

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática de Software

Trabajo Fin de Grado

Cloud Computing y migración de una aplicación de riego a AWS

Jesús Barquero Cuadrado

Julio, 2023

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática de Software

Trabajo Fin de Grado

Cloud Computing y migración de una aplicación de
riego a AWS

Autor: Jesús Barquero Cuadrado

Tutor: Pedro José Clemente Martín

Índice

1. Resumen.....	10
2. Introducción	11
2.1. Objetivos	13
2.2. Alcance.....	14
2.3. Planificación	15
3. Estado del arte	17
3.1. Virtualización	18
3.1.1. Virtualización hace unos años.....	18
3.1.2. Ventajas de la virtualización.....	19
3.1.3. Hipervisores.....	20
3.2. Contenedores	22
3.2.1. Docker	22
3.2.2. Arquitectura Docker.....	23
3.2.3. Construcción de imágenes con Dockerfiles	25
3.2.4. Docker Compose.....	26
3.2.5. Máquinas virtuales vs. Contenedores	27
3.3. Computación en la nube o “Cloud Computing”	29
3.3.1. Modelo Cloud	29
3.3.2. Tipos de nube o modelos de implementación de la nube.....	30
3.3.3. Niveles o modelos de servicio.....	32
3.3.4. Proveedores de nube	34
3.4. Amazon Web Services	36
3.4.1. Características de AWS	36
3.4.2. Infraestructura global.....	38
3.4.3. Precios de AWS	39
3.4.4. Servicios AWS.....	41
3.5. Herramientas de infraestructura como código o “IaC”	49
3.5.1 Importancia de las IaC	50
3.5.2. Terraform como herramienta de IaC	50
4. Refactorización de una aplicación web basada en microservicios para su despliegue en cloud.....	52
4.1. Diagrama de la arquitectura	53
4.1.1. Servicio Front-End.....	53
4.1.2. Servicios API REST y sus bases de datos	57
4.2. Estructura del código y tecnologías de la aplicación web	57
4.2.1. Angular	57
4.2.2. Spring Boot.....	60
4.2.3. Tipos de bases de datos utilizadas	64
5. Propuesta de arquitectura en Cloud	68
5.1. Diagrama de la arquitectura Cloud	68
5.2. Razonamiento detrás de la elección de AWS	69
5.3. Explicación de la arquitectura.....	69
5.4. Automatización del despliegue con Terraform	72
6. Desarrollo	73

6.1. Refactorización de la aplicación para el despliegue de los microservicios en contenedores Docker	73
6.1.1. Modificación de las URL de conexión y archivos de propiedades para la configuración de la aplicación	75
6.1.2. Creación de los Dockerfiles.....	84
6.1.3. Construcción de las imágenes	89
6.1.4. Definición del lanzamiento de contenedores con Docker-Compose	89
6.1.5. Datos y estructuras de las bases de datos al lanzarse la aplicación.....	96
6.1.6. Subida de imágenes al repositorio de Docker-Hub.....	97
6.2. Migración y despliegue de la infraestructura en AWS.....	98
6.2.1. Infraestructura de red	100
6.2.2. Servicios de almacenamiento	115
6.2.3. EC2 Bastión o máquina administradora.....	122
6.2.4. Servicios de bases de datos: RDS y EC2	125
6.2.5. Clúster ECS y despliegue de microservicios	137
6.2.6. EC2 Front-End	144
6.2.7. Balanceador de carga ALB	147
6.3. Transformación IaC de la infraestructura a través de Terraform	153
6.3.1. Estructura del proyecto de Terraform	153
6.3.2. Detalles de implementación de los ficheros Terraform	154
7. Procedimiento para la ejecución de la aplicación local y de la infraestructura de Terraform.....	168
7.1. Aplicación local	168
7.2. Infraestructura de Terraform	169
8. Pruebas.....	172
8.1. Funcionalidad auxiliar de comprobación de estado de las API REST	172
8.2. Comprobación del despliegue	173
8.2.1. Despliegue local de contenedores	173
8.2.2. Despliegue de servicios en AWS	180
8.2.3. Despliegue con Terraform de los servicios AWS.....	195
8.3. Comprobación de la funcionalidad de las APIs desplegadas	213
8.3.1. API del servicio Sensor	213
8.3.2. API del servicio Plots.....	216
8.3.3. API del servicio Statistics	218
8.3.4. API del servicio Authentication.....	219
9. Lecciones aprendidas y errores	221
10. Conclusiones	223
11. Trabajos futuros.....	224
12. Manual de instalación y de usuario.....	225
12.1. Creación de la cuenta de AWS	225
12.2. Instalación Terraform	226
12.2.1. Creación de un usuario con credenciales AWS para Terraform.....	227
12.3. Instalación de Docker	229
12.4. Creación de cuenta en DockerHub	230
13. Bibliografía	231

Índice de ilustraciones

Ilustración 1 - Planificación Gantt	16
Ilustración 2 - Antes y después de la virtualización (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2016a)	19
Ilustración 3 - Hipervisores de tipo 1 y 2 (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2016b).....	21
Ilustración 4 - Arquitectura Docker (Muñoz, 2016).....	23
Ilustración 5 - Fases construcción de un contenedor (JFrog, 2021)	25
Ilustración 6 - Máquinas virtuales y su hipervisor (Docker, 2021)	27
Ilustración 7 - Contenedores sobre el sistema operativo (Docker, 2021)	28
Ilustración 8 - Niveles servicio en la nube (Nazir et al., 2020)	34
Ilustración 9 - Cuadrante mágico de Gartner (Gartner, 2022).....	35
Ilustración 10 - Infraestructura global AWS (Amazon Web Services, Inc., 2014b)	38
Ilustración 11 - Zonas disponibilidad (José Manuel M., 2019).....	39
Ilustración 12 - Capa gratuita (Amazon Web Services, Inc., 2013).....	41
Ilustración 13 - Tipos de servicios AWS (Medrano, 2022).....	41
Ilustración 14 – Diagrama VPC (Amazon Web Services, Inc., 2022a)	42
Ilustración 15 - Destino de montaje EFS (Amazon Web Services, Inc., 2015)	46
Ilustración 16 – ECS AWS (Amazon Web Services, Inc., 2023b)	48
Ilustración 17 – ALB AWS (Amazon Web Services, Inc., 2016).....	49
Ilustración 18 - Diagrama de arquitectura de la aplicación base	53
Ilustración 19 - Página inicial aplicación	54
Ilustración 20 - Parcela catastro	54
Ilustración 21 - Parcela inicial página	55
Ilustración 22 – Login	55
Ilustración 23 – Lista de parcelas	56
Ilustración 24 – Lista de sensores.....	56
Ilustración 25 – Visualización de estadísticas de datos de los sensores	56
Ilustración 26 - Estructura proyecto “tfm-front-end”	58
Ilustración 27 - Fichero “plot-service.service.ts”	59
Ilustración 28 - Dependencias "node_modules"	59
Ilustración 29 - Fichero "package.json".....	60
Ilustración 30 - Properties sensor.....	61
Ilustración 31 - Properties plots	62
Ilustración 32 – Estructura del proyecto plots	62
Ilustración 33 - Clase plot.....	63
Ilustración 34 - Repositorio plot	63
Ilustración 35 – Servicio de plot implementado	63
Ilustración 36 - Controlador plot	64
Ilustración 37 - Triángulo CAP (Mason, 2013)	66
Ilustración 38 - Propuesta de arquitectura cloud.....	68
Ilustración 39 – Ficheros service de “tfm-front-end”	77
Ilustración 40 - Index.html	79
Ilustración 41 - Carpeta "application.properties" de Plots	81
Ilustración 42 - Carpeta "libs" con el archivo "jar".....	85

Ilustración 43 - Carpeta ficheros Cassandra	87
Ilustración 44 - Lista de imágenes Docker.....	89
Ilustración 45 - Estructura Docker-Compose.....	90
Ilustración 46 - Ejecución de contenedores local y lista de volúmenes.....	96
Ilustración 47 - Perfil Docker-Hub	97
Ilustración 48 - Creación de par de claves.....	100
Ilustración 49 - VPC AWS.....	101
Ilustración 50 - Configuración DNS	101
Ilustración 51 - Tabla VPC	102
Ilustración 52 – IGW AWS	102
Ilustración 53 - Subredes AWS	103
Ilustración 54 - IP elástica AWS.....	105
Ilustración 55 - NAT AWS	106
Ilustración 56 - Tablas AWS.....	107
Ilustración 57 - Tabla internet AWS	107
Ilustración 58 - Asociacion tabla internet aws.....	108
Ilustración 59 - Tablas databases AWS.....	108
Ilustración 60 - Asociación tabla database AWS.....	109
Ilustración 61 - Bastión sg AWS	110
Ilustración 62 – Servicios sg AWS.....	111
Ilustración 63 - Front sg AWS	112
Ilustración 64 - Databases sg AWS.....	113
Ilustración 65 - NFS sg AWS	114
Ilustración 66 - ALB sg AWS	115
Ilustración 67 - Punto s3 AWS	116
Ilustración 68 - Ruta s3 tabla Internet AWS.....	117
Ilustración 69 - Ruta s3 tabla databases AWS	117
Ilustración 71 - Bucket s3 carpeta cassandra AWS.....	118
Ilustración 70 - Bucket s3 AWS.....	118
Ilustración 73 - Bucket s3 carpeta cql AWS.....	119
Ilustración 72 - Bucket s3 carpeta bash AWS.....	119
Ilustración 74 - Rol s3 AWS.....	120
Ilustración 75 - EFS AWS.....	121
Ilustración 76 - Destino montaje AWS	121
Ilustración 77 - EC2 Bastión.....	122
Ilustración 78 - SSH EC2 Bastión.....	125
Ilustración 79 - Comprobación directorios EFS.....	125
Ilustración 80 - Comando "df"	125
Ilustración 81 - Grupo subredes AWS.....	127
Ilustración 82 - RDS posgres AWS	128
Ilustración 83 - RDS mysql AWS	129
Ilustración 84 - EC2 mongo AWS.....	130
Ilustración 85 - Carpeta cassandra AWS.....	133
Ilustración 86 - EC2 cassandra AWS.....	133
Ilustración 87 - Clúster ECS AWS	138
Ilustración 88 - Instancias ECS AWS	139
Ilustración 89 - Instancia ECS 1	140

Ilustración 90 - Instancia EC2 de ECS (1/2)	140
Ilustración 91 - Instancia ECS 2	141
Ilustración 92 - Instancia EC2 ECS (2/2)	141
Ilustración 93 - Lista tareas AWS	142
Ilustración 94 - Env vars AWS.....	142
Ilustración 95 - Colocación tareas AWS	144
Ilustración 96 - EC2 Front AWS.....	145
Ilustración 97 - Grupos destino AWS.....	147
Ilustración 98 - ALB AWS.....	151
Ilustración 99 - Listeners ALB AWS.....	152
Ilustración 100 - Estructura Terraform.....	154
Ilustración 101 - Terraform init	169
Ilustración 102 - Terraform validate.....	170
Ilustración 103 - Terraform plan	170
Ilustración 104 - Terraform apply	170
Ilustración 105 - Terraform destroy	171
Ilustración 106 - Comprobación de contenedores en ejecución	173
Ilustración 107 - Comprobación de volúmenes	173
Ilustración 108- Logs Front local	174
Ilustración 109 - Front en navegador.....	175
Ilustración 110 - Logs Plots local	175
Ilustración 111 - Consulta mongo local (1/2).....	176
Ilustración 112 - Consulta mongo local (2/2).....	176
Ilustración 113 - Logs Sensor local	177
Ilustración 114 - Consulta postgres local	177
Ilustración 115 - Logs Authentication local.....	178
Ilustración 116 - Consulta mysql local.....	179
Ilustración 117 - Logs Statistics local	179
Ilustración 118 - Consulta cassandra local (1/2).....	180
Ilustración 119 - Consulta cassandra local (2/2).....	180
Ilustración 120 - Conexión bastión AWS	181
Ilustración 121 - Consulta postgres AWS	181
Ilustración 122 - Consulta mysql AWS.....	182
Ilustración 123 - Conexión SSH bastión con par de claves	183
Ilustración 124 - Conexión ec2 mongo	183
Ilustración 125 - Consulta mongo AWS (1/2).....	183
Ilustración 126 - Consulta mongo AWS (2/2).....	184
Ilustración 127 - Mongo volumen	184
Ilustración 128 - Mongo EFS.....	185
Ilustración 129 - Conexión ec2 cassandra	185
Ilustración 130 - Consulta cassandra AWS.....	185
Ilustración 131 - Cassandra volumen	186
Ilustración 132 - EFS Cassandra	186
Ilustración 133 – Docker ps y logs Front (1/2)	187
Ilustración 134 – Docker ps y logs Front (2/2)	187
Ilustración 135 - Front AWS navegador	188
Ilustración 136 - Panel instancias ECS.....	188

Ilustración 137 - Panel tareas ECS.....	189
Ilustración 138 - Docker ps instancia ECS 1.....	189
Ilustración 139 - Plot logs AWS	190
Ilustración 140 - Docker ps instancia ECS 2.....	190
Ilustración 141 - Logs sensor AWS	191
Ilustración 142 - Logs Auth AWS (1/2)	191
Ilustración 143 - Logs Auth AWS (2/2)	192
Ilustración 144 - Logs Stats AWS.....	192
Ilustración 145 - Destino front	193
Ilustración 146 - Destino plots.....	193
Ilustración 147 - Destino sensor.....	194
Ilustración 148 - Destino Authentication	194
Ilustración 149 - Destino statistics	195
Ilustración 150 - VPC tf.....	195
Ilustración 151 - IGW tf	196
Ilustración 152 - Subredes tf.....	196
Ilustración 153 - IP Elástica tf.....	197
Ilustración 154 - NAT tf	197
Ilustración 155 - Tabla 1 tf	198
Ilustración 156 - Tabla 2 tf	198
Ilustración 157 - Tabla 3 tf	198
Ilustración 158 - Asociación 1 tf.....	198
Ilustración 159 - Asociación 2 tf	199
Ilustración 160 - Punto enlace S3 tf.....	199
Ilustración 161 – Grupo seguridad 1 tf.....	199
Ilustración 162 - Grupo seguridad 2 tf	200
Ilustración 163 - Grupo seguridad 3 tf	200
Ilustración 164 - Grupo seguridad 4 tf	200
Ilustración 165 - Grupo seguridad 5 tf	200
Ilustración 166 - Grupo seguridad 6 tf	201
Ilustración 167 - Grupo seguridad 7 tf	201
Ilustración 168 - Bucket S3 1 tf.....	201
Ilustración 169 - Bucket S3 2 tf.....	202
Ilustración 170 - Bucket S3 3 tf.....	202
Ilustración 171 - Bucket S3 4 tf.....	202
Ilustración 172 - Rol S3 tf	203
Ilustración 173 - Rol ECS tf	203
Ilustración 174 - EFS tf.....	204
Ilustración 175 - Destino montaje tf.....	204
Ilustración 176 - Instancias EC2 tf.....	205
Ilustración 177 - EC2 1 tf	205
Ilustración 178 - EC2 2 tf	205
Ilustración 179 - EC2 3 tf	206
Ilustración 180 - EC2 4 tf	206
Ilustración 181 - EC2 5 tf	206
Ilustración 182 - EC2 6 tf	207
Ilustración 183 - RDS mysql tf	207

Ilustración 184 - RDS postgres tf.....	208
Ilustración 185 - Grupo subred tf	208
Ilustración 186 - ECS tf	209
Ilustración 187 - Tareas tf	209
Ilustración 188 - ALB tf	210
Ilustración 189 - Listeners ALB tf.....	210
Ilustración 190 - Grupos destino tf.....	211
Ilustración 191 - Grupo 1 tf (1/2)	211
Ilustración 192 - Grupo 1 tf (2/2)	211
Ilustración 193 - Grupo 2 tf (1/2)	211
Ilustración 194 - Grupo 2 tf (2/2)	212
Ilustración 195 - Grupo 3 tf (1/2)	212
Ilustración 196 - Grupo 3 tf (2/2)	212
Ilustración 197 - Grupo 4 tf (1/2)	212
Ilustración 198 - Grupo 4 tf (2/2)	213
Ilustración 199 - Grupo 5 tf (1/2)	213
Ilustración 200 - Grupo 5 tf (2/2)	213
Ilustración 201 - sensor addSensor	214
Ilustración 202 - sensor getSensorList 1.....	214
Ilustración 203 - sensor getSensorList 2.....	215
Ilustración 204 - sensor getSensorBySensorId	215
Ilustración 205 - plots getPlotsByOwner.....	216
Ilustración 206 - plots getPlotsByCadastralReference.....	217
Ilustración 207 - plots addPlotByReference	217
Ilustración 208 - plots checkOwnerRequest.....	218
Ilustración 209 - statistics operación	219
Ilustración 210 - authentication createAuthenticationToken	219
Ilustración 211 - authentication RequestBody.....	220
Ilustración 212 - Inicio registro aws	225
Ilustración 213 - Registro en aws	226
Ilustración 214 - Instalación Terraform.....	227
Ilustración 215 - Terraform version.....	227
Ilustración 216 - IAM aws.....	228
Ilustración 217 - Usuario	228
Ilustración 218 - Nombre usuario y clave.....	228
Ilustración 219 - Permisos administrador	229
Ilustración 220 - Credenciales Terraform AWS	229
Ilustración 221 - Instalación Docker.....	230
Ilustración 222 - Cuenta Docker-Hub	230

1. Resumen

Actualmente se podría decir que nos encontramos en uno de los puntos de mayor auge en cuanto a la utilización de las tecnologías de computación en la nube, sobre todo por parte de las organizaciones del sector TI.

Las empresas proveedoras dedicadas a la computación en la nube ponen a disposición de sus clientes recursos virtuales tanto de red, hardware, software o de infraestructura informática sin necesidad de que estos tengan que interactuar con los recursos físicamente, solo a través de Internet y bajo demanda. Esto ha permitido a las empresas dedicadas al desarrollo de software liberarse sustancialmente de la carga que supone el mantenimiento y actualización de su propia infraestructura física local, junto con la posibilidad de aprovechar este conjunto flexible de recursos para migrar sus aplicaciones, inicialmente desplegadas en entornos locales, a una infraestructura cloud mucho más potente, lo que lleva a una mejora de rendimiento y eficiencia que se traducirá en un servicio de calidad para los clientes.

Para reflejar el proceso que sigue una aplicación hasta su funcionamiento en la nube, este Trabajo de Fin de Grado tiene como objetivo realizar la migración de una aplicación web inicial ya desarrollada, basada en microservicios y desplegada en un entorno local hacia una infraestructura en la nube que se construirá haciendo uso de los servicios de Amazon Web Service (AWS). El proceso de desarrollo seguido para realizar esta tarea se estructurará en tres fases: refactorizar la aplicación para su funcionamiento a través de contenedores Docker, migración de la aplicación refactorizada hacia una infraestructura construida en AWS y finalmente automatizar el proceso de despliegue de la infraestructura a través de ficheros Terraform.

2. Introducción

Según el Instituto Nacional de Estándares y Tecnología (NIST) de Estados Unidos, la computación en la nube o cloud computing, “es un modelo que permite el acceso a la red de forma adecuada, en cualquier lugar y bajo demanda, con el fin de compartir recursos de cómputo con un esfuerzo mínimo de administración o interacción del proveedor del servicio” (Mell & Grance, 2011). Dicho de otra manera, es la entrega de recursos tecnológicos bajo demanda, ya sean aplicaciones, datos, almacenamiento o servicios compartidos a través de Internet con la filosofía de pagar únicamente por el uso de los recursos. Entre las empresas proveedoras más importantes que suministran servicios en la nube están Amazon Web Service, Microsoft Azure y Google.

Uno de los problemas principales a los que se enfrentan las empresas de desarrollo software tradicionales, además de preocuparse por ofrecer el mejor servicio posible a sus clientes, es el coste económico que supone encargarse directamente del mantenimiento, administración, configuración y actualización de sus propios servidores e infraestructura física que está detrás de las aplicaciones. Además, esta tarea exige de un equipo de profesionales dedicados a la resolución de los problemas que pudieran surgir a nivel técnico, actualizaciones, averías... (Salesforce, 2017). A este gasto económico también se incluye una situación muy común dada en este tipo de empresas, y es la sustitución de los equipos locales que van quedando obsoletos con paso de los años por otras máquinas más potentes.

Debido a la problemática explicada anteriormente, existe una demanda creciente de infraestructura, ya sea a través de recursos hardware o equipos virtualizados, que permitan el despliegue de aplicaciones a gran escala, con unos costos reducidos y que garantice una alta disponibilidad de servicio (Izaguirre Tapia, 2022). La computación en la nube es muy atractiva para las empresas por varias razones económicas y a nivel de desarrollo de aplicaciones: requiere una inversión de infraestructura muy baja al no ser necesario construir y mantener un sistema físico a gran escala. Las máquinas virtualizadas proporcionadas por los proveedores en la nube serán mucho más potentes, mantendrán unos costos de computación basados en el uso muy bajos y los clientes solo pagarán por la infraestructura utilizada. Al mismo tiempo, el tiempo de ejecución de las aplicaciones que demandan un procesamiento intensivo a nivel de datos y operaciones se reduce drásticamente debido a la computación paralela y distribuida que ofrece el cloud con la posibilidad de utilizar tantos servidores como sea necesario para que las aplicaciones puedan responder de manera óptima a los costos y las restricciones de tiempo. Además, los desarrolladores, al disponer a una

infraestructura cloud flexible, también disfrutan de la posibilidad de poder diseñar y desplegar infraestructuras a través herramientas de infraestructura como código de manera automatizada para crear diversos entornos de desarrollo, pruebas o producción donde pueden observar cómo se comportarán las aplicaciones en sus diversas fases de desarrollo (Marinescu, 2013).

Una de las soluciones más utilizadas por las empresas para implantar el uso de la computación en la nube es a través de la migración o traslado de las aplicaciones, datos e infraestructura local actual directamente hacia un entorno cloud. Es por este motivo que, el presente Trabajo de Fin de Grado, realizará una propuesta de migración de una aplicación desplegada en un entorno local hacia una infraestructura en la nube, como forma de abordar una situación real a la que se enfrentan las empresas actualmente. La aplicación inicial que se busca migrar ha sido proporcionada por el tutor del TFG y consiste en una aplicación web basada en microservicios con una temática agrícola en el que se busca controlar el riego a través de sensores sobre cultivos situados en parcelas. El proveedor cloud utilizado será Amazon Web Service (Amazon Web Services, Inc., 2023a) para levantar toda la infraestructura de red, servicios almacenamiento, instancias... y demás recursos para una correcta migración de la aplicación. Esta tarea se realizará a través de un proceso de desarrollo que comenzará con la refactorización del código de aplicación, empaquetando los microservicios para su funcionamiento a través de contenedores Docker (Docker, 2022) dentro del equipo local. Después, se construirá la infraestructura de AWS para migrar la aplicación y sus microservicios dentro de esta. Finalmente, la infraestructura se podrá replicar automatizando el proceso de despliegue a través de Terraform (Terraform by HashiCorp, 2023), que será la herramienta de infraestructura como código escogida para cumplir con este cometido.

2.1. Objetivos

El **objetivo principal** de este proyecto es construir una infraestructura en la nube utilizando AWS y documentar todo el proceso seguido para la migración de una aplicación web hacia un entorno cloud. Esta infraestructura resultante podrá utilizarse para el despliegue tanto de la aplicación de la que se dispone como de otras con una estructura similar basada en microservicios.

Este objetivo a su vez puede desglosarse en los siguientes objetivos específicos:

1. **Estudio de una aplicación web basada en microservicios:** este será primer paso antes de comenzar con la construcción de la infraestructura, puesto que será esencial conocer primero cómo se comporta la aplicación desplegada de forma local para poder conocer qué recursos cloud se deberán definir para su funcionamiento en la nube. Se estudiarán las tecnologías implicadas en el funcionamiento de la aplicación inicial, entre las que se encuentran servicios API REST basados en el framework de Spring Boot, un servicio Front-End basado en Angular y diversos tipos de bases de datos como MySQL, PostgreSQL, MongoDB y Cassandra.
2. **Refactorización del código de aplicación local para el despliegue de sus microservicios a través de contendores Docker:** algunos de los microservicios de la aplicación ya contaban con ficheros Dockerfile y Docker-Compose para su funcionamiento a través de contenedores puesto que forma parte de su diseño inicial. Por lo tanto, se entiende como refactorización tanto a la preparación de las partes funcionales del código como a la modificación de las direcciones de conexión entre los servicios para adaptarse a esta tecnología de contenedores. Además, se elaborarán los ficheros Dockerfile restantes junto a los ficheros de configuración y scripts necesarios para preparar el despliegue de la aplicación tanto a nivel local como en la nube.
3. **Creación de una infraestructura AWS:** se deberá escoger, configurar y comprobar el despliegue de los diferentes servicios y recursos de AWS necesarios para ejecutar una aplicación basada en microservicios. Esto implica la creación de las redes, subredes, sistemas de almacenamiento, máquinas virtuales, sistemas de bases de datos...

4. **Automatización del despliegue de la infraestructura a través de ficheros Terraform:** dentro de estos ficheros de código estarán definidos todos los recursos que conformarán la infraestructura creada para poder replicar su despliegue de forma automatizada.
5. **Documentación del proceso de migración y lecciones aprendidas:** todo el proceso seguido para la migración y despliegue de la aplicación quedará documentado en la fase de desarrollo y pruebas del proyecto. Adicionalmente, se añadirá un apartado con los aprendizajes y errores cometidos durante la elaboración del proyecto a nivel organizativo junto con recomendaciones para la migración y despliegue de aplicaciones basadas en microservicios.

2.2. Alcance

Lo principal es que la infraestructura de AWS sea funcional, comprobando que todos sus componentes estén operativos y que la aplicación refactorizada pueda, por lo menos, ejecutarse correctamente en forma de contenedores dentro de esta. No se modificarán ni se implementarán nuevas funcionalidades en el código de la aplicación, solo se modificarán partes necesarias como las URL de conexión o ficheros de propiedades para que puedan funcionar correctamente tanto en el entorno local como en la nube.

La aplicación inicial no está desarrollada por completo ni se dispone de ningún tipo de documentación auxiliar, por lo que muchas partes no están implementadas del todo o pueden no funcionar como deberían. En consecuencia, el despliegue de la infraestructura primará sobre los posibles errores provocados por funcionalidades concretas de la aplicación, las cuales no tendrán un peso importante en el proyecto.

Entre las funcionalidades de la aplicación web original, esta encuentra con un sistema en tiempo real para recoger la información de riego forma más rápida y precisa. Esta parte no estará incluida en el proyecto, ya que añade una complejidad extra debido a la utilización de tecnologías como MQTT, Kafka, Apache Spark... las cuales no aportan valor a los objetivos del proyecto, donde se destaca la infraestructura por encima de otras funcionalidades extra. Solo se trabajará con la aplicación base.

2.3. Planificación

El proyecto se ha desarrollado en un periodo de tiempo de un año de duración:

- **Fecha de inicio:** Abril 2022.
- **Fecha de fin:** Junio 2023.

Se hará uso de un “Diagrama de Gantt” para visualizar las fases implicadas en la elaboración del proyecto a lo largo de los meses de una manera gráfica. Cada fase tendrá una serie de tareas que se han realizado en dicha fase (Ilustración 1):

- **Estudio inicial**
 - Recogida de los requisitos iniciales del proyecto tras las primeras reuniones con el tutor del Trabajo de Fin de Grado.
 - Estudio del código de la aplicación base.
 - Investigación de todas las tecnologías que componen la aplicación.
 - Estudio de los diferentes servicios de AWS para la construcción de la infraestructura.
- **Pruebas con la aplicación inicial y AWS**
 - Pruebas de ejecución en local de la aplicación.
 - Estudio del uso de contenedores a través de Docker.
 - Aprender a trabajar con la infraestructura de AWS.
- **Puesta en marcha de la aplicación base**
 - Refactorización y despliegue parcial de la aplicación base utilizando contenedores.
 - Selección de las partes funcionales de la aplicación con las que se trabajará.

- Despliegue parcial de la aplicación base en AWS.
 - Documentar los conceptos teóricos y prácticos aprendidos hasta el momento.
- **Corrección de errores**
- Corrección de los errores que surgen durante el proceso de refactorización de la aplicación.
 - Corregir los errores durante el despliegue parcial de la aplicación en AWS a nivel de infraestructura.
- **Despliegue completo y documentación final**
- Despliegue total de la aplicación en local.
 - Despliegue total de la infraestructura funcional de AWS.
 - Elaboración de los ficheros Terraform para automatizar el despliegue.
 - Finalización de la documentación.



Ilustración 1 - Planificación Gantt

3. Estado del arte

Este apartado abordará el estado del arte actual de la computación en la nube a lo largo de las siguientes secciones:

3.1. Virtualización: definición del concepto de virtualización, cómo aparece su uso en el ámbito empresarial, ventajas de su utilización y explicación de los hipervisores que permiten la virtualización.

3.2. Contenedores: definición de contenedor, qué es Docker, su arquitectura, cómo se construyen las imágenes a partir de ficheros Dockerfile, qué es un fichero Docker-Compose y las diferencias de los contenedores con respecto a las máquinas virtuales.

3.3. Computación en la nube o “Cloud Computing”: concepto de computación en la nube, cómo aparece este modelo de computación en la nube, tipos de nube, tipos de servicio y empresas proveedoras principales.

3.4. Amazon Web Services: qué es AWS, sus características, cómo es su infraestructura física global, precios por utilización de los servicios y presentación de los servicios de AWS utilizados en la fase de desarrollo del proyecto para la construcción de la infraestructura en la nube.

3.5. Herramientas de infraestructura como código o “IaC”: definición de las herramientas “IaC”, la importancia de su uso al trabajar con infraestructuras en la nube y presentación de Terraform al ser la herramienta que se utilizará para automatizar el despliegue de la infraestructura construida en AWS.

Antes de abordar el concepto de computación en la nube dentro de este estado del arte, es necesario entender qué es la virtualización puesto que, gracias al desarrollo de esta tecnología, ha hecho posible que se pueda disponer de todos los servicios, recursos e infraestructura informática que proporciona en la nube a través de la red.

[3.1. Virtualización](#)

La virtualización se puede definir como la tecnología que representa la abstracción o representación lógica de recursos físicos. Permite la creación de versiones virtuales de recursos o dispositivos hardware, ya sea un servidor, un sistema operativo, un dispositivo de almacenamiento o una red para ser utilizados a través de máquinas virtuales. Además, es posible ejecutar varias máquinas virtuales, cada una con su correspondiente sistema operativo, dentro de una misma máquina, por lo que se logra un aprovechamiento eficaz del hardware (Kumar & Charu, 2015).

[3.1.1. Virtualización hace unos años](#)

Tradicionalmente, la práctica más común extendida en el ámbito empresarial de la informática era acumular gran cantidad de servidores físicos y pilas TI de un solo proveedor en los que se solían ejecutar una sola aplicación o servicio informático por cada servidor, donde los usuarios no tenían mucha posibilidad de trabajar en conjunto pudiendo acceder simultáneamente a una misma computadora. Esta última práctica fue evolucionando de forma más temprana con el uso de técnicas como la computación de tiempo compartido, permitiendo que una gran cantidad de usuarios pudieran interactuar y utilizar una computadora de forma simultánea, y más tarde con la aparición de sistemas operativos multiusuario y multitarea como Linux, basado en UNIX. Sin embargo, la virtualización se desarrollaría con más fuerza más adelante como una solución natural al problema que suponía tanto actualizar como renovar los equipos, el alto coste económico y el aprovechamiento de muchos recursos físicos de una manera ineficiente, por ejemplo, con la práctica de “un servidor, una aplicación” mencionada anteriormente (Red Hat, Inc, 2023). En la siguiente imagen (Ilustración 2) puede apreciarse el antes y después de utilizar la virtualización para alojar varias aplicaciones dentro de un mismo hardware a través de máquinas virtuales:

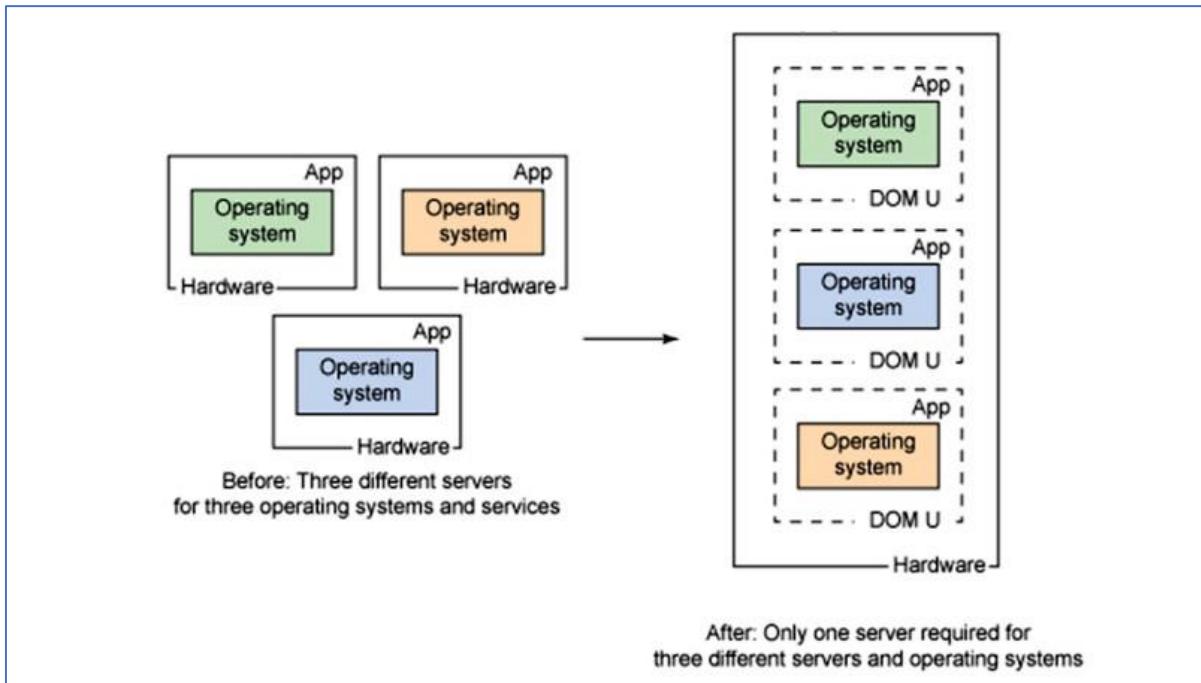


Ilustración 2 - Antes y después de la virtualización (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2016a)

3.1.2. Ventajas de la virtualización

Con el uso de la virtualización se puede interactuar de forma flexible con los recursos hardware. Los servidores físicos consumen electricidad, ocupan espacio de almacenamiento y necesitan mantenimiento. Muchas veces el acceso a estos está limitado por la proximidad física y el diseño de la red. La virtualización resuelve todas estas limitaciones con su capacidad de abstraer la funcionalidad del hardware físico en el software. Permite administrar, utilizar y mantener la infraestructura de hardware como una aplicación en la web. Para las organizaciones, proporciona beneficios como los siguientes (Amazon Web Services, Inc., 2022b):

- Utilización eficiente de los recursos y ahorro en los costes

Se aprovecha mucho mejor los recursos hardware disponibles. Por ejemplo, en vez de ejecutar un solo servidor en un equipo, ese mismo equipo puede ejecutar diversas máquinas virtuales con servidores al mismo tiempo. Esto consigue reducir el número de servidores físicos del centro de datos y disminuye los costes, ahorrando en electricidad, generadores, refrigeración... Ahora, tener diversas máquinas podrían aprovecharse para formar un clúster con un software que coordine las máquinas virtuales desplegadas en estas para garantizar alta disponibilidad de los servicios ante un fallo de hardware o software.

- Administración automatizada de las máquinas virtuales e infraestructuras

Con el uso de herramientas de software, facilita la tarea del administrador de sistemas al poder utilizar herramientas y la programación para definir plantillas que despliegan máquinas virtuales, incluso configurar sus entornos de una manera eficiente y automatizada. Esta automatización no se reduce solo a máquinas virtuales, sino también a infraestructuras enteras que se pueden desplegar de manera repetida, coherente, flexible ante cambios y con una configuración ya especificada previamente, evitando así errores manuales (IBM, 2021a).

- Gestión de recuperación ante desastres

La recuperación ante desastres causados por ataques cibernéticos o desastres naturales y meteorológicos, pueden inhabilitar de forma temporal o permanente la actividad de la empresa ante la posibilidad de que los centros de datos del área afectada queden inutilizados o pierdan toda la conectividad. Con la utilización de entornos virtualizados, el tiempo de respuesta ante estos eventos y la capacidad de recuperación mejoran enormemente permitiendo así la continuidad del negocio y de la actividad. Su aplicación permite que las imágenes de las máquinas virtuales se puedan desplegar y ejecutar de forma distribuida en otros centros de datos remotos debido a la portabilidad de estas, y utilizando un balanceo de carga entre las máquinas, se consigue la continuidad de servicio junto con la repartición del uso de recursos. También, los datos pueden estar replicados y distribuidos en diversas localizaciones para ser después reconstruidos en el sistema. Por último, las redes pueden ser reconfiguradas lógicamente, a mano o autónomamente a través de software, de forma remota sin necesidad de mover ningún tipo de cable en los centros de datos (VMware, Inc., 2020).

[3.1.3. Hipervisores](#)

La base para entender cómo funciona la virtualización es el hipervisor. Es el software de virtualización es un intermediario, una capa que está entre las máquinas virtuales y el hardware subyacente. Provee una interfaz que permite y coordina los accesos a los recursos físicos solicitados por parte de los servicios virtualizados para que cada uno obtenga lo que necesite, desde potencia de procesamiento y acceder a ficheros de los dispositivos de almacenamiento, como a la interfaz de red. Se conoce como anfitrión o “host” a la máquina donde se encuentra instalado el hipervisor (y a su sistema operativo como “sistema operativo host”) y se le llama huésped o “guest” a las

máquinas virtuales que se encuentran funcionando dentro del anfitrión. Se diferencian dos tipos de hipervisores (Eder, 2016):

- Tipo 1

Hipervisores que se ejecutan directamente en el hardware del sistema (Ilustración 3). También se conoce como virtualización de metal desnudo o “bare metal” para representar que no hay ningún software que se encuentre entre el hipervisor y el hardware. Algunos de los hipervisores de tipo 1 más relevantes son Hyper-V de Microsoft o XenServer.

- Tipo 2

Hipervisores que se ejecutan en un sistema operativo host (Ilustración 3). Aprovecha el sistema operativo para proporcionar utilidades como soporte a dispositivos de E/S, administración de memoria, gestión de procesos... propios del SO o incluso poder realizar “snapshots” (copias del estado del sistema) y proporcionar interfaces gráficas. Entre los más conocidos están KVM, implementada en el kernel de Linux, y VirtualBox de Oracle.

