página para nuevos recursos en API Gateway

Después de crear el recurso, se necesita editarlo para asignarle que es una integración del tipo HTTP Proxy y la URL del punto de enlace. Para la URL se ingresa el DNS público de la instancia EC2 a la que se quiere acceder utilizando el proxy proporcionado por API Gateway.

Figura 147: Página para editar recursos en API Gateway

Hasta este punto se tiene la API Gateway creada. Sin embargo, es necesario implementarla para que entre en funcionamiento. Para ello, se entra al menú de acciones y a la opción de implementar API como se muestra en la figura 148.

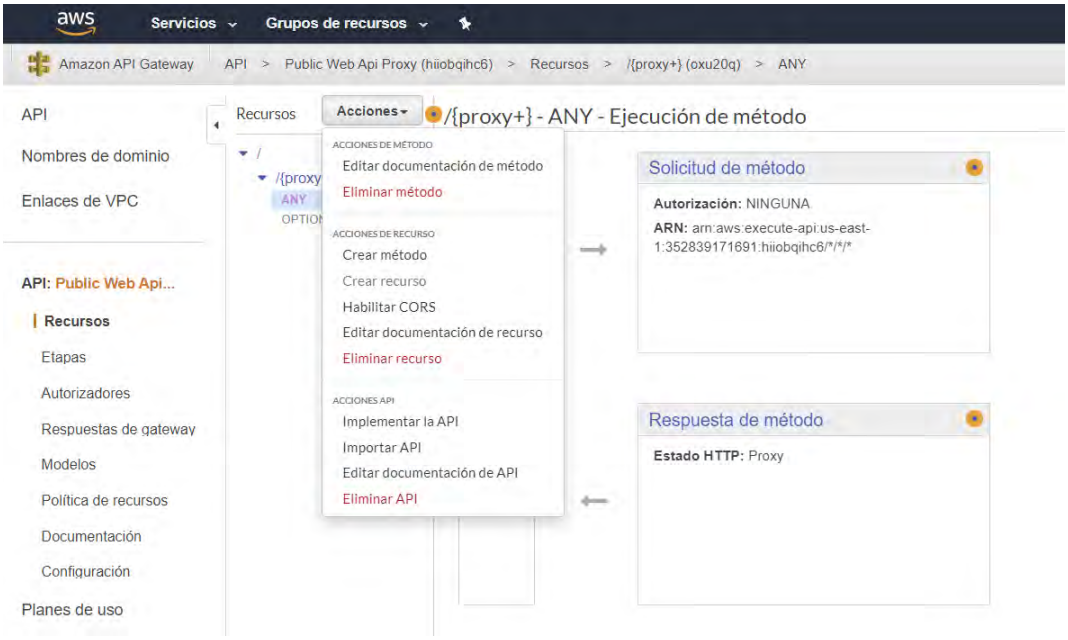

Figura 148: Página para editar API en API Gateway

Después de entrar a la opción de implementar API se muestra la página de la figura 149. En esta página se captura en tipo de implementación, nombre de la fase, descripción de la etapa y descripción de la implementación.

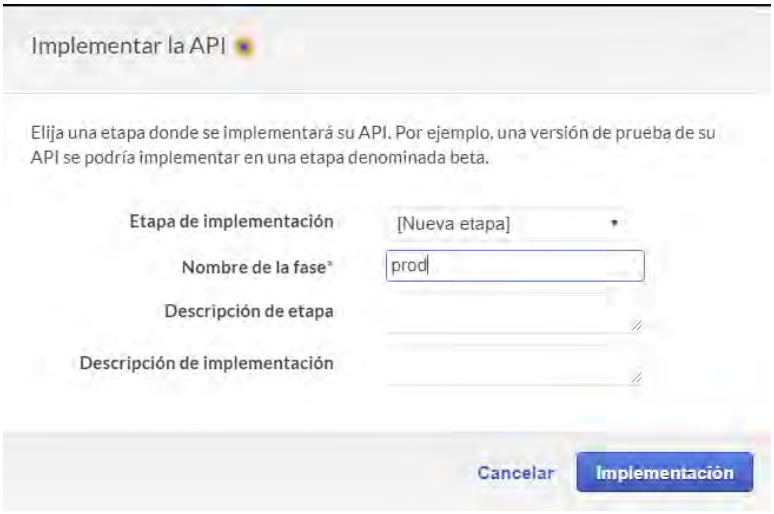

Figura 149: Página para implementar API en API Gateway

Para finalizar, en la figura 150 se muestra la URL que se generó después de crear y poner en marcha el servicio de Gateway API. Esta URL es la que permitirá acceder al microservicio de la instancia EC2 sin exponer el DNS público.

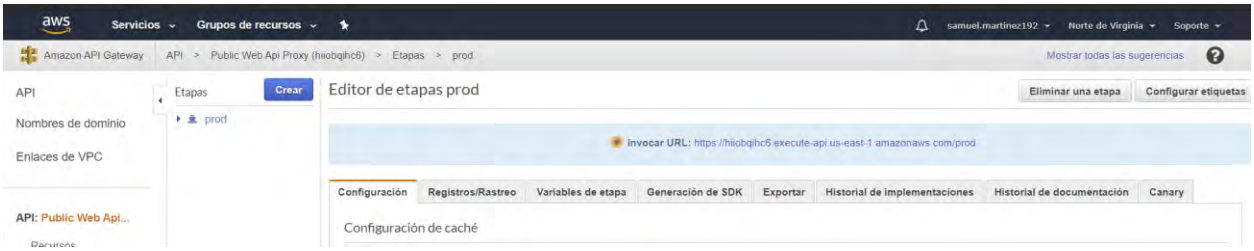

Figura 150: Página para editar etapas en API Gateway

Para comprobar que la instancia EC2 y el servicio de Gateway API están en correcto funcionamiento se ingresa la URL en el navegador web que prefiera el usuario. En la figura 151 se muestra la URL y como parámetro /health el cual verifica que el microservicio en EC2 se encuentra operativo y listo para responder solicitudes.

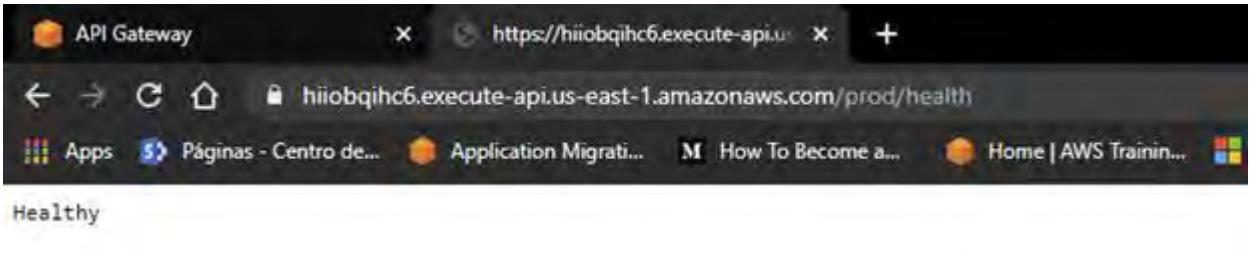

Figura 151: Navegador Web con URL del Proxy en API Gateway

Anexo XIV: Creación de imagen Docker

La décima cuarta sección en la implementación del caso de aplicación describe el proceso a seguir para la creación de imágenes en .NET Core. Para comenzar es necesario seleccionar un componente del cual se desea crear una imagen Docker. En este punto se puede seleccionar como componente un microservicio, servicio o aplicación. Posterior a seleccionar el componente se necesita crear un archivo DockerFile de dos posibles maneras. La primera manera es crear un archivo manualmente el cual tiene que contener los comandos necesarios que se utilizara para crear la imagen Docker. La segunda manera es usar un IDE para generar el archivo DockerFile como se muestra en la figura 152.

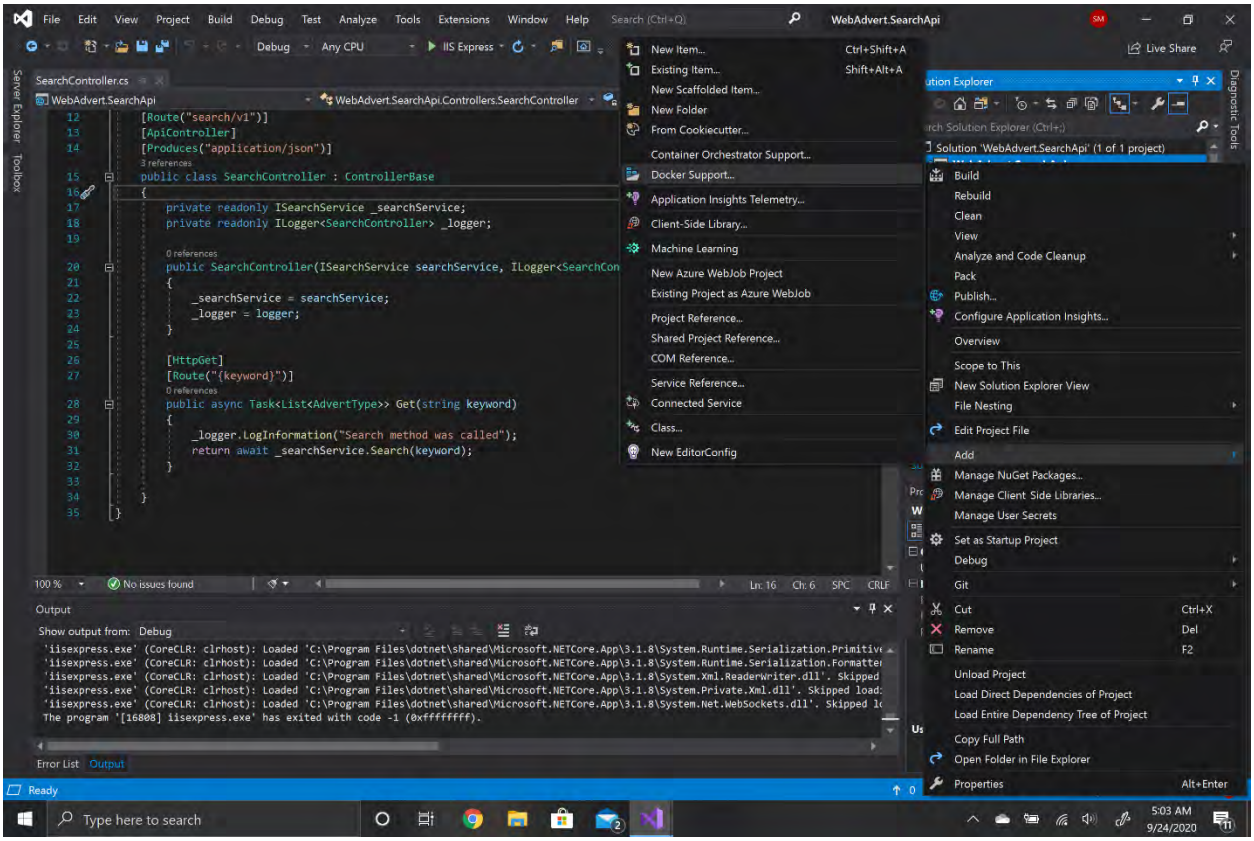

Figura 152: Soporte para Docker en Visual Studio 2019

El segundo paso después de agregar el soporte a Docker utilizando Visual Studio es seleccionar un sistema operativo base para el archivo DockerFile. Para este punto se puede pensar que es necesario seleccionar el sistema operativo base que se está usando en el entorno de desarrollo. Sin embargo, se puede crear un archivo DockerFile para un sistema operativo diferente al que se está utilizando. Para el microservicio SearchAPI se decidió utilizar Linux como sistema base por la reducción de costos y memoria en comparación de una imagen para Windows. En la figura 153 se muestra los comandos que contiene DockerFile y serán usados por Docker para construir una imagen para Linux en .NET Core.

Dockerfile

```
1  #See https://aka.ms/containerfastmode to understand how Visual Studio uses this Docker
2
3  FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base
4  WORKDIR /app
5  EXPOSE 80
6
7  FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build
8  WORKDIR /src
9  COPY ["WebAdvert.SearchApi.csproj", ""]
10 RUN dotnet restore "./WebAdvert.SearchApi.csproj"
11 COPY . .
12 WORKDIR "/src/."
13 RUN dotnet build "WebAdvert.SearchApi.csproj" -c Release -o /app/build
14
15 FROM build AS publish
16 RUN dotnet publish "WebAdvert.SearchApi.csproj" -c Release -o /app/publish
17
18 FROM base AS final
19 WORKDIR /app
20 COPY --from=publish /app/publish .
21 ENTRYPOINT ["dotnet", "WebAdvert.SearchApi.dll"]
```

Figura 153: Archivo DockerFile

El tercer paso para crear la imagen Docker es utilizar la terminal para indicarle a Docker en donde se encuentra el archivo DockerFile para construir la imagen. Para ello se entra al directorio en donde se encuentra el microservicio con el comando que se muestra en la figura 154.

Figura 154: Comando para acceder a la ruta del archivo DockerFile

El cuarto paso es verificar que se encuentre el archivo DockerFile que se creó en los pasos anteriores. Para verificar que se encuentre el archivo se ingresa el comando que presenta la figura 155.

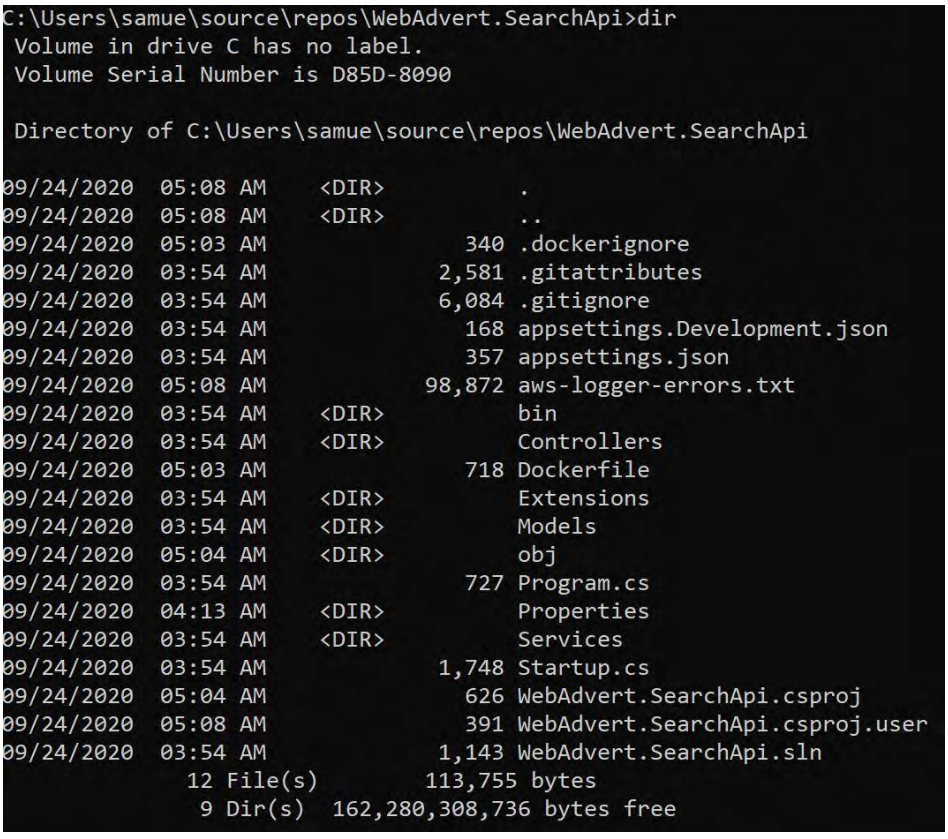

Figura 155: Comando para ver los archivos en la ruta seleccionada

El quito paso es construir la imagen Docker que será utilizada por la instancia del contenedor Docker. La imagen se construye con el comando “Docker build” como se ve en la figura 156.

```
C:\Users\samue\source\repos\WebAdvert.SearchApi>docker build . -t searchapi
```

Figura 156: Comando para crear imágenes Docker

El sexto paso es esperar que Docker construya la imagen y después validar que se haya construido sin errores. Si se creó correctamente el archivo DockerFile y se tiene una instalación correcta de Docker en el entorno de desarrollo se debe mostrar un mensaje indicando que se ha construido la imagen correctamente como se ve en la figura 157.

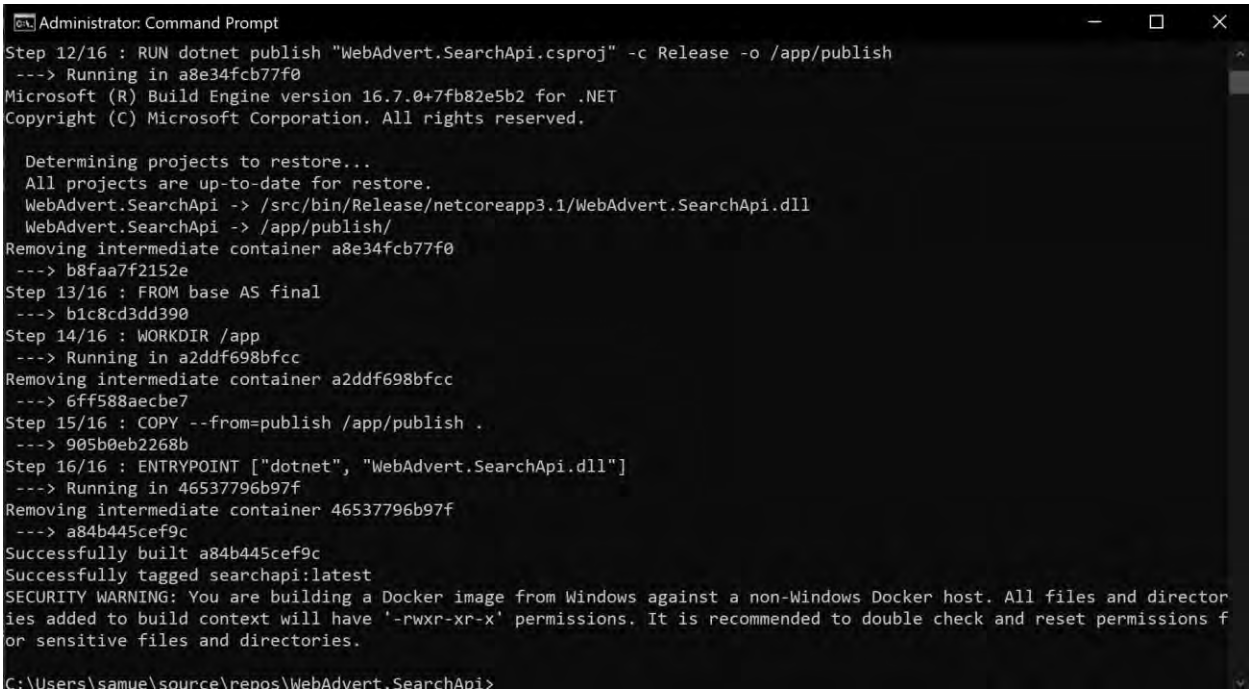

Figura 157: Ejecución del comando Docker build

Anexo XV: Configuración de ECS

La décima quinta sección en la implementación del caso de aplicación explica cómo crear un repositorio en AWS ECR para almacenar imágenes Docker de las cuales se pueden crear instancias de contenedores. Para iniciar con el proceso de creación de un repositorio en ECR es necesita entrar a la consola de administración de AWS y buscar el servicio de ECS. Después de entrar al servicio de ECS se presenta la página de bienvenida de la figura 158. En esta página se entra al enlace de repositorios para ir al siguiente punto.