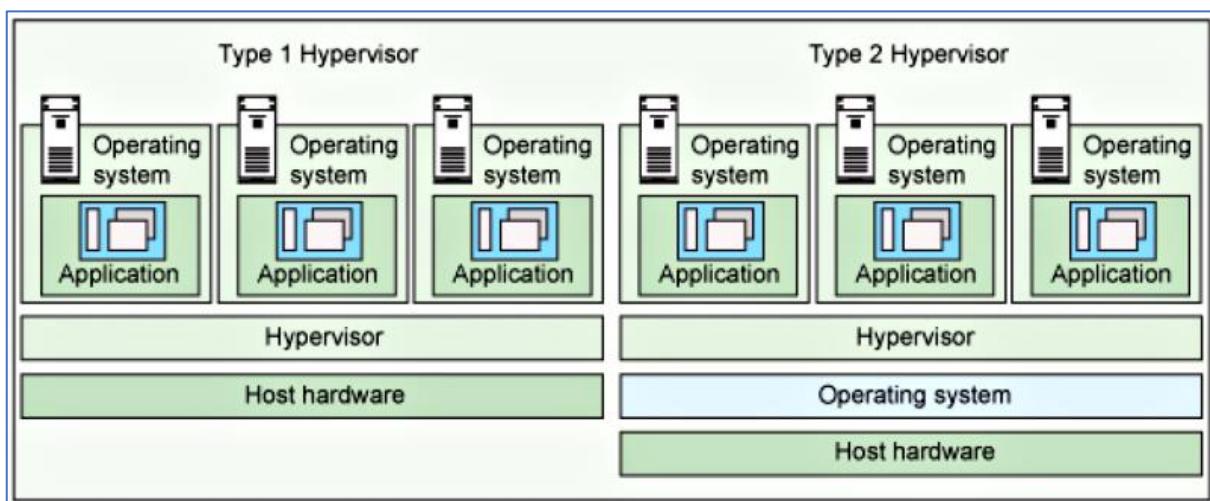


Ilustración 3 - Hipervisores de tipo 1 y 2 (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2016b)

Como se ha podido observar, la virtualización es la tecnología que hace posible la computación en la nube puesto que los servicios y recursos informáticos bajo demanda son suministrados por parte de los proveedores cloud a sus clientes gracias a ella. Estos proveedores se encargan de administrar sus propios centros de datos, y mediante la creación de entornos virtuales proporcionan servicios que satisfacen las necesidades de infraestructura mediante la utilización de los recursos hardware subyacentes de estos centros.

3.2. Contenedores

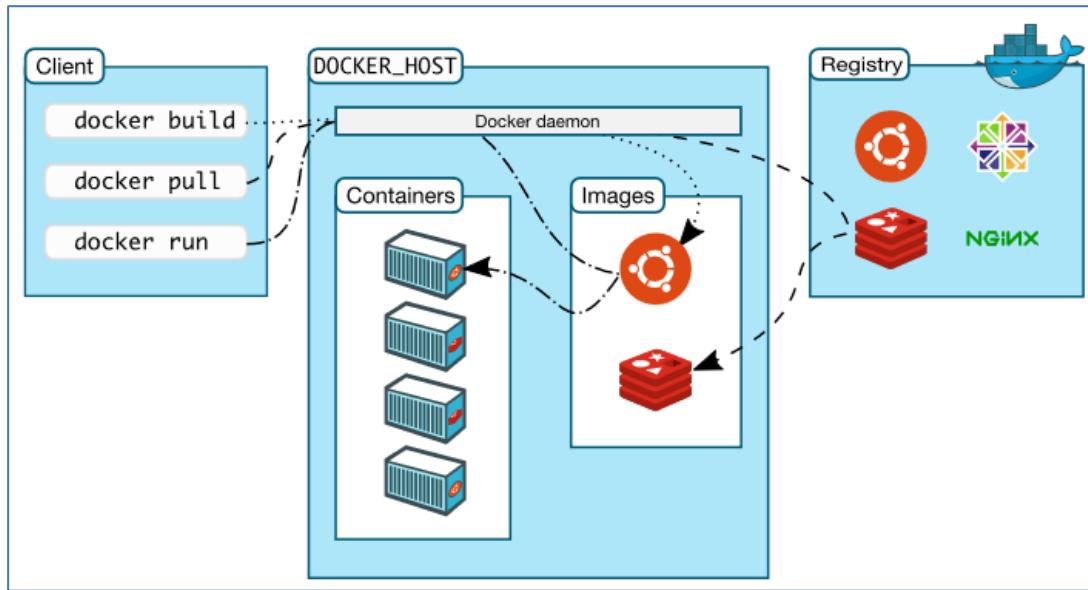
Los contenedores son un método utilizado en las tecnologías de desarrollo software muy usado desde hace ya varios años en las que las grandes empresas tecnológicas que ofrecen servicios de nube y virtualización como Microsoft, Amazon, Oracle, Google, IBM... han estado apostando muy fuerte. Es un tipo de virtualización en el nivel de sistema operativo o ligera, y que, combinado su uso junto a las infraestructuras en la nube, lleva el despliegue de aplicaciones a otro nivel siendo una de las herramientas más potentes actualmente.

Un contenedor es una unidad de software estandarizada que empaqueta el código de una aplicación junto a todas sus bibliotecas, archivos de configuración y dependencias que se necesitan para permitir su ejecución de una forma consistente, rápida e independiente de donde se ejecute. Esto permite a los desarrolladores implementar, probar y desplegar aplicaciones en cualquier entorno informático. Tienen la característica de ser livianos además de proporcionar una infraestructura que no cambia al empaquetar completamente el software de una aplicación. Esta aplicación o servicio se transforma en una imagen de contenedor. Ahora, la aplicación se puede probar como una unidad de software individual mediante una instancia de esa imagen de contenedor en un sistema operativo host sin necesidad de realizar cambios en esta o solo cambios mínimos de configuración (Microsoft Azure, 2020).

3.2.1. Docker

Actualmente, Docker (Docker, 2022) es una plataforma de código abierto que se ha convertido en el estándar de facto en la gestión de contenedores. Es cierto que Docker no ha sido la primera solución en cuanto a la creación de soluciones para el aislamiento software de aplicaciones, pero actualmente su plataforma y contenedores son lo más utilizado en el mercado. Además, cuenta con un enorme repositorio de imágenes llamado Docker Hub (Docker Hub, 2014) donde los vendedores de software, proyecto open-source y la comunidad de desarrolladores ponen a disposición de los usuarios sus imágenes de contenedores junto a un control de versiones de cada uno de ellos.

3.2.2. Arquitectura Docker



Como se puede apreciar en la imagen anterior (Ilustración 4), la arquitectura Docker está basada en un modelo cliente-servidor con tres componentes principales: Docker Engine, el cliente de Docker y el registro Docker-Hub (Avi, 2019):

1. Docker Engine

Es la parte principal de la arquitectura, es un proceso demonio o “daemon” también conocido como “dockerd” que escucha las solicitudes de la API de Docker y se encarga de tareas como la administración de objetos Docker como las imágenes, contenedores, redes, volúmenes... También proporciona una interfaz de línea de comandos (CLI) para interactuar con el demonio.

2. Cliente de Docker

Es la parte de la arquitectura con la que los usuarios interactúan con el demonio de Docker para utilizar su línea de comandos. Cliente y demonio se comunican entre sí mediante un API REST de forma flexible, es decir, no tienen por qué ejecutarse los dos en una misma máquina ya que el cliente se puede conectar al servicio Docker, aunque se encuentre en un sistema remoto.

- Imágenes

Objeto Docker con todas las instrucciones necesarias para ejecutar contenedores a partir de esta. Es encargado de empaquetar un servicio u aplicación junto a todas las dependencias necesarias su funcionamiento como

librerías, código, configuraciones o ficheros... Pueden importarse directamente del registro de Docker, modificar una imagen ya existente utilizando comandos de Docker sobre su contenedor para exportar su estado a una nueva imagen y definir imágenes personalizadas a partir de ficheros Dockerfile. Las imágenes se construyen a partir de un conjunto de capas apiladas una encima de otra de forma jerárquica, son los pasos a partir de los cuales se construyen, y cada una de ellas, incluyen los cambios o modificaciones que se van produciendo. Estas capas se comparten entre las imágenes parecidas, por lo que se acelera su proceso de creación. Solo se recrearán las capas que hayan sufrido alguna modificación (JFrog, 2021).

- Contenedores

Son instancias o ejecuciones de las imágenes de la aplicación que se comportan como entornos seguros y aislados, lo que permite que una misma máquina pueda ejecutar varios contenedores. Como diferenciador con respecto a las imágenes, digamos que estas serían equivalentes a un fichero ejecutable y los contenedores serían similares a los distintos procesos que intervienen en el proceso de ejecución de dicho fichero. Los contenedores añaden a las capas de la imagen que las crean su propia capa de contenedor o capa de escritura, que almacena los cambios que se realizan en el contenedor durante su ejecución. Esta capa los diferencia de las imágenes, permitiendo mantener un estado independiente con respecto a los demás contenedores.

- Volúmenes

Objetos que cumplen la función de persistir los datos utilizados por los contenedores ya que sus datos se almacenan en la capa de escritura, y como esa capa no persiste automáticamente, se pierden al finalizarlos. El contenido dentro de los volúmenes existe con independencia del ciclo de vida de los contenedores. Las soluciones que ofrece Docker son los volúmenes y los “bind mounts”, siendo la primera la más recomendable debido a su portabilidad y la manera más segura de compartir los volúmenes entre varios contenedores, además de poder ser administrado con Docker. La segunda opción, permite montar directorios ya existentes de la máquina host en un contenedor, pero no se pueden administrar con Docker. Hay una última opción disponible para Linux llamados “tmpfs mounts” recomendado para guardar ficheros de datos sensibles, puesto que almacenan los datos en memoria en vez de en disco y desaparecen cuando el contenedor se elimina.

3. Registros Docker

Docker-Hub es el registro público oficial donde se almacenan las imágenes de Docker, con la posibilidad de poder permitir a usuario la creación de sus propios repositorios privados. Cuando se importa una imagen Docker en una máquina, se almacena en el registro de imágenes local, de forma que, si se desea crear un contenedor a partir de esa imagen, el daemon chequeará el registro local para localizarlo. En el caso de que no se localice la imagen y se esté intentando utilizar una imagen inexistente en el equipo, la descargará directamente del registro de Docker-Hub.

3.2.3. Construcción de imágenes con Dockerfiles

Como puede verse en la siguiente imagen (Ilustración 5), el proceso de construcción de un contenedor se organiza en tres pasos (Aula de Software Libre de la UCO, 2022):

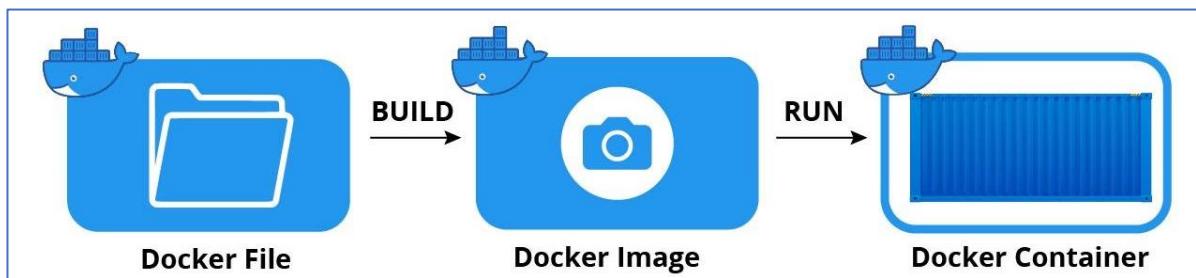


Ilustración 5 - Fases construcción de un contenedor (JFrog, 2021)

1. Escritura del Dockerfile.

El primer paso será la elaboración del fichero Dockerfile con el conjunto de instrucciones para compilar y ejecutar la imagen (FROM, WORKDIR, RUN, COPY, EXPOSE...). Algunas de estas instrucciones son: especificar la imagen base a partir de la que se creará la nueva imagen (FROM), ejecución de comandos de instalación de paquetes (RUN), definición de los puertos que se expondrán en el contenedor (EXPOSE)...

2. Construcción de la imagen definida en el Dockerfile

Una vez que el fichero esté creado se construirá la imagen utilizando el comando “docker build”. Las imágenes tienen un sistema de etiquetado que se usa mediante el argumento “-t” para identificar las imágenes y controlar sus versiones.

3. Ejecución del contenedor a partir de la imagen creada

Finalmente se lanzarán los contenedores mediante el comando “docker run”. Se acompaña con diversos argumentos como por ejemplo “-p” para especificar los puertos del contenedor y la máquina host que se expondrán, “-e” para pasar variables de entorno o incluso “-v” para montar volúmenes al contenedor consiguiendo la persistencia de datos.

3.2.4. Docker Compose

Herramienta de Docker para definir y lanzar aplicaciones que utilizan varios contenedores al mismo tiempo. Es un fichero de tipo YAML donde se puede definir las configuraciones de los contenedores, volúmenes, redes y demás elementos para su lanzamiento en conjunto mediante un simple comando “docker-compose up”. Puede lanzar tanto contenedores a partir de las imágenes guardadas en el equipo como imágenes disponibles en el repositorio público (Docker, 2023).

Los contenedores pueden interactúan entre si mediante su nombre de contenedor. Los que están definidos dentro del fichero reciben el nombre de servicios, y se comunicarán entre ellos por este nombre cuando se desplieguen mediante Compose.

Estas son algunas características clave que hacen Compose una herramienta muy efectiva para el despliegue de contenedores:

- Trabajar con múltiples entornos aislados en una misma máquina

Compose utiliza el nombre del proyecto donde se lanzan los contenedores para aislar los entornos y sus contenedores entre sí, permitiendo crear múltiples copias de un mismo entorno para probar configuraciones o añadir nuevos contenedores en el caso de estar probando una nueva versión de la aplicación y no se deseé alterar la versión estable actual. Incluso si los entornos utilizan los mismos nombres para los servicios contenedores, estos pueden existir sin interferirse unos con otros porque estarán identificados por nombres de proyecto diferentes.

- Preservar los volúmenes utilizados por los servicios

Mantiene los volúmenes definidos utilizados para poder ser usados por otros servicios contenedores. Cuando se ejecuta “docker-compose up”, si encuentra algún contenedor de ejecuciones anteriores, copia los volúmenes

del contenedor antiguo al nuevo. Este proceso garantiza que no se pierdan los datos creados en los volúmenes.

- Solo recrea los contenedores que han recibido cambios

Se almacena en caché la configuración utilizada para crear un contenedor, de forma que, al reiniciar un servicio que no ha cambiado se reutilizan los contenedores existentes. Esto permite realizar cambios en el entorno de desarrollo con gran rapidez.

- Utilización de variables de entorno

Se pueden utilizar ficheros con variables de entorno “.env” y variables de entorno del equipo para poder crear distintos entornos de ejecución personalizados.

3.2.5. Máquinas virtuales vs. Contenedores

Las máquinas virtuales y contenedores tienen algunas características en común, pero existen grandes diferencias entre estas dos tecnologías. Como ya se ha explicado anteriormente en el apartado de virtualización de este documento, las máquinas virtuales utilizan un software llamado hipervisor instalado en la máquina física que permite la virtualización del hardware junto a sus recursos de computación para ejecutar varias instancias de sistemas operativos en una misma máquina, cada una reservando su parte de recursos (Ilustración 6). Cada máquina virtual tiene un sistema operativo propio junto a sus aplicaciones y bibliotecas, pudiendo estar ejecutándose paralelamente cada una con su memoria, almacenamiento... y estando aisladas unas de otras. No obstante, normalmente se necesitan ordenadores potentes para correr varias máquinas virtuales al mismo tiempo.

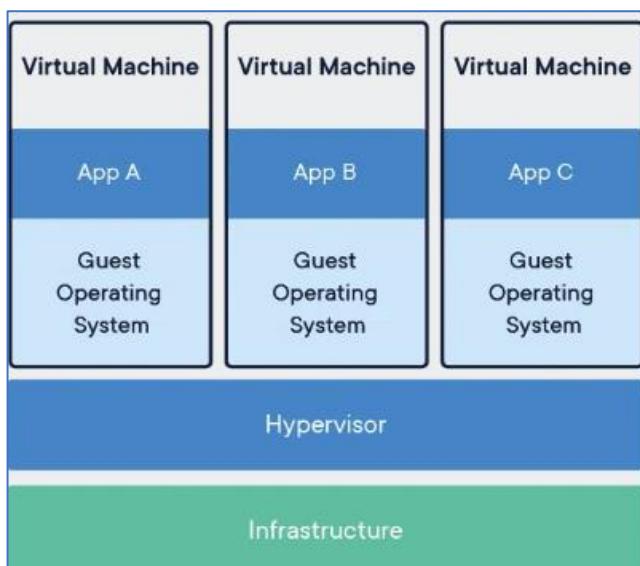


Ilustración 6 - Máquinas virtuales y su hipervisor (Docker, 2021)

Los contenedores utilizan o virtualizan el sistema operativo host, comparten el mismo núcleo del sistema operativo (kernel) de la máquina donde se ejecutan en vez de tener cada uno en un sistema operativo propio como las máquinas virtuales (Ilustración 7). No reservan recursos hardware, si no que todos los recursos se encuentran compartidos entre todos los contenedores, por lo que cada uno utiliza un entorno aislado pudiendo ser tratados ahora como procesos individuales del sistema. Por un lado, su portabilidad les permite ejecutarse en cualquier lugar, independientemente del sistema operativo e infraestructura, por otro lado, están limitados a un tipo de sistema operativo en su creación, por lo que un contenedor hecho para ejecutarse en un Linux no puede ejecutarse en Windows (Microsoft Azure, 2020). Las aplicaciones dentro de contenedores arrancan mucho más rápido que las máquinas virtuales al no tener que emular el hardware completo al arrancar el sistema, y al ser más ligeras, permiten un mayor rendimiento y escalabilidad al poder ser desplegadas un número elevado de instancias a gran escala (Eder, 2016).

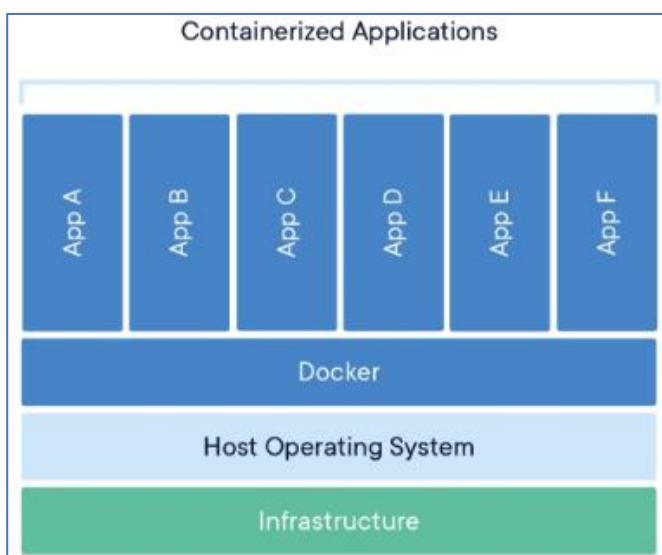


Ilustración 7 - Contenedores sobre el sistema operativo (Docker, 2021)

En este apartado hay que mencionar a Kubernetes, una herramienta software de código abierto muy importante ampliamente utilizada por las empresas para la orquestación de los contenedores. Se encarga de mantener funcionando las aplicaciones basadas en contenedores, una tarea muy compleja puesto que los contenedores pueden estar alojados en distintos servidores o máquinas virtuales. Kubernetes organiza o agrupa las máquinas virtuales en clústeres y ofrece una manera de programar e implementar estos contenedores a través de una API para que se ejecuten en esas máquinas en función de los recursos disponibles y de los requisitos propios del contenedor. Además, se encarga de administrar el ciclo de vida de los mismos y escalar los contenedores, los cuales son agrupados en forma de “pods” (unidad básica con la que opera Kubernetes), hacia un estado deseado (Microsoft Azure, 2019).

Aunque se hayan explicado sus diferencias, no existe como tal un enfrentamiento entre ambas, al contrario, el utilizarlas de forma complementaria permite desplegar y gestionar aplicaciones de una manera flexible y utilizar los recursos de una manera más eficiente y segura. Son una forma de virtualización clave en cuanto al desarrollo de aplicaciones y con su uso en entornos en la nube, permite aprovechar eficazmente esta tecnología.

3.3. Computación en la nube o “Cloud Computing”

La computación en nube es una tecnología gestionada por un "proveedor en nube" externo para ofrecer a los clientes servicios en cualquier lugar, en cualquier momento y en diversas circunstancias gracias a los distintos formatos de privacidad o tipos de nube pública, privada o híbrida entre otros. El cliente solo paga por los recursos informáticos que usa o reserva, realizando un control del gasto de recursos supervisando su uso para medir e informar de manera transparente a los usuarios. Es un tipo de sistema paralelo y distribuido en el que grupos de servidores remotos se conectan a la red y entre sí para permitir el almacenamiento centralizado y compartido de datos, realizar tareas de procesamiento de datos y brindar acceso en línea a servicios o recursos de Tecnologías de la Información (IT). Este modelo de nube ha surgido como fruto de la maduración tanto de Internet como de las tecnologías que la utilizan, ante la proliferación del uso, cada vez mayor, de dispositivos y equipos conectados entre sí y el uso de la inmensa cantidad de aplicaciones y servicios en la red emergentes. La nube permite a los usuarios ejecutar tareas informáticas sin necesidad de entender la tecnología subyacente sin la preocupación de tener que invertir o mantener grandes piezas de hardware gracias al uso de las infraestructuras de los proveedores cloud (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2018; Shukur et al., 2020).

3.3.1. Modelo Cloud

En los últimos años, el uso de la computación en nube ha abarcado muchos sectores de la vida diaria como por ejemplo en el gobierno, las empresas, la industria, los negocios y los mercados. Aparece en un momento crítico para la industria TI puesto que, cada vez se es más consciente de que los recursos físicos, sistemas e infraestructuras informáticas estaban llegando a un límite. Esto es debido a la constante aceleración del ritmo de vida de la sociedad y los negocios los cuales necesitan estos recursos tecnológicos para seguir progresando. El límite del que se habla por la creciente demanda de tecnología se debe, entre otros, a estos factores (Projekt Cloud Computing, Universidad Tecnológica de Breslavia, 2018; Shukur et al., 2020):

- La ingente cantidad de datos, transacciones y dispositivos digitales está poniendo a prueba las infraestructuras y operaciones informáticas existentes.
- Con el crecimiento exponencial de la demanda de sistemas de comunicación y servicios están apareciendo limitaciones en el ancho de banda de la red y la capacidad de almacenamiento.
- Las ineficiencias de suministro de servicio durante los picos de demanda están presionando directamente al sector energéticos.

Para afrontar estos límites, el modelo de computación en la nube se caracteriza por su propuesta de utilizar una “economía innovadora” impulsada por Internet para aprovechar sus tecnologías subyacentes y escalarlas en función del uso. Brinda una visión de los servicios en la red de calidad, con alta disponibilidad y capaz de satisfacer la necesidad de acceso a estos en tiempo real y de forma dinámica. También permite no solo reducir los costes, sino aumentar en gran medida la productividad mediante el uso de la virtualización, aprovisionamiento flexible de recursos y la gestión energética.

No obstante, no todo son ventajas y soluciones, también aparecen nuevos retos a los que se tienen que enfrentar los proveedores de la nube. La complejidad que supone de por si mantener el servicio de computación en la nube en cuanto a escalabilidad, seguridad, demanda de recursos..., junto al uso de la virtualización y brindar servicios en Internet a la carta, requiere de una constante supervisión y monitorización del rendimiento de las infraestructuras, su capacidad, diseño, y gestión de soluciones basadas en la nube.

3.3.2. Tipos de nube o modelos de implementación de la nube

Los tipos de nube que existen están directamente relacionados con el nivel de privacidad y el control que las empresas quieren tener de la infraestructura cloud que usan. Por el hecho que la información y los recursos estén en servidores en la nube provoca que los proveedores planteen varios modelos de nubes, para que las empresas clientes puedan satisfacer sus necesidades de negocio. Destacan los tipos de nube pública, privada e híbrida (Tavbulatova et al., 2020):

Nube pública

Es un tipo de infraestructura diseñada para proporcionar los mismos recursos y servicios informáticos a múltiples empresas sin que haya un control

exhaustivo en cuanto a la localización información. El proveedor cloud es el propietario de la infraestructura y es el encargado de su mantenimiento, poniéndola a disposición del resto de usuarios públicamente a través de Internet, por lo tanto, el cliente no posee control físico sobre la infraestructura. Algunos proveedores que ofrecen nube pública son: Windows Azure, Google o AWS.

Ventajas:

- Uso de la nube sencillo, bajo demanda y eficaz.
- Cantidad ilimitada de recursos informáticos.
- Seguridad a nivel físico y de software mediante el uso de grandes centros de datos.
- Solo se requiere acceso a Internet para acceder a los recursos, sin ningún proceso adicional.
- Pago flexible a través de tarifas por tiempo de uso en la nube.
- Reduce los costes hardware y software de la empresa.
- Estabilidad y alta disponibilidad de la infraestructura.

Desventajas:

- No tener el control total sobre la infraestructura.
- Dependencia total a Internet con acceso remoto a los recursos.
- Almacenamiento de datos sensibles y de los clientes gestionado por el proveedor.
- Dependencia del proveedor de servicios, también conocido como “vendor lock-in”.

Nube privada

Esta opción está destinada al uso exclusivo de la infraestructura por parte de una sola empresa, idealmente, se organiza la nube privada entorno a las necesidades estructurales de la organización en cuanto a sus departamentos, usuarios, clientes y proveedores... No se comparte con otras empresas, y la propia nube puede estar ubicada en un proveedor de servicios externo o dentro del centro de datos de la empresa permitiendo un mayor control físico sobre la infraestructura. Entre los proveedores que ofrecen una solución privada están: Dell Active System, VMware y OpenStack entre otros.

Ventajas:

- Control total de los aspectos de la infraestructura.
- Incremento el rendimiento y velocidad de transferencia de datos

- Optimización y escalado de los propios recursos.
- Mayor protección de los datos y almacenamiento.
- Control personalizado de los pagos ante los gastos de la propia infraestructura.

Desventajas:

- Cuanto más control físico de la infraestructura, mayor esfuerzo de mantenimiento.
- Necesidad de invertir en hardware y software con licencia.
- Riesgo ante desastres a nivel físico.
- Mayor número de recursos físicos, más los humanos, que hay que gestionar.

Nube híbrida

Combina las características de la nube pública y privada, donde el servicio se puede ofrecer en parte de forma pública, como el acceso a herramientas de desarrollo, y al mismo tiempo poder solicitar también, por ejemplo, parte de la infraestructura como el almacenamiento de manera privada para guardar datos sensibles. Las ventajas y desventajas engloban las ya explicadas anteriormente en los otros tipos de nube. Algunos de los proveedores con servicio de nube híbrida son: Microsoft Azure, AWS o VMware.

3.3.3. Niveles o modelos de servicio

Existen diferentes opciones en función del tipo de servicio en la nube y la implicación que tiene el cliente en cuanto a la gestión de la misma (Ilustración 8). La forma más común de clasificarlos es mediante los tres niveles de servicio conocidos como infraestructura como servicio, plataforma como servicio y software como servicio (Patel & Kansara, 2021):

Infraestructura como servicio (IaaS)

IaaS es uno de los tipos de servicio en la nube con mayor flexibilidad. Proporciona toda la parte de infraestructura informática virtualizada, aprovisionada y gestionada a través de Internet. Los proveedores de IaaS gestionan toda la parte hardware de la infraestructura (servidores, espacio de almacenamiento de datos, etc.) en un centro de datos, permitiendo a los clientes personalizar totalmente el uso de estos recursos virtualizados para satisfacer sus necesidades específicas. El cliente puede adquirir, instalar, configurar y gestionar el software, ya sean sistemas operativos, middleware, aplicaciones y herramientas de desarrollo. Es una solución donde solo se

paga por el consumo de los recursos utilizados: espacio de disco, tiempo de CPU, espacio de base de datos, tasa de transferencia de datos... Algunos ejemplos de IaaS son: Microsoft Azure, Amazon Web Services (AWS), Cisco Metacloud o Google Compute Engine (GCE).

Plataforma como servicio (PaaS)

PaaS proporciona el framework necesario para crear, probar, desplegar, gestionar y actualizar el software. Utiliza la parte de la infraestructura al igual que IaaS, pero incluyendo los sistemas operativos, middleware, herramientas de desarrollo y sistemas de gestión de bases de datos. Utilizar las PaaS es muy útil para cualquier empresa que desarrolle software y aplicaciones basadas en la web, ya que aumenta su productividad al tener que preocuparse solo de desarrollar. Muchas de las herramientas necesarias para desarrollar en múltiples dispositivos o plataformas (ordenadores, dispositivos móviles, navegadores...) pueden ser bastante caras, por lo que abstrae a los clientes de este hardware. Algunos ejemplos de PaaS son: AWS Elastic Beanstalk, Apache Stratos, Google App Engine o Microsoft Azure.

Software como servicio (SaaS)

SaaS es una solución software completamente desarrollada y lista para ser comprada y utilizada directamente a través de Internet mediante suscripciones a servicios. El proveedor gestiona toda la pila de recursos informáticos, entrega aplicaciones completas como servicio permitiendo una abstracción completa tanto del hardware como de la plataforma garantizando que el software esté disponible cuando y donde los clientes lo necesiten. Muchas aplicaciones SaaS funcionan directamente a través del navegador, eliminando la necesidad de descargas o instalaciones. Las aplicaciones SaaS permiten a las empresas ponerse en marcha rápidamente sin la necesidad de adquirir o implantar el hardware y el software para ofrecer sus servicios empresariales. Entre algunas aplicaciones actuales están: Microsoft Office 365, Cisco, Salesforce o Google Apps.

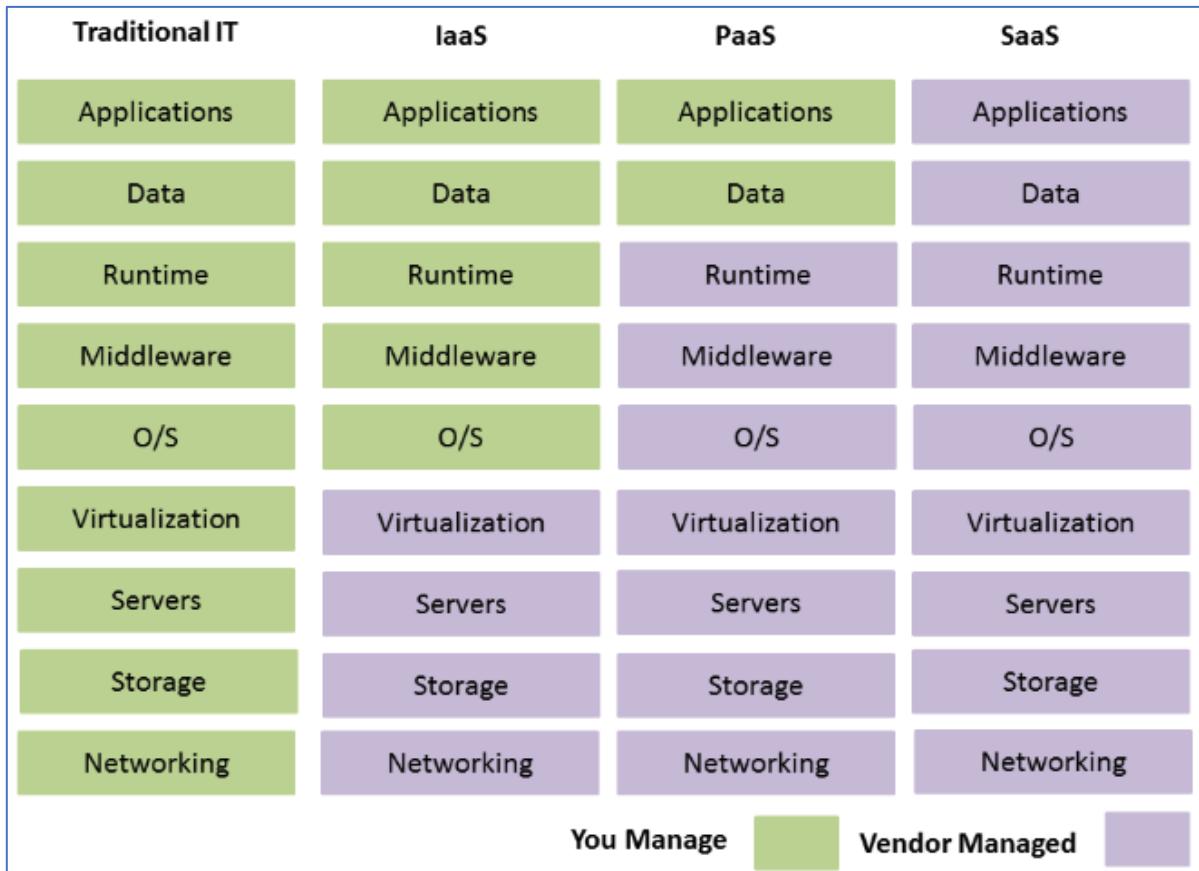


Ilustración 8 - Niveles servicio en la nube (Nazir et al., 2020)

3.3.4. Proveedores de nube

Dentro del mercado de servicios de computación en la nube existen una gran competitividad entre los diversos proveedores. Se explicarán las características de sus principales participantes mediante un “Cuadrante mágico de Gartner”. Este realiza una comparativa entre los principales competidores según una serie de características, apoyada por un gran trabajo de investigación sobre el mercado, ofreciendo una visión panorámica de las posiciones de las empresas en un “ranking”. Algunos de los criterios que utiliza para comparar a los competidores son la variedad de niveles de servicio que ofrecen, la calidad de los recursos de sus infraestructuras, ofertas, interfaces de autoservicio... (Gartner, 2022).

Estos son los proveedores más relevantes actualmente según el cuadrante:

1. Amazon Web Services

Líder actual del mercado, se centra en ser un proveedor con una gran variedad de servicios de TI, que van desde servicios nativos en la nube y de entornos de tipo edge hasta ERP y volúmenes de trabajo de misión crítica. AWS tiene

una visión de futuro en la que busca ampliar el tamaño del mercado mediante el paso a nuevos territorios como el 5G privado y las asociaciones con las telecomunicaciones. Las operaciones de AWS son globales. Sus clientes tienen diversos perfiles, que abarcan tamaños, sectores y lugares.

2. Microsoft

Se encuentra en el segundo puesto de los líderes, es un proveedor muy competitivo en todo tipo de servicios, que incluyen nube ampliada y la computación de entorno edge. Azure es adoptado generalmente por organizaciones centradas en Microsoft. Microsoft invierte en la nube híbrida y multinube, y se enfoca en la realización de mejoras arquitectónicas y de seguridad en su plataforma. Sus operaciones están geográficamente diversificadas, y sus clientes suelen ser empresas medianas y grandes.

3. Google

Tercer puesto líder del mercado, también muy competente en todos sus servicios. Invierte en los modelos de servicio IaaS y PaaS, pero sigue ampliando tanto sus capacidades como el tamaño y alcance de sus operaciones en el mercado. Estas se encuentran diversificadas geográficamente y sus clientes tiende a ser empresas emergentes y grandes empresas.



Ilustración 9 - Cuadrante mágico de Gartner (Gartner, 2022)

Mediante este repaso a la computación en la nube, ahora se conoce un poco más su contexto y características generales, junto con las diversas tecnologías que la han hecho posible como la virtualización y otras como los contenedores que utilizan el enfoque cloud de manera efectiva gracias a su capacidad de aprovechamiento de recursos hardware en el despliegue de aplicaciones. A continuación, se ha escogido el proveedor de Amazon Web Services (AWS) para indagar en la forma de trabajar con una infraestructura cloud y conocer cómo son sus servicios.

[3.4. Amazon Web Services](#)

En el sector de la computación en la nube, Amazon Web Services (AWS) destaca como el proveedor líder actual y más ampliamente utilizado. Se ha ganado su posición a lo largo de varios años gracias a su extensa gama de productos y servicios a nivel global basados en la nube. Estos servicios abarcan una amplia variedad de áreas, incluyendo recursos informáticos, bases de datos, almacenamiento, herramientas de análisis, infraestructura de redes, dispositivos móviles, herramientas de desarrollo, gestión de sistemas, Internet de las cosas (IoT), seguridad y aplicaciones empresariales (Amazon Web Services, Inc., 2021).

La principal ventaja de AWS es la rápida disponibilidad de estos recursos, que pueden ser desplegados en cuestión de segundos. Además, se ofrecen precios flexibles basados en el pago por uso, lo que permite a los usuarios utilizar los servicios según sus necesidades y en el momento que lo deseen. Cuenta con 200 servicios disponibles, entre los cuales abarca desde almacenamiento de datos hasta herramientas de implementación, y desde directorios hasta entrega de contenido.

La capacidad de aprovisionamiento rápido de nuevos servicios sin incurrir en costos de capital iniciales brinda a grandes corporaciones, startups, pequeñas y medianas empresas, así como a clientes del sector público, acceso a los recursos fundamentales necesarios para adaptarse rápidamente a los cambiantes requisitos empresariales.

[3.4.1. Características de AWS](#)

A modo de resumen, se explorarán algunas de las características y ventajas que supone trabajar con AWS (Amazon Web Services, Inc., 2021):

- **Tres modelos de servicio:** AWS dispone de tres modelos de servicios compatibles entre ellos que proporcionan diferentes soluciones, según las necesidades de la empresa o del proyecto. Estas son:

Infraestructura como servicio (IaaS), Plataforma como servicio (PaaS) y Software como servicio (SaaS).

- **Modelo de implementación híbrida:** esta estrategia permite integrar de manera eficiente la infraestructura y las aplicaciones, combinando recursos en la nube pública y privada. Este enfoque es comúnmente utilizado para ampliar y adaptarse a las necesidades de infraestructura de las organizaciones. La parte correspondiente a la implementación de nube privada para proporcionar recursos dedicados se conoce como implementación “on-premise” o bajo demanda.
- **Pago de gastos fijos a gastos flexibles:** en lugar de realizar inversiones significativas en infraestructura tecnológica sin conocer su utilización real, se paga únicamente por los recursos informáticos que se consuma y en el momento en que se necesiten.
- **Economía a gran escala:** con la adopción de servicios en la nube, los gastos flexibles pueden resultar más económicos que los gastos de mantener un sistema propio. A medida que el uso de la nube aumenta y atrae a miles de clientes, los proveedores como AWS logran alcanzar un tipo de economía competitiva gran escala, lo que se traduce en precios más bajos por uso.
- **Capacidad de recursos ajustable:** al tomar decisiones sobre la capacidad que necesitará la aplicación antes de implementarla, a menudo se termina obteniendo recursos caros que no se utilizan realmente o se descubre que la capacidad que tienen es limitada. Este problema desaparece al poder acceder a la capacidad que se necesite, ya sea grande o pequeña, y ajustar los recursos en cuestión de minutos.
- **Mayor velocidad y flexibilidad:** al tener un catálogo de servicios actualizable con las últimas versiones disponibles, se reduce el tiempo necesario para acceder a estos nuevos recursos. Esto resulta en una mejora significativa en la agilidad de las empresas, ya que los costos y el tiempo requeridos para realizar pruebas y desarrollar se reducen en gran medida.
- **Ahorro en el mantenimiento de centros de datos:** libera a las organizaciones del mantenimiento de la infraestructura para que estas se dediquen a las actividades de empresa que generen valor y a sus clientes.

- **Infraestructura a nivel global:** gracias a la amplia extensión de su infraestructura permite desplegar las aplicaciones a lo largo de múltiples regiones de todo el mundo, permitiendo ofrecer una menor latencia y una mejor experiencia para los clientes.

3.4.2. Infraestructura global

AWS utiliza una infraestructura de red a nivel global en expansión que da servicio a más de un millón de clientes activos en más de 240 países y territorios. Se consigue una menor latencia, un mayor rendimiento y la garantía de que los datos residirán únicamente en la región de AWS especificada. También proporciona una gran flexibilidad al poder desplegar instancias, bases de datos y otros servicios AWS en varias zonas de disponibilidad al mismo tiempo dentro de sus correspondientes regiones (Amazon Web Services, Inc., 2021).

La infraestructura está compuesta por 25 regiones y 80 zonas de disponibilidad:

1. Regiones

Las regiones de AWS son las ubicaciones físicas en el mundo donde se dispone de varias zonas de disponibilidad (Ilustración 10). Cada región se ha diseñado para que esté totalmente aislada de las demás regiones, consiguiendo la mejor tolerancia a errores y estabilidad posibles.