Figura 158: Página de inicio ECS

Después de entrar a la sección de repositorios AWS muestra los repositorios existentes y un botón para crear nuevos repositorios de imágenes Docker. Dado que no existen repositorios se creará uno nuevo como se ve en la figura 159.

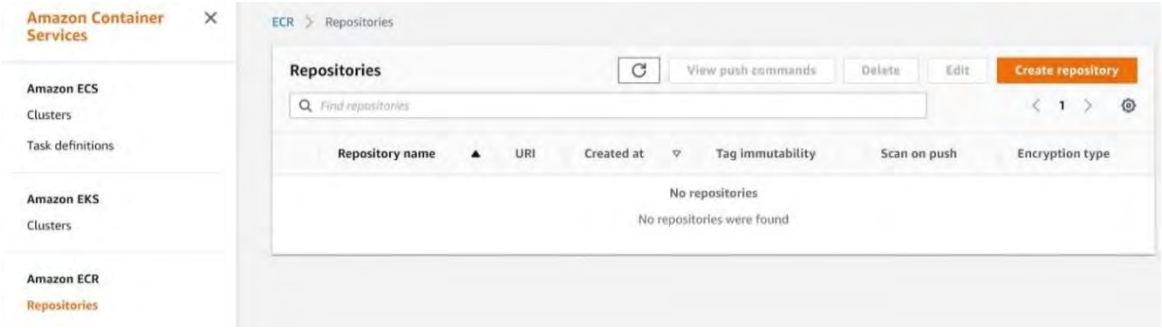

Figura 159: Página repositorios ECR

El primer paso para crear el repositorio en ECR es asignar un nombre para el repositorio y determinar si se requieren habilitar las opciones de inmutabilidad de etiquetas y el escaneo al momento de subir imágenes. Para el caso de aplicación se determinó no utilizar estas opciones como se muestra en la figura 160.

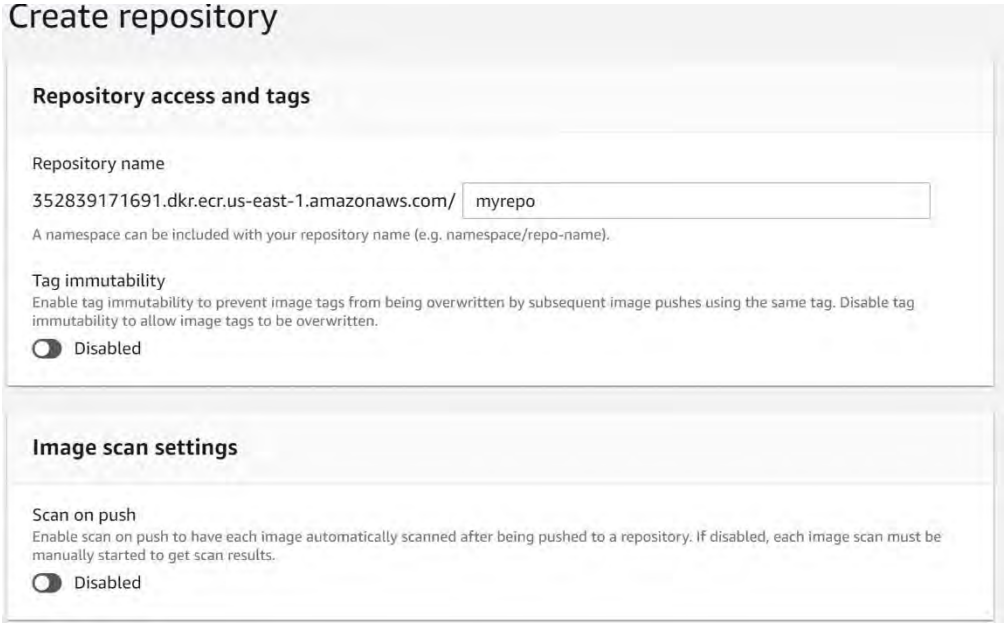

Figura 160: Página para crear repositorios ECR

El segundo paso es decidir si es necesario habilitar la tecnología de KMS para encriptar las imágenes Docker que se suban al repositorio ECR. Dado que solo se subirá una imagen de un microservicio se optó por no habilitar esta opción y continuar la creación del repositorio, tal como se muestra en la figura 161.

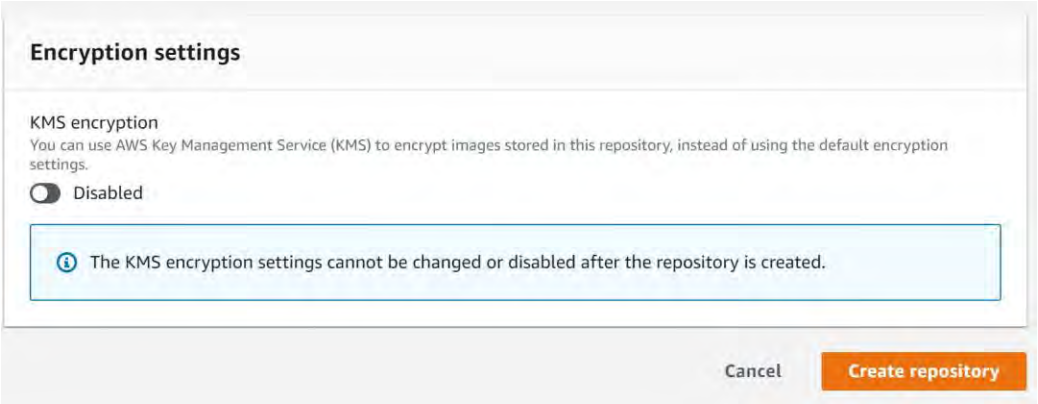

Figura 161: Página para crear repositorios ECR

Después de crear el repositorio en AWS ECR se mostrará el mensaje de la figura 162 indicando que se ha creado correctamente un repositorio en ECR con el nombre de “myrepo”. El siguiente paso es subir la imagen Docker al repositorio que se acaba de crear. Para ello se entra al enlace que se muestra en la columna “nombre de repositorio” en la figura 162.

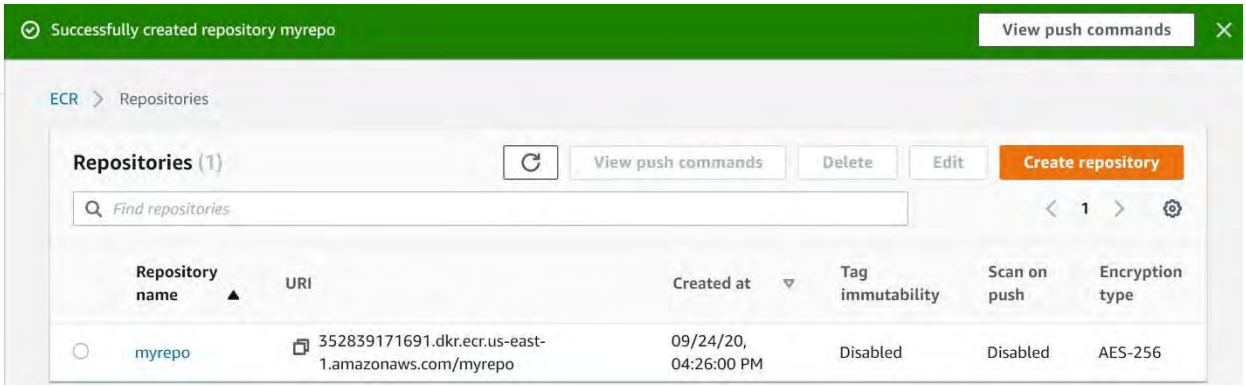

Figura 162: Página para ver y editar repositorios ECR

Ya que se entró al repositorio se muestra el botón de la figura 163 con el nombre de “ver comandos para subir”. Estos comandos se deben ingresar en la terminal de elección del usuario. Para este punto es necesario tener Docker y AWS CLI instalados en la maquina en donde se ejecutarán los comandos.

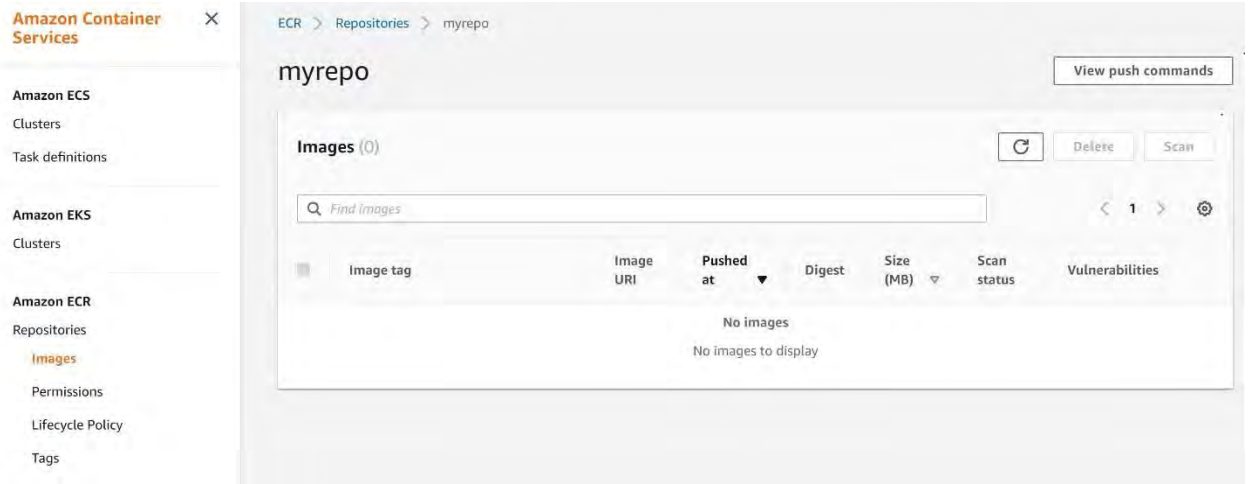

Figura 163: Página para editar repositorios ECR

La siguiente página que se muestra en la figura 164 se divide en comandos para macOS/Linux y Windows. La selección de cual tipo de comandos utilizar dependerá del sistema operativo base de la imagen Docker. Dado que para el caso de aplicación se creó una imagen para un sistema operativo base Linux se utilizará esta opción.

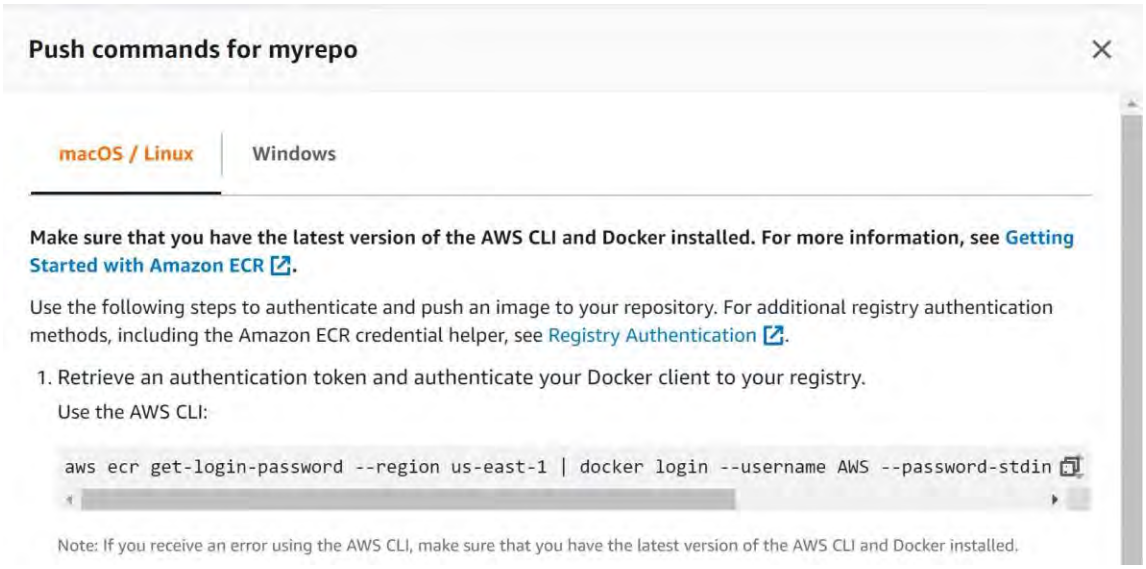

Figura 164: Comandos para subir imagen Docker a ECR

Continuando con la misma página se muestran el resto de los comandos que se necesitan ingresar de manera secuencial para subir la imagen Docker al repositorio. En la figura 165 se ven los comandos dos, tres y cuatro.

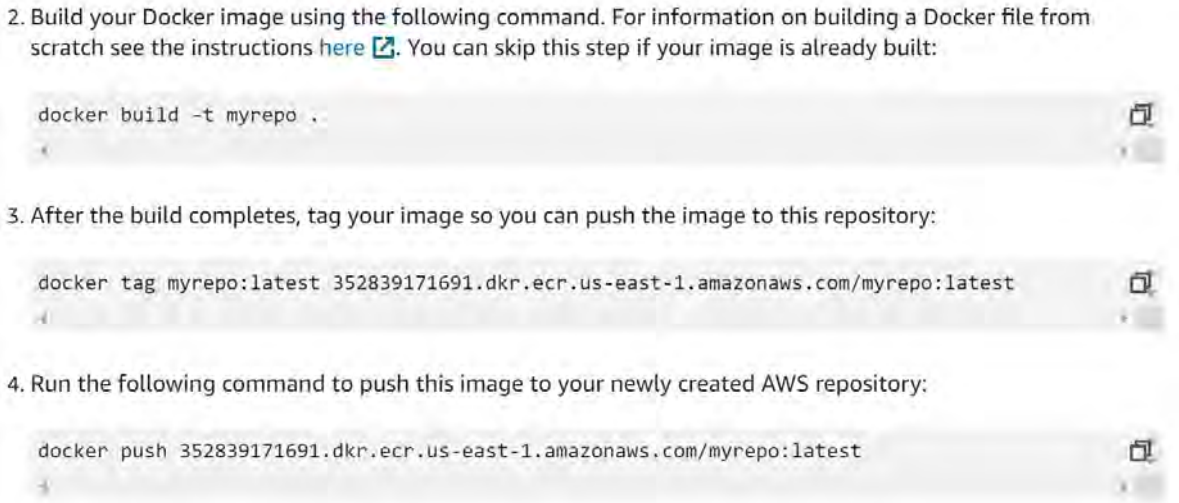

Figura 165: Comandos para subir imagen Docker a ECR

En las siguientes figuras se muestra la ejecución de los comandos para subir la imagen al repositorio ECR. El primer paso es obtener un token de autentificación y con ello autentificar el cliente Docker al registro ECR como se ve en la figura 166.

```
C:\Users\samue\source\repos\WebAdvert.SearchApi>aws ecr get-login-password --region us-east-1 | docker login --username
AWS --password-stdin 352839171691.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded

C:\Users\samue\source\repos\WebAdvert.SearchApi>
```

Figura 166: Comando para obtener token de autentificación

El segundo paso es construir la imagen Docker en caso de que no se haya construido previamente. Este comando se puede omitir en caso de que se haya construido la imagen en pasos previos como se ve en la figura 167.

```
C:\Users\samue\source\repos\WebAdvert.SearchApi>docker build -t myrepo .
```

Figura 167: Comando para construir imagen Docker

El tercer paso es etiquetar la imagen para de esta manera poder identificar la imagen y posteriormente subir al repositorio. La ejecución de este comando aun y cuando es correcta no muestra ningún mensaje o notificación como se ve en la figura 168.

```
C:\Users\samue\source\repos\WebAdvert.SearchApi>docker tag myrepo:latest 352839171691.dkr.ecr.us-east-1.amazonaws.com/my
repo:latest

C:\Users\samue\source\repos\WebAdvert.SearchApi>
```

Figura 168: Comando para etiquetar imagen Docker

El cuarto paso es ejecutar el comando para subir la imagen Docker al repositorio ECR que se creó recientemente en los pasos anteriores. La ejecución del comando mostrara un mensaje indicando que se ha subido la imagen correctamente como se ve en la figura 169.

```
:\Users\samue\source\repos\WebAdvert.SearchApi>docker push 352839171691.dkr.ecr.us-east-1.amazonaws.com/myrepo:latest
The push refers to repository [352839171691.dkr.ecr.us-east-1.amazonaws.com/myrepo]
abac079a9984: Pushed
542f26aa533d: Pushed
f4f010e6e005: Pushed
e86e0c2642d: Pushed
966e4a1df68a: Pushed
08e8c8e2420b: Pushed
97cab4339852: Pushed
latest: digest: sha256:428aa4ccf5f00d5fe90f6ceb9b001586fee5934055352d6a7d4522c585aa937d size: 1793
```

Figura 169: Comando para subir imagen Docker a ECR

Anexo XVI: Configuración de Fargate

La décima sexta sección de la implementación del caso de aplicación explica cómo crear una instancia de contenedor de la imagen Docker mediante el uso de AWS Fargate. El primer paso para comenzar es entrar la consola de administración de AWS y buscar al servicio de ECS. Después de entrar al servicio de ECS se entra al enlace de clústeres como se ve en la figura 170. En esta página es donde se creará un clúster en el cual se ejecutará por el momento una sola instancia del contenedor Docker.

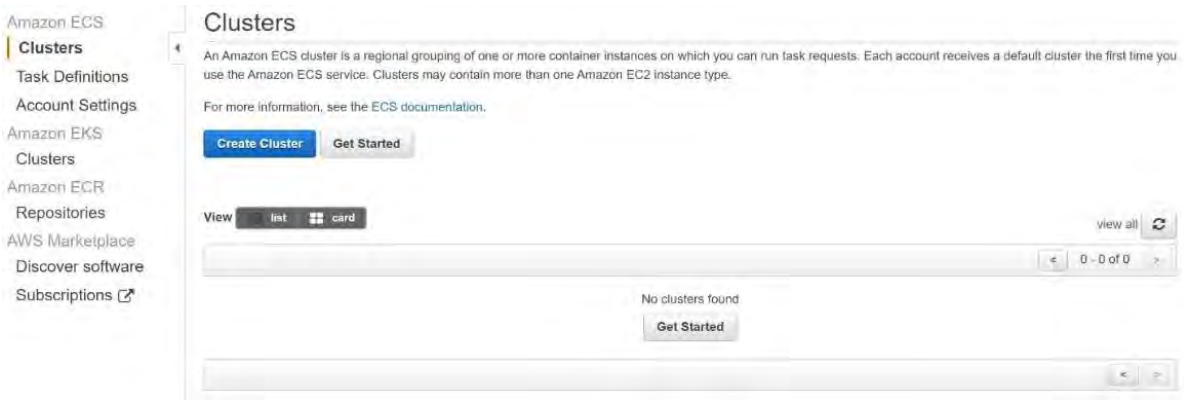

Figura 170: Página de Clusters en ECS

El segundo paso requiere seleccionar una plantilla en la que se tienen tres opciones. La primera opción hace uso de la nueva tecnología de AWS Fargate. La segunda opción requiere una configuración manual de una instancia EC2 para Linux. La tercera opción es igual a la segunda, pero con la diferencia que es para sistemas Windows. Para el caso de aplicación se utilizó la primera opción por la automatización de recursos y aprovisionamiento que se obtiene al utilizar esta tecnología.