Ilustración 10 - Infraestructura global AWS (Amazon Web Services, Inc., 2014b)

2. Zonas de disponibilidad

Las zonas de disponibilidad están formadas por uno o varios centros de datos discretos alojados en instalaciones independientes dentro de una región y conectadas entre sí con baja latencia (Ilustración 11). Cada zona tiene su propio sistema de alimentación ininterrumpida (SAI), redes con conectividad redundante y las instalaciones están protegidas ante desastres naturales como inundaciones. Permiten operar a las bases de datos y aplicaciones con una disponibilidad, tolerancia a errores y escalabilidad superior que si se dispusiera solo de un centro de datos.

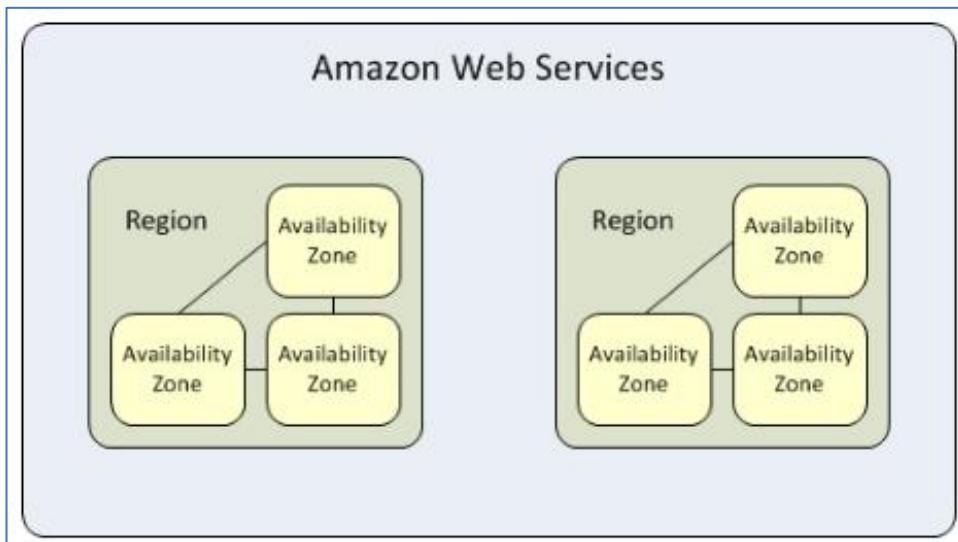


Ilustración 11 - Zonas disponibilidad (José Manuel M., 2019)

3.4.3. Precios de AWS

Amazon proporciona un sistema de pago por uso sin suscripciones, lo que significa que en el momento que se deje de necesitar un servicio, no ata al usuario a tarifas extra ni costos adicionales por contratos. De esta manera, las empresas pueden realizar un control exhaustivo de los costes al consumir servicios (Amazon Web Services, Inc., 2013).

Existen diversos servicios para controlar los costes:

1. **AWS Cost Explorer**: informes con los que se puede analizar los costes totales de los servicios que se están utilizando.
2. **Informes de instancias reservadas “on-premise”**: administrar y monitorear las instancias reservadas de tipo EC2, RDS, Amazon ElasticSearch...

3. **Presupuestos AWS:** herramienta de creación de presupuestos para estimar el costo que supondrá utilizar una serie de servicios. Tiene modalidad mensual, trimestral o anual.
4. **Saving Plans:** es un plan de suscripción donde, a cambio por parte del usuario de comprometerse a utilizar una cantidad de cómputo específica, obtendrá un gran ahorro de hasta un 70%.
5. **Informe de uso y costo:** simple visualización de los costos y cantidad de utilización de servicios con filtros por horas, días, semanas, meses y años.

También se proporciona herramientas calculadoras, una para **calcular los precios** y estimar presupuestos y otra llamada **calculadora de coste total de propiedad (TCO)** para analizar y evaluar el ahorro en la utilización de una serie de productos y servicios AWS.

Para finalizar con los precios de AWS, la siguiente opción es una de las más interesantes puesto que ha ayudado en gran medida a la realización de la parte práctica de este proyecto, y es la capa gratuita:

Capa gratuita

Es una lista con más de sesenta productos que pueden probarse de forma gratuita sin coste alguno (Ilustración 12). Dependiendo del producto, hay tres categorías de capa gratuita:

1. **Gratis para siempre:** permite a los clientes nuevos y existentes utilizar los productos sin coste alguno durante 12 meses dentro de unos límites de tiempo o gasto de recursos de computación. Por ejemplo, las instancias EC2 tienen un límite mensual de 750h de cómputo que gastan al estar encendidas, pero no cuentan las horas si el servicio se para temporalmente. Es una gran opción para comenzar a trabajar con AWS.
2. **Doce meses de uso gratuito:** oferta de uso gratis durante doce meses a partir de la fecha de suscripción, también con límites de cómputo.
3. **Pruebas:** ofertas exclusivas que comienza desde el primer uso hasta que expire el plazo temporal de prueba.

Detalles del nivel gratuito

Filtrar por: [Buscar productos de nivel gratuito](#) [Borrar todos los filtros](#)

Tipo de nivel		Computación		Almacenamiento		Base de datos	
<input type="checkbox"/> Destacado	<input type="checkbox"/> 12 meses de uso gratuito	Nivel gratuito	12 MESES GRATIS	Nivel gratuito	12 MESES GRATIS	Nivel gratuito	12 MESES DE USO GRATUITO
<input type="checkbox"/> Gratis para siempre	<input type="checkbox"/> Pruebas	Amazon EC2 750 horas al mes		Amazon S3 5 GB de almacenamiento estándar		Amazon RDS 750 horas al mes de uso de la base de datos (se aplica a motores de bases de datos)	
Categorías de productos		Capacidad de cómputo de tamaño variable en la nube.		Infraestructura de almacenamiento de objetos segura, duradera y escalable.		Servicio de bases de datos relacional administrada para MySQL, PostgreSQL...	
<input checked="" type="checkbox"/> Informática		750 horas al mes de uso de Instancias		5 GB de almacenamiento estándar			
<input checked="" type="checkbox"/> Contenedores							
<input type="checkbox"/> Interacción con clientes							
<input checked="" type="checkbox"/> Base de datos							
<input type="checkbox"/> Herramientas para desarrolladores							
<input type="checkbox"/> Informática para usuarios finales							
<input type="checkbox"/> Servicios de frontend web y móviles							
<input type="checkbox"/> Tecnología para videojuegos							
<input type="checkbox"/> Internet de las cosas							
<input type="checkbox"/> Machine Learning							
<input type="checkbox"/> Administración y control							
<input type="checkbox"/> Servicios multimedia							

Base de datos		Computación		Análisis	
Nivel gratuito	GRATUITO PARA SIEMPRE	Nivel gratuito	GRATUITO PARA SIEMPRE	Nivel gratuito	PRUEBA GRATUITA
Amazon DynamoDB 25 GB de almacenamiento		AWS Lambda 1 millón solicitudes gratuitas al mes		Amazon Redshift 2 meses prueba gratuita	
Base de datos NoSQL rápida y flexible		Servicio informático que ejecuta su		almacenamiento de datos rápido, sencillo y rentable.	

Ilustración 12 - Capa gratuita (Amazon Web Services, Inc., 2013)

3.4.4. Servicios AWS

Actualmente, AWS proporciona más de **175 servicios combinables** entre sí agrupados en **categorías** como puede ser: informática (EC2, Fargate o Lambda), bases de datos (RDS o Aurora), almacenamiento (buckets S3 o sistemas de ficheros EFS) y servicios de tipo contenedor (clúster ECS o registro ECR) entre otros. En la siguiente ilustración (Ilustración 13) pueden apreciarse estas categorías de una forma más visual:



Ilustración 13 - Tipos de servicios AWS (Medrano, 2022)

Al tratarse de un número bastante elevado de servicios para ir explicándolos todos a grandes rasgos, se explicarán los que se han utilizado en la realización de la parte práctica del proyecto durante la migración de la aplicación a AWS:

VPC

Amazon Virtual Private Cloud o “VPC” (Amazon Web Services, Inc., 2022a) forma parte de categoría “Redes y entrega de contenido” y permite crear una porción aislada y lógica dentro las zonas de disponibilidad para desplegar recursos de AWS en una red virtual personalizada y privada. El usuario tiene un control total sobre el entorno de la red virtual, definiendo, en primer lugar, el rango de direcciones IP de la VPC para los recursos desplegados dentro de esta puedan comunicarse entre sí. Después se podrían crear las subredes, también con su rango de direcciones, configurar tablas de enrutamiento y crear gateways de red entre otros muchos recursos de red. Admite direcciones tanto IPv4 como IPv6 para garantizar un acceso seguro y simple a los recursos y aplicaciones (Ilustración 14).

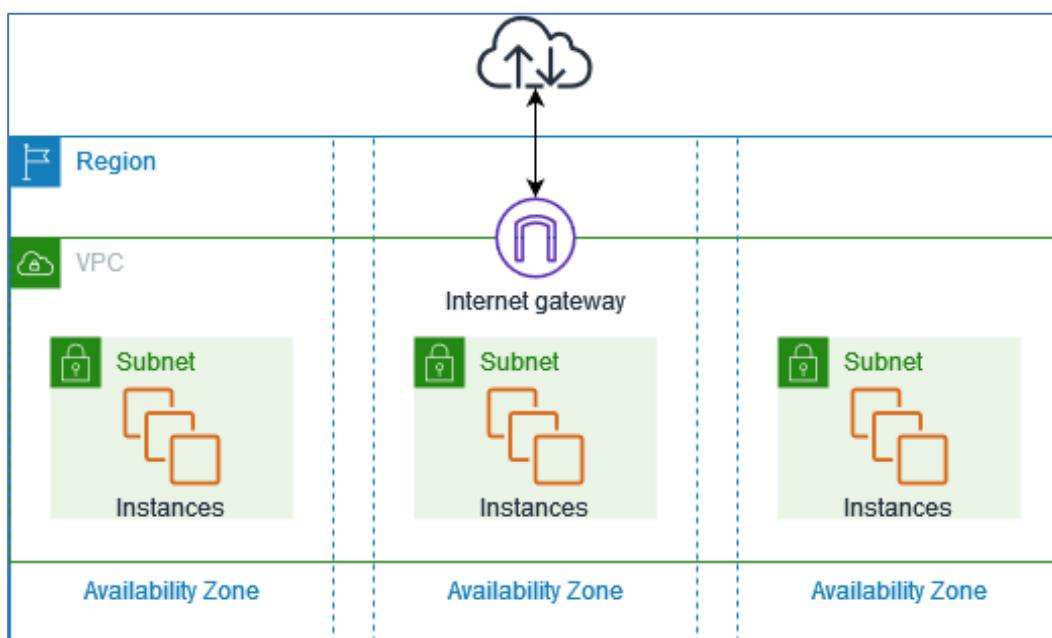


Ilustración 14 – Diagrama VPC (Amazon Web Services, Inc., 2022a)

Dentro del servicio de VPC pueden crearse los recursos de red necesarios para gestionar la red de la infraestructura:

Subredes

Son rangos de direcciones IP personalizadas formadas a partir del rango de direcciones de la VPC y que residen en las zonas de disponibilidad disponibles en esta. Como recursos que se despliegan en estas pueden estar en

diferentes zonas de disponibilidad, si alguna zona falla, solo una parte de los recursos estarían deshabilitados.

Los dos tipos de subredes más relevantes dependen de cómo se configuren sus tablas de enrutamiento:

1. **Pública:** la tabla de enrutamiento posee una ruta directa a través de una puerta de enlace a Internet o Internet Gateway.
2. **Privada:** por defecto todas las subredes son privadas y no poseen una ruta hacia Internet. Para poder acceder Internet es necesario definir una ruta a través de una NAT Gateway.

Internet Gateway

Puertas de enlace hacia Internet para permitir la comunicación entre la VPC e Internet. Posee una alta disponibilidad, redundancia y sin restricciones de ancho de banda.

NAT Gateway e IP Elástica

Permiten que los recursos de las subredes privadas puedan tener acceso a Internet y acceder a servicios externos, pero estos no pueden iniciar una conexión con los recursos de la subred. Pueden ser públicas (por defecto) o privadas, la primera se comporta tal y como se ha explicado y la segunda solo pueden conectarse a otras VPC o a la red en las instalaciones a través de una NAT privada.

Se asocian a una subred y para poder crearse utilizan otro recurso llamado **IP elástica**, que son IPs estáticas que se asocian a la cuenta y permanecen activas hasta que se liberan. Estas últimas sirven para mantener un recurso con una IP que no cambia en el tiempo, puesto que AWS les asigna diferentes direcciones IP públicas a lo largo del tiempo.

Tablas de enrutamiento

Son los conjuntos de reglas o rutas que determinan la dirección del tráfico de red. Se asignan a las subredes o a la propia VPC.

Punto de enlace de VPC

Recurso de AWS para establecer conexiones entre la VPC y otros recursos de forma privada sin tener que pasar por la red pública. Existen muchos tipos

de conexiones que habilitan la conexión a distintos servicios, como por ejemplo un punto de enlace hacia servicios de almacenamiento S3 Bucket para acceder sus objetos solo desde dentro de la VPC.

Identity Access Management (IAM)

Identity and Access Management o “IAM” (Amazon Web Services, Inc., 2014a) es el servicio web de AWS utilizado para administrar de forma segura y centralizada los permisos (políticas) de acceso a los recursos, qué recursos tienen acceso a determinados recursos y qué permisos poseen las distintas identidades (usuarios) dentro de una cuenta. Por defecto, al crear una cuenta el usuario se identifica como el usuario raíz de esta con permisos de acceso completo a todos los recursos y servicios. En un entorno empresarial no es una práctica recomendada realizar tareas cotidianas con este usuario raíz, se recomienda una correcta gestión de usuarios y permisos en función de las tareas tenga que realizar cada uno.

Usuarios

Entidad que representa al usuario humano o carga de trabajo a los que se le asigna una serie de políticas o rol. Necesita un nombre y credenciales de acceso. Como ejemplo, para que herramientas de IaC como Terraform pueda acceder a una cuenta de AWS, necesita de un usuario y sus credenciales para poder acceder a esta junto, además de permisos de administrador para desplegar servicios.

Roles

Entidad con permisos específicos parecidos a los usuarios en cuanto a la asignación de permisos para determinar lo que una identidad puede o no hacer. Se diferencian de estos en no necesitar credenciales y en su finalidad para poder asociarse a los usuarios. De esta forma, creando un rol y asignándole unas políticas, se obtiene una especie de paquete de permisos que puede asignarse a otro recurso IAM.

Perfiles de instancia IAM

Un perfil es una especie de contenedor de roles. Cuando se asignan permisos a las instancias EC2 se realiza a partir de estos perfiles.

Amazon S3 Bucket

Amazon Simple Storage Service o “S3” (Amazon Web Services, Inc., 2012c) es el servicio de almacenamiento de objetos perteneciente a la categoría de “Almacenamiento” que ofrece escalabilidad, seguridad, disponibilidad de datos y un gran rendimiento. Su finalidad es guardar cualquier cantidad de datos de archivos multimedia como fotos, videos, datos, archivos necesarios para alojar un sitio web estático, copias de seguridad, aplicaciones, ficheros de configuración... Proporciona una forma de controlar el acceso a los objetos en forma de políticas de permisos muy sencillas de implementar a partir de ficheros. Los buckets se localizan a nivel de región, no se encuentran como tal dentro de una VPC.

Elastic File System (EFS)

Amazon Elastic File System o “EFS” (Amazon Web Services, Inc., 2015) es el servicio perteneciente a la categoría de “Almacenamiento” que proporciona la creación de sistemas de ficheros para almacenar datos de forma elástica y sin servidor. Escala a medida que se añaden y eliminan archivos, está diseñado para soportar petabytes de información sin interrumpir la actividad de las aplicaciones que lo utilizan. Utilizan el protocolo NFS y su uso más común es montar estos sistemas de archivos dentro las EC2 para su uso compartido.

El acceso al EFS se produce a partir de los **destinos de montaje** (Ilustración 15). El sistema de archivos se monta mediante la utilización de su DNS, el cual se resuelve en la dirección IP del destino de montaje en la misma zona de disponibilidad que la instancia EC2. Se puede crear un punto de montaje en cada zona de disponibilidad dentro de una región, aun así, en el caso de que una VPC cuente con múltiples subredes en una zona de disponibilidad, es posible crear un único punto de montaje en solo una de ellas. Todas las instancias EC2 en dicha zona de disponibilidad compartirán ese mismo punto de montaje.

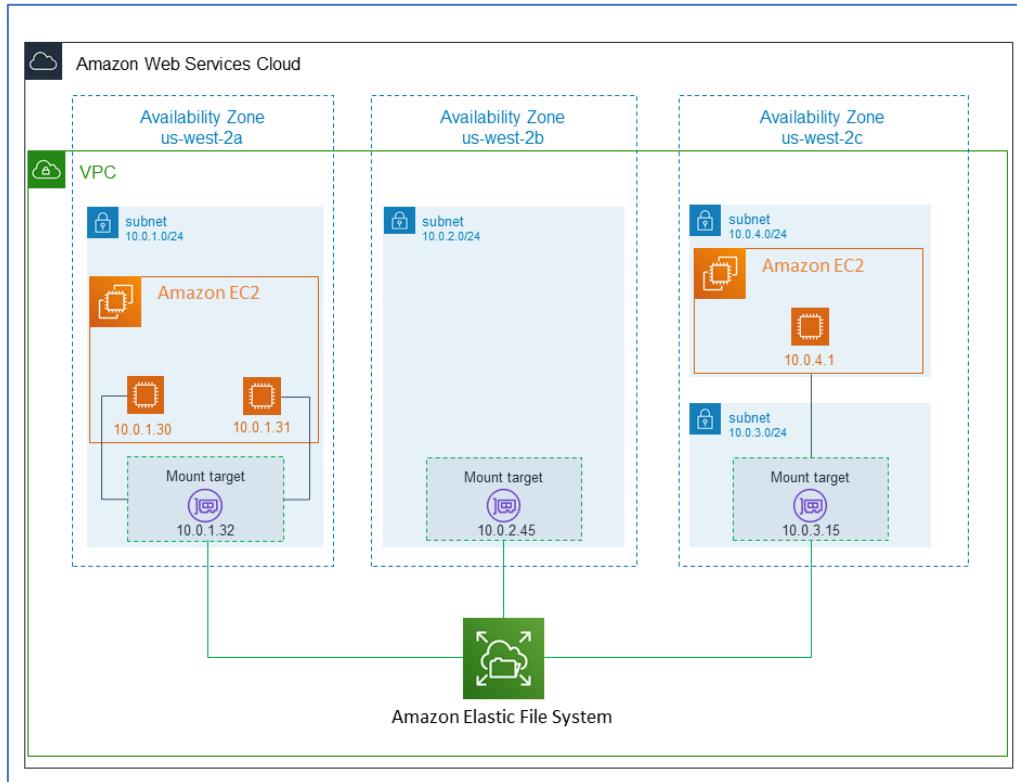


Ilustración 15 - Destino de montaje EFS (Amazon Web Services, Inc., 2015)

Instancias EC2

Amazon Elastic Compute Cloud o “EC2” (Amazon Web Services, Inc., 2012b) pertenece a la categoría de servicios “Informática” y es capaz de proporcionar capacidad de computación escalable a través de instancias de máquinas virtuales en la nube. Elimina la necesidad de disponer del hardware pudiendo lanzar tantos servidores virtuales como se necesite según se busque escalar horizontalmente, desplegando muchas máquinas, o verticalmente para conseguir mayores prestaciones y potencia de computación por máquina que ejecuta la instancia. Son altamente configurables tanto después de su creación como durante gracias a su función de datos de usuario para ejecutar scripts.

Una de las características más importantes es que dispone de una variedad enorme de imágenes de máquina de Amazon (AMI), administradas y mantenidas por AWS, con todo tipo de sistemas operativos (Linux, Windows, Mac...) configurados con las herramientas software necesarias para empezar a trabajar con ellas inmediatamente. También se puede seleccionar el tipo de hardware del host utilizado para la instancia, con mayor cantidad de CPUs, memoria, almacenamiento... A mayor potencia de cómputo, mayor será su gasto por hora.

Grupos de seguridad

Cada instancia EC2 tendrá uno o varios grupos de seguridad. Actúan como un firewall virtual para controlar el tráfico entrante y saliente a través de reglas. Estas reglas se pueden personalizar para habilitar el tráfico según la dirección de IP, puerto, protocolo o permitir las conexiones provenientes de otro grupo de seguridad.

Relational Database Service (RDS)

Amazon Relational Database Service o “RDS” (Amazon Web Services, Inc., 2012b) pertenece a la categoría de “Bases de datos” brinda un servicio para el despliegue y configuración de una base de datos relacional en la nube. Las RDS se encargan de la administración de bases de datos con una capacidad de computación escalable y segura. Es parecido a utilizar una instancia EC2 puesto que también se puede escoger la potencia de computación a través de la elección de un tipo de instancia, pero con la particularidad de que su uso está exclusivamente dedicado al uso de bases de datos. Libera al usuario de instalaciones, actualizaciones o tareas de administración, al contrario que las EC2. Persisten los datos gracias a instantáneas con copias de seguridad a lo largo del tiempo y mediante su opción de implementación multi-AZ, es capaz de desplegar una instancia en espera alojada en otra zona de disponibilidad para atender una mayor cantidad de tráfico o atender las conexiones si la instancia principal falla. Los motores de base de datos a los que da soporte son: MariaDB, Microsoft SQL Server, MySQL, Oracle y PostgreSQL.

Grupos de subredes

Un conjunto de al menos dos subredes que permite especificar las zonas de disponibilidad que desea utilizar para lanzar instancias de bases de datos. Es requerido para la creación de las RDS aunque no se desplieguen en varias zonas.

Elastic Container Service (ECS)

Amazon Elastic Container Service o “ECS” (Amazon Web Services, Inc., 2023b) es un servicio de la categoría “Contenedores” para la orquestación de contenedores escalable y administrado completamente por AWS. Es compatible con los contenedores Docker, puede acceder a Docker-Hub para importarlos de sus repositorios y también puede utilizar el registro de contenedores ECR propio de Amazon. Su funcionamiento es el siguiente: un **clúster** está formado por un grupo de instancias EC2 o instancias de contenedor (llamadas así en el contexto de ECS) gracias a un **agente ECS**

que se ejecutará dentro de estas, las cuales se desplegarán inmediatamente tras de crear el clúster en la VPC y subredes especificadas. No interesa la configuración de la propia máquina sino priorizar la disponibilidad inmediata de la infraestructura EC2, de forma que se puedan levantar y destruir los servicios contenedores alojados en las máquinas según se necesite. Los contenedores se lanzan de una manera rápida y automatizada a través de **tareas ECS**. En la definición de tareas se importan y configuran los contenedores de la misma forma en la que se haría con Docker para ser ejecutados sobre las instancias del clúster. Si se desea tener alguna tarea siempre en funcionamiento, se pueden definir **servicios** para mantener en ejecución un número determinado instancias de una definición de tarea

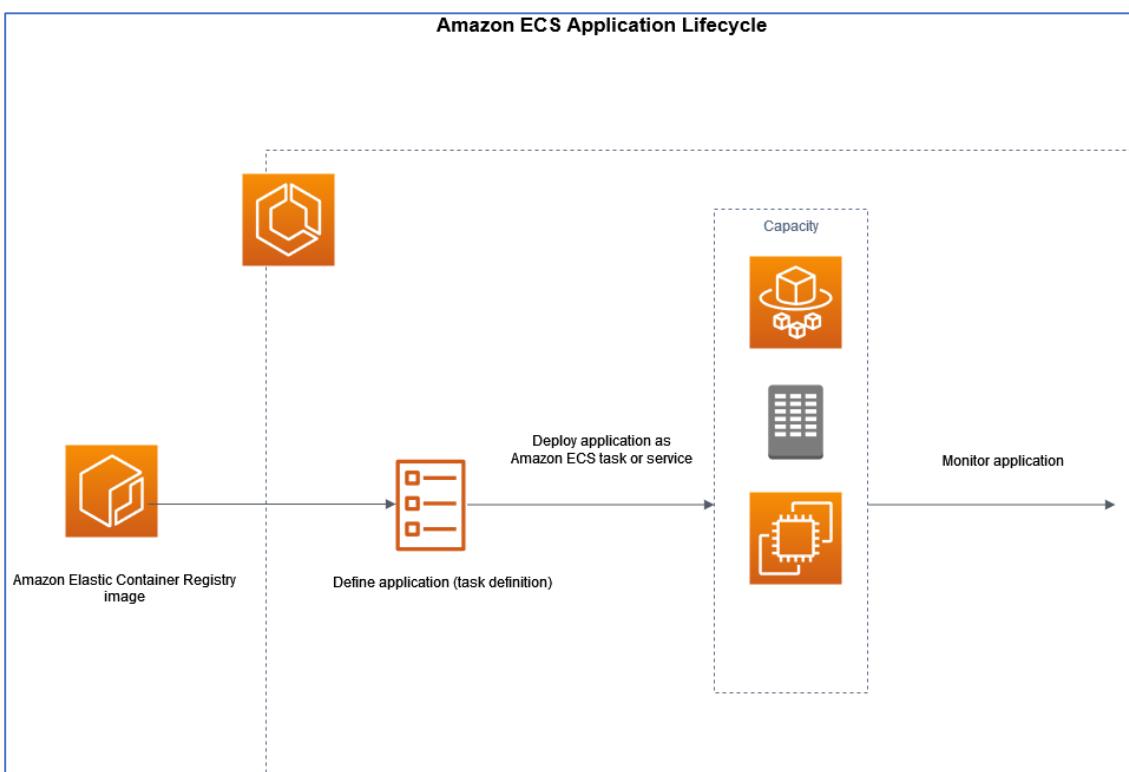


Ilustración 16 – ECS AWS (Amazon Web Services, Inc., 2023b)

Balizadores de carga (Application Load Balancer)

Elastic Load Balancing o “balanceadores de carga” (Amazon Web Services, Inc., 2016) pertenecientes a la categoría “Redes y entrega de contenido” de AWS, distribuyen el tráfico entrante de forma automática entre varios destinos, que pueden ser de tipo EC2, contenedores y direcciones IP localizadas en una o múltiples zonas de disponibilidad. Como el tráfico puede variar a lo largo del tiempo, los balanceadores se escalan y se adaptan a diversas cargas de trabajo.

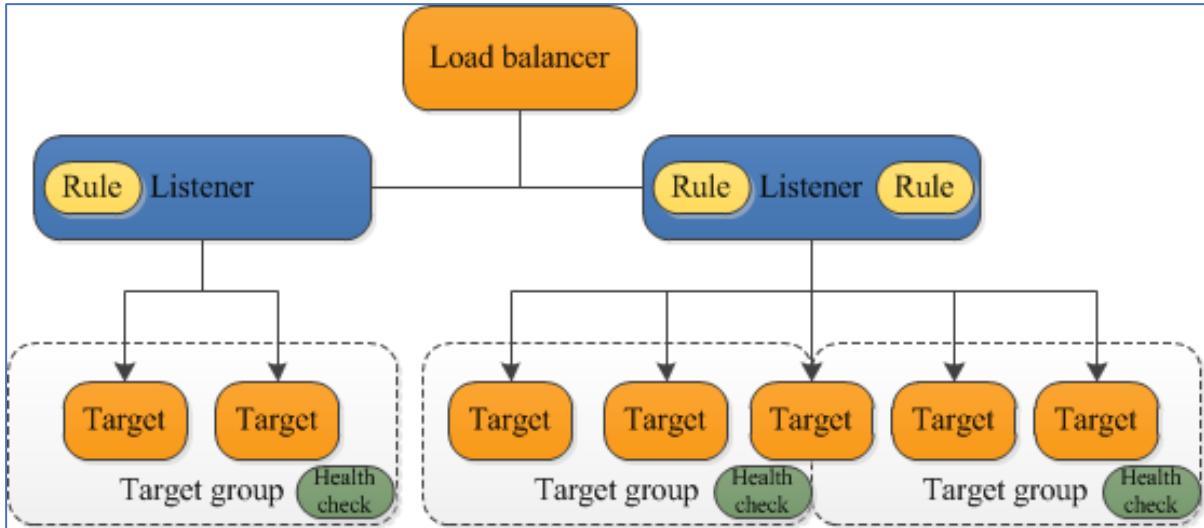


Ilustración 17 – ALB AWS (Amazon Web Services, Inc., 2016)

Uno de los tipos de balanceadores más utilizados es el de aplicación o ALB (Ilustración 17). Permite agregar diversos agentes de escucha o “listeners” para atender el tráfico HTTP o HTTPS entrante por varios puertos al mismo tiempo. Mediante los grupos de destino, dirigirá las peticiones entrantes detectadas por los “listeners” y las redirigirá a los destinos registrados en estos grupos por el puerto especificado. El “listener” asociará este grupo a una regla para que se produzca la redirección. Según el tipo de configuración seleccionado, solo los destinos saludables o alcanzables recibirán el tráfico. Para detectar la salud del destino, el grupo enviará una petición de comprobación de salud a una ruta específica.

Como se ha podido ver en este apartado, Amazon Web Services proporciona una potente solución para resolver casi cualquier necesidad de servicio o infraestructura a través del cloud. Para finalizar este estado del arte, hay otra tecnología muy importante a conocer, sobre todo si se trabaja con este tipo de entornos en la nube, las herramientas de infraestructura como código.

3.5. Herramientas de infraestructura como código o “IaC”

Debido al auge de la virtualización junto a la aparición de los proveedores cloud se empezó a explotar los diferentes niveles o modelos de servicio en la nube. Entre estos modelos se encuentran los servicios de infraestructura como servicio (IaaS). A medida que las plataformas de computación en la nube crecían sus infraestructuras se volvían cada vez más complejas. Esto provocó una necesidad de poder gestionar y configurar con agilidad estas infraestructuras complejas en la nube. Aquí es cuando aparece la idea de infraestructura como código (IaC) como forma de automatizar los flujos de

trabajo software para reducir la complejidad de la administración de sistemas en la nube (Atlassian, 2022).

La infraestructura como código es un proceso que aplica las prácticas recomendadas de desarrollo DevOps y CI/CD para la gestión de la configuración de infraestructuras en la nube y sus recursos (máquinas virtuales, redes o bases de datos entre otros). Se modela la infraestructura para la automatización de su despliegue a través de ficheros de texto con código, por lo que se pueden almacenar en repositorios como GitHub y mantener un control de versiones.

[*3.5.1 Importancia de las IaC*](#)

Es una tecnología importante puesto que ayuda a resolver las diferencias entre entornos. Muchas veces las aplicaciones en la nube se despliegan en distintos tipos de entornos a lo largo de su ciclo de vida, como es el caso de los llamados entornos de desarrollo, producción y pruebas. Estos entornos suelen presentar diferencias entre sí, lo que provoca que durante el trabajo manual de administración o desarrollo puede dar lugar a errores concretos sobre la aplicación sin tener en cuenta una visión global de su infraestructura, dando lugar a otras indecencias, ralentización en las entregas... Las IaC permiten trabajar con una visión completa de la infraestructura de la aplicación, pudiéndose adaptar automáticamente a los cambios de configuración producidos en cualquier recurso de esta, mejorando la supervisión y visibilidad de las tareas de administración manual de sistemas mediante el uso de código de forma flexible (Atlassian, 2022).

[*3.5.2. Terraform como herramienta de IaC*](#)

Se trata de una herramienta de infraestructura como código creado por la empresa HashiCorp. Tiene un tipo de codificación declarativa, lo que permite describir la infraestructura que se quiere construir mediante un lenguaje de alto nivel. Después de definir la infraestructura, Terraform elabora un plan de acciones para orquestar cómo se va a crear la infraestructura definida y lo ejecuta. Actualmente es una de las herramientas de IaC más utilizadas en el mercado, sobre todo en entornos cloud híbridos por su capacidad de responder ante los cambios de configuración, aunque también puede trabajar con infraestructura de entornos locales (IBM, 2023).

Al ser de código abierto con una gran comunidad de desarrolladores y proveedores cloud que crean plugins, nuevas funcionalidades y mejoras. No se encuentra atado a ningún proveedor, por lo que puede conectarse tanto a la nube de Azure, AWS, Google... para trabajar con sus infraestructuras.

Además, posee la característica de suministrar “infraestructura inmutable”, por lo que, si se produce algún cambio de configuración en el entorno, esta se sustituye por otra nueva sin que haya problemas de compatibilidad entre recursos por actualizaciones u otras causas.

Los ficheros de Terraform se identifican por utilizar una extensión “.tf” y dentro de estos se definen los distintos recursos o “resources” de la infraestructura. Por ejemplo, si se trabaja con el proveedor de AWS se podrán crear recursos como redes VPC mediante “resource aws_vpc”, instancias de máquinas virtuales EC2 con “resource aws_instance” entre otras. También se utilizan los módulos de recursos, que son colecciones de recursos conectados que realizan una acción común. Un ejemplo sería el módulo de VPC de AWS de Terraform para crea una VPC completa con sus subredes, NAT Gateway y demás recursos de red (Babenko, 2021).

4. Refactorización de una aplicación web basada en microservicios para su despliegue en cloud

En este apartado se presentará el proyecto base utilizado en el desarrollo de la parte práctica del proyecto. Primero se conocerán los componentes y funcionalidades de la aplicación, para después explicar las tecnologías y particularidades de los servicios a nivel de código.

La funcionalidad básica del proyecto consiste en una aplicación web basada en microservicios destinada a la agricultura, donde los usuarios podrán controlar del riego sobre parcelas a través de sensores que recabarán información sobre la temperatura, presión, precipitaciones... y otros parámetros atmosféricos que puedan afectar a las cosechas dentro de las parcelas. Mantiene un registro sobre la información referente a las parcelas como el propietario, la lista completa de sensores dentro de las parcelas, información geoespacial para localizarla en un mapa interactivo y la referencia catastral, que es un identificador oficial y obligatorio de los inmuebles. También almacena la información sobre los dispositivos sensores: el propietario, referencia catastral de su parcela, coordenadas de latitud y longitud para poder ser localizado y el tipo de información que recoge el sensor. La información recabada por los sensores también se almacenará de cara a su visualización dentro de la aplicación de forma sencilla y a través de gráficos. Finalmente, como es una aplicación web, proporcionará un registro y autenticación de los usuarios.

La aplicación estará conectada mediante su servicio encargado de las parcelas a la web de la “Sede electrónica del Catastro” del Gobierno de España (Ministerio de Hacienda y Función Pública, 2009). En esta web se encuentran registradas de forma oficial todas las propiedades, las cuales serán en este caso parcelas, a través de sus identificadores de referencia catastral. La aplicación verificará que las parcelas añadidas se encuentren registradas dentro del catastro.

En la propuesta original del proyecto, se cuenta con un sistema en tiempo real para recoger la información de riego forma más rápida y precisa. Esta parte no se ha incluido en el estudio, por lo que se trabajará con la funcionalidad base explicada anteriormente. Las razones más importantes detrás de esta elección son: la complejidad añadida de las tecnologías implicadas en el sistema en tiempo real y que el proyecto no se encuentra implementado completamente.

4.1. Diagrama de la arquitectura

Las funcionalidades o servicios del sistema están implementadas siguiendo la filosofía de microservicios. Estos son unidades funcionales concretas e independientes que todas juntas ofrecen la funcionalidad completa de la aplicación, pero que por sí solas pueden actualizarse sin que esto afecte al resto en una arquitectura de microservicios.

Los componentes de la arquitectura de la aplicación se explicarán a partir del siguiente diagrama (Ilustración 18):

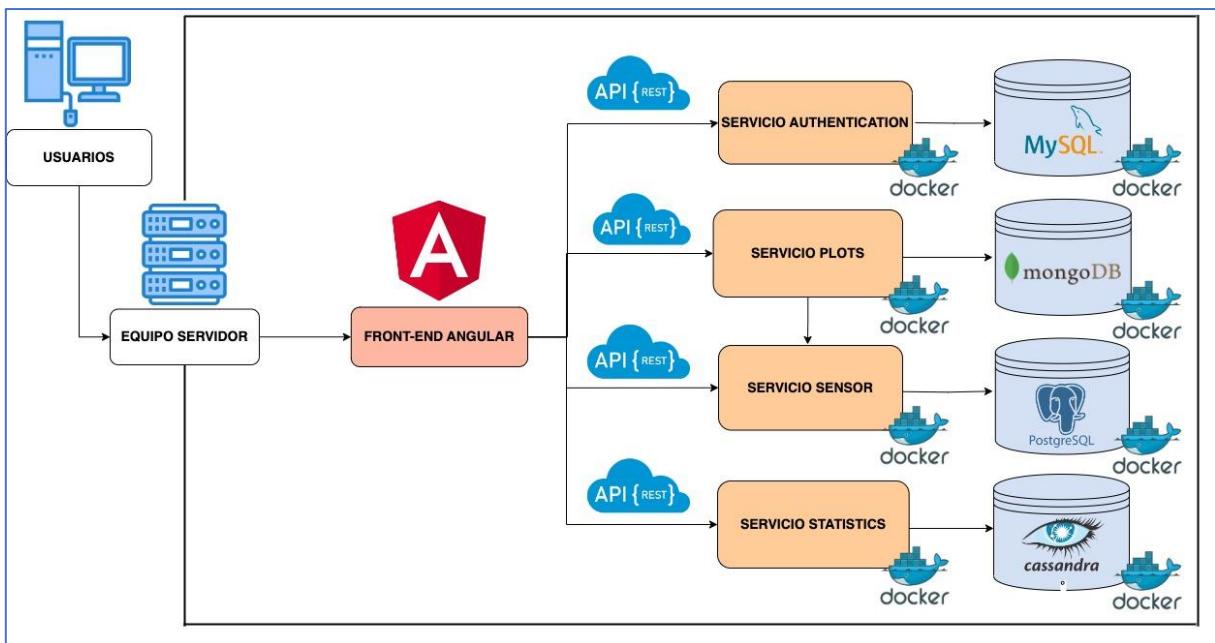


Ilustración 18 - Diagrama de arquitectura de la aplicación base

4.1.1. Servicio Front-End

El servicio Front-End permite navegar por la aplicación hacia el resto de los servicios. Está desarrollado con el framework para el desarrollo de aplicaciones web **Angular** (Angular, 2015). Su página principal mostrará un mapa interactivo (Ilustración 19) para visualizar las parcelas cuando sean registradas en la aplicación:

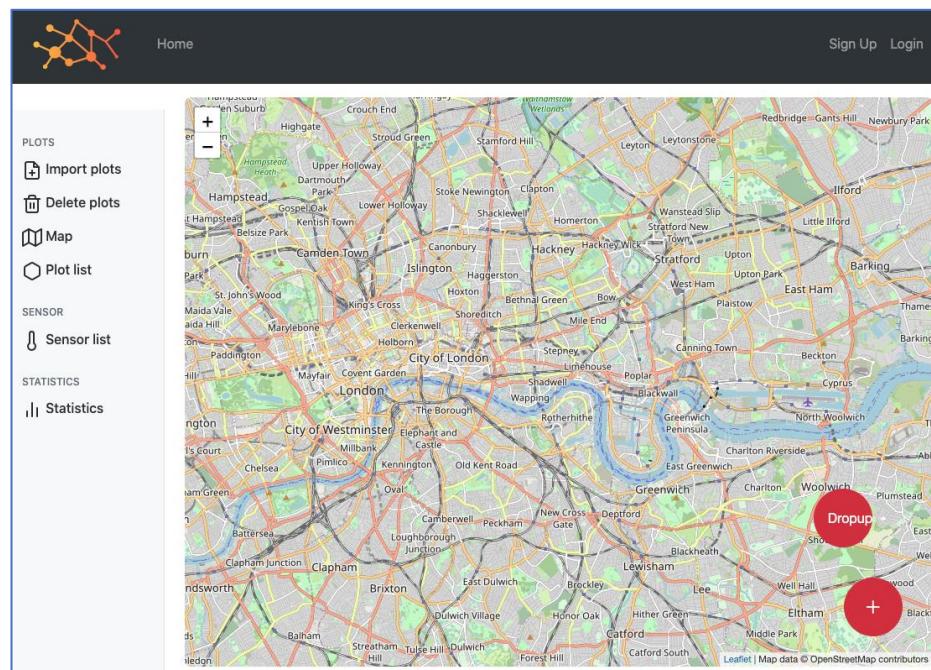


Ilustración 19 - Página inicial aplicación

La parcela inicial que se encuentra registrada en la aplicación está localizada en la parte oeste de la localidad de Arroyo de la Luz en la provincia de Cáceres (Ilustración 20). Como esta parcela existe y se encuentra registrada en el catastro, es posible consultar sus datos en el apartado de buscador de propiedades

(https://www1.sedecatastro.gob.es/CYCBienInmueble/OVCConCiud.aspx?Ur_bRus=R&RefC=10022A010000380000WZ&esBice=&RCBice1=&RCBice2=&DenoBice=&from=OVCBusqueda&pest=rc&RCCompleta=10022A01000038000WZ&final=&del=10&mun=22#).