Figura 171: Página para crear clusters en ECS

En el tercer paso se necesita asignar un nombre al clúster, configurar un VPC, añadir una etiqueta y decidir si habilitar las métricas de CloudWatch para obtener datos de funcionamiento del servicio. Para el caso de aplicación solamente se asignó el nombre del clúster como se ve en la figura 172.

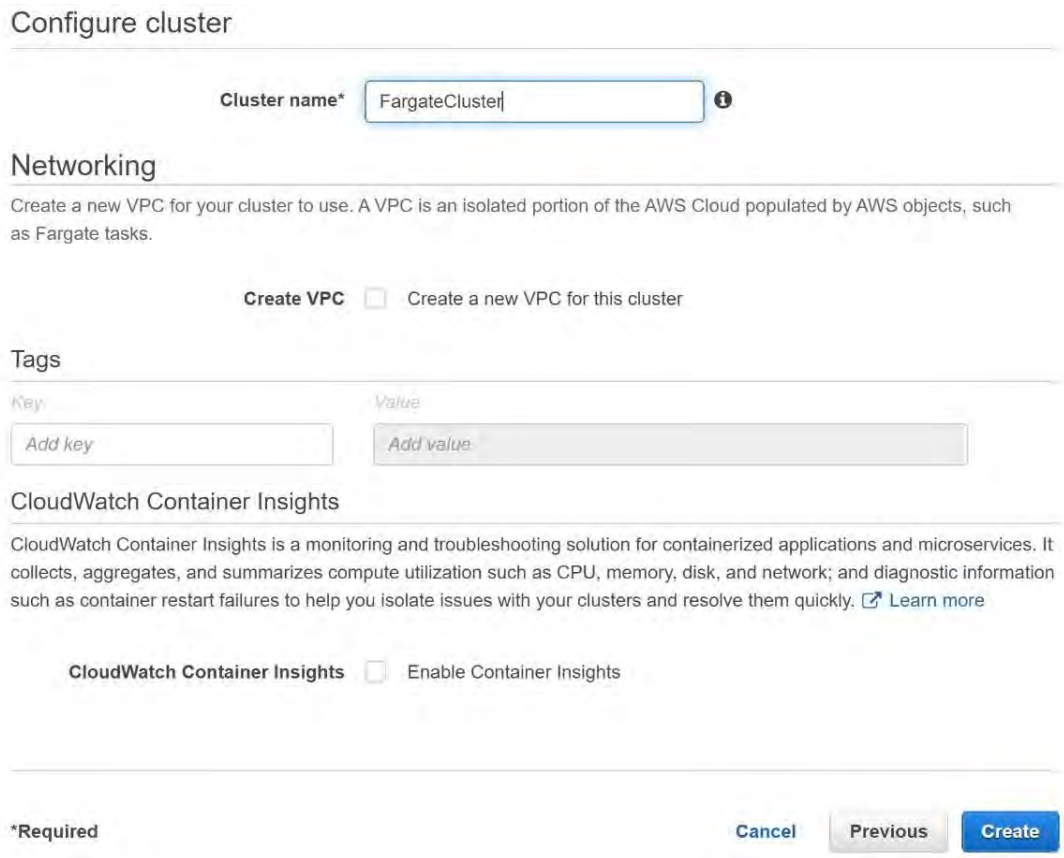

Figura 172: Página para crear clusters en ECS

El cuarto y último paso para la configuración del clúster es esperar a que AWS muestre un mensaje notificando que se ha creado correctamente un clúster y que se están inicializando las instancias de contenedor como se ve en la figura 173.

Figura 173: Notificación de creación de cluster en ECS

Ahora bien, hasta el punto anterior se tiene un clúster sin tareas creadas. Esto quiere decir que es un clúster que no tiene instancias de contenedores en ejecución. Para crear una instancia de contenedor se necesita crear una tarea en la que se asigne el contenedor como se mostrara en los siguientes puntos.

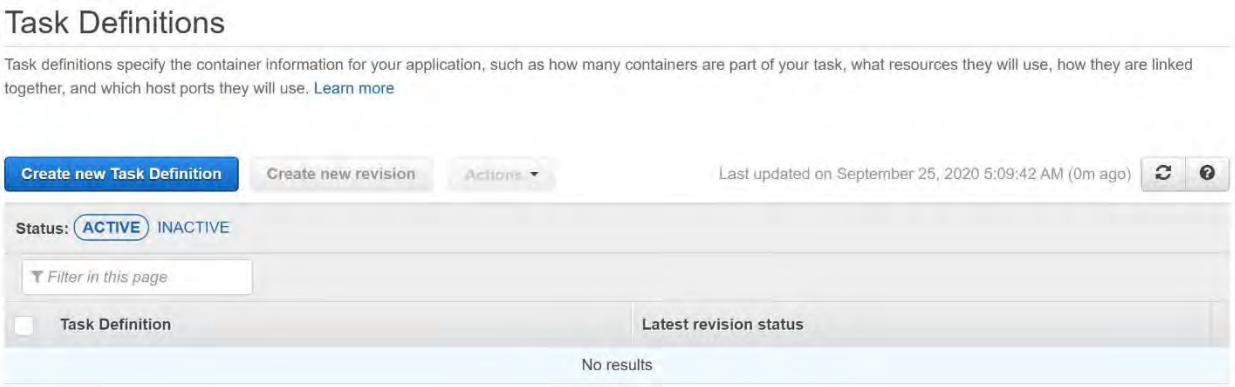

Figura 174: Definición de tareas en ECS

El primer punto para crear una tarea en ECS es seleccionar un tipo de lanzamiento para el cual se tienen dos opciones. La primera opción es utilizar Fargate la cual automatiza la administración de la infraestructura. La segunda opción es utilizar EC2 la cual requiere administrar y configurar la infraestructura en EC2. Para el caso de aplicación se utilizará la primera opción.

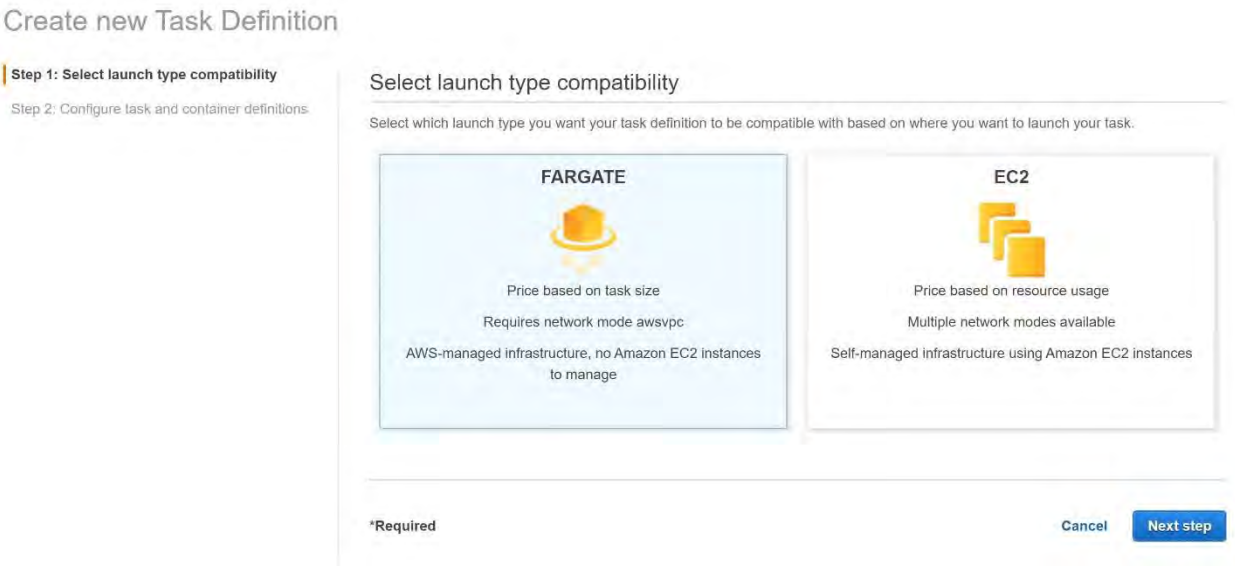

Figura 175: Página para crear tareas en ECS

El segundo punto muestra una página en donde se necesita realizar varias configuraciones. La primera configuración es asignar un nombre a la tarea y en caso de ser necesario se puede configurar un rol y un modo red. Para el caso de aplicación se dejaron estas dos últimas opciones en blanco como se muestra en la figura 176.

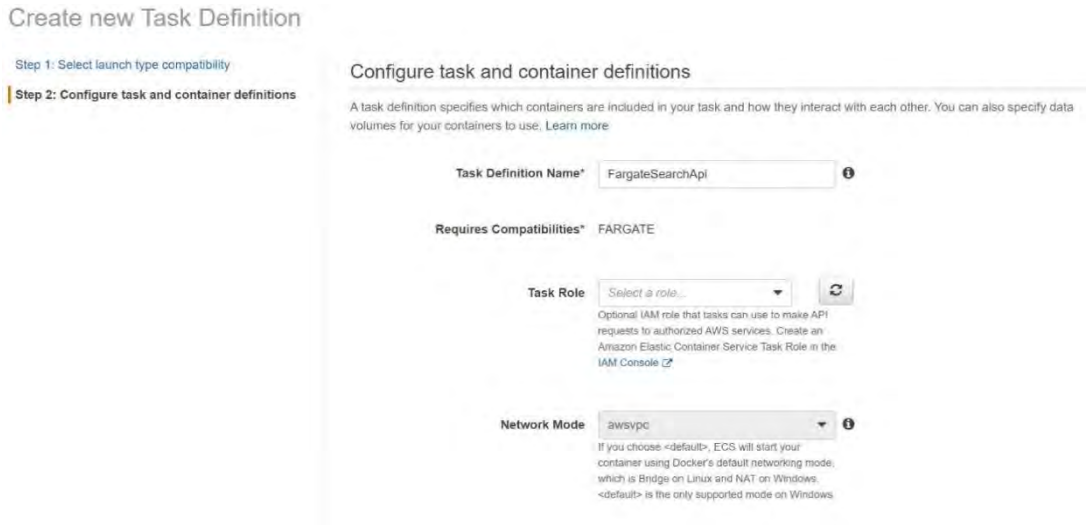

Figura 176: Página para crear tareas en ECS

El tercer punto requiere definir el tamaño de la tarea, es decir cuanta memoria y CPU se utilizan para ejecutar la tarea. En la figura 177 se muestra que se asignó 0.5 CPU y 1 GB de memoria ya que se usara la tarea para ejecutar un pequeño microservicio.

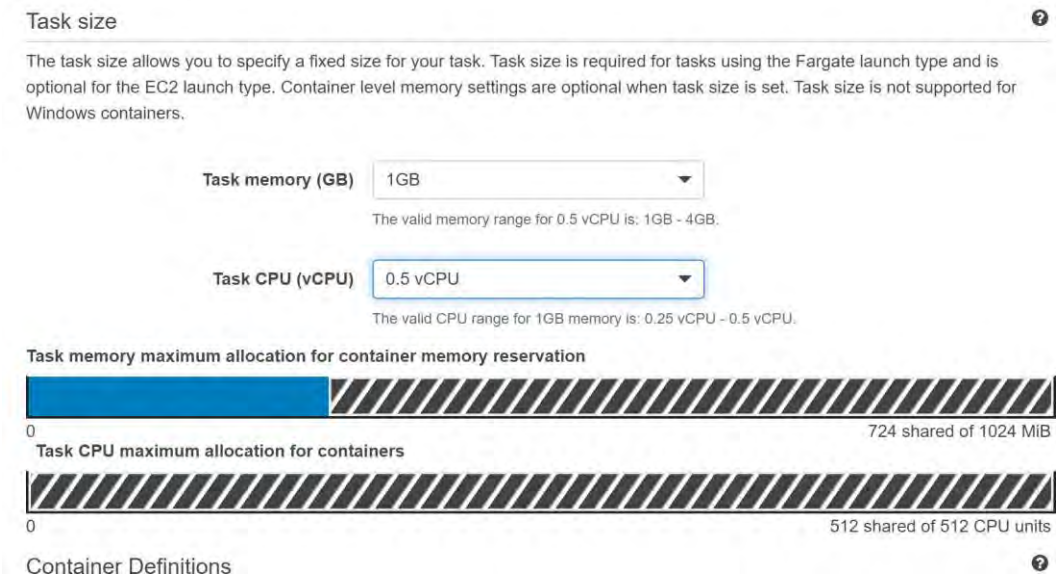

Figura 177: Página para crear tareas en ECS

Para el cuarto punto se necesita añadir el contenedor que se ejecutara en la tarea que se está defiendo en esta sección. Para ello se da clic al botón agregar contenedor como se puede ver en la figura 178.

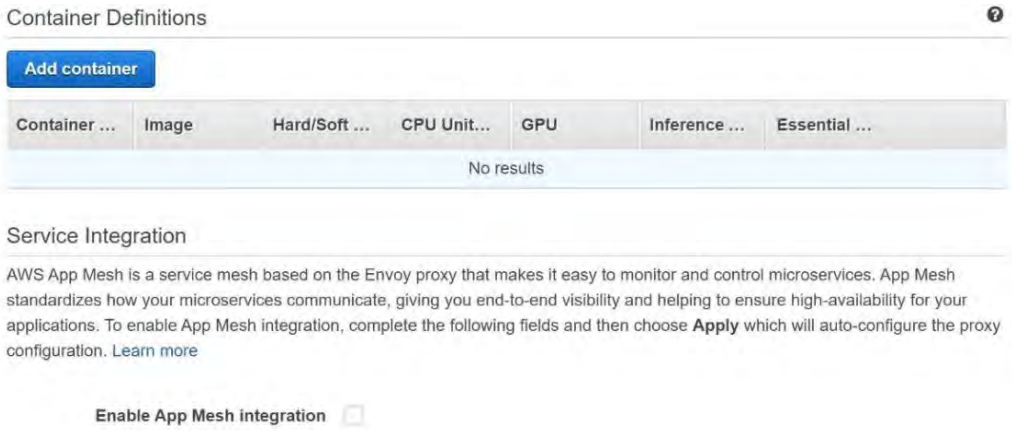

Figura 178: Página para crear tareas en ECS

Después de dar clic al botón de añadir contenedor se muestra la página de la figura 179. En esta página se asignó un nombre al contenedor, se especificó la URL con el repositorio ECR en donde se encuentra la imagen Docker y para finalizar se defino un límite de memoria y un mapeo de puertos.

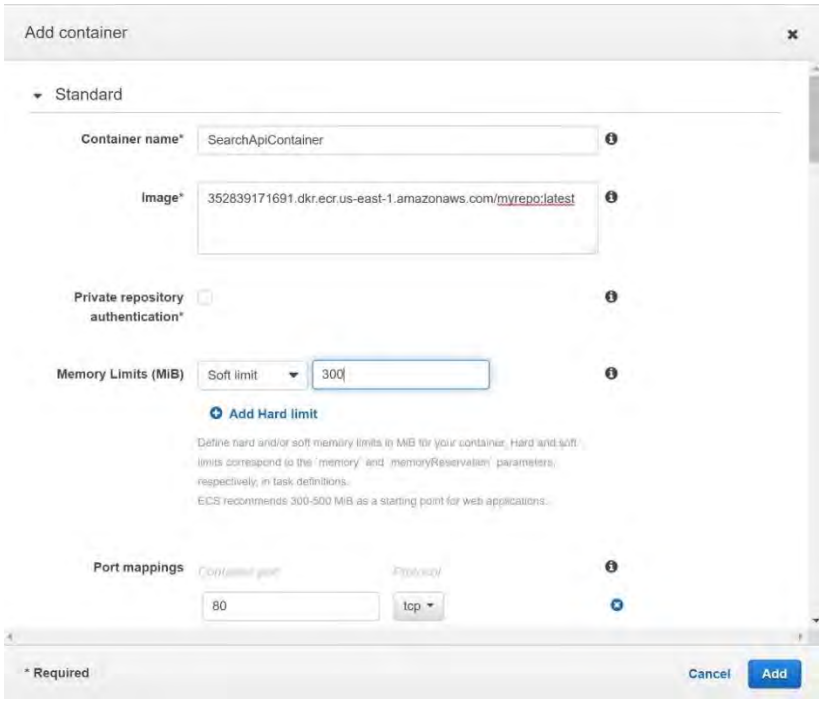

Figura 179: Página para crear tareas en ECS

En el quito punto se pueden realizar configuraciones más avanzadas para el contenedor, tal como un añadir un proxy, volúmenes de almacenamiento adicionales o bien agregar un servicio adicional para guardar logs de eventos. Todas estas opciones se dejaron en blanco como se muestra en la figura 180.

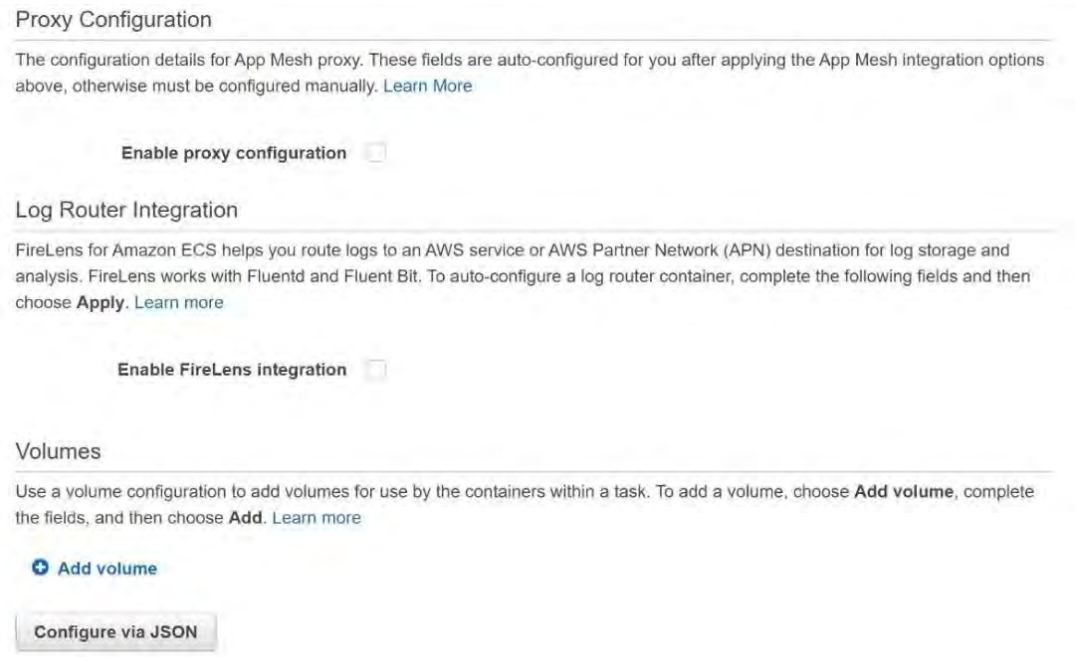

Figura 180: Página para crear tareas en ECS

En sexto y último punto para crear la tarea es esperar a que el asistente de AWS muestre un mensaje indicando que se han definido los estados de la tarea y se encuentra en proceso de lanzar la tarea como se puede ver en la figura 181.