Ilustración 20 - Parcels data card

En la siguiente imagen (Ilustración 21) puede apreciarse cómo queda coloreada la parcela dentro del mapa a partir del objeto geométrico que forman sus coordenadas (atributo geodata).

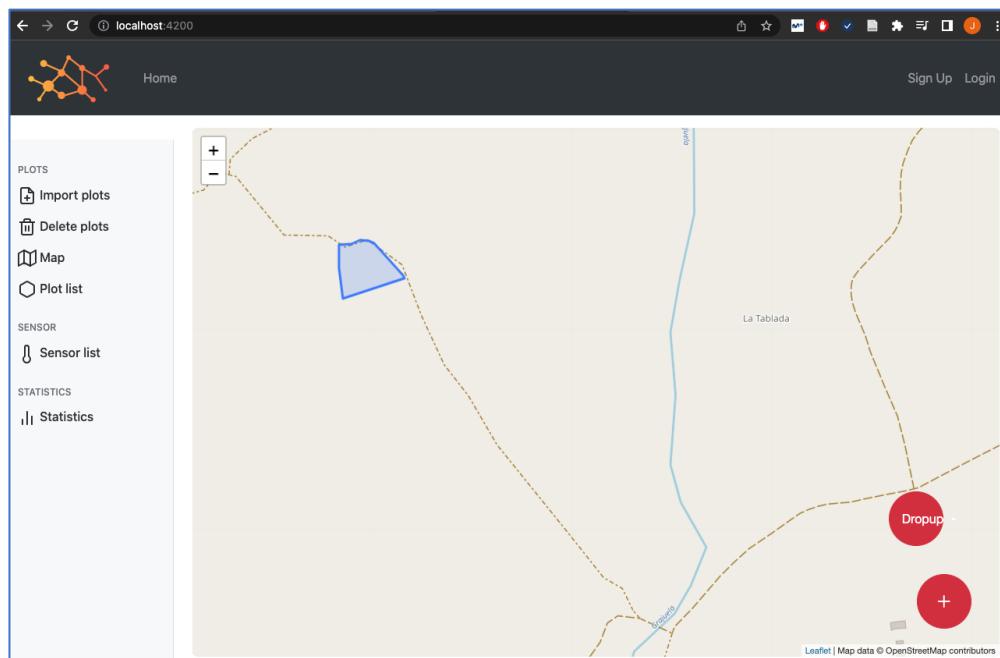


Ilustración 21 - Parcela inicial página

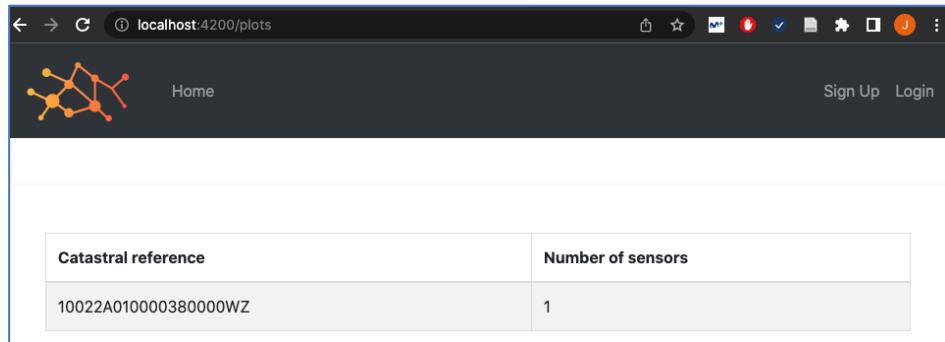
También cuenta con un menú lateral para navegar a las demás páginas de la aplicación:

- **Sign Up / Login:** registro y loguearse en la aplicación (Ilustración 22).

A screenshot of a web application interface. At the top, there is a navigation bar with icons for back, forward, refresh, and other browser controls. The URL 'localhost:4200/register' is visible. On the right side of the header are 'Sign Up' and 'Login' buttons. Below the header, there is a central form area. The form has a logo at the top left and the word 'Sign up' in bold. It contains three input fields: 'Username' with placeholder 'username', 'Email' with placeholder 'example@example.com', and 'Password' with placeholder 'password'. Below these fields is a large blue 'Sign Up' button. The entire form is enclosed in a light gray rounded rectangle.

Ilustración 22 – Login

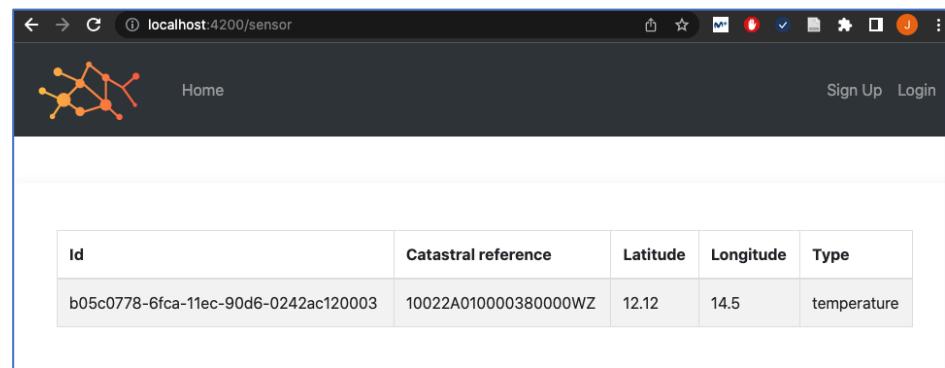
- **Plot list:** listar las parcelas registradas (Ilustración 23).



Catastral reference	Number of sensors
10022A01000380000WZ	1

Ilustración 23 – Lista de parcelas

- **Sensor list:** listar los sensores registrados (Ilustración 24).



Id	Catastral reference	Latitude	Longitude	Type
b05c0778-6fca-11ec-90d6-0242ac120003	10022A01000380000WZ	12.12	14.5	temperature

Ilustración 24 – Lista de sensores

- **Statistics:** mostrar la información recogida por los sensores en forma de gráficos (Ilustración 25).

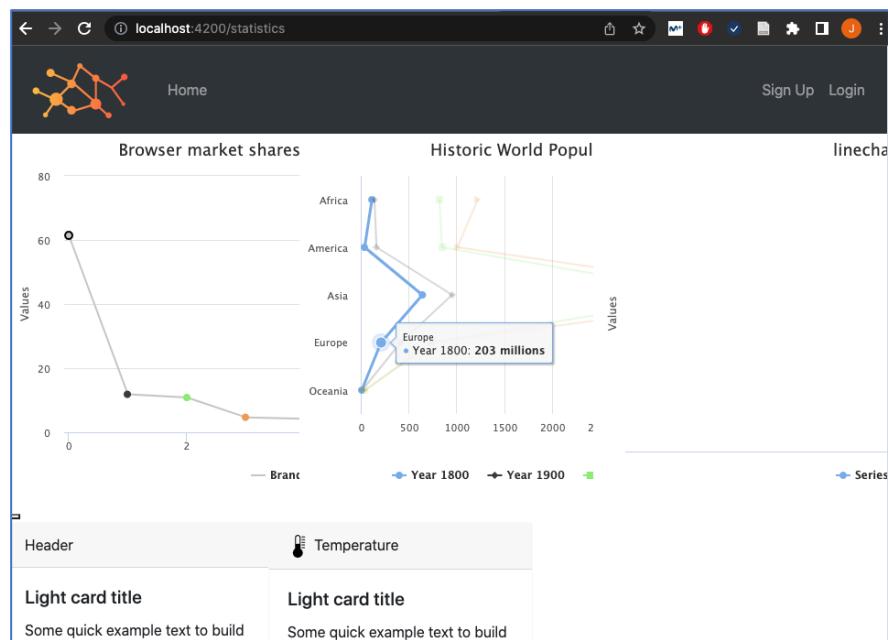


Ilustración 25 – Visualización de estadísticas de datos de los sensores

4.1.2. Servicios API REST y sus bases de datos

Los microservicios serán en este caso aplicaciones con funcionalidad **API REST**, conectado cada uno a una **base de datos**, a partir de la cual se podrá almacenar y consultar los datos. Están desarrollados mediante el framework para el desarrollo de aplicaciones web y microservicios **Spring Boot** (Spring Boot, 2021).

1. Funcionalidad de registro y autentificación de los usuarios dentro de la aplicación se realizará mediante el **servicio Authentication** conectado a una base de datos de **MySQL**.
2. Funcionalidad de guardado y consulta de los datos referentes a las parcelas o “plots” se realizará con el **servicio Plots** conectado a una base de datos **MongoDB**.
3. Funcionalidad de guardado y consulta de los datos de sensores lo realizará el **servicio Sensor** conectado a una base de datos **PostgreSQL**.
4. Funcionalidad de guardado y envío de los datos de estadísticas de sensores para su visualización se realizará mediante el **servicio Stadistics** conectado a una base de datos **Cassandra**.

4.2. Estructura del código y tecnologías de la aplicación web

Se procede a presentar las tecnologías utilizadas por los microservicios, así como la información más importante a nivel de código sobre su funcionamiento. Las imágenes con archivos o código que aparecerán en este apartado son de la versión inicial de la aplicación web antes de realizar la refactorización, documentada más adelante en la sección de desarrollo del proyecto.

4.2.1. Angular

El servicio Front-End está desarrollado con Angular (Angular, 2015).

Es un framework, y a su vez una plataforma de desarrollo, basado en componentes para la creación de aplicaciones web escalables desarrollada por Google. Está construida sobre **TypeScript**, un superset de JavaScript, lo que quiere decir que puede ejecutar programas escritos con JavaScript y al mismo tiempo es un lenguaje de programación propio que potencia enormemente el uso de Javascript (Hernández, 2018). También utiliza el

patrón de diseño modelo-vista-controlador (MVC) y trabaja junto a **NodeJS**, una librería y entorno de ejecución de JavaScript a nivel de servidor cuyo uso se complementa muy bien con Angular (Gonçalves, 2021).

Ficheros importantes de Angular en el proyecto

Entre todos los ficheros que componen el proyecto “tfm-front-end” (Ilustración 26), destacan los ficheros que se encuentran en la carpeta “src/app/_services” con extensión “...service.ts”. Son los encargados de realizar las peticiones HTTP hacia las direcciones de los servicios API REST para utilizar sus diferentes operaciones GET/POST.

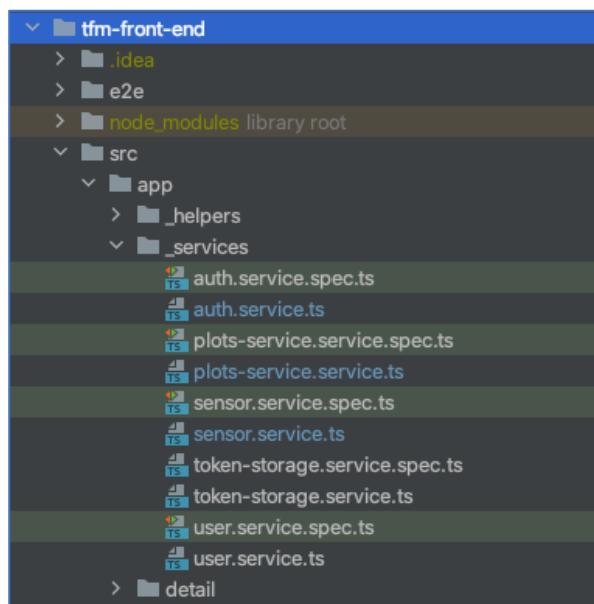


Ilustración 26 - Estructura proyecto “tfm-front-end”

Como ejemplo, en la siguiente imagen (Ilustración 27) se podrá observar el fichero “plot-service.service.ts” donde aparecerá una de las operaciones más importantes definida dentro de este. En la clase “PlotsServiceService” se encuentra la función “getPlots()” que utiliza un cliente HTTP para realizar una petición de tipo GET al servicio Plots que devolverá la lista de parcelas en su URL correspondiente: “<http://localhost:8080/plots/owner/pepe>”.

```

1   import { Injectable } from '@angular/core';
2   import { HttpClient } from '@angular/common/http';
3   import * as Rx from "rxjs/Rx";
4   import { from, Observable, throwError } from 'rxjs';
5   import { map, catchError } from 'rxjs/operators';
6   import { Layer } from '../layer';
7   import { TokenStorageService } from './token-storage.service';
8
9   @Injectable({
10     providedIn: 'root'
11   })
12 export class PlotsServiceService {
13
14   constructor(private httpClient: HttpClient, private tokenStorage: TokenStorageService) {}
15
16   getPlots(): Observable<Layer[]> {
17     //this.tokenStorage.getUser();
18     return this.httpClient.get<Layer[]>(`http://localhost:8080/plots/owner/pepe`).
19       pipe(
20         map((data: Layer[]) => {
21           console.log(data);
22           return data;
23         }), catchError( error => {
24           return throwError( 'Something went wrong!' );
25         })
26       )
27     }
28   }

```

Ilustración 27 - Fichero “plot-service.service.ts”

NodeJS y NPM

Uno de los puntos fuertes de utilizar **NodeJS** en conjunto con Angular es su manejador de paquetes **npm** “Node Package Manager”. Es un repositorio online de paquetes para NodeJS y a su vez una herramienta para la terminal muy útil mediante la cual se realizan tareas de instalación de dependencias, compilación y ejecución de la aplicación (freeCodeCamp.org, 2021).

Dentro del proyecto “tfm-front-end”, las dependencias y paquetes necesarios para su funcionamiento se encuentran en la carpeta “**node_modules**” (Ilustración 28.)

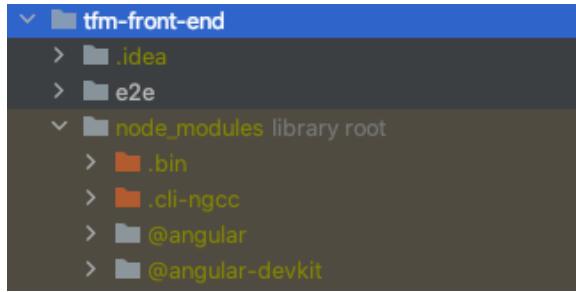


Ilustración 28 - Dependencias “node_modules”

El fichero “**package.json**” servirá para listar todos estos paquetes que se necesita, y se podrán instalar todos juntos al mismo tiempo con el comando “**npm install**”. Al instalar nuevas dependencias se crea también un archivo

“package-lock.json” con una copia del árbol “node_modules” exacto para poder generar arboles idénticos en instalaciones posteriores.

La siguiente imagen (Ilustración 29) muestra la estructura del “package.json” donde se puede ver la lista de dependencias en el apartado “dependencies”:

```
{  
  "name": "tfm-front-end",  
  "version": "0.0.0",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "~11.2.1",  
    "@angular/common": "~11.2.1",  
    "@angular/compiler": "~11.2.1",  
    "@angular/core": "~11.2.1",  
    "@angular/forms": "~11.2.1",  
    "@angular/platform-browser": "~11.2.1",  
    "@angular/platform-browser-dynamic": "~11.2.1",  
    "@angular/router": "~11.2.1",  
    "core-js": "3.10.0",  
    "rxjs": "6.6.7",  
    "tslib": "2.3.1",  
    "zone.js": "0.11.4"  
  }  
}
```

Ilustración 29 - Fichero “package.json”

Otro apartado importante del fichero es el de “scripts”, donde se pueden establecer las diferentes operaciones de Angular “ng” a las de npm. La más importante es “**npm start**” ya que a partir de esta se realizará un “ng serve” que **lanzará la aplicación en un servidor desplegado en el puerto 4200** por defecto.

4.2.2. Spring Boot

Los servicios Authentication, Plots, Sensor y Statistics están desarrollados con Spring Boot (Spring Boot, 2021).

Java Spring Boot es un framework de código abierto utilizado para el desarrollo de microservicios y aplicaciones web autónomas que se ejecutan en una máquina Java (JVM). Es una versión del framework de Spring original más fácil de usar y más rápido de instalar y configurar. Existe una herramienta web llamada “Spring Boot Initializr” que crea proyectos mediante un simple formulario, los configura e instala las dependencias necesarias para comenzar a trabajar con un proyecto inicial (Spring Initializr, 2020). Es compatible con las herramientas de compilación Gradle y Maven, con las que se pueden instalar dependencias de forma sencilla, muchas de ellas disponibles en Spring para trabajar con bases de datos y otras tareas. Estas se definen en el

fichero “build.gradle” junto a otras características de compilación del proyecto (IBM, 2021b).

Conexión con las bases de datos

Los servicios del proyecto realizan conexiones a sus correspondientes bases de datos, y en Spring Boot su configuración se realiza en los ficheros “application.properties”. Para la configuración de conexión se utilizan operaciones “Spring Data” del propio Spring Boot en los servicios Plots y Statistics para su conexión con MongoDB y Cassandra respectivamente. La configuración de conexiones con bases de datos relacionales realizadas por los servicios Sensor y Authenticacion con PostgreSQL y MySQL, la realizarán Spring y algunas herramientas JPA e Hibernate también disponibles para Spring encargadas de realizar algunas operaciones extra.

Java Persistence API (JPA) es la API de persistencia para Java EE, se utiliza como un framework de Java y se encarga del manejo de datos relacionales procedente de aplicaciones. Hibernate implementa JPA, y es un framework que facilita el mapeo de atributos entre una base de datos tradicional y el modelo de datos de la aplicación (Vergara, 2016).

Por ejemplo, las “application.properties” del servicio Sensor para conectarse a PostgreSQL (Ilustración 30), utilizarán operaciones como “spring.datasource.url” para tareas como establecer la URL de conexión, y otras como “spring.jpa.hibernate.ddl-auto” para mapear los objetos del modelo y crear los esquemas automáticamente al conectarse con la base de datos.

```
1  spring.jpa.database=POSTGRESQL
2  spring.datasource.platform=postgres
3  spring.datasource.url=jdbc:postgresql://localhost:5432/sensors
4  spring.datasource.username=postgres
5  spring.datasource.password=password
6  spring.jpa.show-sql=true
7  spring.jpa.generate-ddl=true
8  spring.jpa.hibernate.ddl-auto=update
9  spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
10 server.port=8081
```

Ilustración 30 - Properties sensor

Otro ejemplo sería la configuración de conexión del servicio Plots con MongoDB (Ilustración 31), que utiliza operaciones tipo “spring.data.mongodb” para realizar esta tarea.

```
1  spring.data.mongodb.host=localhost
2  spring.data.mongodb.port=27017
3  spring.data.mongodb.username=admin
4  spring.data.mongodb.password=example
5  spring.data.mongodb.database=test
6  spring.data.mongodb.authentication-database=admin
```

Ilustración 31 - Properties plots

Elementos de Spring Boot en los servicios

La estructura de los elementos de Spring Boot que interaccionan entre sí para poder utilizar las operaciones API REST de tipo GET/POST de los servicios es la misma o muy parecida en todos los servicios implementados. Se pondrá como ejemplo el proyecto del servicio Plots (Ilustración 32):

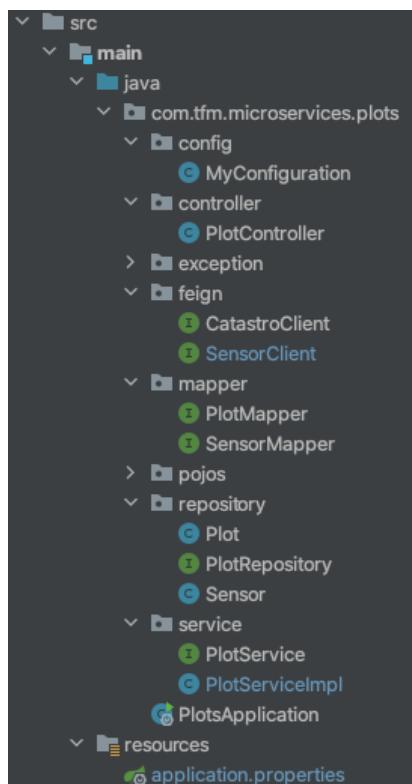


Ilustración 32 – Estructura del proyecto plots

Los elementos básicos de Spring son:

1. **Repositorio y modelo:** capa de persistencia y acceso a datos.

En primer lugar, dentro de la carpeta “repository” se etiqueta la clase modelo Plot básica con @Data o @Entity para indicar que la clase junto a sus atributos se transformen en una entidad de base de datos o entidad persistida. Hay que anotar un atributo con @Id para indicar la clave primaria de la entidad en la

base de datos. Al iniciar los servicios se crearán sus correspondientes tablas, documentos... (Ilustración 33).

```
@Data  
@Document(collection = "layers")  
public class Plot {  
  
    @Id  
    private String id;  
    private String owner;  
    private String catastralReference;  
    private FeatureCollection geoData;  
    private List<Sensor> sensorList;
```

Ilustración 33 - Clase plot

También estará el repositorio (Ilustración 34), una interfaz encargada de definir las operaciones que realizarán consultas con la base de datos. Extenderá el tipo de repositorio que necesite en función de la base de datos con la que se trabaje, en este caso es MongoDB con un “MongoRepository” estableciendo la clase Plot como la entidad que se almacenará (Spring Boot, 2014).

```
public interface PlotRepository extends MongoRepository<Plot, String> {
```

Ilustración 34 - Repositorio plot

2. Servicio: capa de servicios que utilizará la capa de repositorio.

Es la interfaz donde se definen las operaciones o servicios que utilizará el controlador. Se implementa en una clase las funcionalidades, utilizando la clase repositorio definida anteriormente para poder interactuar con la base de datos (Ilustración 35).

```
@Service  
public class PlotServiceImpl implements PlotService {  
  
    10 usages  
    @Autowired  
    PlotRepository plotRepository;
```

Ilustración 35 – Servicio de plot implementado

3. Controlador: capa de controlador que utilizará la capa de servicios.

Finalmente, la última capa será la encargada de transformar las operaciones de la clase servicio en operaciones GET y POST de la API REST para poder ser utilizadas por los clientes, mapeando las peticiones a determinadas rutas según el tipo de operación (Ilustración 36). Por ejemplo, la URL

“/plots/owner/{owner}” se encargará de devolver la lista de parcelas en función del valor del propietario recogido en la variable de ruta {owner}, definida en la etiqueta @GetMapping de la función.

```
@RestController
@RequestMapping(value = "plots")
public class PlotController {

    5 usages
    @Autowired
    PlotService plotService;

    ± sergio
    @GetMapping(value = "/owner/{owner}")
    @ResponseBody
    public ResponseEntity<List<PlotResponse>> getPlotsByOwner(@PathVariable String owner) {
        List<PlotResponse> plots = plotService.getPlotsByOwner(owner);
        return ResponseEntity.ok(plots);
    }
}
```

Ilustración 36 - Controlador plot

4.2.3. Tipos de bases de datos utilizadas

Los microservicios de la aplicación web utilizan tanto bases de datos relacionales como no relacionales o NoSQL. A continuación, se explicarán sus características más importantes y se indicará el tipo de estructuras de datos que almacena cada una:

1. Bases de datos relacionales

Los sistemas gestores de bases de datos relacionales permiten almacenar y administrar datos estructurados a través de sus operaciones CRUD (Create, Read, Update y Delete). Utilizan un modelo cliente-servidor donde el usuario realiza consultas sobre el servidor de base de datos utilizando el lenguaje SQL. Su forma de organizar los datos es mediante tablas, donde cada dato queda guardado en registros con una estructura de filas y columnas. Estas tablas se identifican por una clave primaria fija y claves foráneas con las que se relacionan con otras tablas (Google Cloud, 2022).

MySQL

Motor de base de datos relacional desarrollado por Oracle (MySQL, 2011) que cuenta con una licencia de código abierto y otra comercial. Es un gestor que destaca por su facilidad de uso, una escalabilidad que permite trabajar con volúmenes tanto pequeños como masivos de datos, rápida, multiplataforma y segura. Cuenta con varias herramientas de gestión como “mysqladmin” por la línea de comandos y programas gráficos como “MySQL Workbench” o el cliente gráfico “PHPmyadmin”.

PostgreSQL

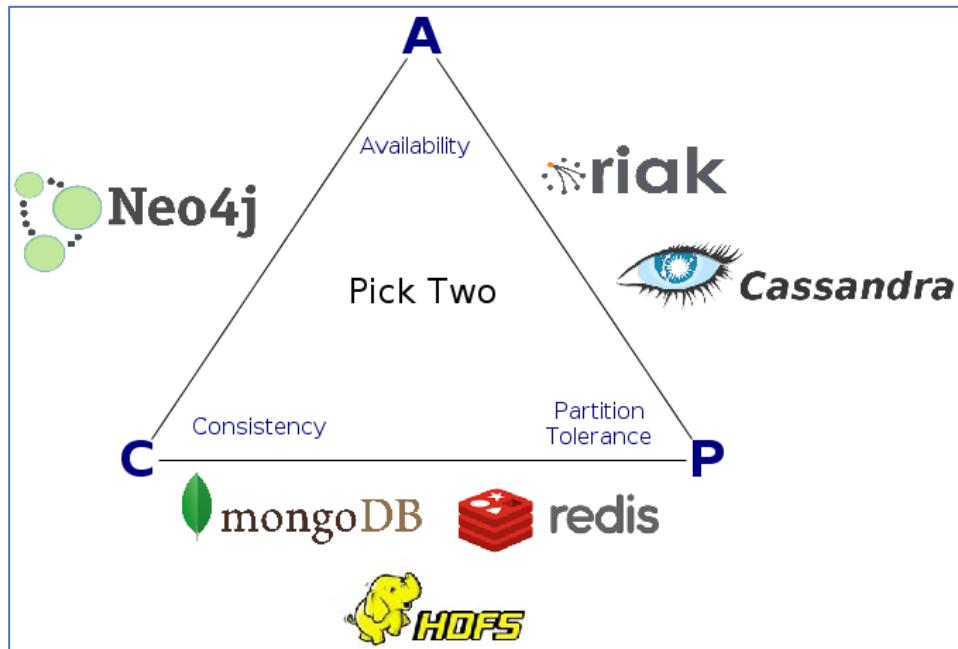
Gestor de base de datos relacionales orientada a objetos que puede utilizar tanto los tipos de datos tradicionales como objetos (PostgreSQL, 2008). Es de código abierto, multiplataforma, con una gran escalabilidad vertical, utiliza el lenguaje SQL y es capaz de extender su funcionalidad enormemente al ser compatible con otros lenguajes de programación como: PL/pgSQL, PL/Perl, PL/Java, PL/Python, PL/PHP... (ayudaley, 2020)

2. **Bases de datos NoSQL**

Sistemas de gestión de bases de datos con un modelo al relacional. Estas no poseen un esquema de datos fijo como las bases de datos tradicionales ya que se trabaja con datos no estructurados, es decir, pueden no contener los mismos atributos. En función del tipo de base de datos, utilizan una gran variedad de tipos de modelos de datos como clave-valor, documentos y grafos, optimizados para el rendimiento y la escala. Destacan por su capacidad de operar de forma distribuida, lo que permite la replicación de datos en múltiples servidores-nodos consiguiendo una gran escalabilidad horizontal (Amazon Web Services, Inc., 2012a).

Una forma de organizar el tipo de bases de datos NoSQL es a través del Teorema CAP, el cual expone que ninguna base de datos puede garantizar tener las siguientes tres características que forman CAP (Mason, 2013):

1. **Consistency** (Consistencia): todos los nodos deben garantizar tener la misma información.
2. **Disponibilidad** (Availability): el sistema debe poder atender peticiones a pesar de que algún nodo deje de funcionar.
3. **Tolerancia a particionado de red** (Partition Tolerance): el sistema debe seguir operando, aunque existan problemas de conexión entre los nodos.



MongoDB

Base de datos distribuida, multiplataforma, con un modelo de datos basado en documentos y diseñada para desarrolladores de aplicaciones modernas y para la era de la nube (MongoDB, 2022). Destaca por su escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos.

Su modelo de datos está orientado a documentos, es decir, en vez de guardar los datos en tablas con registros de filas y columnas fijas, los guarda en estas estructuras de datos llamadas documentos con una estructura JSON. Cada documento tiene su clave con un valor que la identifica, y pueden contener muchos pares de clave-valor distintos, o pares de clave-matriz, o incluso documentos anidados. Los documentos se almacenan en colecciones, las cuales no tienen un esquema fijo (Paramio, 2011).

Cassandra

Base de datos descentralizada en múltiples nodos, donde ningún nodo es superior (maestro) a otro, sino que actúan en diferentes roles. Cada nodo es un coordinador, puede atender la solicitud el mismo y solicitar los datos que necesite a otros nodos. Pueden extenderse de forma física a través de ubicaciones separadas, utiliza la replicación de datos y las particiones para escalar infinitamente las lecturas y escrituras.

El modelo de datos es de tipo grafo, define su estructura en entidades y sus relaciones entre ellas basadas en la teoría de grafos (nodos y aristas). Antes

de almacenar los datos, se especifica un “keyspace” para indicar el número de nodos donde se replicarán los datos. Dentro de este se definen los datos en forma de tablas con sus filas y columnas al igual que las bases relacionales, pero de una forma más compleja debido a su sistema de particiones. La clave primaria está formada por una clave de partición obligatoria y otra de agrupación opcional. Cada dato forma parte de una partición según su clave, y si tiene clave de agrupación, varios datos pueden agruparse dentro de una partición según su valor de columna de agrupación. A su vez, estas particiones pueden estar distribuidas en varios nodos (Apache Cassandra, 2009).

5. Propuesta de arquitectura en Cloud

Una vez analizada la aplicación base con sus tecnologías y arquitectura, se procede a presentar la siguiente propuesta de infraestructura utilizando el proveedor cloud de AWS, que permitirá ejecutar esta aplicación local en un entorno en la nube (Ilustración 38).

5.1. Diagrama de la arquitectura Cloud

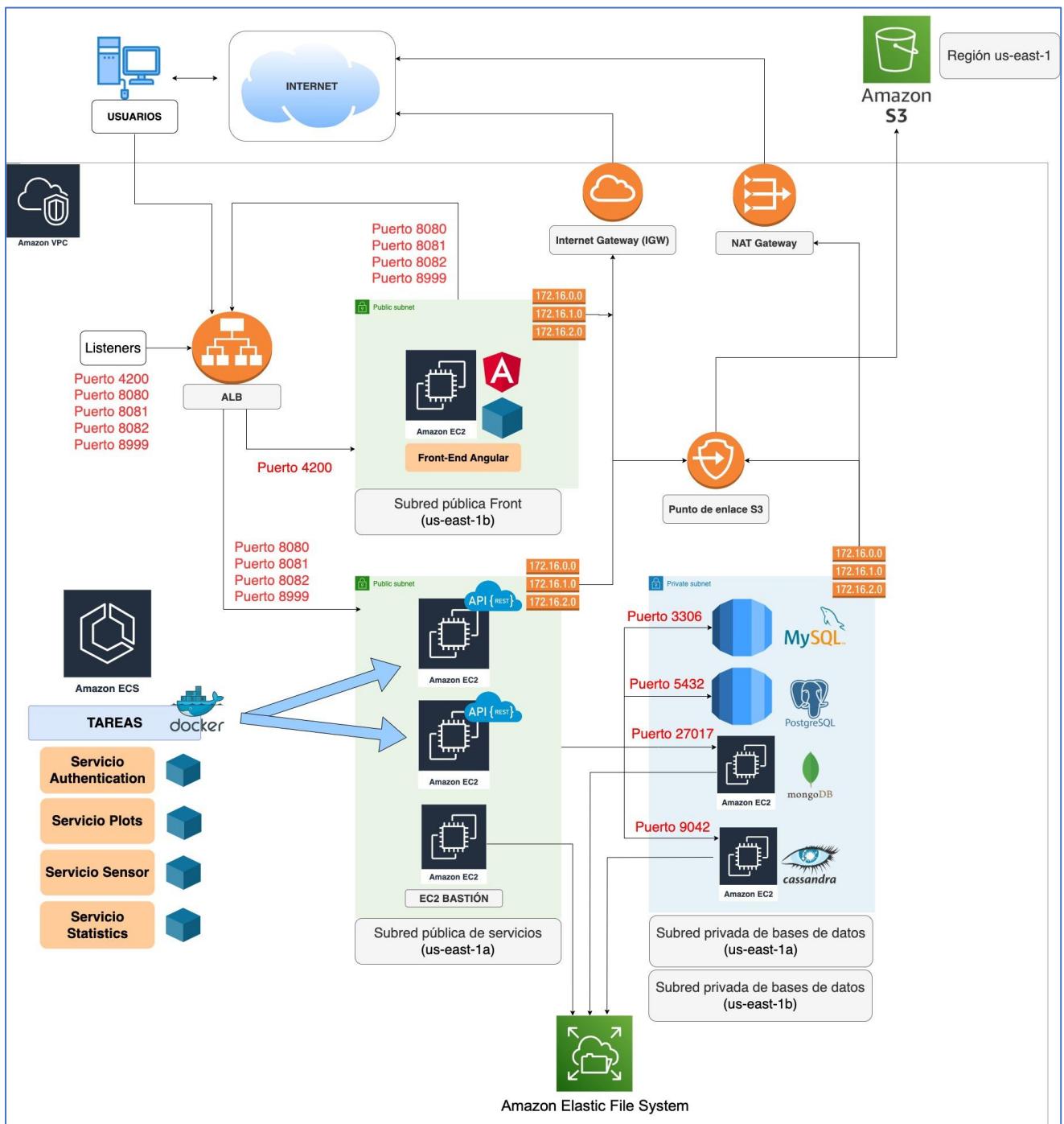


Ilustración 38 - Propuesta de arquitectura cloud

5.2. Razonamiento detrás de la elección de AWS

La razón principal por la que se ha llevado a escoger AWS frente a otros proveedores cloud es que ya se estaba familiarizado con la infraestructura y su entorno de servicios, por lo que era la elección más llamativa entre todos los proveedores.

Otras de las razones de su elección son:

- Importancia en el mercado y su amplia utilización por todo tipo empresas.
- Un completo catálogo de servicios para trabajar con infraestructuras.
- Mucho contenido didáctico para aprender conceptos teóricos y prácticos de sus servicios, tanto dentro de la propia plataforma de AWS como en multitud de páginas con contenido multimedia, ejemplos de código...
- Capa gratuita que ofrece una gran cantidad de servicios que se pueden utilizar sin ningún tipo de coste o con un coste por horas muy reducido, lo que es una opción ideal para iniciarse con las infraestructuras cloud.

5.3. Explicación de la arquitectura

Con respecto a la arquitectura anterior, el usuario ya no interactuará con un solo servidor físico donde se encuentran desplegados todos los contenidos de microservicios, sino que estos estarán distribuidos por diferentes máquinas virtuales y servicios de bases de datos alojados en la infraestructura de AWS. A su vez, estas máquinas y bases de datos se encontrarán dentro de una infraestructura de red que permitirá la comunicación de los servicios entre sí y con el exterior (Internet).

Infraestructura de red

La red donde se despliegan los componentes de la infraestructura será una **VPC**. Esta es una red privada dentro de AWS en la que se podrán crear otras **subredes** que asignarán direcciones IP a los servicios dentro de esta. Las subredes tendrán sus correspondientes **tablas de enrutamiento** que les permitirán comunicarse con el resto de la red y con Internet a través de un **Internet Gateway** y un servicio de **NAT Gateway**.

La infraestructura de AWS estará localizada en la **región del Norte de Virginia “us-east-1”** y las subredes utilizarán sus **zonas de disponibilidad “us-east-1a” y “us-east-1b”**.

Lista de subredes:

1. Subred pública de servicios (us-east-1a)

Es la subred principal donde se desplegarán las máquinas virtuales en instancias EC2 para lanzar los contenedores con los microservicios de la aplicación. Tendrá acceso a Internet a través del Internet Gateway. Adicionalmente, se desplegará una máquina EC2 denominada **“EC2 Bastión”**. La finalidad de esta instancia bastión será realizar tareas de configuración de otros servicios desplegados dentro de la infraestructura de una forma directa. Es una práctica que se suele realizar dentro de este tipo de infraestructuras en la nube para conectarse mediante SSH y un par de claves a las máquinas dentro de las subredes de la VPC, tanto públicas, pero sobre todo privadas a través del bastión situado dentro de la propia VPC, puesto que desde fuera no se podría acceder a las EC2 privadas de forma directa. Tras realizar las tareas de configuración pertinentes, se puede destruir la máquina para evitar posibles problemas de seguridad.

2. Subred pública con el servicio Front (us-east-1b)

Destinada a alojar una instancia EC2 con el servicio Front-End de la aplicación, la cual podrá redirigir el tráfico al resto de microservicios situados en la subred de servicios. También usará la Internet Gateway para acceder a la red.

3. Subred privada con las bases de datos (us-east-1a)

No será accesible desde fuera de la VPC puesto que alojará los servicios de bases de datos de la aplicación ejecutados en servicios **RDS** y dentro de **instancias EC2** mediante contenedores.

4. Subred privada con las bases de datos (us-east-1b)

Se desplegará otra red con las mismas características que la anterior situada en la zona de disponibilidad “us-east-1a”, pero esta vez dentro la zona de disponibilidad “us-east-1b”. Estas subredes se asociarán para formar un grupo de subredes, que será necesario para poder crear las bases de datos RDS (requisito de creación). Aunque no se dará el caso en esta aplicación, generalmente estos grupos se definen para aprovechar la capacidad de

replicación de datos que tienen las instancias RDS en varias zonas de disponibilidad.

Contenedores de microservicios

Los contenedores Docker con servicios Front-End y API REST de la aplicación se encontrarán almacenados en un **repositorio de Docker-Hub** para ponerlos a disposición de la infraestructura. Estos serán utilizados por el servicio de **Amazon ECS**, que será el encargado de levantar las instancias EC2 dentro de la subred de servicios para lanzar y configurar sobre estos los contenedores importados del repositorio.

Servicios de bases de datos

Los **servicios RDS** serán los encargados de levantar las bases de datos relacionales utilizadas de MySQL y PostgreSQL. También se seguirán utilizando contenedores para desplegar los servicios de MongoDB y Cassandra dentro de **instancias EC2**. Las RDS permiten por defecto la **persistencia de datos** a través de copias de seguridad o “snapshots” automáticas, pero las EC2 no cuentan con ello. Es por eso que se requerirá de un servicio EFS externo que utilizarán las instancias para persistir los datos, el cual será explicado más adelante.