Figura 181: Notificación de tarea creada en ECS

Para concluir con la configuración del servicio AWS Fargate se requiere ejecutar la tarea que se definido en los puntos anteriores en el clúster de Fargate. Para ello se entra al clúster "FargateSearchApi" y en el botón de acciones se muestra un menú desplegable para poner en ejecución a la tarea que contiene el contenedor como se puede ver la figura 182.

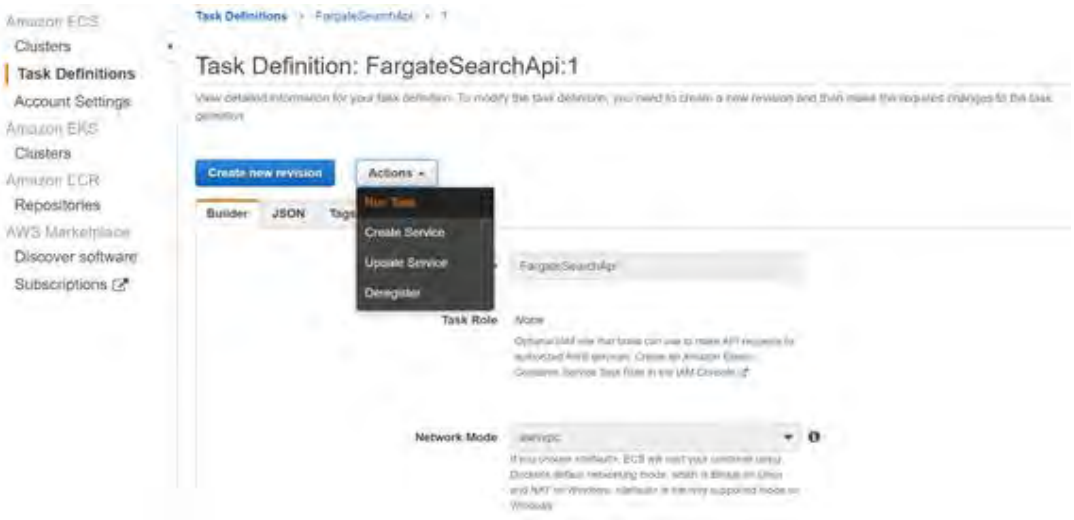

Figura 182: Página para ejecutar tareas en ECS

Inmediatamente después de dar clic a la opción de ejecutar tarea se muestra la página de la figura 183. En esta página se definió que tipo de lanzamiento usara la ejecución de la tarea, la versión de la plataforma, el clúster en donde se ejecutara la tarea y la cantidad de tareas que se estarán en ejecución en el clúster.

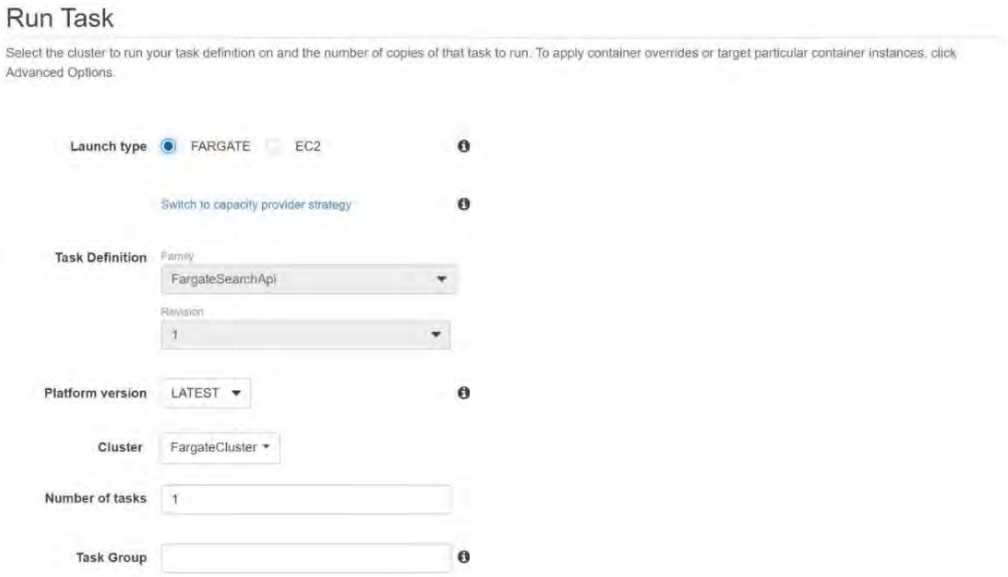

Figura 183: Página para editar tarea en ejecución en ECS

Continuando con la misma página de configuración se seleccionó el VPC y la subred en la cual estará en ejecución la tarea, así como el grupo de seguridad y la asignación automática direcciones IP como se puede ver en la figura 184.

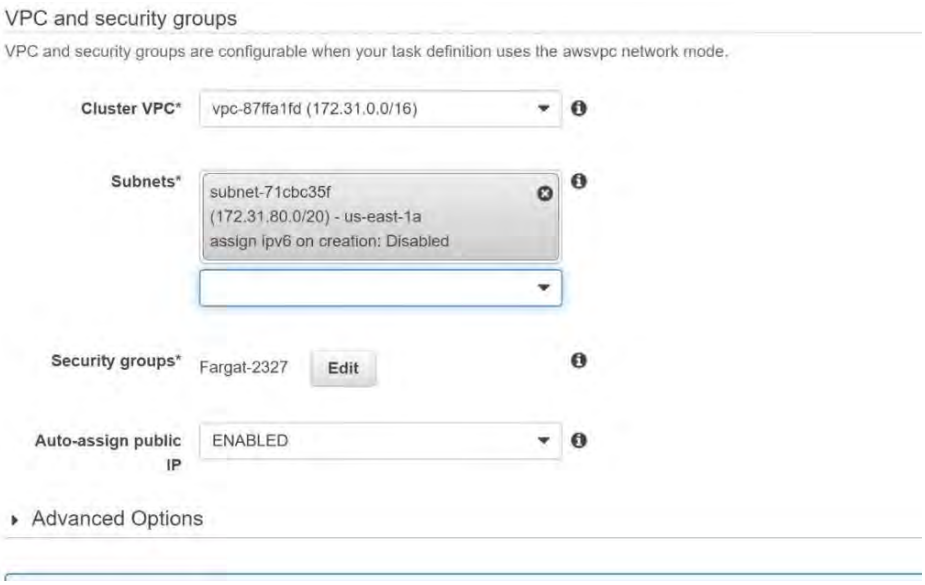

Figura 184: Página para editar tarea en ejecución en ECS

Listo, con las configuraciones realizadas anteriormente se terminó la configuración del servicio AWS Fargate. En la figura 185 se puede ver que existe un clúster con el nombre "FargateCluster", el cual tiene un estatus de activo y está ejecutando una tarea que contiene la definición de un contenedor Docker.

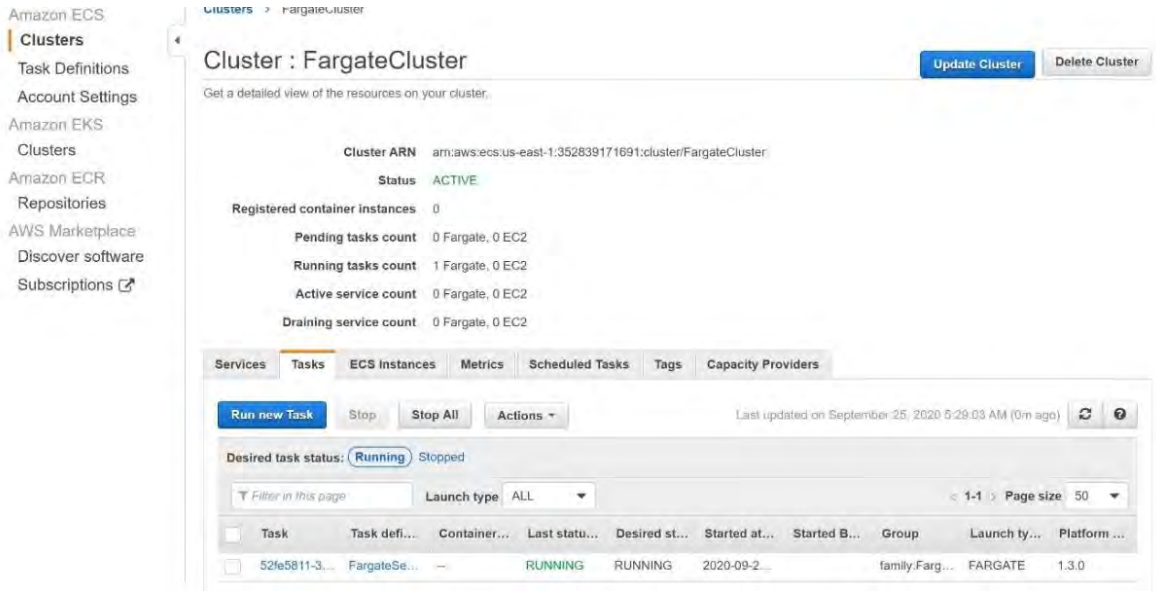

Figura 185: Página para ver y editar clusters en ECS

Referencias

[1] L. Da Xu, E. L. Xu y L. Li, «Industry 4.0: state of the art and future trends,» *International Journal of Production Research,* vol. 56, nº 8, p. 2941, 2018.