Servicios de almacenamiento

Para el almacenamiento de la información dentro de la infraestructura se dispondrá de un **bucket S3** con el que se almacenará todos los ficheros de configuración que utiliza la aplicación. Como el S3 no se encuentra en la VPC sino a nivel regional, las instancias EC2 accederán a este bucket de forma segura mediante un **punto de enlace S3** (Amazon Web Services, Inc., 2014c) que permitirá que sólo puedan acceder al contenido del bucket estas instancias dentro de la VPC. Otro servicio que se utilizará es **Amazon Elastic File System o EFS**, capaz de definir sistemas de ficheros portátiles. Se pondrán a disposición de las bases de datos desplegadas en instancias EC2 para que puedan guardar los datos en otra localización por si alguna instancia se borra en algún momento.

Conducción del tráfico de la aplicación

El usuario interactuará directamente con un servicio **balanceador de carga de tipo ALB o “Application Load Balancer”** que se encontrará escuchando las peticiones entrantes a través de “**listeners**” en distintos puertos. En función del puerto donde reciba la petición, la redigirá a su servicio destino. El

servicio Front-End también utilizará este ALB para interactuar con los servicios API REST.

Para acceder a la página principal el usuario se conectará a esta por el puerto 4200 utilizando el ALB, lo que le redirirá al servicio Front-End. Dentro del Front, para navegar por las diferentes páginas realizando las diferentes peticiones a las API REST de los servicios de parcelas, sensores... utilizará el ALB, que estará escuchando las peticiones tanto en el puerto 4200 como los demás puertos donde están desplegados los microservicios:

- **Puerto 8080:** servicio Plots.
- **Puerto 8081:** servicio Sensor.
- **Puerto 8082:** servicio Statistics.
- **Puerto 8999:** servicio Authentication.

[**5.4. Automatización del despliegue con Terraform**](#)

Primero se desplegará la infraestructura a mano utilizando la interfaz gráfica de AWS, configurando uno a uno sus servicios. Para poder desplegarla de forma automática posteriormente, se utilizará la herramienta de infraestructura como código de Terraform. Mediante esta herramienta, se pueden definir todos los recursos de AWS para que Terraform los despliegue automáticamente al mismo tiempo. Cuando se deseé dejar de utilizar la infraestructura, solo habrá que ejecutar un comando y se destruirán todos los recursos creados.

6. Desarrollo

La arquitectura en AWS propuesta tomará forma a través de un proceso de desarrollo que comenzará con la ejecución de la aplicación y sus componentes en un entorno local a través de contenedores, tal y como estaba planteado el proyecto, y que poco a poco se irá convirtiendo en un sistema de contenedores con microservicios que funcionarán dentro de una infraestructura en la nube. Finalmente, para automatizar el despliegue de la infraestructura se utilizarán ficheros Terraform como herramienta de infraestructura como código (IaC).

Este desarrollo se estructura en tres fases:

1. Refactorización de la aplicación para el despliegue de los microservicios en contenedores en Docker.
2. Migración y despliegue de la infraestructura en AWS.
3. Transformación IaC de la infraestructura a través de Terraform.

6.1. Refactorización de la aplicación para el despliegue de los microservicios en contenedores Docker

En esta primera fase se asegurará que todos los microservicios se ejecuten en contenedores Docker puesto que es la manera a través de la cual se desplegarán posteriormente en la infraestructura de AWS dentro de instancias EC2. Mediante el uso de contenedores se empaquetará cada aplicación junto a sus dependencias, asegurando que se ejecutarán siempre de la misma manera y serán compatibles entre distintos entornos, independientemente de la máquina o infraestructura donde se ejecute, siempre sea compatible con Docker.

Para el proceso de contenerización de los microservicios, se procederá con la definición de los ficheros Dockerfile restantes de la aplicación, puesto que ya se contaba con algunos de ellos. Estos se sitúan dentro de la raíz de cada proyecto correspondiente al servicio o base de datos que vaya a desplegarse, con el conjunto de comandos necesarios para formar la imagen Docker y configurar el contenedor en función de las necesidades de la aplicación.

Los Dockerfiles que ya se encontraban implementados corresponden a los servicios de Plots y Authentication. Como ejemplo, se mostrará a continuación el Dockerfile del servicio Plots inicial:

Dockerfile inicial del servicio Plots

```
FROM adoptopenjdk/openjdk11:alpine-jre
VOLUME /tmp
ARG DEPENDENCY=build
RUN echo ${DEPENDENCY}
COPY ${DEPENDENCY}/libs/plots-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Este fichero es válido y puede crear una imagen Docker para desplegar el servicio Plots correctamente. Sin embargo, se han realizado algunas modificaciones para optimizar su rendimiento, puesto que algunas instrucciones no son necesarias y pueden ser simplificadas. Aunque todos los ficheros Dockerfile se encuentran explicados en detalle más adelante durante el desarrollo, en el siguiente fichero se pueden observar las modificaciones realizadas al Dockerfile inicial del servicio Plots:

Dockerfile inicial del servicio Plots modificado

```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY build/libs/plots-0.0.2-SNAPSHOT.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Las instrucciones “VOLUME”, “ARG” y “RUN” pueden ser eliminadas puesto que las instrucciones importantes son “COPY” y “ENTRYPOINT”. La primera instrucción copia el archivo ejecutable de Java con extensión “.jar” dentro del contenedor para poder ejecutar el servicio tras lanzar el contenedor mediante el segundo comando.

Una vez que se hayan definido todos los Dockefiles, se crearán sus imágenes correspondientes. Con estas imágenes, se generarán un total de cinco servicios web y cuatro bases de datos que serán desplegadas a través de contenedores:

- **Servicios:**

1. **Tfm-front-end**: servicio front-end, puerto 4200.
2. **Plots**: servicio API REST, puerto 8080.
3. **Sensor**: servicio API REST, puerto 8081.
4. **Authentication**: servicio API REST, puerto 8999.
5. **Statistics**: servicio API REST, puerto 8082.

- **Bases de datos:**

1. **Postgres**, puerto 5432 y conectado al servicio **Sensor**.
2. **Mongo**, puerto 27017 y conectado al servicio **Plots**.
3. **Mysql**, puerto 3306 y conectado al servicio **Authentication**.
4. **Cassandra**, puerto 9042 y conectado al servicio **Statistics**.

Además, se utilizará un fichero “Docker-Compose” donde se coordinará la creación y el lanzamiento de los contenedores a partir sus imágenes previamente creadas o partiendo de versiones oficiales importadas directamente de DockerHub. Cada servicio contenedor creado se lanzará en conjunto con su respectiva base de datos.

No obstante, antes de la construcción de los contenedores, se ha de refactorizar el código necesario para que los microservicios funcionen de forma correcta con esta tecnología y puedan comunicarse correctamente entre ellos, en este caso, para que los servicios de API REST se conecten a las bases de datos.

• **Comprobación del despliegue de los contenedores**

La comprobación del despliegue local de la infraestructura se ha realizado en el apartado [8. Pruebas](#) del documento, más concretamente en el [8.2.1. Despliegue local de contenedores](#).

6.1.1. Modificación de las URL de conexión y archivos de propiedades para la configuración de la aplicación

En una aplicación basada en microservicios que interactúan entre sí, ya estén desplegados tanto en un equipo o máquina virtual local como remota, el código de esta debe ser flexible a la hora de realizar las conexiones hacia las máquinas host o direcciones independientemente de donde se alojen los microservicios.

Al principio, todos los servicios de la aplicación utilizaban la dirección IP o host con valor de “localhost” para todos los endpoints, puesto que todos ellos estaban pensados para su despliegue en el propio equipo local. El servicio Front redireccionaba a las URLs de las APIs utilizando este host, de la misma manera que los servicios de API REST lo usaban para conectarse a sus correspondientes bases de datos. No solo es un problema a la hora de establecer conexiones entre los microservicios si estos estuvieran alojados en una máquina remota, sino que provoca problemas dentro del propio entorno local al utilizar contenedores debido a que sus “hostnames” son diferentes incluso cuando se despliegan en la misma máquina. Por ejemplo, si un contenedor de base de datos utiliza el nombre de “mysql”, el servicio que se

conecte a él (también desplegado en la misma máquina) debe utilizar el nombre del contenedor como nombre de host dentro de la URL de conexión:

```
Uso de URL de conexión incorrecto:  
jdbc:postgresql://localhost:5432/sensors
```

```
Uso de URL de conexión correcto:  
jdbc:postgresql://mysql:5432/sensors
```

La solución empleada para la resolver este problema es la utilización de variables de entorno dentro del código junto a un fichero de extensión “.env”, que contiene un listado de las variables de entorno con las direcciones host que se va a utilizar. Los contenedores que alojarán los servicios API REST y el Front, importarán estas variables de entorno en la fase de ejecución, por lo que estarán disponibles para su invocación cuando el código de aplicación lo requiera. Las variables contendrán el valor de cada host utilizado por la aplicación, de forma que si cambia el host de localización solo habría que modificar el valor de las variables dentro del fichero sin tener que modificar el resto del código de la aplicación.

El fichero encargado de esta tarea es “env_vars.env”, estará situado dentro de la raíz del directorio de trabajo y será utilizado por todos los servicios contenedores de la aplicación que lo necesiten. Contendrá los siguientes valores en el despliegue local:

- Fichero **env_vars.env** para el despliegue en local

```
# Endpoint Front-End  
FRONT_ENDPOINT=localhost  
# Servicio cliente dentro de Plots  
SENSOR_CLIENT=sensor  
# Endpoints Bases de Datos  
POSTGRES_HOST=postgres  
MYSQL_HOST=mysql  
MONGODB_HOST=mongo  
CASSANDRA_HOST=cassandra
```

De esta forma, y retomando el ejemplo anterior, el servicio que utiliza el contenedor “mysql” ahora utilizaría la variable “\${POSTGRES_HOST}” para conectarse al servicio independientemente de la dirección del host:

```
jdbc:postgresql://${POSTGRES_HOST}:5432/sensors
```

A continuación, se muestra el código de aplicación que se ha modificado:

Servicio Tfm-front-end

Dentro del proyecto de Angular correspondiente al Front-End, se identifican los ficheros con las clases donde se realizan las peticiones HTTP, con sus correspondientes URL, hacia los demás servicios API REST (Ilustración 39). En este caso, se encuentran implementados dentro de la ruta “src/app/_services” en forma de clases con sus correspondientes métodos. La modificación del código se ha realizado sobre las URL mediante las cuales la clase “HttpClient” invocada realiza las operaciones GET, POST y DELETE en cada método.

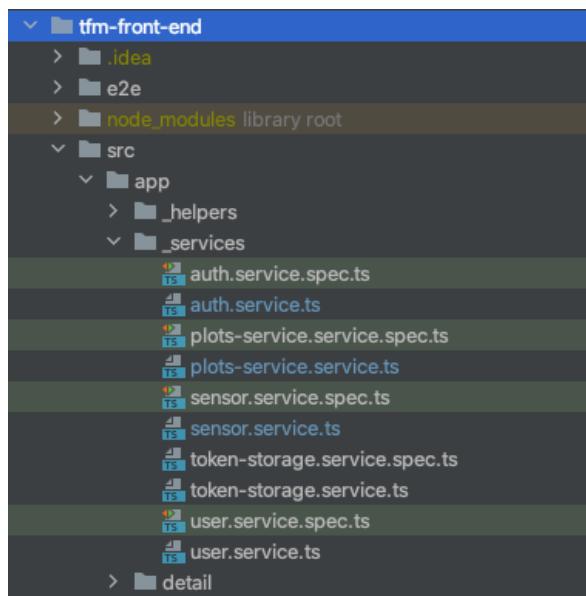


Ilustración 39 – Ficheros service de “tfm-front-end”

Se ha utilizado la variable de entorno “FRONT_ENDPOINT” del fichero .env con el valor “localhost” dentro de las URL para redireccionar a los distintos servicios puesto que se encuentran desplegados en la misma máquina local. Para poder ser utilizada dentro del código, se ha propuesto una solución alternativa debido a que la versión de Angular del proyecto no era compatible con el uso de otros métodos más sencillos para invocar variables de entorno, como sería el caso del módulo de Node “dotenv” (npm, 2023).

La solución se ha obtenido a partir del siguiente enlace:

- <https://stackoverflow.com/questions/66541451/passing-environment-variables-into-angular-docker-container>

El primer paso es la definición del script “docker-env-js-to-script.sh” situado dentro de la raíz del proyecto Front:

- Script **docker-env-js-to-script.sh**

```
#!/bin/bash
set -eu pipefail

echo $FRONT_ENDPOINT

# Capture all environment variables starting with APP_ and make JSON
# string from them
ENV_JSON=$(jq --compact-output --null-input 'env |
with_entries(select(.key | startswith("FRONT_"))))'

# Escape sed replacement's special characters: \, &, /.
# No need to escape newlines, because --compact-output already removed
# them.
# Inside of JSON strings newlines are already escaped.
ENV_JSON_ESCAPED=$(printf "%s" "${ENV_JSON}" | sed -e 's/[\\&]/\\\\&/g')

find ./src -name "index.html" -exec sed -i "s/<noscript id=\"env-
insertion-point\"></noscript>/<script>var env=${ENV_JSON_ESCAPED}; if
(global === undefined) { var global = window;}</script>/g" '{}' '
exec "$@"
```

Este script se ejecutará al lanzar el contenedor del Front, momento en el que se capturarán todas las variables de entorno del fichero “env_vars.env” que Docker pasará al propio contenedor. A continuación, guardará en la variable “ENV_JSON” las variables de entorno cuyo nombre comience por “FRONT_” en formato JSON y se formatean correctamente para su uso posterior en la variable “ENV_JSON_ESCAPED”. Finalmente, el último comando añadirá las variables guardadas en el “index.html”, modificando el fichero y escribiendo sobre este la definición de la variable “var env=\${ENV_JSON_ESCAPED}”. Esta podrá ser invocada más adelante en el código de la aplicación para obtener el valor de “FRONT_ENDPOINT”.

Para insertar la variable “var env” dentro del “index.html”, el comando lo que hace es detectar una determinada línea de texto dentro del fichero y sustituirla con otro texto. Esto requiere de la modificación del “index.html” para añadir la línea “`<noscript id="env-insertion-point"></noscript>`” que indica el texto que hay que sustituir por la variable “env”.

- Fichero **index.html**

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    ...
<noscript id="env-insertion-point"></noscript>
</head>
...

```

Más adelante cuando se ejecute el contenedor, puede observarse que el nuevo “index.html” ahora contiene la variable de entorno (Ilustración 40):

The screenshot shows the Postman interface with a GET request to `http://localhost:4200`. In the Headers tab, there is a 'Query Params' section with a single entry: 'Key' under 'Value'. In the Body tab, the response is displayed in Pretty format, showing the generated HTML code for `index.html`. The code includes a `<noscript>` block that contains the value of the `FRONT_ENDPOINT` environment variable.

```

1  <!doctype html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <title>TfmFrontEnd</title>
7      <base href="/">
8      <meta name="viewport" content="width=device-width, initial-scale=1">
9      <link rel="icon" type="image/x-icon" href="favicon.ico">
10     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
11         integrity="sha384-Vkoo8x4CGs03+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuH0f23Q9Ifjh" crossorigin="anonymous" />
12     <script>
13         var env={"FRONT_ENDPOINT":"localhost"}; if (global === undefined) { var global = window; }
14     </script>
15     <link rel="stylesheet" href="styles.css">

```

Ilustración 40 - Index.html

Una vez realizado este proceso, se podrá utilizar la variable de entorno “`FRONT_ENDPOINT`” haciendo referencia a la variable global “`window`”, que invocará la variable “`env`” definida en el fichero “`index.html`”. Esta variable “`env`” a su vez contendrá la variable “`FRONT_ENDPOINT`”.

A continuación, se mostrarán los cambios realizados en las URL de los ficheros service para utilizar la variable:

- Archivo: **auth.service.ts**

Está formado por la clase “AuthService” y contiene los siguientes métodos encargados de enviar las peticiones hacia el servicio **Authentication**:

```
login(username: string, password: string): Observable<any> {
  return this.http.post
(`http://` + window.env.FRONT_ENDPOINT + ':8999/auth/authenticate',
<resto del código>
}
```

```
register(username: string, password: string): Observable<any> {
  return this.http.post
(`http://` + window.env.FRONT_ENDPOINT + ':8999/auth/register',
<resto del código>
}
```

- Archivo: **plots-service.service.ts**

Está formado por la clase “PlotsServiceService” y contiene los siguientes métodos encargados de enviar las peticiones hacia el servicio **Plots**:

```
getPlots(): Observable<Layer[]> {
  //this.tokenStorage.getUser();
  return this.httpClient.get<Layer[]>
(`http://` + window.env.FRONT_ENDPOINT + ':8080/plots/owner/pepe').
<resto del código>
}
```

```
getPlotByCatastralReference(catastralReference: String):
Observable<Layer> {
  return this.httpClient.get<Layer>
(`http://` + window.env.FRONT_ENDPOINT +
`8080/plots/${catastralReference}`).
<resto del código>
}
```

```
addPlot(owner:String, formData: FormData) {
  this.httpClient.post<any>
(`http://` + window.env.FRONT_ENDPOINT + `8080/plots/file/${owner}`).
<resto del código>
}
```

- Archivo: **sensor.service.ts**

Está formado por la clase “SensorService” y contiene los siguientes métodos encargados de enviar las peticiones hacia el servicio **Sensor**:

```
getSensor(sensorId: String): Observable<Sensor[]> {
    return this.httpClient.get<Sensor[]>
(`http://` + window.env.FRONT_ENDPOINT + `:8081/sensor/${sensorId}`)
    <resto del código>
}
```

```
deleteSensor(sensorId: String) {
    return this.httpClient.delete<any>
(`http://` + window.env.FRONT_ENDPOINT + `:8081/sensor/${sensorId}`)
    <resto del código>
}
```

```
getSensors(catastralReference: String): Observable<Sensor[]> {
    return this.httpClient.get<Sensor[]>
(`http://` + window.env.FRONT_ENDPOINT + `:8081/sensor`, {
    <resto del código>
})
```

```
addSensor(sensorData: SensorRequest) {
    this.httpClient.post<any>
(`http://` + window.env.FRONT_ENDPOINT + `:8081/sensor`,
    <resto del código>}
}
```

El resto de servicios, al estar implementados con el framework de Spring Boot la configuración de la aplicación, URLs de conexión... recae en los archivos “application.properties”, situados dentro de la ruta “src/main/resources” en cada proyecto como puede observarse en la siguiente imagen tomando como ejemplo al servicio Plots (Ilustración 41):

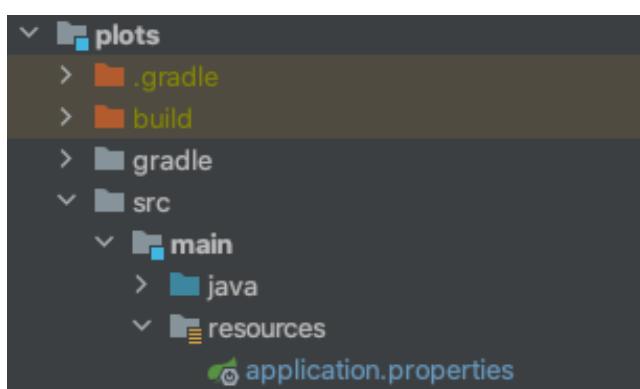


Ilustración 41 - Carpeta "application.properties" de Plots

Se utilizarán las variables de entorno definidas en el fichero .env y se modificará el código de cada servicio de la siguiente forma:

Servicio Plots

Este servicio se desplegará en el puerto 8080 y se conectará a una base de datos MongoDB. Mediante la utilización de “Spring Data MongoDB” se realiza la conexión a la base de datos indicando el host con valor \${MONGODB_HOST} y el puerto 27017. Se conectará a la base de datos “test” con el usuario “admin” y contraseña “example”, estableciendo también la base de datos “admin” con la que autentificará el usuario con el que se realiza la conexión.

```
spring.data.mongodb.host=${MONGODB_HOST}
spring.data.mongodb.port=27017
spring.data.mongodb.username=admin
spring.data.mongodb.password=example
spring.data.mongodb.database=test
spring.data.mongodb.authentication-database=admin
server.port=8080
```

Servicio Sensor

Este servicio se desplegará en el puerto 8081 y se conectará a una base de datos PosgreSQL. La conexión se realizará utilizando el conector JDBC, estableciendo el valor de host \${POSTGRES_HOST}, el puerto 5432 y la base de datos “sensors”. Esta base de datos será creada posteriormente en el fichero “docker-compose” al lanzar la base de datos a través de una consulta SQL (explicada más adelante), puesto que PosgreSQL no soporta operaciones para crear bases de datos en caso de que no se encuentren creadas previamente. Finalmente, el usuario de conexión será “postgres” y la contraseña “password”.

Utilizará algunas operaciones de JPA e Hibernate que definen cómo se comportará la base de datos al conectar este servicio:

- **spring.jpa.database=POSGRESQL:** indicará que se realizará la conexión a una base de datos de tipo PostgreSQL.
- **spring.jpa.show-sql=true:** activan los logs de las operaciones.
- **spring.jpa.generate-ddl=true:** inicializa los esquemas de la base de datos al iniciarse.
- **spring.jpa.hibernate.ddl-auto=update:** crea los esquemas que se va a utilizar, como las tablas y sus columnas, o los actualiza si se han realizado cambios.
- **spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true:** se utiliza para resolver un error de JPA en postgres para que funcione correctamente.

```
spring.jpa.database=POSTGRESQL
spring.datasource.url=jdbc:postgresql://${POSTGRES_HOST}:5432/sensors
spring.datasource.username=postgres
spring.datasource.password=password
spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
server.port=8081
```

Servicio Authentication

Este servicio se desplegará en el puerto 8999 y realizará la conexión con una base de datos MySQL. Se realiza la conexión mediante el conector JDBC al host con valor \${MYSQL_HOST}, en el puerto 3306 y la base de datos “users”, la cual será creada si no existía previamente. El servicio se conectará con el usuario “root” y la contraseña “password”.

Las operaciones JPA e Hibernate utilizadas son:

- `spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver`: se establece el driver de MySQL que se va a utilizar.
- `spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect`: referenciar el dialecto que necesita el Hibernate para trabajar con la base de datos. Se especifica en función de la versión de MySQL utilizada, en este caso es la 5.7.
- `spring.jpa.show-sql=true`: activan los logs de las operaciones.
- `spring.jpa.generate-ddl=true`: inicializa los esquemas de la base de datos al iniciarse.
- `spring.jpa.hibernate.ddl-auto=update`: crea los esquemas que se va a utilizar, como las tablas y sus columnas, o los actualiza si se han realizado cambios.

```
jwt.secret=javainuse
spring.datasource.url=jdbc:mysql://${MYSQL_HOST}:3306/users?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=password
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
server.port=8999
```

Servicio Statistics

Este servicio se desplegará en el puerto 8082 y se conectará a una base de datos de Cassandra. Mediante la utilización de “Spring Data Cassandra” se realiza la conexión a la base de datos, indicando el host con valor \${CASSANDRA_HOST} y el puerto 9042. Se conectará al nodo del keyspace “sensorKeyspace” con el usuario “cassandra”, la contraseña “cassandra” y el centro de datos “datacenter1”.

Se crean los esquemas si no estaban creados previamente mediante la operación:

- spring.data.cassandra.schema-action=create_if_not_exists

```
spring.data.cassandra.keyspace-name=sensorKeyspace  
spring.data.cassandra.contact-points=${CASSANDRA_HOST}  
spring.data.cassandra.port=9042  
spring.data.cassandra.username=cassandra  
spring.data.cassandra.password=cassandra  
spring.data.cassandra.schema-action=create_if_not_exists  
spring.data.cassandra.local-datacenter=datacenter1  
server.port=8082
```

6.1.2. Creación de los Dockerfiles

Una vez realizadas las modificaciones del código de aplicación necesarios, se crearán sus imágenes Docker a partir de los ficheros Dockerfile.

Con el empaquetado del código de los microservicios se crearán los ejecutables que utilizarán los contenedores. En el caso del Front, más que empaquetar se copiarán todos los ficheros dentro del contenedor, y en el caso de los servicios API REST, se crearán los archivos .jar pertinentes con la funcionalidad del servicio. Para ello, al estar utilizando Gradle como herramienta de automatización y construcción del código, se hará uso del siguiente comando dentro de la ruta correspondiente a cada proyecto API REST:

```
./gradlew bootJar
```

Esto generará dentro de la ruta “build/libs” de cada proyecto su ejecutable correspondiente (Ilustración 42).

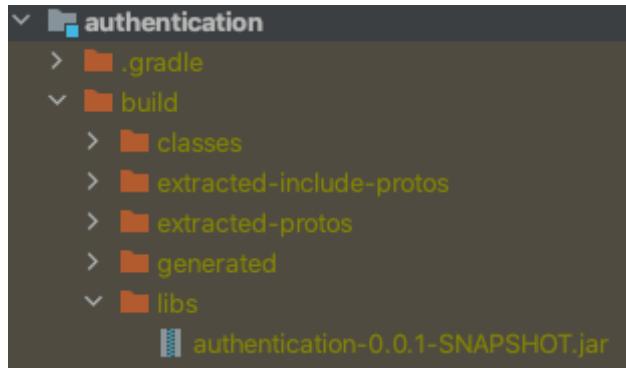


Ilustración 42 - Carpeta "libs" con el archivo ".jar"

A continuación, antes de proceder a la creación de las imágenes Docker para generar los contenedores, es necesario configurar sus correspondientes ficheros Dockerfile. No se elaborarán los Dockerfile para todos los microservicios puesto que algunas imágenes se importarán de DockerHub directamente al lanzar los contenedores con el fichero Docker-Compose. Este será el caso de las bases de datos de MongoDB, MySQL y PostgreSQL.

Dockerfile del servicio Tfm-front-end

Para poder ejecutar el front es necesario que la imagen tenga instalado NodeJS, por lo que esta imagen será construida partir de una imagen “node:16” mediante la instrucción FROM. Estará configurada con todo lo necesario para usar de los comandos de node y npm necesarios.

```
FROM node:16
WORKDIR /app
RUN curl -L -o /usr/bin/jq
https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
RUN chmod u+rx /usr/bin/jq
COPY ["package.json", "package-lock.json", "./"]
RUN npm install --save --legacy-peer-dep && npm cache clean --force
COPY docker-env-js-to-script.sh /
RUN ["chmod", "+x", "/docker-env-js-to-script.sh"]
COPY . /app
EXPOSE 4200
ENTRYPOINT ["/docker-env-js-to-script.sh"]
CMD ["npm", "start"]
```

En el proceso de construcción se crea directorio de trabajo “/app” dentro del contenedor mediante la instrucción WORKDIR para establecer el directorio donde se van a ejecutar el resto de instrucciones del Dockerfile. Se instala la herramienta “jq” con RUN y se le da permisos de ejecución (este comando se utiliza en el script de configuración). Seguidamente se copian con COPY los archivos “package.json” y “package-lock.json” del proyecto con todas las dependencias dentro del directorio para después instalarlas con el comando “npm install”. También se copia el script “docker-env-js-to-script.sh” explicado anteriormente y se le da permisos de ejecución. Luego se copian todos los

ficheros del proyecto (carpeta actual “.”) al directorio “/app” y se realiza un EXPOSE para especificar a Docker que el contenedor se expondrá en el puerto 4200. Finalmente, el script de configuración se establece en la instrucción ENTRYPOINT para ejecutarse tras iniciar el contenedor y con CMD se especifica el comando “npm start” para iniciar la aplicación.

De forma complementaria, se establece un fichero “.dockerignore” situado en la raíz del proyecto con una lista de carpetas y archivos que se descartarán al ejecutar el Dockerfile.

- Fichero **.dockerignore**

```
# Node
node_modules
npm-debug.log
```

La carpeta “node_modules” con las dependencias actuales del proyecto no se incluirá en el contenedor puesto que dentro de este ya se instalarán a partir del “package.json” con “npm start”.

Los siguientes ficheros Dockerfile corresponden con los servicios API REST. Sus procesos de construcción son idénticos: se necesita Java 11 para poder ejecutar los .jar generados, por lo que se importa la imagen “adoptopenjdk/openjdk11:alpine-jre” en el FROM. Se realiza una copia del ejecutable, almacenado como se ha explicado previamente en la ruta “build/libs”, dentro del directorio raíz del contenedor con el nombre “app.jar”. Finalmente, se realiza un EXPOSE de cada servicio en el puerto donde se despliega, y se establece el comando “java -jar /app.jar” en el ENTRYPOINT para ejecutar la aplicación cuando se inicie el contenedor.

Estos son sus Dockerfiles:

Dockerfile del servicio Plots

```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY build/libs/plots-0.0.2-SNAPSHOT.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerfile del servicio Sensor

```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY build/libs/sensor-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerfile del servicio Authentication

```
FROM adoptopenjdk/openjdk11:alpine-jre
WORKDIR /app
COPY build/libs/authentication-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8999
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerfile del servicio Statistics

```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY build/libs/statistics-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8082
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Por último, se ha realizado un Dockerfile para Cassandra debido a que necesita configuración para inicializarse correctamente:

Dockerfile del servicio Cassandra

El Dockerfile configurará la base de datos para el correcto funcionamiento del servicio Statistics, puesto que necesita que dentro de Cassandra se encuentre creado un keyspace. Para ello, se han elaborado los ficheros “cassandra_init-entrypoint.sh” e “init_keyspace.cql”. Tanto el Dockerfile como los ficheros utilizados estarán localizados en la carpeta “cassandra” dentro de la raíz del proyecto (Ilustración 43):

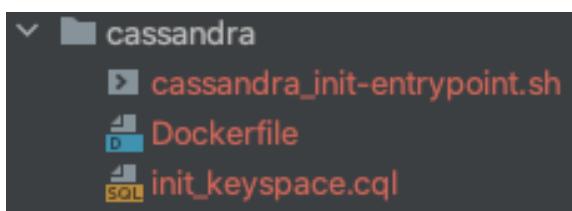


Ilustración 43 - Carpeta ficheros Cassandra

```
FROM cassandra:4.1.0
COPY cassandra_init-entrypoint.sh /
COPY init_keyspace.cql /docker-entrypoint-initdb.d/
RUN chmod +x cassandra_init-entrypoint.sh
EXPOSE 9042
ENTRYPOINT ["/cassandra_init-entrypoint.sh"]
```

Se utiliza una imagen de Cassandra con versión 4.1.0. Los ficheros de configuración se copian dentro del contenedor, el primer el script se dirigirá a la carpeta raíz y el segundo el fichero .sql a la carpeta “docker-entrypoint-

initdb.d". Luego se realiza el EXPOSE al puerto 9042 del contenedor donde escuchará el servicio y se ejecuta el script como entrada del ENTRYPOINT, pero antes se le da permisos de ejecución con el comando "chmod" dentro del RUN.

- Fichero **init_keyspace.cql**

Consulta de creación del keyspace "sensorKeyspace" en el caso de que no exista previamente gracias a la condición de creación "IF NOT EXISTS".

```
CREATE KEYSPACE IF NOT EXISTS sensorKeyspace
    WITH REPLICATION = {
        'class' : 'SimpleStrategy',
        'replication_factor' : 1
    };
```

- Fichero **cassandra_init-entrypoint.sh**

El script se ha obtenido del repositorio de GitHub oficial de Cassandra (<https://github.com/docker-library/cassandra>), más concretamente del fichero "docker-entrypoint.sh" encargado de inicializar y poner en marcha el contenedor por defecto en el ENTRYPOINT al construir la imagen. Solo se ha añadido el siguiente código al final del fichero:

```
for f in docker-entrypoint-initdb.d/*; do
    case "$f" in
        *.sh)      echo "$0: running $f"; . "$f" ;;
        *.cql)     echo "$0: running $f" && until cqlsh -f "$f"; do >&2
echo "Cassandra is unavailable - sleeping"; sleep 2; done &;;
        *)        echo "$0: ignoring $f" ;;
    esac
    echo
done

exec "$@"
```

Consiste en un bucle que recorre los ficheros dentro de la carpeta "docker-entrypoint-initdb.d" creada durante el Dockerfile ejecutando los ficheros con extensión .sh y .cql. Al copiar el fichero "init_keyspace.cql" en esta carpeta se ejecutará la consulta de creación del keyspace, y hasta que no termine la operación, no se devolverá el control al script original "docker-entrypoint" por defecto (exec "@"). De esta manera se pueden realizar operaciones sobre la base de datos justo después de su configuración inicial.

6.1.3. Construcción de las imágenes

Para crear las imágenes se ejecuta el comando “docker build” dentro de la raíz de cada proyecto (“.”) donde se encuentran los Dockerfiles. En el mismo comando, se etiquetan a través de la opción “-t” para identificar la imagen junto a su versión. Además, como más adelante se subirán a DockerHub es necesario que cumpla el siguiente formato:

```
docker build -t repositorio/imagen:tag .
```

Como puede verse el comando anterior, para etiquetar una imagen se necesita especificar un repositorio (nombre de usuario en DockerHub), el nombre de la imagen y una versión (por defecto la más reciente o “latest”). No obstante, no es necesario adaptarse completamente a este formato, pero es recomendable.

Los siguientes comandos crean las imágenes de los servicios, etiquetándolas con la versión “latest”:

```
docker build -t jesusbc/sensor:latest .
docker build -t jesusbc/plots:latest .
docker build -t jesusbc/tfm-front-end:latest .
docker build -t jesusbc/statistics:latest .
docker build -t jesusbc/authentication:latest .
docker build -t jesusbc/cassandra:latest .
```

Para ver la lista de imágenes creadas (Ilustración 44) se utiliza el siguiente comando:

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jesusbc/tfm-front-end	latest	fa435917621d	3 weeks ago	2.09GB
jesusbc/statistics	latest	0091b296608e	3 weeks ago	185MB
jesusbc/plots	latest	196335597c84	3 weeks ago	186MB
jesusbc/authentication	latest	812c51ed6721	3 weeks ago	219MB
jesusbc/sensor	latest	8c751adfd5a2	3 weeks ago	198MB
jesusbc/cassandra	latest	e60f3e2420d7	4 weeks ago	355MB

Ilustración 44 - Lista de imágenes Docker

6.1.4. Definición del lanzamiento de contenedores con Docker-Compose

Mediante el fichero Docker-Compose, se configurarán todos servicios de contenedor y sus parámetros de lanzamiento como su imagen, el puerto tanto del contenedor como de la máquina host donde se desplegará, variables de entorno... Esto es equivalente a lanzar uno a uno los contenedores mediante el comando estándar “docker run” (también especificando los puertos, la

imagen...). Además, dentro del fichero se permite la definición de volúmenes para la persistencia de datos de los contenedores. En la siguiente figura (Ilustración 45) se puede ver la estructura del fichero Docker-Compose con el listado de los nombres de servicios de contenedores y volúmenes utilizados:

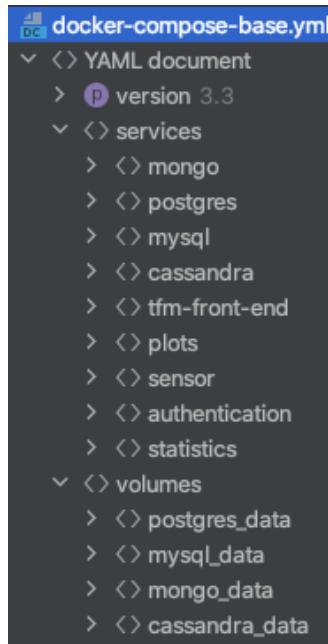


Ilustración 45 - Estructura Docker-Compose

A continuación, se mostrará primero la configuración de lanzamiento compose de los servicios Front y API REST y después los servicios de bases de datos junto a sus volúmenes:

Servicio contenedor Tfm-front-end

El servicio tfm-front-end se desplegará en el puerto 4200. En Docker, la redirección de puertos se indica mediante dos puntos “：“, donde el puerto de la izquierda corresponde al de la máquina host y el de la derecha al contenedor. De esta manera, el contenedor dentro de Docker puede ser accedido desde Internet a través del puerto expuesto de la máquina host.

```
tfm-front-end:
  image: jesusbc/tfm-front-end:latest
  container_name: tfm-front-end
  ports:
    - "4200:4200"
  env_file:
    - env_vars.env
```

Tanto a este contendor como el resto de los servicios definidos en el fichero, a excepción de los servicios de bases de datos, se les podrá pasar la lista de variables de entorno del fichero “env_vars.env” gracias a la opción “env_file”.

Servicio contenedor Plots

El servicio Plots se desplegará en el puerto 8080. Se lanzará el contenedor una vez que el servicio de MongoDB llamado **mongo** se encuentre levantado, puesto que para realizar una conexión con este la base de datos debe estar primero activa.

```
plots:
  image: jesusbc/plots:latest
  container_name: plots
  ports:
    - "8080:8080"
  depends_on:
    - mongo
  env_file:
    - env_vars.env
```

Servicio contenedor Sensor

El servicio Sensor se desplegará en el puerto 8081. Se lanzará el contenedor una vez que el servicio de PostgreSQL llamado **postgres** se encuentre levantado.

```
sensor:
  image: jesusbc/sensor:latest
  container_name: sensor
  ports:
    - "8081:8081"
  depends_on:
    - postgres
  env_file:
    - env_vars.env
```

Servicio contenedor Authentication

El servicio Autenticacion se desplegará en el puerto 8999. Se lanzará el contenedor una vez que el servicio de MySQL llamado **mysql** se encuentre levantado.

```
authentication:
  image: jesusbc/authentication:latest
  container_name: authentication
  ports:
    - "8999:8999"
  depends_on:
    - mysql
  env_file:
    - env_vars.env
```

Servicio contenedor Statistics

El servicio Statistics se desplegará en el puerto 8082. Se lanzará el contenedor una vez que el servicio de Cassandra llamado **cassandra** se encuentre levantado. Tiene la opción “restart: always” activada para que se levante el servicio constantemente si en algún caso se deja de ejecutar. Esto es debido a que Cassandra tarda un par de minutos en iniciarse, lo que provoca que el servicio Statistics falle al establecer la conexión hasta que finalmente se conecta con la base de datos.

```
statistics:  
  image: jesusbc/statistics:latest  
  container_name: statistics  
  ports:  
    - "8082:8082"  
  restart: always  
  depends_on:  
    - cassandra  
  env_file:  
    - env_vars.env
```

Una vez configurados los servicios API REST, se procede con la configuración de lanzamiento de los contenedores de bases de datos. Estos utilizarán **volúmenes Docker** para la persistencia de datos. El montaje de los volúmenes se realiza sobre la ruta donde cada base de datos almacena sus datos. Es importante el uso de volúmenes ya que, si el servicio contenedor se destruyera, también lo harían los datos. De esta manera los datos estarán a disposición del contenedor cada vez que se vuelta a lanzar dentro de la ruta de montaje.

La ruta por defecto en el equipo donde Docker crea los volúmenes es “var/lib/docker/volumes” y su definición dentro del compose es la siguiente:

```
volumes:  
  postgres_data:  
  mysql_data:  
  mongo_data:  
  cassandra_data:
```

Servicio contenedor Mongo

El servicio Mongo utilizará una imagen mongo:5.0.7 y será accesible por el puerto 27017. Dentro del apartado de volúmenes se realizará el montaje del volumen “mongo_data” sobre la ruta “/data/db” donde MongoDB almacena los datos.