[2] Z. R. Alashhab, M. Anbar, M. Mahinderjit, Y.-B. Leau, Z. Ali y S. Abu, «Impact of coronavirus pandemic crisis on technologies and cloud computing applications,» *Journal of Electronic Science and Technology,* pp. 10059-10071, 2020.

[3] B. Borcherding, «Why Enterprises Are Accelerating Cloud Adoption,» Forbes, 2020.

[4] L. Eagle, «Going Hybrid Demand For Cloud And Managed Services Across Asia-Pacific,» *451 Research,* pp. 1-25, 2019.

[5] A. Banijamali, O. Pekka, P. Kuvaja y M. Oivo, «Software architectures of the convergence of cloud computing and the Internet of Things: A systematic literature review,» *Information and Software Technology,* vol. 122, pp. 1-24, 2020.

[6] C. Duffy, «From the brink of bankruptcy to a 1,300% stock gain: How this CEO turned around her company,» CNN Business, 2020.

[7] IMCO Staff, «Cómputo en la Nube: Nuevo Detonador para la Competitividad de México,» Instituto Mexicano para la Competitividad A.C., México, 2012.

[8] Altexsoft, «Legacy System Modernization: How to Transform the Enterprise for Digital Future,» *Altexsoft Software R&D Engineering,* pp. 1-34, 2018.

[9] A. Khayer, S. Talukder, Y. Bao y N. Hossain, «Cloud computing adoption and its impact on SMEs' performance for cloud supported operations: A dual-stage analytical approach,» *Technology in Society,* vol. 60, pp. 1-30, 2020.

[10] M. Mihalec, «Five Reasons More Businesses Are Choosing Cloud,» Forbes, 2020.

[11] C. Pahl, «Containerization,» *Irish Centre for Cloud Computing and Commerce,* vol. 2, nº 3, pp. 24-31, 2015.

[12] C. Espinoza, Metodología de investigación tecnológica, Huancayo, Perú: Imagen Grafica, 2010.

[13] P. Mell y T. Grance, «The NIST Definition of Cloud Computing,» National Institute of Standards and Technology, Gaithersburg, 2011.

[14] D. Rountree y I. Castrillo, The Basics of Cloud Computing : Understanding the Fundamentals of Cloud Computing in Theory and Practice, USA: Elsevier, 2014.

[15] D. Marinescu, Cloud ComputingTheory and Practice, Waltham: ElSevier, 2018.

[16] L. Columbus, «Roundup Of Cloud Computing Forecasts And Market Estimates, 2018,» Forbes, 2018.

[17] C. Pahl, A. Brogi, J. Soldani y P. Jamshidi, «Cloud Container Technologies: A State-of-the-Art Review,» *IEEE TRANSACTIONS ON CLOUD COMPUTING,* vol. 7, nº 3, pp. 677-692, 2019.

[18] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah y P. Merle, «Elasticity in Cloud Computing: State of the Art and Research Challenges,» *IEEE TRANSACTIONS ON SERVICES COMPUTING,* vol. 11, nº 2, pp. 430-447, 2018.

[19] M. Souppaya, J. Morello y K. Scarfone, «Application Container Security Guide,» *NIST Special Publication 800-190,* pp. 1-63, 2017.

[20] S. Sultan, I. Ahmad y T. Dimitriou, «Container Security: Issues, Challenges, and the Road Ahead,» *IEEE Access,* vol. 7, pp. 52976-52996, 2019.

[21] M. Pittenger, «Addressing the security challenges of using containers,» *Network Security,* vol. 2016, nº 12, pp. 5-8, 2016.

[22] A. Martin, S. Raponi, T. Combe y R. Di Pietro, «Docker ecosystem – Vulnerability Analysis,» *Computer Communications,* vol. 122, pp. 30-43, 2018.

[23] A. Potdar, D. Narayan , S. Kengond y M. Moin, «Performance Evaluation of Docker Container and Virtual Machine,» *Procedia Computer Science,* vol. 171, pp. 1419-1428, 2020.

[24] N. Marathe, A. Gandhi y J. Shah, «Docker Swarm and Kubernetes in Cloud Computing Environment,» de *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, India, 2019.

[25] P. Ganney, S. Pisharody y E. Claridge, «Software Engineering,» de *Clinical Engineering*, London, 2014, pp. 133-170.

[26] J. Verona, Practical DevOps, Birmingham: Packt Publishing, 2016.

[27] M. Soni, DevOps for Web Development, Birmingham: Packt Publishing, 2016.

[28] L. Bass, P. Clements y R. Kazman, Software Architecture in Practice, Westford: Addison-Wesley, 2013.

[29] M. Richards, Software Architecture Patterns, Sebastopol: O'Reilly Media, Inc, 2015.

[30] S. Farshidi, S. Jansen y J. Martijn, «Capturing software architecture knowledge for pattern-driven design,» *The Journal of Systems & Software,* vol. 169, pp. 1-18, 2020.

[31] A. Sharma, M. Kumar y S. Agarwal, «A Complete Survey on Software Architectural Styles and Patterns,» *4th International Conference on Eco-friendly Computing and Communication Systems, ICECCS,* vol. 70, pp. 16-28, 2015.

[32] P. Seda, P. Masek, J. Sedova, M. Seda, J. Krejci y J. Hosek, «Efficient Architecture Design for Software as a Service in Cloud Environments,» *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT),* pp. 1-6, 2018.

[33] A. Sarkar y A. Shah, Learning AWS : Design, Build, and Deploy Responsive Applications Using AWS Cloud Components, Birmingham: Packt Publishing, 2018.

[34] Y. Raheja, G. Borgese y N. Felsen, Effective DevOps with AWS : Implement Continuous Delivery and Integration in the AWS Environment, Birmingham: Packt Publishing, 2018.

[35] A. Alshazly, M. ElNainay, A. El-Zoghabi y M. Abougabal, «A cloud software life cycle process (CSLCP) model,» *Ain Shams Engineering Journal,* pp. 1-14, 2020.

[36] A. Banijamali, O.-P. Pakanen, P. Kuvaja y M. Oivo, «Software architectures of the convergence of cloud computing and the Internet of Things: A systematic literature review,» *Information and Software Technology,* vol. 122, pp. 1-24, 2020.

[37] S. Aleem, R. Batool, F. Ahmed y A. Masood, «Design guidelines for SaaS development process,» *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON),* pp. 825-831, 2018.

[38] S. Imenda, «Is There a Conceptual Difference between Theoretical and Conceptual Frameworks?,» *Journal of Social Sciences,* vol. 38, nº 2, pp. 185-195, 2014.

[39] J. Cito, P. Leitner, T. Fritz y H. Gall, «The Making of Cloud Applications – An Empirical Study on Software Development for the Cloud,» *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering,* pp. 393-403, 2015.

[40] «PaaS - Black or White: An Investigation into Software Development Model for Building Retail Industry SaaS,» *2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion,* pp. 285-287, 2017.

[41] S. Nalchigar y E. Yu, «Business-driven data analytics: A conceptual modeling framework,» *Data & Knowledge Engineering,* vol. 117, pp. 1-14, 2018.

[42] A. Rodrigues, «Model-driven engineering: A survey supported by the unified conceptual model,» *Computer Languages, Systems & Structures,* vol. 43, nº 1477-8424, pp. 139-155, 2015.

[43] K. Olmos-Sanchez y J. Rodas-Osollo, «Requirements Engineering Based on Knowledge Management: Theoretical Aspects and a Practical Proposal,» *International Journal of Software Engineering and Knowledge Engineering,* vol. 27, nº 08, pp. 1199-1233, 2017.

[44] A. Olivé, Conceptual Modeling of Information Systems, Barcelona: Springer, 2007.

[45] A. Kropp y R. Torre, «Docker: containerize your application,» de *Computing in Communication Networks From Theory to Practice*, Dresden, Computing in Communication Networks, 2020, pp. 231-244.

[46] N. Haile y J. Altmann, «Evaluating investments in portability and interoperability between,» *Future Generation Computer Systems,* vol. 78, pp. 224-241, 2018.

[47] E. Loukisa, M. Janssenb y I. Mintchev, «Determinants of software-as-a-service benefits and impact on firm performance,» *Decision Support Systems,* vol. 117, pp. 38-47, 2019.

[48] G. Aceto, V. Persico y A. Pescapé, «Industry 4.0 and Health: Internet of Things, Big Data, and Cloud Computing for Healthcare 4.0,» *Journal of Industrial Information Integration,* vol. 18, pp. 1-13, 2020.

[49] P. Modisane y O. Jokonya, «Evaluating the benefits of Cloud Computing in Small, Medium and Micro-sized Enterprises (SMMEs),» *Procedia Computer Science,* vol. 181, pp. 784-792, 2021.