```
mongo:
  image: mongo:5.0.7
  container_name: mongodb
  volumes:
    - mongo_data:/data/db
    - ./init-mongodb.js:/docker-entrypoint-initdb.d/init-mongodb.js:ro
  environment:
    - MONGO_INITDB_ROOT_USERNAME=admin
    - MONGO_INITDB_ROOT_PASSWORD=example
  ports:
    - "27017:27017"
```

Dentro de este apartado también se pueden montar ficheros dentro de otras rutas del contenedor. En este caso se almacena el script “init-mongodb.js” en la carpeta “docker-entrypoint-initdb.d” para la configuración inicial de la base de datos. Este fichero init se almacenará en dicho directorio debido a que, muchas imágenes Docker oficiales de servicios de bases de datos utilizan esta carpeta “initdb.d” para que el usuario almacene los ficheros con las operaciones que desee ejecutar al lanzar el contenedor.

- Fichero **init-mongo.js**

```
db.createUser(
{
  user: "admin",
  pwd: "example",
  roles: [
    {
      role: "readWrite",
      db: "test"
    }
  ]
);

db = new Mongo().getDB("test");
db.createCollection('layers', { capped: false });
db.layers.insert({
  "_id" : ObjectId("611d1ad5dfb24e64eb994bce"),
  "owner" : "pepe",
  "catastralReference" : "10022A010000380000WZ",
  "geoData" : {
    <resto del código>
```

Este script creará un usuario “admin” con contraseña “example” dentro de la base de datos “test” a la que se conectará el servicio Plots, además de crear la colección “layers” donde se insertará el primer documento de parcela inicial.

Otro detalle importante en el compose es la especificación de las variables de entorno MONGO_INITDB_ROOT_USERNAME y MONGO_INITDB_ROOT_PASSWORD. Son utilizadas en conjunto para definir un usuario root junto a su contraseña en la base de datos de autenticación llamada “admin” propia de mongo. Estas deben ser especificadas debido al complejo sistema de autenticación de usuarios de MongoDB, una tarea necesaria para que cuando se conecte el servicio Plots tenga privilegios suficientes para operar en ella.

Servicio contenedor Postgres

El servicio Postgres utilizará la última versión de la imagen de la base de datos y será accesible por el puerto 5432. Dentro del apartado de volúmenes se realizará el montaje del volumen “postgres_data” sobre la ruta “/var/lib/postgresql/data” donde PostgreSQL almacena los datos.

```
postgres:
  image: postgres
  container_name: postgres
  volumes:
    - postgres_data:/var/lib/postgresql/data
    - ./init-postgres.sql:/docker-entrypoint-initdb.d/init-postgres.sql
  environment:
    - POSTGRES_PASSWORD=password
    - POSTGRES_USER=postgres
  ports:
    - '5432:5432'
```

En las variables de entorno se especificará el nombre y contraseña del usuario raíz. El establecimiento de un valor de contraseña para la variable “POSTGRES_PASSWORD” es obligatorio.

Dentro de la carpeta “docker-entrypoint-initdb.d” del contenedor se introducirá el fichero “init-postgres.sql”, que contendrá una consulta para crear la base de datos “sensors” en el momento de su inicialización, de la misma forma con la que se inició MongoDB anteriormente.

- Fichero **init-postgres.sql**

```
SELECT 'CREATE DATABASE sensors'  
WHERE NOT EXISTS (SELECT FROM pg_database WHERE datname =  
'sensors') \gexec
```

Fichero con una consulta SQL para crear la base de datos “sensors” en caso de que no existiera previamente gracias al parámetro “\gexec”. PostgreSQL no soporta operaciones “create if not exist” para una base de datos, por lo que esta consulta permite simular este tipo de operación.

La consulta del fichero “init-postgres.sql” se ha obtenido a partir del siguiente enlace:

- <https://stackoverflow.com/questions/18389124/simulate-create-database-if-not-exists-for-postgresql>

Servicio contenedor Mysql

El servicio Mysql utilizará una imagen mysql:5.7 y será accesible por el puerto 3306. Dentro del apartado de volúmenes se realizará el montaje del volumen “mysql_data” sobre la ruta “/var/lib/mysql” donde MySQL almacena los datos.

```
mysql:  
  image: mysql:5.7  
  container_name: mysql  
  volumes:  
    - mysql_data:/var/lib/mysql  
  environment:  
    MYSQL_ROOT_PASSWORD: 'password'  
  ports:  
    - '3306:3306'
```

Se deberá definir obligatoriamente la variable de entorno “MYSQL_ROOT_PASSWORD” para especificar el valor de contraseña del usuario raíz.

Servicio contenedor Cassandra

El servicio Cassandra utilizará la imagen “jesusbc/cassandra:latest” construida previamente y será accesible por el puerto 9042. Dentro del apartado de volúmenes se realizará el montaje del volumen “cassandra_data” sobre la ruta “/var/lib/cassandra” donde Cassandra almacena los datos.

```
cassandra:
  image: jesusbc/cassandra:latest
  container_name: cassandra
  volumes:
    - cassandra_data:/var/lib/cassandra
  ports:
    - "9042:9042"
```

Tras la refactorización y configuración del código del proyecto este se encontrará listo para su despliegue. Para lanzar los contenedores se utilizará el fichero “docker-compose-base.yml” del proyecto con el siguiente comando:

```
docker-compose -f docker-compose-base.yml up
```

Mediante el comando “docker ps” se podrá observar la lista de contenedores para ver que se han ejecutado correctamente. También se podrá ver la lista con los volúmenes creados con el comando “docker volumen ls” (Ilustración 46).

```
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d88fa21858da jesusbc/sensor:latest "java -jar app.jar" 14 minutes ago Up 14 minutes 0.0.0.0:8081->8081/tcp sensor
1585a7119d99 jesusbc/plots:latest "java -jar app.jar" 14 minutes ago Up 14 minutes 0.0.0.0:8080->8080/tcp plots
cc2f9fe8c6d7 jesusbc/authentication:latest "java -jar app.jar" 14 minutes ago Up 14 minutes 0.0.0.0:8999->8999/tcp authentication
64d74e88778fa jesusbc/statistics:latest "java -jar app.jar" 14 minutes ago Up 18 minutes 0.0.0.0:8082->8082/tcp statistics
4965f97a1db4 jesusbc/tfm-front-end:latest "/docker-env-js-to-s..." 14 minutes ago Up 14 minutes 0.0.0.0:4200->4200/tcp tfm-front-end
f13203eab3d1 jesusbc/cassandra:latest "/cassandra_init.s..." 14 minutes ago Up 14 minutes 7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp cassandra
7a369bbddcc mongo:5.0.7 "docker-entrypoint.s..." 12 days ago Up 14 minutes 0.0.0.0:27017->27017/tcp mongodb
c7aa478ad123 mysql:5.7 "docker-entrypoint.s..." 2 months ago Up 14 minutes 0.0.0.0:3306->3306/tcp, 33060/tcp mysql
e5eb54a1e82b postgres "docker-entrypoint.s..." 2 months ago Up 14 minutes 0.0.0.0:5432->5432/tcp postgres
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker volume ls
DRIVER VOLUME NAME
local 3cdf818c0efec971e3b7abe112b6aadeaa7f33ee17844c65d38a8eb1a84b4ec
local 72439b7330e260cf6d72daaebe82d7a32ea5b0d61243170edeef025596956e94
local f33ed6f7391e5c6965a759880acac75e2b3ac6e44f4f688975dc6c861746f354
local tfm_cassandra_data
local tfm_mongo_data
local tfm_mysql_data
local tfm_postgres_data
MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ]
```

Ilustración 46 - Ejecución de contenedores local y lista de volúmenes

6.1.5. Datos y estructuras de las bases de datos al lanzarse la aplicación

Se mostrará una lista con el estado inicial que tendrán los servicios de bases de datos tras el lanzamiento de la aplicación una vez realizada la refactorización:

Base de datos MongoDB

- Usuario “admin” creado tanto en su base de datos de autorización “admin” como en la de “test” donde el servicio Plots se conecta.
- Colección “layers” para guardar las parcelas
- Una parcela inicial base dentro de “layers” con referencia catastral “10022A010000380000WZ”.

Base de datos Cassandra

El keyspace inicial “sensorkeyspace” y la tabla “statisticdata” dentro del keyspace.

Base de datos PostgreSQL

Base de datos “sensors” y la tabla “sensor” dentro de esta.

Base de datos MySQL

Base de datos “users” y la tabla “user” dentro de esta.

6.1.6. Subida de imágenes al repositorio de Docker-Hub

Para finalizar, se subirán las imágenes creadas a un repositorio propio público en Docker-Hub para su almacenamiento (Ilustración 47). Estas quedarán preparadas para su importación utilizando AWS en la siguiente fase del desarrollo.

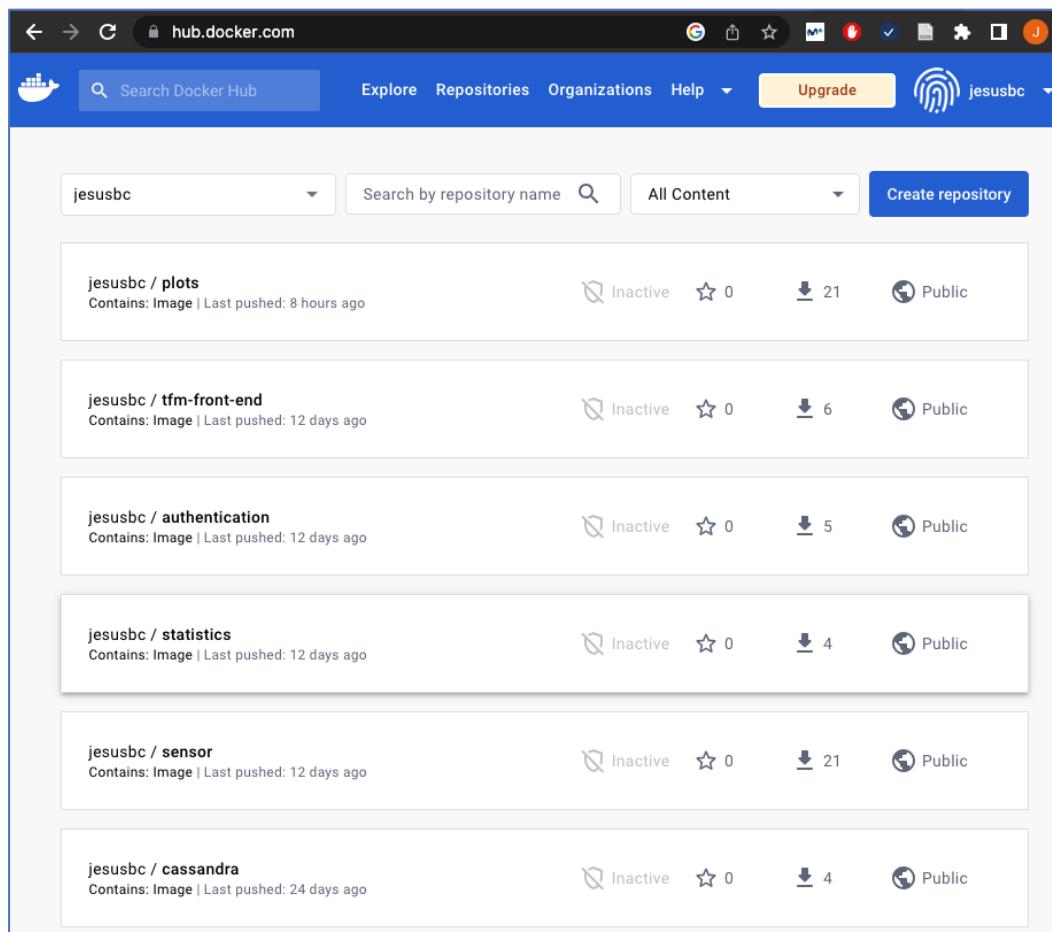


Ilustración 47 - Perfil Docker-Hub

Para subir las imágenes al repositorio, primero habrá que iniciar sesión con el usuario de Docker-Hub, y una vez logueado, se utiliza el comando “docker pull” para cada imagen que se desea subir:

```
docker push jesusbc/sensor:latest  
docker push jesusbc/plots:latest  
docker push jesusbc/tfm-front-end:latest  
docker push jesusbc/statistics:latest  
docker push jesusbc/authentication:latest  
docker push jesusbc/cassandra:latest
```

Para importarlas, se utiliza el comando “docker pull” acompañado del nombre de la imagen que se deseé descargar.

```
docker pull jesusbc/sensor:latest  
docker pull jesusbc/plots:latest  
docker pull jesusbc/tfm-front-end:latest  
docker pull jesusbc/statistics:latest  
docker pull jesusbc/authentication:latest  
docker pull jesusbc/cassandra:latest
```

6.2. Migración y despliegue de la infraestructura en AWS

En esta segunda fase del desarrollo se realizará la migración de la aplicación refactorizada en la primera fase a la nube de AWS. Para ello se explicarán todos los pasos a seguir para crear una infraestructura funcional con la que se podrán desplegar aplicaciones basadas en microservicios y contenedores. En cuanto a los servicios de AWS utilizados, se irán creando de forma ordenada a lo largo de este apartado, desde la creación de la VPC o red privada donde se desplegarán el resto de los servicios, como las subredes, máquinas EC2, sistemas de almacenamiento...

- Fichero **env_vars.env** para el despliegue en la nube:

El fichero de variables de entorno “env_vars.env” volverá a utilizarse con nuevos valores para los hosts. Como se explicó anteriormente durante el despliegue local, los hosts que se estaban utilizando eran tanto “localhost” como los propios nombres de los contenedores Docker ejecutándose en la misma máquina y red. Ahora, antes de lanzar los contenedores con los microservicios API REST y Front de la aplicación en AWS, sólo habría que modificar los valores de los hosts del fichero una vez estuviera desplegada toda la infraestructura. Estos valores se corresponden con las direcciones IP o DNS de las máquinas donde estén desplegados los servicios de base de datos, de cara a realizar la conexión con sus correspondientes servicios API REST. En el caso del servicio Front, se utilizará un servicio de balanceo de carga que permitirá redireccionar las peticiones hacia los demás

microservicios. Este proceso se automatizará al utilizar Terraform, donde se modificará este fichero de forma dinámica a medida que despliega la infraestructura.

```
# Endpoint Front-End
FRONT_ENDPOINT=ALB-principal-1848667940.us-east-1.elb.amazonaws.com
# Servicio cliente dentro de Plots
SENSOR_CLIENT=ALB-principal-1848667940.us-east-1.elb.amazonaws.com
# Endpoints Bases de Datos
POSTGRES_HOST=sensor-postgres.cvt9cc7dxazz.us-east-1.rds.amazonaws.com
MYSQL_HOST=authentication-mysql.cvt9cc7dxazz.us-east-1.rds.amazonaws.com
MONGODB_HOST=10.0.2.168
CASSANDRA_HOST=10.0.2.34
```

- **Región y Zonas de disponibilidad**

La región de AWS donde se trabajará será el **Norte de Virginia (us-east-1)** y sus zonas de disponibilidad elegidas serán **us-east-1a** principalmente y **us-east-1b**.

- **Comprobación del despliegue AWS**

La comprobación del despliegue de la infraestructura se ha realizado en el apartado de [8. Pruebas](#) del documento, más concretamente en [8.2.2. Despliegue de servicios en AWS](#)

- **Par de claves (SSH)**

De cara a realizar conexiones con las instancias de máquinas EC2 se creará un **par de claves**. Consiste en una clave pública que se almacena en la máquina EC2 y otra clave privada que el desarrollador deberá almacenar ya que será necesaria para establecer una conexión SSH con las instancias EC2. De esta manera se añade una capa más de seguridad permitiendo conexiones más seguras.

Para su creación, habrá que dirigirse a la sección de “EC2”, y dentro de esta al apartado de “Par de claves” dentro de AWS. El tipo de encriptación de la clave será RSA y el formato de tipo “.pem” ya que se está trabajando con Linux, en el caso de utilizar Windows se recomienda seleccionar “.ppk”. Cada vez que se establezca una conexión con una máquina EC2 se deberá proporcionar esta clave, por lo que habrá que guardarla en el equipo.

Archivo resultante: clave_maquinas_TFG.pem (Ilustración 48).

The screenshot shows the 'Create key pair' step in the AWS EC2 console. It includes fields for 'Name' (set to 'clave_maquinas_TFG'), 'Type' (set to 'RSA'), 'Format' (set to '.pem'), and an 'Add tag' button.

EC2 > Pares de claves > Crear par de claves

Crear par de claves Información

Par de claves
Un par de claves, compuesto por una clave privada y una clave pública, es un conjunto de credenciales de seguridad que se utilizan para demostrar su identidad cuando se conecta a una instancia.

Nombre
clave_maquinas_TFG
El nombre puede incluir hasta 255 caracteres ASCII. No puede incluir espacios al principio ni al final.

Tipo de par de claves Información
 RSA
 ED25519

Formato de archivo de clave privada
 .pem
Para usar con OpenSSH
 .ppk
Para usar con PuTTY

Etiquetas: *opcional*
No hay etiquetas asociadas a este recurso.

Agregar nueva etiqueta
Puede agregar hasta 50 etiquetas más.

Ilustración 48 - Creación de par de claves

A continuación, se estructurará el despliegue de la infraestructura en los siguientes apartados:

6.2.1. Infraestructura de red

La construcción de una red es fundamental para que todos los elementos y servicios creados dentro de esta puedan comunicarse entre sí y con Internet. También permite la administración de esta al poder gestionar qué redes serán privadas, públicas, restringir o habilitar las direcciones IP que pueden acceder a un determinado servicio, el tipo de protocolo permitido... Es la base a partir de la cual se construirá el resto de la infraestructura.

Lista de servicios de red que se van a utilizar:

- VPC: 1
- Internet Gateway: 1
- IP Elástica: 1
- NAT Gateway: 1
- Tablas de enrutamiento: 3
- Subredes: 4
- Grupos de seguridad: 5

Estos servicios se encuentran en el **apartado “VPC”** dentro de AWS, a excepción de la IP elástica y los grupos de seguridad que están en el **apartado “EC2”**.

VPC

La VPC o “Virtual Private Cloud” creará una red virtual privada a partir de la cual se desplegarán la mayor parte de los servicios, además de ser el primer paso necesario para poder crear el resto de los recursos de red (Ilustración 49).

The screenshot shows the AWS VPC console interface. At the top, there is a header with the title "Sus VPC (1/2) Información" and a "Create VPC" button. Below the header is a search bar labeled "Filtrar las VPC". A table lists two VPCs:

Name	ID de la VPC	Estado	CIDR IPv4	CIDR IPv6
-	vpc-0159f42f2353df40e	Available	172.31.0.0/16	-
vpc-TFG	vpc-0cd81c07aefa4362d	Available	10.0.0.0/16	-

Below the table, a specific VPC is selected: "vpc-0cd81c07aefa4362d / vpc-TFG". The "Detalles" tab is active. The details shown include:

- ID de la VPC: vpc-0cd81c07aefa4362d
- Estado: Available
- Tenencia: Default
- Conjunto de opciones de DHCP: dopt-004e64d0b5e69221a
- Nombre de host de DNS: Habilitado
- Resolución de DNS: Habilitado
- CIDR IPv4: 10.0.0.0/16
- Grupos de reglas del firewall de DNS: Tabla de enruteamiento principal rtb-039bc241d753cde2 / tabla-rutas-VPC
- ACL de red principal: acl-0f7ff635b2c86f54f
- Métricas de uso de direcciones de red: Desactivado
- Grupo IPv6: -
- CIDR IPv6 (grupo de bordes de red): -
- ID de propietario: 690031176274

Ilustración 49 - VPC AWS

Características durante la creación de la VPC:

- **Nombre:** vpc-principal
- **Bloque de CIDR IPv4:** 10.0.0.0/16

Adicionalmente, se habilitarán las características de **Nombres de host de DNS** y **Resolución DNS** para todos los recursos creados dentro de la VPC en el apartado “Acciones > Editar la configuración de VPC”. Se activarán estas opciones ya que otros recursos de AWS utilizan el DNS, además de la dirección IP numérica, para tener una mayor flexibilidad a la hora de direccionar recursos (Ilustración 50).

The screenshot shows the "Configuración de DNS" (DNS Configuration) section. It contains two checked checkboxes:

- Habilitar la resolución de DNS [Información](#)
- Habilitar nombres de host DNS [Información](#)

Ilustración 50 - Configuración DNS

De forma automática, se generará una tabla de enruteamiento principal por defecto al crear la VPC. Se le dará nombre a esta para identificarla y solo

tendrá registrada una ruta, en este caso “10.0.0.0/16 – Local” para permitir el tráfico solo dentro de la VPC por el momento (Ilustración 51).

The screenshot shows the AWS Management Console interface for managing VPC routing tables. At the top, there's a search bar with 'search: vpc' and a 'Borrar filtros' (Clear filters) button. Below the search bar is a table header with columns: Name, ID de tabla de enrutamiento, Asociaciones de subredes, Asociaciones de borde, Prin..., VPC. A single row is selected: 'tabla-rutas-VPC' with ID 'rtb-0309bc241d753cde2'. The 'VPC' column shows 'Sí' and the ID 'vpc-0cd81c07aefa4362d | vpc-TFG'. Below the table, a detailed view for 'rtb-0309bc241d753cde2 / tabla-rutas-VPC' is shown. It has tabs for 'Detalles' (selected), 'Rutas' (selected), 'Asociaciones de subredes', 'Asociaciones de borde', 'Propagación de rutas', and 'Etiquetas'. Under the 'Rutas' tab, there's a table with one entry: 'Destino' (10.0.0.0/16), 'Destino' (local), 'Estado' (Activ), and 'Propagada' (No).

Ilustración 51 - Tabla VPC

Internet Gateway

Esta puerta de enlace permitirá la comunicación entre la VPC e Internet. Esta se asociará más adelante a la tabla de enrutamiento que utilizarán las subredes públicas para tener conexión con el exterior de la VPC (Ilustración 52).

The screenshot shows the AWS Management Console interface for managing Internet Gateways. At the top, there's a search bar with 'search: igw' and a 'Borrar filtros' (Clear filters) button. Below the search bar is a table header with columns: Name, ID de gateway de Internet, Estado, ID de la VPC. A single row is selected: 'igw-TFG' with ID 'igw-0fba8b3da4e39bc50'. The 'Estado' column shows 'Attached' and the 'ID de la VPC' column shows 'vpc-0cd81c07aefa4362d | vpc-TFG'. Below the table, a detailed view for 'igw-0fba8b3da4e39bc50 / igw-TFG' is shown. It has tabs for 'Detalles' (selected) and 'Etiquetas'. Under the 'Detalles' tab, there's a table with four columns: 'ID de gateway de Internet' (igw-0fba8b3da4e39bc50), 'Estado' (Attached), 'ID de la VPC' (vpc-0cd81c07aefa4362d | vpc-TFG), and 'Propietario' (690031176274).

Ilustración 52 – IGW AWS

Características durante la creación del IGW:

- **Nombre:** igw-principal
- **Conectada a la VPC:** vpc-principal

Tras crear el IGW se debe asociar con una VPC mediante “Acciones > Conectar a la VPC”.

Subredes

Dentro del rango de direcciones de la VPC se definirán algunas subredes dentro de la misma región en dos zonas de disponibilidad: us-east-1a y us-east-1b. La razón de utilizar dos zonas en vez de solo una, se debe a que algunos recursos las necesitan para funcionar correctamente, como es el caso del balanceador de carga ALB y los grupos de subredes necesarios para desplegar las RDS. Otra de las razones para utilizar varias zonas de disponibilidad para desplegar los recursos de AWS es que, si debido a algún problema algunas de las zonas dejan de dar servicio por algún tipo de problema, tener distribuidos o replicados estos recursos en varias zonas reduciría el impacto al poder seguir dando continuidad de servicio a los clientes.

Es necesario recordar que las subredes al ser creadas son privadas por defecto. Para que una subred sea pública, es necesario crear un Internet Gateway y añadir una ruta dentro de su tabla de enrutamiento que dirija el tráfico hacia este Gateway. Dicho esto, se han creado dos subredes públicas y dos privadas, tanto en la zona de disponibilidad us-east-1a como us-east-1b (Ilustración 53).

Subredes (10) Información						
<input type="checkbox"/>	Name	ID de subred	Estado	VPC	CIDR IPv4	
<input checked="" type="checkbox"/>	subred-servicios-1a	subnet-0cc3734c40221676d	Available	vpc-0cd81c07aefa4362d vpc...	10.0.0.0/24	
<input type="checkbox"/>	subred-front-1b	subnet-02ff8267200d84ab4	Available	vpc-0cd81c07aefa4362d vpc...	10.0.1.0/24	
<input type="checkbox"/>	subred-databases-1a	subnet-0506b3973e1b527d6	Available	vpc-0cd81c07aefa4362d vpc...	10.0.2.0/24	
<input type="checkbox"/>	subred-databases-1b	subnet-042c9bce890cf1ee6	Available	vpc-0cd81c07aefa4362d vpc...	10.0.3.0/24	

Ilustración 53 - Subredes AWS

1. Subred de servicios (us-east-1a)

Esta red será pública, es decir, los recursos desplegados dentro de esta podrán ser accedidos desde fuera de la VPC a través de la IGW. Dentro de la misma se desplegarán las máquinas virtuales EC2 con los servicios contenedores principales de la aplicación: Plots, Sensors, Authentication y Statistics.

Características durante la creación de la subred:

- **VPC:** vpc-principal
- **Nombre:** subred-servicios-1a
- **Zona de disponibilidad:** us-east-1a
- **Bloque de CIDR IPv4:** 10.0.0.0/24

2. Subred Front (us-east-1b)

Esta subred pública se aprovechará principalmente para desplegar el servicio de Front-End de la aplicación. Adicionalmente, se desplegará dentro de esta también el recurso de NAT Gateway creado más adelante.

Características durante la creación de la subred:

- **VPC:** vpc-principal
- **Nombre:** subred-front-1b
- **Zona de disponibilidad:** us-east-1b
- **Bloque de CIDR IPv4:** 10.0.1.0/24

3. Subred de bases de datos (us-east-1a)

Será una subred privada la cual tendrá acceso a Internet a través de una NAT Gateway que se creará posteriormente. Dentro de esta, se desplegarán los servicios de bases de datos con máquinas RDS y EC2 de una forma más segura sin exponer los datos de la aplicación directamente fuera de la VPC.

Características durante la creación de la subred:

- **VPC:** vpc-principal
- **Nombre:** subred-databases-1a
- **Zona de disponibilidad:** us-east-1a
- **Bloque de CIDR IPv4:** 10.0.2.0/24

4. Subred de bases de datos (us-east-1b)

Al igual que la subred anterior, esta también será privada y servirá para desplegar los servicios de bases de datos. También es necesario disponer de dos subredes en distintas zonas de disponibilidad de cara a la creación de un grupo de subredes, requerido durante la definición de las RDS.

Características durante la creación de la subred:

- **VPC:** vpc-principal
- **Nombre:** subred-databases-1b
- **Zona de disponibilidad:** us-east-1b
- **Bloque de CIDR IPv4:** 10.0.3.0/24

IP Elástica

De cara a la posterior creación de una NAT Gateway se necesita definir un recurso de IP elástica. Es una dirección IP estática proporcionada por AWS que no cambia con el tiempo, al contrario que las IP públicas autoasignadas al crear por ejemplo una instancia EC2, lo que permite direccionar a la misma IP a un recurso asignado (Ilustración 54).

Direcciones IP elásticas (1/1)				
<input type="button" value="C"/> Acciones ▾ <input type="button" value="Asignar la dirección IP elástica"/>				
<input type="text"/> Filtrar direcciones IP elásticas				
✓ Name	Dirección IPv4 asig...	Tipo	ID de asignación	Re
ip-elastica-nat	3.218.225.253	IP pública	eipalloc-0397c2a3e371ee992	-

Resumen

Dirección IPv4 asignada <input type="text" value="3.218.225.253"/>	Tipo <input type="text" value="IP pública"/>	ID de asignación <input type="text" value="eipalloc-0397c2a3e371ee992"/>	Registro DNS inverso -
ID de asociación <input type="text" value="eipassoc-00cc499762bd6fc7a"/>	Ámbito <input type="text" value="VPC"/>	ID de la instancia asociada -	Dirección IP privada <input type="text" value="10.0.1.99"/>
ID de la interfaz de red eni-0983347ef0eedea23	ID de la cuenta del propietario de la interfaz de red <input type="text" value="690031176274"/>	DNS público <input type="text" value="ec2-3-218-225-253.compute-1.amazonaws.com"/>	ID de la gateway NAT nat-05254623edc53e252 (nat-gateway-principal)
Grupo de direcciones <input type="text" value="Amazon"/>	Grupo fronterizo de red <input type="text" value="us-east-1"/>		

Ilustración 54 - IP elástica AWS

Características durante la creación de la IP Elástica:

- **Nombre:** ip-elastica-nat
- **Grupo fronterizo de red:** us-east-1

Esta IP actuará dentro de la región (grupo fronterizo) para direccionar a la NAT Gateway que se creará a continuación.

NAT Gateway

Este tipo de puerta de enlace permitirá a las subredes privadas el acceso a Internet para poder conectarse a servicios fuera de la VPC, pero con la seguridad de que los servicios externos no podrán iniciar una conexión con los recursos internos (Ilustración 55).

Name	ID de gateway NAT	Tipo de con...	Estado	Mensaje de est...	Direcció...
nat-gateway-principal	nat-05254623edc53e252	Public	Available	-	3.218.225.253

Ilustración 55 - NAT AWS

Características durante la creación de la NAT Gateway:

- **Nombre:** nat-gateway-principal
- **Subred asociada:** subred-front-1b
- **Tipo de conectividad:** Pública
- **Asignación IP elástica:** ip-elastica-nat

El tipo de conectividad pública permite que las instancias pertenecientes a las subredes privadas puedan acceder a Internet, impidiendo las conexiones entrantes no deseadas de Internet. No se escoge la opción privada puesto que solo permitiría la conexión con otras VPC o a la red interna.

Tablas de enrutamiento

Además de la tabla de enrutamiento generada durante la creación de la VPC, se crearán dos tablas más para dirigir correctamente el tráfico hacia las subredes según sus necesidades (Ilustración 56).

Tablas de enrutamiento (4) Información						
<input type="checkbox"/>	Name	ID de tabla de...	Asociacione...	Asocia...	Prin...	VPC
<input type="checkbox"/>	tabla-rutas-VPC	rtb-0309bc241d...	-	-	Sí	vpc-0cd81c07aefaa4362d vpc-principal
<input type="checkbox"/>	tabla-rutas-internet	rtb-0e36f8d957c...	2 subredes	-	No	vpc-0cd81c07aefaa4362d vpc-principal
<input type="checkbox"/>	tabla-rutas-databases	rtb-08e72b887f4...	2 subredes	-	No	vpc-0cd81c07aefaa4362d vpc-principal

Ilustración 56 - Tablas AWS

Para su creación, habrá que establecer el nombre la tabla y asociarla a una VPC. En este momento se añaden las rutas hacia el Internet Gateway y al NAT Gateway creados anteriormente en sus correspondientes tablas para dar conectividad con el exterior. Para ello, se establecerán las rutas mediante el apartado “Acciones > Editar rutas”.

1. Tabla de rutas con Internet Gateway

Esta tabla contendrá dos rutas, una para dirigir el tráfico dentro de la VPC a través de “10.0.0.0/16 – Local” y otra que lo dirija hacia el IGW “0.0.0.0/0 – igw-0fba8b3da4e39bc50” (Ilustración 57).

Tablas de enrutamiento (1/4) Información						
<input type="checkbox"/>	Name	ID de tabl...	Asociaciones de...	Asocia...	Prin...	VPC
<input type="checkbox"/>	tabla-rutas-VPC	rtb-0309bc241d...	-	-	Sí	vpc-0cd81c07aefaa4362d vpc-principal
<input checked="" type="checkbox"/>	tabla-rutas-internet	rtb-0e36f8d957c...	2 subredes	-	No	vpc-0cd81c07aefaa4362d vpc-principal
<input type="checkbox"/>	tabla-rutas-databases	rtb-08e72b887f4...	2 subredes	-	No	vpc-0cd81c07aefaa4362d vpc-principal

rtb-0e36f8d957c694157 / tabla-rutas-internet																		
Detalles	Rutas	Asociaciones de subredes	Asociaciones de borde	Propagación de rutas	Etiquetas													
Rutas (3) <div style="display: flex; justify-content: space-between;"> Editar rutas Filtrar rutas Ambos < 1 > </div> <table border="1"> <thead> <tr> <th>Destino</th> <th>Destino</th> <th>Estado</th> <th>Propagada</th> </tr> </thead> <tbody> <tr> <td>10.0.0.0/16</td> <td>local</td> <td>Activ</td> <td>No</td> </tr> <tr> <td>0.0.0.0/0</td> <td>igw-0fba8b3da4e39bc50</td> <td>Activ</td> <td>No</td> </tr> </tbody> </table>							Destino	Destino	Estado	Propagada	10.0.0.0/16	local	Activ	No	0.0.0.0/0	igw-0fba8b3da4e39bc50	Activ	No
Destino	Destino	Estado	Propagada															
10.0.0.0/16	local	Activ	No															
0.0.0.0/0	igw-0fba8b3da4e39bc50	Activ	No															

Ilustración 57 - Tabla internet AWS

Las subredes públicas creadas utilizarán esta tabla de enrutamiento. Para asociar la tabla a las subredes se realiza mediante “Acciones > Editas asociaciones de subredes” (Ilustración 58).

The screenshot shows the AWS Route Tables list with three entries:

Name	ID de tabla de enrutamiento	Asociaciones de subredes	Asociaciones de borde	Propagación de rutas	VPC
tabla-rutas-VPC	rtb-0309bc241d753cde2	-	-	Sí	vpc-0cd81c
tabla-rutas-internet	rtb-0e36f8d957c694157	2 subredes	-	No	vpc-0cd81c
tabla-rutas-databases	rtb-08e72b887f4768c20	2 subredes	-	No	vpc-0cd81c

Below the list, a detailed view of the selected route table (rtb-0e36f8d957c694157) is shown. The 'Asociaciones de subredes' tab is active, displaying two explicit subnet associations:

Name	ID de subred	CIDR IPv4	CIDR IPv6
subred-servicios-1a	subnet-0cc3734c40221676d	10.0.0.0/24	-
subred-front-1b	subnet-02ff8267200d84ab4	10.0.1.0/24	-

Ilustración 58 - Asociacion tabla internet aws

2. Tabla de rutas con NAT Gateway

Las rutas que contendrá esta tabla permitirán el tráfico hacia la VPC mediante la regla “10.0.0.0/16 – Local” y hacia la NAT Gateway a través de “0.0.0.0/0 – Nat-05254623edc53e252” (Ilustración 59).

The screenshot shows the AWS Route Tables list with three entries:

Name	ID de tabla de enrutamiento	Asociaciones de subredes	Asociaciones de borde	Propagación de rutas	VPC
tabla-rutas-VPC	rtb-0309bc241d753cde2	-	-	Sí	vpc-0cd81c
tabla-rutas-internet	rtb-0e36f8d957c694157	2 subredes	-	No	vpc-0cd81c
tabla-rutas-databases	rtb-08e72b887f4768c20	2 subredes	-	No	vpc-0cd81c

Below the list, a detailed view of the selected route table (rtb-08e72b887f4768c20) is shown. The 'Rutas' tab is active, displaying three routes:

Destino	Destino	Estado	Propagada
10.0.0.0/16	local	Activado	No
0.0.0.0/0	nat-05254623edc53e252	Activado	No

Ilustración 59 - Tablas databases AWS

Esta tabla se asociará con las subredes privadas (Ilustración 60):

The screenshot shows the AWS Route Tables interface. At the top, there's a header for 'Tablas de enruteamiento (1/4)'. Below it is a search bar labeled 'Filtrar tablas de enruteamiento' and a toolbar with 'Acciones' and 'Crear tabla de enruteamiento' buttons. A table lists three route tables: 'tabla-rutas-VPC', 'tabla-rutas-internet', and 'tabla-rutas-databases'. The 'tabla-rutas-databases' row is selected, indicated by a checked checkbox. The table columns include 'Name', 'ID de tabla...', 'Asociaciones...', 'Asociac...', 'Prin...', and 'VPC'. The 'VPC' column shows 'vpc-0cd81c07aefa4362d | vpc-principal' for all three entries. Below the table, a specific route table ('rtb-08e72b887f4768c20 / tabla-rutas-databases') is selected. The 'Asociaciones de subredes' tab is active, showing two explicit subnet associations:

Name	ID de subred	CIDR IPv4	CIDR IPv6
subred-databases-1a	subnet-0506b3973e1b527d6	10.0.2.0/24	-
subred-databases-1b	subnet-042c9bce890cf1ee6	10.0.3.0/24	-

Ilustración 60 - Asociación tabla database AWS

Grupos de seguridad

Cada grupo de seguridad actuará como un firewall virtual que controlará el tráfico de las instancias o servicios que la utilicen a través de una serie de reglas definidas en función de los puertos, protocolos, IPs... a los que se les permitirá el paso. Estas reglas dependerán del tipo de función que realice la instancia o servicio.

1. Grupo de seguridad EC2 Bastión

El tráfico de entrada permitido será por el puerto 22 para conexiones SSH con la instancia EC2 de cara al acceso por parte del administrador (Ilustración 61).

Características de creación del grupo de seguridad:

- **Nombre:** ec2-bastion-sg
- **Descripción:** Grupo de seguridad para la EC2 Bastión
- **VCP:** vpc-principal
- **Reglas de entrada:**
 - o SSH – TCP – Puerto 22 – Tráfico de origen 0.0.0.0

Name	ID del grupo de seguridad	Nombre del grupo	ID de la VPC	Descripción
sg-02ce550a437b84c22	ec2-bastion-sg	vpc-0cd81c07aefa4362d	Grupo de seguridad para...	

Reglas de entrada (1/1)	Administrador de etiquetas	Editar reglas de entrada													
<table border="1"> <thead> <tr> <th>Nombre</th> <th>ID de la regla de...</th> <th>Versió...</th> <th>Tipo</th> <th>Protocolo</th> <th>Intervalo de puertos</th> <th>Origen</th> </tr> </thead> <tbody> <tr> <td>sgr-00c03987b07e...</td> <td>IPv4</td> <td>SSH</td> <td>TCP</td> <td>22</td> <td>0.0.0.0/0</td> </tr> </tbody> </table>	Nombre	ID de la regla de...	Versió...	Tipo	Protocolo	Intervalo de puertos	Origen	sgr-00c03987b07e...	IPv4	SSH	TCP	22	0.0.0.0/0		
Nombre	ID de la regla de...	Versió...	Tipo	Protocolo	Intervalo de puertos	Origen									
sgr-00c03987b07e...	IPv4	SSH	TCP	22	0.0.0.0/0										

Ilustración 61 - Bastión sg AWS

2. Grupo de seguridad de los servicios EC2

El tráfico de entrada permitido será a través de los puertos necesarios donde se desplegarán los servicios principales de la aplicación dentro de instancias EC2: Plots (8080), Sensors (8081), Statistics (8082) y Authentication (8999). Además, se permitirá la conexión SSH en el puerto 22 por parte de las instancias desplegadas con el grupo de seguridad “ec2-bastion-sg”, lo que provoca que la máquina EC2 Bastión pueda conectarse. Adicionalmente, se habilitarán las conexiones HTTPS en el puerto 443 para las conexiones que utilizan el servicio Authentication y su base de datos MySQL (Ilustración 62).

Características de creación del grupo de seguridad:

- **Nombre:** servicios-sg
- **Descripción:** Grupo de seguridad para las EC2 de servicios
- **VCP:** vpc-principal
- **Reglas de entrada:**
 - o SSH – TCP – Puerto 22 – Tráfico de origen “ec2-bastion-sg”
 - o HTTPS – TCP – Puerto 443 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 8080 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 8081 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 8082 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 8999 – Tráfico de origen 0.0.0.0

Name	ID de la regla de...	Versió...	Tipo	Protoc...	Intervalo...	Origen
sgr-0d8f1bf03554f...	-	SSH	TCP	22		sg-02ce550a437b
sgr-016e61906dd7...	IPv4	TCP perso...	TCP	8080		0.0.0.0/0
sgr-01afc43228b3...	IPv4	TCP perso...	TCP	8081		0.0.0.0/0
sgr-040b2f1f34045...	IPv4	TCP perso...	TCP	8082		0.0.0.0/0
sgr-0e6f419d0e83f...	IPv4	TCP perso...	TCP	8999		0.0.0.0/0

Ilustración 62 – Servicios sg AWS

3. Grupo de seguridad del EC2 Front

El tráfico de entrada permitido será a través del puerto 4200 donde se desplegará el servicio front-end dentro de una máquina EC2 y la conexión por el puerto 22 de las instancias del grupo de seguridad “ec2-bastion-sg” (Ilustración 63).

Características de creación del grupo de seguridad:

- **Nombre:** front-sg
- **Descripción:** Grupo de seguridad para el EC2 front
- **VCP:** vpc-principal
- **Reglas de entrada:**
 - SSH – TCP – Puerto 22 – Tráfico de origen “ec2-bastion-sg”
 - TCP – Puerto 4200 – Tráfico de origen 0.0.0.0

The screenshot shows the AWS Security Groups console. At the top, there's a header with 'Grupos de seguridad (1/1)' and a 'Información' link. Below the header are buttons for 'Acciones' (Actions), 'Exportar los grupos de seguridad a CSV' (Export security groups to CSV), and 'Crear grupo de seguridad' (Create security group). A search bar labeled 'Filtrar grupos de seguridad' (Filter security groups) is present, along with a filter button 'search: front' and a 'Quitar los filtros' (Remove filters) button.

The main table lists one security group:

	Name	ID del grupo de segu...	Nombre del grup...	ID de la VPC	Descripción
<input checked="" type="checkbox"/>	-	sg-092e5b0159190fb4	front-sg	vpc-0cd81c07aefa4362d	Grupo de seguridad pa...

Below the table are tabs for 'Detalles', 'Reglas de entrada' (selected), 'Reglas de salida', and 'Etiquetas'. The 'Reglas de entrada' tab shows two inbound rules:

	Name	ID de la regla de...	Versión d...	Tipo	Protoc...	Interv...	Origen
<input type="checkbox"/>	-	sgr-05b088361d5b...	IPv4	TCP personal...	TCP	4200	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0eeecb80c7edc...	-	SSH	TCP	22	sg-02ce55c

Ilustración 63 - Front sg AWS

4. Grupo de seguridad de las bases de datos RDS y EC2

El tráfico de entrada permitido será a través de los puertos necesarios donde se desplegarán los servicios de bases de datos: MongoDB (27017), MySQL (3306), PostgreSQL (5432) y Cassandra (9042). Además, se permite la conexión por el puerto 22 de las instancias del grupo de seguridad “ec2-bastion-sg” (Ilustración 64).

Características de creación del grupo de seguridad:

- **Nombre:**
- **Descripción:** Grupo de seguridad para los servicios de base de datos.
- **VCP:** vpc-principal
- **Reglas de entrada:**
 - o SSH – TCP – Puerto 22 – Tráfico de origen “ec2-bastion-sg”
 - o TCP – Puerto 27017 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 3306 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 5432 – Tráfico de origen 0.0.0.0
 - o TCP – Puerto 9042 – Tráfico de origen 0.0.0.0

The screenshot shows the AWS Management Console interface for security groups. At the top, it displays 'Grupos de seguridad (1/1) Información'. Below this is a toolbar with icons for Actions, Export to CSV, and Create new security group. A search bar labeled 'Filtrar grupos de seguridad' is present, along with a 'search: data' button and a 'Quitar los filtros' link. The main table lists one security group:

<input checked="" type="checkbox"/>	Name	ID del grupo de segu...	Nombre del grup...	ID de la VPC	Descripción
<input checked="" type="checkbox"/>	-	sg-0e1a3747535dfc3b2	databases-sg	vpc-0cd81c07aefaf4362d	Grupo de seguridad de...

Below this, another section titled 'Reglas de entrada (5)' is shown. It includes a toolbar with icons for Actions, Manage tags, and Edit rules. A search bar labeled 'Filtrar reglas de grupo de seguridad' is available. The table lists five ingress rules:

<input type="checkbox"/>	Name	ID de la regla de...	Versió...	Tipo	Protoc...	Interv...	Origen
<input type="checkbox"/>	-	sgr-09704dc64bfc8...	IPv4	MySQL/Aur...	TCP	3306	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0ef2abbac4c02...	IPv4	TCP persona...	TCP	9042	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0b05a17c96b8...	IPv4	PostgreSQL	TCP	5432	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0e1dc62759a0...	IPv4	TCP persona...	TCP	27017	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0ede57a564dc...	-	SSH	TCP	22	sg-02ce550a4...

Ilustración 64 - Databases sg AWS

5. Grupo de seguridad del EFS

El tráfico de entrada permitido dentro del grupo de seguridad al que se asociará el sistema de ficheros EFS será por el puerto 2049 destinado a conexiones con protocolo NFS (Ilustración 65).

Características de creación del grupo de seguridad:

- **Nombre:** efs-sg
- **Descripción:** Grupo de seguridad para EFS
- **VPC:** vpc-principal
- **Reglas de entrada:**
 - o NFS – TCP – Puerto 2049 – Tráfico de origen 0.0.0.0

The screenshot shows the AWS Security Groups console. At the top, there is a search bar with the filter "search: efs" and a "Quitar los filtros" (Remove filters) button. Below the search bar is a table header with columns: Name, ID del grupo de segu..., Nombre del grupo..., ID de la VPC, and Descripción. A single row is selected in the table, showing "sg-05e8488398a56d205" as the Name, "efs-sg" as the Group Name, "vpc-0cd81c07aef4362d" as the VPC ID, and "Grupo de se" as the Description. Below the table, the group name "sg-05e8488398a56d205 - efs-sg" is displayed. At the bottom of the screen, there are tabs for "Detalles", "Reglas de entrada" (which is selected), "Reglas de salida", and "Etiquetas".

Reglas de entrada (1)

Name	ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen
-	sgr-0c388c28d765...	IPv4	NFS	TCP	2049	0.0.0.0/0

Ilustración 65 - NFS sg AWS

6. Grupo de seguridad del balanceador de carga ALB

El tráfico de entrada permitido en el ALB abarca todos los puertos correspondientes tanto al servicio Front (4200) como los demás servicios API REST (8080, 8081, 8082 y 8999). El balanceador estará escuchando las peticiones entrantes en cada uno de estos puertos para redirigirlas hacia sus correspondientes servicios a través de los grupos de destino del ALB, cada uno configurado con un puerto distinto y un conjunto de instancias destino, donde se encontrarán desplegados los servicios de la aplicación, a las que se redirigirán las peticiones (Ilustración 66).

Características de creación del grupo de seguridad:

- **Nombre:** ALB-sg
- **Descripción:** Grupo de seguridad para el ALB
- **VCP:** vpc-principal
- **Reglas de entrada:**
 - TPC – Puerto 4200 – Tráfico de origen 0.0.0.0
 - TCP – Puerto 8080 – Tráfico de origen 0.0.0.0
 - TCP – Puerto 8081 – Tráfico de origen 0.0.0.0
 - TCP – Puerto 8082 – Tráfico de origen 0.0.0.0
 - TCP – Puerto 8999 – Tráfico de origen 0.0.0.0

The screenshot shows the AWS Management Console interface for managing security groups. At the top, there's a header for 'Grupos de seguridad (1/2)' with a 'Información' link. Below the header are buttons for 'Acciones' (Actions), 'Exportar los grupos de seguridad a CSV' (Export selected security groups to CSV), and a prominent orange 'Crear grupo de seguridad' (Create security group) button. There's also a search bar labeled 'Filtrar grupos de seguridad' (Filter security groups) and a filter button 'Quitar los filtros' (Remove filters). A search input field shows 'search: ALB-sg'. The main table lists one security group named 'ALB-sg' with its ID and associated VPC. Below the table is another section titled 'Reglas de entrada (5)' (5 incoming rules) with a similar filtering and export/import functionality.

Name	ID del grupo de segu...	Nombre del grupo ...	ID de la VPC	Descripción
<input checked="" type="checkbox"/> -	sg-0c507d06f585599a0	ALB-sg	vpc-0cd81c07aefa4362d	Grupo de se

Name	ID de la regla de...	Versió...	Tipo	Protoc...	Interval...	Origen
<input type="checkbox"/>	sgr-02ac244a2a89...	IPv4	TCP person...	TCP	4200	0.0.0.0/0
<input type="checkbox"/>	sgr-089ddb2f65f18...	IPv4	TCP person...	TCP	8081	0.0.0.0/0
<input type="checkbox"/>	sgr-0f8b1ca8f12ac...	IPv4	TCP person...	TCP	8999	0.0.0.0/0
<input type="checkbox"/>	sgr-02b88de00481...	IPv4	TCP person...	TCP	8080	0.0.0.0/0
<input type="checkbox"/>	sgr-0a7459452113...	IPv4	TCP person...	TCP	8082	0.0.0.0/0

Ilustración 66 - ALB sg AWS

6.2.2. Servicios de almacenamiento

En toda infraestructura debe haber alguna manera de almacenar los datos de la aplicación y los ficheros de configuración necesarios por los servicios de la misma. En este caso, se harán cargo de esta tarea dos tipos de servicios en AWS: un bucket S3 para almacenar el fichero con las variables de entorno que necesitarán los microservicios contenedores y un sistema de ficheros EFS que permitirá la persistencia de los datos al poder guardar los datos procedentes de las instancias EC2 ejecutando los servicios de bases de datos.

Lista de servicios de almacenamiento que se van a utilizar:

- Amazon S3 Bucket: 1
- Elastic File System (EFS): 1

Adicionalmente, se crearán otros dos servicios que no tienen que ver con el almacenamiento pero que son necesarios utilizarlos:

- Punto de enlace de VPC: 1
- Rol IAM AmazonS3FullAccess: 1

Estos servicios se encuentran en el **apartado “S3” y “EFS”** de AWS, a excepción del punto de enlace VPC que se accede en el **apartado “VPC”** y la creación del rol IAM situado en el **apartado “IAM”**.

Punto de enlace de VPC

Este servicio habilitará las conexiones entre la VPC y un tipo de servicio escogido en función del punto de enlace que se defina. En este caso, será un enlace de tipo Gateway hacia un servicio de bucket S3. De esta forma, el acceso al S3 solo se podrá producir a través de este punto de enlace desde la VPC (Ilustración 67).

The screenshot shows the AWS CloudFormation console with the 'Puntos de enlace' (Link Points) page. The table lists one endpoint:

Name	ID de punto de enlace	ID de la VPC	Nombre del servicio
s3-endpoint	vpce-01f584c5ae887dce9	vpc-0cd81c07aefa4362d vpc-principal	com.amazonaws.us-east-1.s3

Below the table, the 'Detalles' (Details) section provides more information:

ID de punto de enlace vpce-01f584c5ae887dce9	Estado Disponible	Hora de creación jueves, 1 de diciembre de 2022, 02:37:20 CET	Tipo de punto de enlace Gateway
ID de la VPC vpc-0cd81c07aefa4362d (vpc-principal)	Mensaje de estado -	Nombre del servicio com.amazonaws.us-east-1.s3	Nombres de DNS privados habilitados No

Ilustración 67 - Punto s3 AWS

Características de creación del punto de enlace:

- **Nombre:** s3-endpoint
- **Categoría de servicio:** Servicios de AWS
- **Servicios:** com.amazonaws.us-east-1.s3 (tipo Gateway).
- **VPC:** vpc-principal
- **Tablas de enrutamiento:**
 - o tabla-rutas-internet
 - o tabla-rutas-databases
- **Política:** Acceso completo a cualquier usuario o servicio dentro de la VPC.

Durante la creación, se asocia automáticamente la ruta del punto de enlace en las tablas de enrutamiento seleccionadas, por lo que estas tendrán una

entrada más para el acceso a un servicio S3: “pl-63a5400a – vpce-01f584c5ae887dce9”.

1. Tabla-rutas-internet asociada a las subredes públicas (Ilustración 68)

The screenshot shows the AWS Route Table configuration for route table 'rtb-0e36f8d957c694157'. The 'Rutas' tab is selected, displaying three routes:

Destino	Destino	Estado
10.0.0.0/16	local	Activo
0.0.0.0/0	igw-0fba8b3da4e39bc50	Activo
pl-63a5400a	vpce-01f584c5ae887dce9	Activo

Ilustración 68 - Ruta s3 tabla Internet AWS

2. Tabla-rutas-databases asociada a las subredes privadas (Ilustración 69)

The screenshot shows the AWS Route Table configuration for route table 'rtb-08e72b887f4768c20'. The 'Rutas' tab is selected, displaying three routes:

Destino	Destino	Estado
10.0.0.0/16	local	Activo
0.0.0.0/0	nat-05254623edc53e252	Activo
pl-63a5400a	vpce-01f584c5ae887dce9	Activo

Ilustración 69 - Ruta s3 tabla databases AWS

Amazon S3 Bucket

Dentro de este servicio de almacenamiento de objetos se subirán los ficheros para ponerlos a disposición de los demás servicios de AWS. Esta es la estructura del bucket (Ilustración 70-73):

db-services-bucket

- Carpeta **cassandra/**
 - o Carpeta **bash/**
 - **cassandra_init.sh:** script para configurar Cassandra.

- Carpeta: **cql/**
 - **init_keyspace.cql**: fichero de cassandra para crear el keyspace inicial de la base de datos.
- **docker-compose-cassandra.yml**: fichero docker-compose para lanzar los contenedores de Cassandra.
- **env_vars.env**: variables de entorno para los contenedores.
- **init-mongodb.js**: script para la configuración inicial de MongoDB.

Amazon S3 > Buckets > db-services-bucket

db-services-bucket Info

Objetos (3)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

C	Copiar URI de S3	Copiar URL	Descargar	Abrir	Eliminar	Acciones
Crear carpeta	Cargar					
<input type="text"/> Buscar objetos por prefijo						
	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento	
<input type="checkbox"/>	cassandra/	Carpeta	-	-	-	
<input type="checkbox"/>	env_vars.env	env	31 May 2023 1:29:41 PM CEST	309.0 B	Estándar	
<input type="checkbox"/>	init-mongodb.js	js	19 May 2023 1:53:07 PM CEST	3.3 KB	Estándar	

Ilustración 71 - Bucket s3 AWS

Amazon S3 > Buckets > db-services-bucket > **cassandra/**

cassandra/

	Nombre	Tipo	Última modificación
<input type="checkbox"/>	bash/	Carpeta	-
<input type="checkbox"/>	cql/	Carpeta	-
<input type="checkbox"/>	docker-compose-cassandra.yml	yml	23 May 2023 3:02:56 PM CEST

Ilustración 70 - Bucket s3 carpeta cassandra AWS

Amazon S3 > Buckets > db-services-bucket > cassandra/ > bash/						
bash/						
	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento	
	cassandra_init.sh	sh	23 May 2023 3:02:57 PM CEST	557.0 B	Estándar	Copiar URI de S3

Ilustración 73 - Bucket s3 carpeta bash AWS

Amazon S3 > Buckets > db-services-bucket > cassandra/ > cql/						
cql/						
	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento	
	init_keyspace.cql	cql	23 May 2023 3:02:58 PM CEST	132.0 B	Estándar	Copiar URI de S3

Ilustración 72 - Bucket s3 carpeta cql AWS

Características de creación del bucket S3:

- **Nombre:** db-services-bucket
- **Región:** us-east-1
- **Propiedad de objetos:** ACL deshabilitadas (para gestionar el acceso al bucket y sus objetos mediante políticas)
- **Configuración de bloqueo de acceso público para este bucket:** Bloquear todo el acceso público.

Este bucket no se define dentro de una VPC sino en la región escogida en su creación. Para evitar que pueda ser accedido desde Internet, se ha escogido la opción de bloquear el acceso público al bucket, pudiendo solo acceder al bucket a partir del punto de enlace (endpoint) creado anteriormente.

Rol IAM AmazonS3FullAccess

Para poder tener los permisos de acceso necesarios al S3 creado se ha de crear un rol dentro del servicio IAM “Identity and Access Management”. Este rol se asignará a las instancias EC2 que necesiten acceder al bucket (Ilustración 74).

The screenshot shows the AWS IAM Roles page with the following details:

- Role Name:** AmazonS3FullAccess
- Description:** Allows EC2 instances to call AWS services on your behalf.
- Actions:** Eliminar (Delete), Editar (Edit)
- Resumen (Summary) section:**

Fecha de creación December 01, 2022, 02:32 (UTC+01:00)	ARN arn:aws:iam::690031176274:role/AmazonS3FullAccess	ARN del perfil de instancias arn:aws:iam::690031176274:instance-profile/AmazonS3FullAccess
Última actividad Hace 5 meses	Duración máxima de la sesión 1 hora	
- Permissions tab:** Permisos, Relaciones de confianza, Etiquetas, Access Advisor, Revocar las sesiones
- Permissions Policies section:**
 - Políticas de permisos (1) Información**: Puede asociar hasta 10 políticas administradas.
 - Buttons: Recargar, Simular, Eliminar, Añadir permisos ▾
 - Search bar: Filtre las políticas por propiedad o nombre de política y pulse Intro.
 - Table headers: Nombre de la política, Tipo, Descripción
 - Data row: + **AmazonS3FullAccess** Adminis... Provides full access to all buckets via the AWS...

Ilustración 74 - Rol s3 AWS

Características de creación del rol:

- **Tipo de entidad:** Servicio de AWS
- **Caso de uso:** EC2 (permite el acceso a servicios AWS a las instancias EC2)
- **Políticas de permisos:** AmazonS3FullAccess
- **Nombre del rol:** AmazonS3FullAccess

Elastic File System (EFS)

Este servicio EFS permite la creación de un sistema de ficheros que se montará dentro de los directorios de las máquinas EC2 donde se desplegarán los servicios contenedores de bases de datos (MongoDB y Cassandra), consiguiendo así la persistencia de los datos independientemente del estado del contenedor (Ilustración 75).

Amazon EFS > Sistemas de archivos > fs-0f2f65bb0548dde2f

efs-databases (fs-0f2f65bb0548dde2f)

General

Modo de rendimiento: Uso general

Modo de desempeño: Transmisión por ráfagas

Administración del ciclo de vida:

- Transición al acceso poco frecuente: Días desde el último acceso: 30
- Transición fuera del acceso poco frecuente: En el primer acceso

Zona de disponibilidad: Estándar

Copias de seguridad automáticas: Activado

Cifrado: f5be0bcc-1c11-486c-9f81-f6332cb9a94f (aws/elasticfilesystem)

Estado del sistema de archivos: Disponible

Nombre de DNS: fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com

Ilustración 75 - EFS AWS

Características de creación del EFS:

- **Nombre:** efs-databases
- **Copias de seguridad:** activadas
- **VPC:** vpc-principal
- **Destino de montaje:**
 - **Zona de disponibilidad:** us-east-1a
 - **ID de la subred:** subnet-0506b3973e1b527d6 (subred-databases-1a)
 - **Dirección IP del destino de montaje:** Asignada automáticamente
 - **Grupos de seguridad:** efs-sg

Virtual Private Cloud (VPC)
Elija la VPC en la que desea que las instancias EC2 se conecten a su sistema de archivos.

Zona de disponibilidad	ID de la subred	Dirección IP	Grupos de seguridad
us-east-1a	subnet-0506b3973	10.0.2.95	Elegir grupos ... sg-05e8488398a5 6d205 efs-sg

Ilustración 76 - Destino montaje AWS

El destino de montaje definido durante la creación del EFS (Ilustración 76) se establecerá en la subred privada de bases de datos (us-east-1a), lo que permitirá el acceso al sistema de ficheros EFS a partir de este punto para todas las instancias EC2 con independencia de la subred donde se encuentren. Se utilizará el nombre de DNS del EFS para realizar su montaje sobre las instancias: **fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com**.

6.2.3. EC2 Bastión o máquina administradora

Esta EC2 se desplegará en la subred de servicios pública (us-east-1a) “subred-servicios-1a” y realizará el montaje del sistema de ficheros EFS para crear los directorios que permitirán la persistencia de datos en las máquinas EC2 que ejecutarán los servicios de bases de datos de MongoDB y Cassandra (Ilustración 77). Como el uso del EFS es compartido, una vez que se creen los directorios, estarán disponibles para las demás EC2 que realicen el montaje del EFS. Este proceso se realizará mediante un script (aparece más adelante al final de la creación del EC2) que se copiará en el apartado de “datos de usuario” durante la creación de la instancia y que se ejecutará al lanzarla.

The screenshot shows the AWS CloudFormation Instances page. At the top, there is a search bar labeled "Buscar instancia por atributo o etiqueta (case-sensitive)" and a filter button "Quitar los filtros". Below the header, a table lists one instance:

Name	ID de la Instancia	Estado de la i...	Tipo de...	Comprobación ...	Estado de la ...
ec2-bastion	i-063675e4921c93de2	En ejecución	t2.micro	2/2 comprobador	Sin alarmas

Below the table, the instance details are shown:

Instancia: i-063675e4921c93de2 (ec2-bastion)

Resumen de instancia

ID de la instancia i-063675e4921c93de2 (ec2-bastion)	Dirección IPv4 pública 44.212.59.92 dirección abierta	Direcciones IPv4 privadas 10.0.0.142
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública ec2-44-212-59-92.compute-1.amazonaws.com dirección abierta
Tipo de nombre de anfitrión Nombre de IP: ip-10-0-0-142.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-0-0-142.ec2.internal	
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.micro	Direcciones IP elásticas -
Dirección IP asignada automáticamente 44.212.59.92 [IP pública]	ID de VPC vpc-0cd81c07aefa4362d (vpc-)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer pa

Ilustración 77 - EC2 Bastión

También se instalará en los datos de usuario de la EC2 los clientes de PostgreSQL y MySQL para poder realizar consultas a los servicios RDS donde se desplegarán estas bases de datos con el fin de comprobar que se hayan creado correctamente todas las bases de datos y tablas.

Características de creación de la instancia EC2 (Modelo Bajo demanda u “On-Demand”):

- **Nombre:** ec2-bastion.
- **Imagen de Amazon (AMI):** Amazon Linux 2023 de uso gratuito.

AMI de Amazon Linux 2023	Apto para la capa gratuita
ami-0889a44b331db0194 (64 bits (x86)) / ami-08fc6fb8ad2e794bb (64 bits (Arm))	
Virtualización: hvm Habilitado para ENA: true Tipo de dispositivo raíz: ebs	

- **Tipo de instancia:** t2.micro de uso gratuito.

t2.micro	Apto para la capa gratuita
Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true	

- **Par de claves:** clave_maquinas_TFG
- **Configuraciones de red:**
 - **VPC:** vpc-principal
 - **Subred:** subred-servicios-1a
 - **Asignar automáticamente la IP pública:** Activar
 - **Grupo de seguridad:** ec2-bastion-sg
- **Detalles avanzados:**
 - **Datos de usuario:** El script utilizado se encarga de realizar el montaje del EFS dentro de la máquina. Los comandos de montaje utilizan la **dirección DNS identificadora del EFS** y definen la carpeta “/efs” dentro del EC2 donde se montará el sistema de ficheros. Después, dentro de la carpeta “/efs” se crean las carpetas “/mongo-efs” y “/cassandra-efs” que serán utilizadas más adelante por las bases de datos EC2. Finalmente, se realizará la instalación de los clientes de MySQL y PostgreSQL para futuras conexiones con estas bases de datos.

El script utilizado dentro de los datos de usuario de la instancia EC2 Bastión es el siguiente:

- **Script EC2 Bastión:**

```
#!/bin/bash

# Instalacion herramientas necesarias
yum install nfs-utils -y
# Directorio de montaje /efs en la carpeta raiz
mkdir /efs
# Montaje EFS en el path indicado /efs
mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvp
ort fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com:/ /efs
# Montaje del EFS automatico al reiniciar la instancia
echo "fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com:/ /efs nfs4
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 0 0" >>
/etc/fstab
# Sleep 20 segundos por si necesita tiempo para montarse
sleep 20
# Crear directorios dentro del EFS donde se almacenaran los datos
cd /
# Directorio MongoDB
mkdir /efs/mongo-efs
# Directorio Cassandra
mkdir /efs/cassandra-efs
# Instalar MySQL 8.0
sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-
1.noarch.rpm
sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y
sudo dnf install mysql-community-server -y
# Instalar PostgreSQL 15
sudo yum install postgresql15.x86_64 -y
```

Para comprobar que se ha realizado el montaje correctamente, se realiza la conexión con la instancia Bastión mediante SSH a través del siguiente comando, que utilizará la clave guardada en el equipo del par de claves conectándose con el usuario “ec2-user” a la dirección EC2 pública de la máquina “@...”, la cual puede variar puesto que AWS va asignando cada cierto tiempo una nueva.

```
ssh -i "clave_maquinas_TFG.pem" ec2-user@ec2-44-212-59-92.compute-
1.amazonaws.com
```

Una vez dentro de la instancia, se puede observar que la carpeta “/efs” se encuentra creada en el directorio raíz y dentro de esta se encuentran las carpetas de “/cassandra-efs” y “/mongo-efs” (Ilustración 78 y 79).

```
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ssh -i "clave_maquinas_TFG.pem" ec2-user@ec2-44-212-59-92.compute-1.amazonaws.com
The authenticity of host 'ec2-44-212-59-92.compute-1.amazonaws.com (44.212.59.92)' can't be established.
ECDSA key fingerprint is SHA256:/C1J+/vuzu7fY90PoCXR9AwI/KR1udnGbW3tYMsDAHo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-212-59-92.compute-1.amazonaws.com' (ECDSA) to the list of known hosts.

A newer release of "Amazon Linux" is available.
  Version 2023.0.20230517;
  Version 2023.0.20230607;
Run "/usr/bin/dnf check-release-update" for full release and version update info
      _#
  ~\_\_ #####_      Amazon Linux 2023
  ~~ \_\#\#\#\_
  ~~ \#\#\#
  ~~ \#/ _-- https://aws.amazon.com/linux/amazon-linux-2023
  ~~ V~, '-->
  ~~ /_/
  ~~ ._. _/
  /_/_/
  /m/
Last login: Mon Jun 12 10:04:54 2023 from 81.43.1.157
[ec2-user@ip-10-0-0-142 ~]$
```

Ilustración 78 - SSH EC2 Bastión

```
[ec2-user@ip-10-0-0-142 ~]$ cd /
[ec2-user@ip-10-0-0-142 /]$ ls
bin dev etc lib local mnt proc run srv tmp var
boot efs home lib64 media opt root sbin sys usr
[ec2-user@ip-10-0-0-142 /]$ cd efs/
[ec2-user@ip-10-0-0-142 efs]$ ls
cassandra-efs mongo-efs
```

Ilustración 79 - Comprobación directorios EFS

Otra manera de comprobar que el directorio “/efs” se encuentra creado es a través del comando “df”, que listará todos los sistemas de ficheros montados en la instancia (Ilustración 80):

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
devtmpfs	4096	0	4096	0%	/dev
tmpfs	486320	0	486320	0%	/dev/shm
tmpfs	194532	2856	191676	2%	/run
/dev/xvda1	8310764	2241704	6069060	27%	/
tmpfs	486324	0	486324	0%	/tmp
fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com:/	9007199254739968	312320	9007199254427648	1%	/efs
tmpfs	97264	0	97264	0%	/run/user/1000

Ilustración 80 - Comando "df"

6.2.4. Servicios de bases de datos: RDS y EC2

Para desplegar las bases de datos de PostgreSQL y MySQL se aprovecharán las ventajas que proporciona el servicio de RDS de AWS que permiten la creación bases de datos relacionales gestionadas completamente por Amazon y con la posibilidad de tener un servicio de copias de seguridad automáticas a través de instantáneas o “snapshots”.

En el caso de las bases de datos no relacionales MongoDB y Cassandra, se recurirá al despliegue de los servicios mediante contenedores Docker en instancias EC2. Su lanzamiento y configuración se realizarán a través de scripts que se definirán en el apartado de “Datos de usuario” dentro de la

creación de las EC2, los cuales se ejecutarán solo una vez durante el primer lanzamiento de la instancia. Si la instancia terminase, se tendría que volver a lanzar la EC2 con este script y se obtendría siempre la misma configuración.

En cuanto al problema de la persistencia de datos ante la posibilidad de que la instancia o el contenedor se destruya, se utilizarán volúmenes Docker para realizar su montaje al sistema de ficheros EFS creado anteriormente. De esta manera, se asignarán dichos volúmenes al lanzar los contenedores para cargar los datos guardados en el EFS. Además, se hará uso de los ficheros subidos al bucket s3 “db-services-bucket” creado anteriormente durante la fase de desarrollo con los que se configurarán las bases de datos (se explicarán más adelante).

Tanto las RDS como las EC2 se desplegarán en las subredes privadas, principalmente en la subred de bases de datos privada (us-east-1a) “subred-databases-1a”. Durante su creación, no se seleccionará la opción de asignar una IP pública para que no puedan ser accedidas estas máquinas desde Internet. Solo podrán realizar conexiones las EC2 donde se desplegarán los microservicios y la EC2 Bastión administradora, ambas dentro de la VPC.

Lista de servicios de bases de datos que se van a utilizar:

- RDS PostgreSQL: 1
- RDS MySQL: 1
- EC2 MongoDB: 1
- EC2 Cassandra: 1

También se creará otro servicio necesario para desplegar las RDS:

- Grupo de subredes: 1

Los servicios se encuentran en los **apartados de “RDS” y “EC2”** de AWS.

Grupo de subredes

Este servicio creará dentro de la VPC un grupo formado por las subredes de bases de datos creadas dentro de las zonas de disponibilidad “us-east-1a” y “us-east-1b”. Es requerido durante el proceso de creación de las instancias de bases de datos RDS (Ilustración 81).

RDS > Grupos de subredes > databases-grupo-subred

databases-grupo-subred

Detalles del grupo de subredes		
ID de la VPC	vpc-0cd81c07aefa4362d	
ARN	arn:aws:rds:us-east-1:690031176274:subgrp:databases-grupo-subred	
Tipos de red compatibles	IPv4	
Descripción	Grupo de subredes para las bases de datos	

Subredes (2)		
Zona de disponibilidad	ID de subred	Bloque de CIDR
us-east-1a	subnet-0506b3973e1b527d6	10.0.2.0/24
us-east-1b	subnet-042c9bce890cf1ee6	10.0.3.0/24

Ilustración 81 - Grupo subredes AWS

Características de creación del grupo de subredes:

- **Nombre:** databases-grupo-subred
- **Descripción:** Grupo de subredes para las bases de datos
- **VPC:** vpc-principal
- **Zonas de disponibilidad:**
 - o us-east-1a
 - o us-east-1b
- **Subredes:**
 - o Subred-databases-1a
 - o Subred-databases-1b

RDS PostgreSQL

El servicio de base de datos PostgreSQL se creará con la misma configuración que cuando se desplegó en la fase de contenedores. Escuchará en el puerto 5432, creará la base de datos “sensors” al levantarse y se especificará el nombre junto con la contraseña del usuario maestro con los valores “postgres” y “password” respectivamente (Ilustración 82).

Identificador de base de datos sensor-postgres	CPU 4.30%	Estado Disponible	Clase db.t3.micro
Rol Instancia	Actividad actual 0.00 sessions	Motor PostgreSQL	Región y AZ us-east-1a

Punto de enlace y puerto	Redes	Seguridad
Punto de enlace sensor-postgres.cvt9cc7dxazz.us-east-1.rds.amazonaws.com	Zona de disponibilidad us-east-1a VPC vpc-principal (vpc-0cd81c07aefa4362d)	Grupos de seguridad de la VPC databases-sg (sg-0e1a3747535dfc3b2) Activo
Puerto 5432	Grupo de subredes databases-grupo-subred	Accesible públicamente No

Ilustración 82 - RDS posgres AWS

Características de creación del RDS:

- **Método de creación:** creación estándar
- **Opciones de motor:** PosgreSQL
- **Versión del motor:** PostgreSQL 14.1-R1
- **Plantillas:** capa gratuita
- **Configuración:**
 - o **Identificador de la instancia:** sensor-postgres
 - o **Nombre de usuario maestro:** postgres
 - o **Contraseña maestra:** password
- **Configuración de la instancia:**
 - o **Clase de instancia de base de datos:** db.t3.micro

db.t3.micro
 2 vCPUs 1 GiB RAM Red: 2085 Mbps
- **Conectividad:**
 - o **VPC:** vpc-principal
 - o **Grupo de subredes:** databases-grupo-subred
 - o **Acceso público:** No (no se asigna una IP pública)
 - o **Grupo de seguridad:** databases-sg
 - o **Zona de disponibilidad:** us-east-1a

- **Puerto de la base de datos:** 5432
- **Configuración adicional:**
 - **Nombre de base de datos inicial:** sensors

RDS MySQL

El servicio de MySQL se configurará también de la misma que en la fase de contenedores. Se desplegará en el puerto 3306, creará la base de datos “users” y para su conexión se ha establecido “root” y “password” como usuario y contraseña (Ilustración 83).

Resumen			
Identificador de base de datos authentication-mysql	CPU <div style="width: 2.96%;">2.96%</div>	Estado Disponible	Clase db.t3.micro
Rol Instancia	Actividad actual <div style="width: 0%;">0 Conexiones</div>	Motor MySQL Community	Región y AZ us-east-1a

Conectividad y seguridad				
Punto de enlace y puerto Punto de enlace authentication-mysql.cvt9cc7dxazz.us-east-1.rds.amazonaws.com Puerto 3306	Redes Zona de disponibilidad us-east-1a VPC vpc-principal (vpc-0cd81c07afea4362d)	Seguridad Grupos de seguridad de la VPC databases-sg (sg-0e1a3747535dfc3b2) Activo Accesible públicamente No		

Ilustración 83 - RDS mysql AWS

Características de creación del RDS

- **Método de creación:** creación estándar
- **Opciones de motor:** MySQL
- **Versión del motor:** MySQL 5.7.41
- **Plantillas:** capa gratuita
- **Configuración:**
 - **Identificador de la instancia:** authentication-mysql
 - **Nombre de usuario maestro:** root
 - **Contraseña maestra:** password

- **Configuración de la instancia:**
 - o **Clase de instancia de base de datos:** db.t3.micro

db.t3.micro
 2 vCPUs 1 GiB RAM Red: 2085 Mbps
- **Conectividad:**
 - o **VPC:** vpc-principal
 - o **Grupo de subredes:** databases-grupo-subred
 - o **Acceso público:** No (no se asigna una IP pública)
 - o **Grupo de seguridad:** databases-sg
 - o **Zona de disponibilidad:** us-east-1a
 - o **Puerto de la base de datos:** 3306
 - o **Configuración adicional:**
 - **Nombre de base de datos inicial:** users

EC2 MongoDB

El servicio de MongoDB será desplegado mediante un contenedor Docker en el puerto 27017 de la máquina EC2. En los datos de usuario, se configurará el lanzamiento del contenedor para que utilice un volumen montado sobre el EFS y ejecutará el script de configuración “init-mongodb.js” utilizado anteriormente en la sección de MongoDB dentro del apartado [6.1.4. Definición del lanzamiento de contenedores con Docker-Compose](#), el cual será descargado del bucket S3 “db-services-bucket” de la infraestructura (Ilustración 84).

Instancias (1/1) Información

Name	ID de la Instancia	Estado de la I...	Tipo d...	Comprobación ...	Estado de la ...
ec2-plots-mongo	i-0895f559974fbbae1	En ejecución	t2.micro	2/2 comprobacion	Sin alarmas

Instancia: i-0895f559974fbbae1 (ec2-plots-mongo)

Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas			
Resumen de instancia Información <table border="1"> <tr> <td>ID de la Instancia i-0895f559974fbbae1 (ec2-plots-mongo)</td> <td>Dirección IPv4 pública -</td> <td>Direcciones IPv4 privadas 10.0.2.168</td> </tr> </table>							ID de la Instancia i-0895f559974fbbae1 (ec2-plots-mongo)	Dirección IPv4 pública -	Direcciones IPv4 privadas 10.0.2.168
ID de la Instancia i-0895f559974fbbae1 (ec2-plots-mongo)	Dirección IPv4 pública -	Direcciones IPv4 privadas 10.0.2.168							

Ilustración 84 - EC2 mongo AWS

Características de creación de la EC2 (Modelo Bajo demanda “On-Demand”):

- **Nombre:** ec2-plots-mongo.
- **Imagen de Amazon (AMI):** Amazon Linux 2023 de uso gratuito.

AMI de Amazon Linux 2023	Apto para la capa gratuita
ami-0889a44b331db0194 (64 bits (x86)) / ami-08fc6fb8ad2e794bb (64 bits (Arm))	
Virtualización: hvm Habilitado para ENA: true Tipo de dispositivo raíz: ebs	

- **Tipo de instancia:** t2.micro de uso gratuito.

t2.micro	Apto para la capa gratuita
Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true	

- **Par de claves:** clave_maquinas_TFG
- **Configuraciones de red:**
 - **VPC:** vpc-principal
 - **Subred:** subred-databases-1a
 - **Asignar automáticamente la IP pública:** Desactivar
 - **Grupo de seguridad:** databases-sg
- **Detalles avanzados:**
 - **Perfil de instancia de IAM:** AmazonS3FullAccess (rol con el permiso necesario para acceder a los archivos del bucket S3).
 - **Datos de usuario:** El script ejecutado realiza el montaje del EFS utilizando la dirección del sistema de ficheros EFS dentro de la carpeta “/efs”, instala Docker y crea el volumen Docker “mongodb-volume” para montarlo sobre la carpeta dentro del EFS reservada para el servicio de MongoDB “/mongo-efs”. Después, mediante el comando “aws s3 cp” (la instancia Linux viene con el cliente de AWS instalado) se accede a la dirección donde se encuentra desplegado el bucket S3 para copiar dentro de la carpeta “/init_mongo” el script “init-mongodb.js” utilizado para inicializar la base de datos. Finalmente, se importará una imagen mongo:5.0.7 de DockerHub y se lanzará el contenedor montando el volumen creado sobre la ruta “/data/db” (donde MongoDB guarda los datos) y el script init sobre la carpeta “/docker-entrypoint-initdb.d” para ejecutar el script al iniciar el contenedor. Es muy importante establecer las variables de entorno “MONGO_INITDB_ROOT_USERNAME” con el usuario junto a “MONGO_INITDB_ROOT_PASSWORD” con la contraseña para que MongoDB autentifique el usuario en su base de datos “admin”.

Este es el script utilizado dentro de los datos de usuario de la instancia EC2 MongoDB:

- **Script EC2 MongoDB:**

```
#!/bin/bash

# 1)
# Instalacion herramientas necesarias
yum install nfs-utils -y
# Directorio de montaje /efs en la carpeta raiz
mkdir /efs
# Montaje EFS en el path indicado /efs
mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport
t fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com:/ efs
# Montaje del EFS automatico al reiniciar la instancia
echo "fs-0f2f65bb0548dde2f.efs.us-east-1.amazonaws.com:/ /efs nfs4
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 0 0" >>
/etc/fstab
# Sleep 10 segundos por si necesita tiempo para montarse
sleep 10

# 2)
# Instalar Docker
yum install docker -y
usermod -a -G docker ec2-user
id ec2-user
newgrp docker
sudo systemctl enable docker.service
sudo systemctl start docker.service

# 3)
# Volumen mongodb-volume creado
docker volume create \
--driver local \
--opt type=nfs \
--opt o=addr=fs-0f2f65bb0548dde2f.efs.us-east-
1.amazonaws.com,nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,re-
trans=2,noresvport \
--opt device=/mongo-efs \
mongodb-volume

# 4)
# Copiar en el equipo el script guardado en el S3
aws s3 cp s3://db-services-bucket/init-mongodb.js /init_mongo/init-
mongodb.js
# Docker image MongoDB version 5.0.7
docker pull mongo:5.0.7
# Ejecutar contenedor con el volumen y el comando "mongod" para iniciar el
server
docker run --name mongodb -d -p 27017:27017 -e
MONGO_INITDB_ROOT_USERNAME='admin' -e MONGO_INITDB_ROOT_PASSWORD='example'
-v mongodb-volume:/data/db -v /init_mongo/init-mongodb.js:/docker-
entrypoint-initdb.d/init-mongodb.js:ro mongo:5.0.7 mongod
```

EC2 Cassandra

El servicio de Cassandra será desplegado mediante un contenedor Docker en el puerto 9042 de la máquina EC2 (Ilustración 86). Dentro de los datos de usuario, se lanzará el contenedor a partir de un fichero docker-compose descargado del bucket S3 “db-services-bucket” de la infraestructura. Dentro del fichero compose se utilizarán otros dos ficheros que también se obtendrán del bucket s3 los cuales estarán implicados en el proceso de configuración de Cassandra: "cassandra_init.sh" e "init_keyspace.cql". El primero es un script para configurar el contenedor, que utilizará el segundo fichero para crear el keyspace inicial dentro de la base de datos. Estos ficheros estarán localizados en la carpeta “cassandra” del bucket S3 (Ilustración 85):

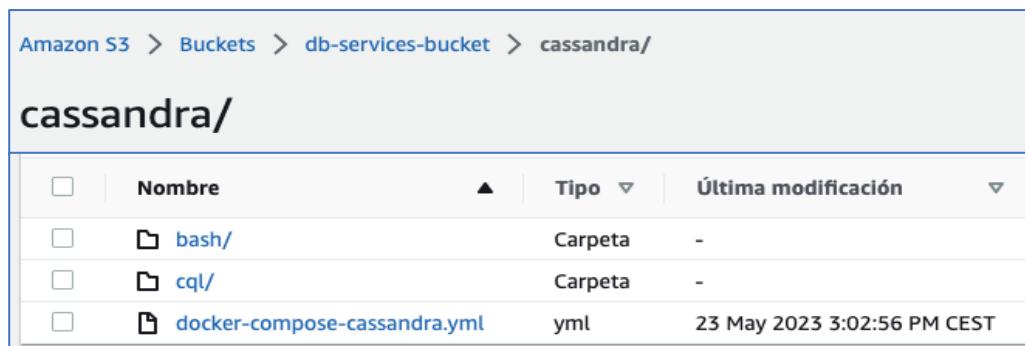


Ilustración 85 - Carpeta cassandra AWS

Dentro de la carpeta “**bash**” se encuentra el fichero **init_keyspace.yml** y dentro de “**cql**” está el fichero **cassandra_init.sh**.



Ilustración 86 - EC2 cassandra AWS

Durante la primera fase del desarrollo, se propuso una solución para el despliegue del contenedor de Cassandra utilizando un solo contenedor mediante una imagen Docker personalizada (apartado [Dockerfile del servicio Cassandra](#)). La solución que se propone para lanzar Cassandra en esta ocasión es muy parecida, pero utilizando dos contenedores, donde uno de ellos levantará el servicio y el otro se conectará a este para configurarlo. Ambas soluciones son válidas, pero la que se aplicará en este apartado posee la ventaja de desplegar el servicio de Cassandra en un primer contenedor independiente que no se bloqueará durante su configuración, la cual realizará el segundo contenedor.

El siguiente fichero Docker-Compose será el encargado de lanzar ambos contenedores:

- **Fichero docker-compose-cassandra.yml**

```
cassandra_db:  
  image: cassandra:4.1.0  
  container_name: cassandra_db  
  volumes:  
    - cassandra_volume:/var/lib/cassandra  
  ports:  
    - "9042:9042"  
  
cassandra_aux:  
  image: cassandra:4.1.0  
  container_name: cassandra_aux  
  volumes:  
    - ./cql:/cql  
    - ./bash:/docker-entrypoint.sh  
  command: >  
    bash ./docker-entrypoint.sh/cassandra_init.sh  
  
volumes:  
  cassandra_volume:  
    driver_opts:  
      type: "nfs"  
      o: "addr=fs-0f2f65bb0548dde2f.efs.us-east-  
1.amazonaws.com,nfsvers=4.1,rsize=1048576,wszie=1048576,hard,timeo=600,r-  
etrans=2,noresvport"  
    device: "/cassandra-efs"
```

El fichero compose levanta dos contenedores, ambas imágenes oficiales cassandra:4.1.0, y crea un volumen. El primer contenedor “cassandra_db” lanzará el servicio principal de Cassandra en el puerto 9042, montando el volumen “cassandra_volume” dentro de su directorio “/var/lib/cassandra” donde la base de datos almacena los datos. En la configuración del volumen se establece la dirección del EFS para realizar el montaje del EFS sobre este, más concretamente sobre la carpeta “/cassandra-efs” reservada para el

servicio de Cassandra. El segundo contenedor también es un servicio Cassandra, pero su función será la de configurar el primer contenedor y finalizar su ejecución. Montará las carpetas “cql” y “bash” junto con sus ficheros de la carpeta “cassandra” dentro del contenedor en los directorios “cql” y “/docker-entrypoint.sh” respectivamente. Cuando se lance el contenedor, ejecutará el script “cassandra_init.sh” en vez del fichero entrypoint que ejecuta la imagen de Cassandra normalmente. Por este motivo, este segundo contenedor se detendrá una vez configurado el contendor principal.

Los ficheros utilizados por el segundo contenedor implicados en la configuración son los siguientes:

- Fichero **cassandra_init.sh**

```
#!/bin/bash

echo " * * * Ejecutando script * * * "
USER_NAME='cassandra'
PASSWORD='cassandra'

while ! cqlsh cassandra_db -u "${USER_NAME}" -p "${PASSWORD}" -e 'describe
cluster' ; do
    echo " * * * Esperando al servicio principal de Cassandra... * * *"
    sleep 1
done

for cql_file in ./cql/*.cql;
do
    cqlsh cassandra_db -u "${USER_NAME}" -p "${PASSWORD}" -f "${cql_file}" ;
    echo " * * * Script ""${cql_file}"" ejecutado !!! * * *"
done
echo " * * * Fin de la ejecución del script"
echo " * * * Parando el contenedor temporalmente... * * * "
```

El primer bucle realiza la conexión con el contenedor de Cassandra principal utilizando el comando “cqlsh” con el usuario por defecto “cassandra” y su contraseña “cassandra”. Este comando utiliza el nombre de host “cassandra_db” para conectarse al contenedor de Cassandra ya que es su nombre de servicio identificadorio dentro del docker-compose, y puesto que ambos contenedores se encuentran desplegados en la misma red de Docker puede ser reconocido. El segundo bucle, una vez que se establece la conexión con el clúster, ejecuta los ficheros contenidos dentro de la carpeta “cql” de extensión “.cql”, donde se encontrará el fichero “init_keyspace.cql” con las consultas a realizar sobre la base de datos.

- Fichero **init_keyspace.cql**

```
CREATE KEYSPACE IF NOT EXISTS sensorKeyspace
    WITH REPLICATION = {
        'class' : 'SimpleStrategy',
        'replication_factor' : 1
    };
```

Este fichero realizará una consulta para la creación del keyspace “sensorKeyspace” en el caso de que no exista previamente gracias a la condición de creación “IF NOT EXISTS”. Es ejecutado por el script explicado anteriormente.

Características de creación de la EC2 (Modelo Bajo demanda u “On-Demand”):

- **Nombre:** ec2-statistics-cassandra
- **Imagen de Amazon (AMI):** Amazon Linux 2023 de uso gratuito.

AMI de Amazon Linux 2023	Apto para la capa gratuita
ami-0889a44b331db0194 (64 bits (x86)) / ami-08fc6fb8ad2e794bb (64 bits (Arm))	
Virtualización: hvm Habilitado para ENA: true Tipo de dispositivo raíz: ebs	

- **Tipo de instancia:** t2.medium de pago

t2.medium	
Familia: t2 2 vCPU 4 GiB Memoria Generación actual: true	

- **Par de claves:** clave_maquinas_TFG
- **Configuraciones de red:**
 - **VPC:** vpc-principal
 - **Subred:** subred-databases-1a
 - **Asignar automáticamente la IP pública:** Desactivar
 - **Grupo de seguridad:** databases-sg
- **Detalles avanzados:**
 - **Perfil de instancia de IAM:** AmazonS3FullAccess (rol con el permiso necesario para acceder a los archivos del bucket S3).
 - **Datos de usuario:** El script ejecutado realiza la instalación de Docker y Docker-Compose para la ejecución del fichero “docker-compose-cassandra.yml” contenido en la carpeta “cassandra” descargada del **bucket S3 “db-services-bucket”** a través del comando “aws s3 cp”. Para ejecutar el comando de docker-compose, primero se situará el directorio de trabajo mediante el comando “cd” en la carpeta “cassandra” puesto que es donde se encuentran todos los ficheros que se utilizan.

Este es el script utilizado dentro de los datos de usuario de la instancia EC2 Cassandra:

- **Script EC2 Cassandra**

```
#!/bin/bash

# 1)
# Instalar Docker
yum install docker -y
usermod -a -G docker ec2-user
id ec2-user
newgrp docker
sudo systemctl enable docker.service
sudo systemctl start docker.service

# 2)
# Copiar en el equipo la carpeta con los ficheros de Cassandra
aws s3 cp s3://db-services-bucket/cassandra cassandra --recursive

# 3)
# Instalar Docker-Compose
sudo dnf -y install wget
sudo curl -s
https://api.github.com/repos/docker/compose/releases/latest | grep
browser_download_url | grep docker-compose-linux-x86_64 | cut -d
'"' -f 4 | wget -qi -
sudo chmod +x docker-compose-linux-x86_64
sudo mv docker-compose-linux-x86_64 /usr/local/bin/docker-compose

# 4)
# Lanzar contenedores con el fichero docker-compose
cd /cassandra
docker-compose -f docker-compose-cassandra.yml up -d
```

6.2.5. Clúster ECS y despliegue de microservicios

El servicio ECS se aprovechará para desplegar los contenedores de la aplicación de una manera rápida y automatizada sin tener que acceder individualmente a cada máquina para lanzarlos. Se creará un clúster con dos instancias EC2 y se utilizarán las herramientas que posee ECS para lanzar los contenedores llamadas tareas. Mediante estas tareas, se podrán importar las imágenes Docker que se subieron a Docker-Hub con la implementación de los microservicios de la aplicación para su posterior lanzamiento. Se creará una tarea para cada servicio API REST de la aplicación: Plots, Sensors, Authentication y Statistics.

Al crear el clúster también se crea automáticamente un grupo de autoescalado para las instancias EC2 desplegadas. Este servicio permite a ECS mantener

siempre levantadas un número de instancias concreto en todo momento en caso de que éstas se detengan. No obstante, no se entrará en detalle en este servicio puesto que se intentará evitar que las instancias estén en constante ejecución para ahorrar en gasto de consumo de recursos de AWS durante el desarrollo del proyecto, por lo que generalmente se destruirá para poder detenerlas cuando sea conveniente.

Lista de servicios ECS que se van a utilizar:

- ECS Clúster: 1
- Tareas: 4

Estos servicios se encuentran disponibles en el **apartado “ECS”** de AWS.

ECS Clúster

Se levantarán dos instancias contenedoras EC2 con AMIs optimizadas para ECS, en las que vendrá instalado el agente de ECS que se ejecutará dentro de las instancias a través de un contenedor Docker y que permitirá la conexión de estas al clúster. Estarán desplegadas en la subred de servicios pública “us-east-1^a” (Ilustración 87).



Ilustración 87 - Clúster ECS AWS

En la siguiente figura (Ilustración 89) se puede observar que ambas instancias tienen disponibles todos los recursos de CPU (1024) y de Memoria (970).

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there are tabs for 'Servicios', 'Tareas', 'Instancias de ECS' (which is selected), 'Métricas', 'Tareas programadas', 'Etiquetas', and 'Proveedores de capacidad'. Below the tabs, a message states: 'Una instancia de Amazon ECS es una instancia externa registrada mediante ECS Anywhere o una instancia de Amazon EC2. Para registrar una instancia externa, seleccione Registrar instancias externas y siga los pasos. [Más información](#). Para registrar una instancia de Amazon EC2, puede utilizar la consola de Amazon EC2. [Más información](#)'. Below this is a button 'Registrar instancias externas' and a dropdown menu 'Acciones'. A status bar indicates 'Última actualización realizada a las mayo 30, a. m. 11:13:54 a. m. (hace 1 minutos)'. On the right, there are icons for refresh, settings, and help. The main area shows a table with columns: 'Instancia de co...', 'Instancia de E...', 'Zona de dis...', 'Instancia ...', 'Agente conectado', 'Estado', 'Recuent...', 'CPU disp...', and 'Memoria'. The table contains three rows of data. At the bottom left, it says 'Filtrar por atributos (haga clic o pulse la flecha hacia abajo para ver las opciones de filtro)'. At the bottom right, there are buttons for '1-2' and 'Tamaño de la página' set to 50.

Ilustración 88 - Instancias ECS AWS

Características de creación del clúster:

- **Plantilla de clúster:** EC2 Linux + Redes (tipo de infraestructura a utilizar en el clúster)
- **Nombre:** cluster-app
- **Configuración de la instancia:**
 - o **Modelo de aprovisionamiento:** Instancia bajo demanda
 - o **Tipo de instancia:** t2.micro

t2.micro	Apto para la capa gratuita		
Familia: t2	1 vCPU	1 GiB Memoria	Generación actual: true

 - o **Cantidad de instancias:** 2
 - o **ID de la AMI de EC2:**
 - [ami-0695f2f761eeb04e6] Amazon ECS-optimized Amazon Linux 2 AMI (64 bits (x86))
 - o **Par de claves:** clave_maquinas_TFG
- **Redes:**
 - o **VPC:** vpc-principal
 - o **Subredes:** subred-servicios-1a
 - o **Asignar automáticamente una IP pública:** Habilitado
 - o **Grupo de seguridad:** servicios-sg
 - o **Rol de IAM de instancia de contenedor:** ecsInstanceRole (Rol necesario para que el agente instalado dentro de las EC2 pueda realizar las llamadas a la API de Amazon ECS correctamente)

Primera instancia de contenedor creada (Ilustración 89):

The screenshot shows the AWS ECS console with the following details:

Clústeres > cluster-app > Instancia de contenedor: 1f293cc490b745b28a2222369cfb04f6

Instancia de contenedor : 1f293cc490b745b28a2222369cfb04f6

Detalles tab is selected.

	Recursos	Registrado	Disponible
Clúster cluster-app			
Instancia de EC2 i-0c81cdb358b1bee23	CPU	1024	1024
Sistema operativo Linux	Memoria	970	970
Zona de disponibilidad us-east-1a	Puertos	5 puertos	
DNS privado ip-10-0-0-45.ec2.internal			
IP privada 10.0.0.45			
Estado ACTIVE			
Recuento de tareas en ejecución 0			
Versión del agente 1.71.1			
Versión de Docker 20.10.22			

Anular el registro and **Actualizar el agente** buttons are at the top right.

Ilustración 89 - Instancia ECS 1

La instancia de contenedor se corresponde con la instancia EC2 indicada a través de su identificador (Ilustración 90):

The screenshot shows the AWS EC2 Instances page with the following details:

Instancias (1/1)

Estado de la instancia: En ejecución

Detalle de la instancia: i-0c81cdb358b1bee23 (1 ECS Instance - EC2ContainerService-cluster-app)

Resumen de instancia:

ID de la Instancia i-0c81cdb358b1bee23 (1 ECS Instance - EC2ContainerService-cluster-app)	Dirección IPv4 pública 3.95.21.58 dirección abierta	Direcciones IPv4 privadas 10.0.0.45
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública ec2-3-95-21-58.compute-1.amazonaws.com dirección abierta

Ilustración 90 - Instancia EC2 de ECS (1/2)

Segunda instancia de contenedor creada (Ilustración 91):

The screenshot shows the AWS CloudWatch Metrics interface for an ECS container instance. At the top, it displays the path: Clústeres > cluster-app > Instancia de contenedor: 23c94817eceb4beb91f38ac46efd302c. Below this, the title "Instancia de contenedor :" and the identifier "23c94817eceb4beb91f38ac46efd302c" are shown. There are two buttons on the right: "Anular el registro" and "Actualizar el agente". The main area contains two tabs: "Detalles" (selected) and "Etiquetas". The "Detalles" tab displays the following information:

	Recursos	Registrado	Disponible
Clúster	cluster-app		
Instancia de EC2	i-0a1f60acf77b46271	CPU	1024
Sistema operativo	Linux	Memoria	970
Zona de disponibilidad	us-east-1a	Puertos	5 puertos
DNS privado	ip-10-0-0-50.ec2.internal		
IP privada	10.0.0.50		
Estado	ACTIVE		
Recuento de tareas en ejecución	0		
Versión del agente	1.71.1		
Versión de Docker	20.10.22		

Ilustración 91 - Instancia ECS 2

La instancia de contenedor se corresponde con la instancia EC2 indicada a través de su identificador (Ilustración 92):

The screenshot shows the AWS EC2 Instances page. At the top, it displays "Instancias (1/1)" and "Información". Below this, there are buttons for "Conejar", "Estado de la Instancia", "Acciones", "Lanzar instancias", and a dropdown. A search bar says "Buscar instancia por atributo o etiqueta (case-sensitive)". Below the search bar, there are buttons for "2 ECS" and "X", and a link to "Quitar los filtros". The main table lists one instance:

<input checked="" type="checkbox"/>	Name	ID de la instancia	Estado de la i...	Tipo de...	Comprobación ...	Estado de la ...	Zo...
<input checked="" type="checkbox"/>	2 ECS Instance - EC2Cont...	i-0a1f60acf77b46271	En ejecución	t2.micro	2/2 comprobador	Sin alarmas	us...

Below the table, a modal window titled "Instancia: i-0a1f60acf77b46271 (2 ECS Instance - EC2ContainerService-cluster-app)" is open. It shows the "Detalles" tab, which includes a "Resumen de instancia" section with the following details:

ID de la Instancia	Dirección IPv4 pública	Direcciones IPv4 privadas
i-0a1f60acf77b46271 (2 ECS Instance - EC2ContainerService-cluster-app)	3.83.109.80 dirección abierta	10.0.0.50
Dirección IPv6	Estado de la instancia	DNS de IPv4 pública
-	En ejecución	ec2-3-83-109-80.compute-1.amazonaws.com dirección abierta

Ilustración 92 - Instancia EC2 ECS (2/2)

Tareas

Los contenedores se lanzarán a través de cuatro tareas, una por cada contenedor (Ilustración 93):

Estado: ACTIVE INACTIVE		
<input type="checkbox"/>	Definición de tarea	
<input type="checkbox"/>	tarea-authentication	ACTIVE
<input type="checkbox"/>	tarea-plots	ACTIVE
<input type="checkbox"/>	tarea-sensor	ACTIVE
<input type="checkbox"/>	tarea-statistics	ACTIVE

Ilustración 93 - Lista tareas AWS

Este servicio de tareas se comunicará con el bucket S3 de la infraestructura “db-services-bucket” (Ilustración 94) para acceder al fichero “env_vars.env”, utilizado para importar las variables de entorno necesarias dentro de los contenedores con los microservicios durante su lanzamiento.

The screenshot shows the AWS S3 console interface. The top navigation bar shows the path: Amazon S3 > Buckets > db-services-bucket > env_vars.env. Below the path, the file name "env_vars.env" is displayed with an "Info" link. To the right of the file name are four buttons: "Copiar URI de S3", "Descargar", "Abrir", and "Acciones de objetos". Below these buttons are three tabs: "Propiedades" (selected), "Permisos", and "Versiones". The main content area is titled "Información general sobre el objeto". It contains two columns of information:

Propietario	URI DE S3
jesubc97	s3://db-services-bucket/env_vars.env
Región de AWS	Nombre de recurso de Amazon (ARN)
EE. UU. Este (Norte de Virginia) us-east-1	arn:aws:s3:::db-services-bucket/env_vars.env

Ilustración 94 - Env vars AWS

Dentro de la definición de contenedor de la tarea, se importarán los contenedores del repositorio público propio donde se subieron las imágenes con los microservicios desarrollados ya configurados. También se expondrán los puertos correspondientes donde se desplegará cada servicio y se establecerá un límite de memoria flexible de unos 300 MiB, lo que quiere decir que se reserva esa cantidad de memoria para el contenedor. La asignación de memoria deberá adecuarse a las necesidades de los contenedores que se van a ejecutar, y debe establecerse o bien a nivel de tarea, o bien a nivel de contenedor, en este caso se opta por la segunda opción.

Características de creación de las tareas:

1. Tarea del servicio Plots

- **Compatibilidad con el tipo de lanzamiento:** EC2
- **Nombre de la definición de tarea:** tarea-plots
- **Rol de la tarea:** ecsTaskExecutionRole (rol para poder realizar operaciones ECS).
- **Rol de ejecución de tareas:** escTaskExecutionRole
- **Definición de contenedor:**
 - o **Nombre del contenedor:** plots
 - o **Imagen:** jesusbc/plots
 - o **Límites de memoria (MiB):** 300 (límite flexible)
 - o **Asignaciones de puertos:** 8080:8080
 - o **Archivos de entorno:**
 - Origen S3 ARN – Ubicación arn:aws:s3:::db-services-bucket/env_vars.env

2. Tarea del servicio Sensor

- **Compatibilidad con el tipo de lanzamiento:** EC2
- **Nombre de la definición de tarea:** tarea-sensor
- **Rol de la tarea:** ecsTaskExecutionRole (rol para poder realizar operaciones ECS).
- **Rol de ejecución de tareas:** escTaskExecutionRole
- **Definición de contenedor:**
 - o **Nombre del contenedor:** sensor
 - o **Imagen:** jesusbc/sensor
 - o **Límites de memoria (MiB):** 300 (límite flexible)
 - o **Asignaciones de puertos:** 8081:8081
 - o **Archivos de entorno:**
 - Origen S3 ARN – Ubicación arn:aws:s3:::db-services-bucket/env_vars.env

3. Tarea del servicio Authentication

- **Compatibilidad con el tipo de lanzamiento:** EC2
- **Nombre de la definición de tarea:** tarea-authentication
- **Rol de la tarea:** ecsTaskExecutionRole (rol para poder realizar operaciones ECS).
- **Rol de ejecución de tareas:** escTaskExecutionRole
- **Definición de contenedor:**
 - o **Nombre del contenedor:** authentication
 - o **Imagen:** jesusbc/authentication

- **Límites de memoria (MiB)**: 300 (límite flexible)
- **Asignaciones de puertos**: 8999:8999
- **Archivos de entorno**:
 - S3 ARN – arn:aws:s3:::db-services-bucket/env_vars.env

4. Tarea del servicio Statistics

- **Compatibilidad con el tipo de lanzamiento**: EC2
- **Nombre de la definición de tarea**: tarea-statistics
- **Rol de la tarea**: ecsTaskExecutionRole (rol para poder realizar operaciones ECS).
- **Rol de ejecución de tareas**: escTaskExecutionRole
- **Definición de contenedor**:
 - **Nombre del contenedor**: statistics
 - **Imagen**: jesusbc/statistics
 - **Límites de memoria (MiB)**: 300 (límite flexible)
 - **Asignaciones de puertos**: 8082:8082
 - **Archivos de entorno**:
 - Origen S3 ARN – Ubicación arn:aws:s3:::db-services-bucket/env_vars.env

Una vez definidas las tareas, durante el proceso de ejecución se establecerá un tipo de **plantilla de colocación de tarea** (Ilustración 90). Estas plantillas permiten personalizar la forma o estrategia en la que se reparten las tareas entre las instancias contenedoras para ser ejecutadas. La plantilla escogida ha sido “Reparto equilibrado en AZ”, que repartirá las tareas entre las zonas de disponibilidad del clúster (en este caso solo en **us-east-1a**), y dentro de éstas, las repartirá también entre las distintas instancias para que no se desplieguen todas en una sola máquina del clúster. Escogerá una máquina con la suficiente capacidad de memoria disponible para ejecutar el contenedor.



Ilustración 95 - Colocación tareas AWS

6.2.6. EC2 Front-End

El Front-End de la aplicación se desplegará en una instancia EC2 independiente al clúster ECS. La razón principal de esta decisión es el gran tamaño de la imagen y su contenedor resultante, puesto que este consume

una cantidad elevada de memoria dentro de la máquina. Esto provocaba que, al desplegarlo utilizando el clúster, cuando este contenedor se ejecutaba junto con otros servicios contenedores en una misma máquina, causaba problemas dentro de la EC2 de tipo “t2.micro” con unas capacidades de cómputo ligeras (1 CPU / 1 GB de Memoria). Incluso se ha tenido que escoger un tipo de instancia “t2.small” (1 CPU / 2 GB de Memoria) con un poco más de memoria para que funcionara correctamente (Ilustración 96).

Instancias (1/1) Información					
C Conectar		Estado de la instancia ▾		Acciones ▾	
<input type="text"/> Buscar instancia por atributo o etiqueta (case-sensitive)					
front	X	Quitar los filtros			
	Name	ID de la instancia	Estado de la i...	Tipo de...	Comprobación ...
<input checked="" type="checkbox"/>	ec2-front	i-0d0e1f716e888898c	En ejecución	t2.small	2/2 comprobacion

Instancia: i-0d0e1f716e888898c (ec2-front)					
Detalles		Seguridad	Redes	Almacenamiento	Comprobaciones de estado
Resumen de instancia Información					
ID de la instancia	Dirección IPv4 pública			Direcciones IPv4 privadas	
i-0d0e1f716e888898c (ec2-front)	54.144.89.102 dirección abierta			10.0.1.16	
Dirección IPv6	Estado de la instancia			DNS de IPv4 pública	
-	En ejecución			ec2-54-144-89-102.compute-1.amazonaws.com dirección abierta	

Ilustración 96 - EC2 Front AWS

Esta instancia EC2 levantará el servicio Front-End en el puerto 4200 de la máquina y se desplegará en la **subred front (us-east-1b)**.

Características de creación de la instancia EC2 (Modelo Bajo demanda u “On-Demand”):

- **Nombre:** ec2-front
- **Imagen de Amazon (AMI):** Amazon Linux 2023 de uso gratuito.

AMI de Amazon Linux 2023	Apto para la capa gratuita
ami-0889a44b331db0194 (64 bits (x86)) / ami-08fc6fb8ad2e794bb (64 bits (Arm))	
Virtualización: hvm Habilitado para ENA: true Tipo de dispositivo raíz: ebs	

- **Tipo de instancia:** t2.small de pago.

t2.small
Familia: t2 1 vCPU 2 GiB Memoria Generación actual: true

- **Par de claves:** clave_maquinas_TFG

- **Configuraciones de red:**
 - o **VPC:** vpc-principal
 - o **Subred:** subred-front-1b
 - o **Asignar automáticamente la IP pública:** Activar
 - o **Grupo de seguridad:** front-sg
- **Detalles avanzados:**
 - o **Perfil de instancia de IAM:** AmazonS3FullAccess (rol con el permiso necesario para acceder a los archivos del bucket S3).
 - o **Datos de usuario:** El script ejecutado realiza la instalación de Docker y descarga el archivo "env_vars.env" del **bucket S3 "db-services-bucket"** con las variables de entorno que utilizará el contenedor mediante el comando "aws s3 cp". Después, ejecuta el contenedor con el servicio Front descargado a partir del repositorio de Docker-Hub.

Este es el script utilizado dentro de los datos de usuario de la instancia EC2 Front:

- **Script EC2 Front:**

```
#!/bin/bash

# 1)
# Instalar Docker
yum install docker -y
usermod -a -G docker ec2-user
id ec2-user
newgrp docker
sudo systemctl enable docker.service
sudo systemctl start docker.service

# 2)
# Copiar en el equipo el fichero de variables de entorno
aws s3 cp s3://db-services-bucket/env_vars.env env_vars.env

# 3)
# Docker image del Front
docker pull jesusbc/tfm-front-end:latest
# Ejecutar el contenedor
docker run --name tfm-front-end -d -p 4200:4200 --env-file
env_vars.env jesusbc/tfm-front-end:latest
```

6.2.7. Balanceador de carga ALB

Para distribuir el tráfico entre los diferentes destinos o máquinas EC2 desplegadas en las dos zonas de disponibilidad us-east-1a y us-east-1b, se utilizará un balanceador de carga de tipo ALB o “Application Load Balancer”. El ALB tendrá una serie de agentes de escucha o “listeners” escuchando las solicitudes entrantes HTTP, uno por cada puerto donde están desplegados los microservicios. Antes de definir el ALB primero se crearán los grupos de destino, también tantos como puertos utilizan los microservicios, que se encargarán de dirigir las solicitudes a las instancias destino registradas en el grupo utilizando el protocolo y puerto. Una vez que se establecen los grupos se crea la ALB y se asigna a cada “listener” su correspondiente grupo de destino, para cuando se comprueben las peticiones de conexión entrantes por el puerto del “listener” se puedan enviar hacia su correspondiente destino.

Lista de servicios EC2 que se van a utilizar:

- Balanceador de carga: 1
- Grupos de destino: 5

Estos servicios se encuentran disponibles en el **apartado “EC2”** de AWS.

Grupos de destino

Se definirán cinco grupos de destino para dirigir el tráfico a las instancias EC2 donde se encuentran los servicios en los puertos: 4200 (Front), 8080 (Plots), 8081 (Sensor), 8082 (Statistics) y 8999 (Authenticacion). Primero se crean los grupos y después se registrarán los destinos (Ilustración 97).

Target groups (5) Info						
		Actions			Create target group	
		<input type="text"/> Find resources by attribute or tag			< 1 > ⟳	
	Name	ARN	Port	Protocol	Target type	Load balancer
<input type="checkbox"/>	front	arn:aws:elasticloadbalanci...	4200	HTTP	Instance	ALB-principal
<input type="checkbox"/>	plots	arn:aws:elasticloadbalanci...	8080	HTTP	Instance	ALB-principal
<input type="checkbox"/>	sensor	arn:aws:elasticloadbalanci...	8081	HTTP	Instance	ALB-principal
<input type="checkbox"/>	statistics	arn:aws:elasticloadbalanci...	8082	HTTP	Instance	ALB-principal
<input type="checkbox"/>	authentication	arn:aws:elasticloadbalanci...	8999	HTTP	Instance	ALB-principal

Ilustración 97 - Grupos destino AWS

Cada grupo tendrá una configuración de la **comprobación de estado** o “**health check**” en la que se establecerá una ruta concreta a la que se enviarán peticiones periódicamente para comprobar la salud de la instancia destino. Si la respuesta es positiva (status 200-299) ese destino estará saludable o “healthy” y recibirá las peticiones. El servicio Front establece la ruta en su directorio raíz y los servicios API REST en la ruta “/actuator/health”. Para más información sobre estas comprobaciones de estado se recomienda visitar el apartado de [8. Pruebas](#), concretamente la sección [8.1. Funcionalidad auxiliar de comprobación de estado de las API REST](#).

Al crear los grupos se establecen los **Atributos**, en los que se configurarán cómo se escogerá y se redirigirán las solicitudes entrantes hacia los destinos EC2 en función de su salud (saludables o no saludables). Se dejará la configuración por defecto con el equilibrio de carga entre zonas activado, que utilizará la **conmutación por error de DNS**. Esto quiere decir que el tráfico se dirigirá solo a los destinos saludables del grupo.

Características de creación de los grupos de destino:

1. Grupo de destino Front

Redirige las peticiones HTTP a las instancias EC2 en su puerto 4200:
<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:4200/>

- **Tipo de destino:** Instancias EC2
- **Nombre:** front
- **Protocolo y puerto:** HTTP – Puerto 4200
- **VPC:** vpc-principal
- **Comprobación de estado:**
 - o **Protocolo:** HTTP
 - o **Ruta de la comprobación de estado:** /

La ruta escogida corresponde al directorio raíz del servicio Front.

Se registrará al grupo de destino solo la instancia EC2 con el Front desplegado puesto que el servicio no se levantará en otra máquina.

2. Grupo de destino Plots

Redirige las peticiones HTTP a las instancias EC2 en su puerto 8080.
<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8080/>

- **Tipo de destino:** Instancias EC2
- **Nombre:** plots
- **Protocolo y puerto:** HTTP – Puerto 8080

- **VPC:** vpc-principal
- **Comprobación de estado:**
 - o **Protocolo:** HTTP
 - o **Ruta de la comprobación de estado:** /actuator/health

La ruta de comprobación de estado corresponde a la operación del servicio API REST Plots encargada de devolver su estado de salud:

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8080/actuator/health>

Se registrará al grupo de destino las instancias EC2 desplegadas en el clúster ECS ya que el servicio podrá asignarse a cualquiera de las dos máquinas. Marcará uno de los destinos como “healthy” indicando que se ha desplegado el servicio en esa instancia.

3. Grupo de destino Sensor

Redirige las peticiones HTTP a las instancias EC2 en su puerto 8081.

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8081/>

- **Tipo de destino:** Instancias EC2
- **Nombre:** sensor
- **Protocolo y puerto:** HTTP – Puerto 8081
- **VPC:** vpc-principal
- **Comprobación de estado:**
 - o **Protocolo:** HTTP
 - o **Ruta de la comprobación de estado:** /actuator/health

La ruta de comprobación de estado corresponde a la operación del servicio API REST Sensor encargada de devolver su estado de salud:

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8081/actuator/health>

Se registrará al grupo de destino las instancias EC2 desplegadas en el clúster ECS ya que el servicio podrá asignarse a cualquiera de las dos máquinas. Marcará uno de los destinos como “healthy” indicando que se ha desplegado el servicio en esa instancia.

4. Grupo de destino Statistics

Redirige las peticiones HTTP a las instancias EC2 en su puerto 8082.

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8082/>

- **Tipo de destino:** Instancias EC2

- **Nombre:** statistics
- **Protocolo y puerto:** HTTP – Puerto 8082
- **VPC:** vpc-principal
- **Comprobación de estado:**
 - o **Protocolo:** HTTP
 - o **Ruta de la comprobación de estado:** /actuator/health

La ruta de comprobación de estado corresponde a la operación del servicio API REST Statistics encargada de devolver su estado de salud:

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8082/actuator/health>

Se registrará al grupo de destino las instancias EC2 desplegadas en el clúster ECS ya que el servicio podrá asignarse a cualquiera de las dos máquinas. Marcará uno de los destinos como “healthy” indicando que se ha desplegado el servicio en esa instancia.

5. Grupo de destino Authentication

Redirige las peticiones HTTP a las instancias EC2 en su puerto 8999.

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8999/>

- **Tipo de destino:** Instancias EC2
- **Nombre:** statistics
- **Protocolo y puerto:** HTTP – Puerto 8999
- **VPC:** vpc-principal
- **Comprobación de estado:**
 - o **Protocolo:** HTTP
 - o **Ruta de la comprobación de estado:** /actuator/health

La ruta de comprobación de estado corresponde a la operación del servicio API REST Authentication encargada de devolver su estado de salud:

<http://ALB-principal-1848667940.us-east-1.elb.amazonaws.com:8999/actuator/health>

Se registrará al grupo de destino las instancias EC2 desplegadas en el clúster ECS ya que el servicio podrá asignarse a cualquiera de las dos máquinas. Marcará uno de los destinos como “healthy” indicando que se ha desplegado el servicio en esa instancia.

ALB

Se procede con la creación del balanceador de carga de aplicación y la definición de los “listeners” a los que se les asignarán los grupos de destino creados (Ilustración 98).

Cuando se encuentre el ALB disponible con sus “listeners” y grupos de destino asignados, la nueva dirección para acceder al Front utilizará la dirección DNS del ALB en la URL “<http://alb-principal-1848667940.us-east-1.elb.amazonaws.com:4200/>”.

Details	
Load balancer type Application	Status Active
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K
	VPC vpc-0cd81c07aefa4362d
	IP address type IPv4
	Availability Zones
	subnet-0cc3734c40221676d us-east-1a (use1-az2)
	subnet-02ff8267200d84ab4 us-east-1b (use1-az4)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:690031176274:loadbalancer/app/ALB-principal/56a1791f4a4afcfd	DNS name ALB-principal-1848667940.us-east-1.elb.amazonaws.com (A Record)

Ilustración 98 - ALB AWS

El ALB mapeará las subredes de cada zona de disponibilidad donde actuará, siendo éstas la subred de servicios (us-east-1a) donde se encuentran las EC2 desplegadas del clúster ECS con los servicios API REST y la subred Front (us-east-1b) con el servicio EC2 Front.

Características de creación del ALB:

- **Tipo de balanceador de carga:** Application Load Balancer
- **Nombre:** ALB-principal
- **Esquema:** Expuesto a Internet (direcciona las solicitudes de los clientes a través de Internet hasta los destinos)
- **VPC:** vpc-principal
- **Mapeos:**
 - o **Zona de disponibilidad:** us-east-1a
 - **Subred:** subred-servicios-1a
 - o **Zona de disponibilidad:** us-east-1b
 - **Subred:** subred-front-1b
- **Grupos de seguridad:** ALB-sg

- **Agentes de escucha y enrutamiento:**
 - Listener: HTTP:4200 – Acción: Grupo de destino Front
 - Listener: HTTP:8080 – Acción: Grupo de destino Plots
 - Listener: HTTP:8081 – Acción: Grupo de destino Sensor
 - Listener: HTTP:8082 – Acción: Grupo de destino Statistics
 - Listener: HTTP: 8999 – Acción: Grupo de destino Authentication

En la siguiente imagen (Ilustración 99) se puede ver la lista de “listeners” junto con sus correspondientes grupos de destino asignados:

Listeners (5)				
A listener checks for connection requests on its port and protocol. Traffic received by the listener is routed according to its rules.				
	Protocol:Port	Default action	Rules	
<input type="checkbox"/>	HTTP:8080	Forward to target group <ul style="list-style-type: none"> • plots: 1 (100%) • Group-level stickiness: Off 	1 rule	
<input type="checkbox"/>	HTTP:8081	Forward to target group <ul style="list-style-type: none"> • sensor: 1 (100%) • Group-level stickiness: Off 	1 rule	
<input type="checkbox"/>	HTTP:8082	Forward to target group <ul style="list-style-type: none"> • statistics: 1 (100%) • Group-level stickiness: Off 	1 rule	
<input type="checkbox"/>	HTTP:8999	Forward to target group <ul style="list-style-type: none"> • authentication: 1 (100%) • Group-level stickiness: Off 	1 rule	
<input type="checkbox"/>	HTTP:4200	Forward to target group <ul style="list-style-type: none"> • front: 1 (100%) • Group-level stickiness: Off 	1 rule	

Ilustración 99 - Listeners ALB AWS

6.3. Transformación IaC de la infraestructura a través de Terraform

En esta tercera y última fase del desarrollo se ha implementado la infraestructura desplegada en el apartado anterior mediante la herramienta de infraestructura como código “IaC” de Terraform. El proceso de levantar a mano cada uno de los servicios de AWS mediante su interfaz gráfica permite aprender las características de cada uno y su configuración. Es un proceso lento, pero con la ventaja de poder crear los servicios de una manera sencilla aprovechando la potente interfaz gráfica web de AWS, la cual provee de mucha información y enlaces directos al manual de usuario de AWS. Sin embargo, mediante la construcción de los ficheros Terraform, se han desplegado y configurado estos servicios para levantar toda la infraestructura de una sola vez con mayor rapidez. Incluso permite destruir todos los recursos al mismo tiempo, al contrario que usando la interfaz de AWS en donde hay que ir destruyendo cada uno de los servicios de forma individual.

Utilizar la interfaz de AWS puede ser muy efectivo si se cuenta con una infraestructura ya en funcionamiento para poder crear servicios de forma puntual. El uso de Terraform está más orientado al despliegue de infraestructuras completas o un conjunto de servicios al mismo tiempo.

En cuanto al proceso de construcción de los ficheros, se ha podido crear progresivamente la infraestructura y comprobar que todos los servicios desplegados funcionaban en conjunto, pudiendo solucionar problemas puntuales junto con la capacidad de ir agregando cada vez más servicios según se necesitase.

6.3.1. Estructura del proyecto de Terraform

En la siguiente imagen se puede ver la estructura del proyecto de Terraform resultante con todos los ficheros que lo forman (Ilustración 100):

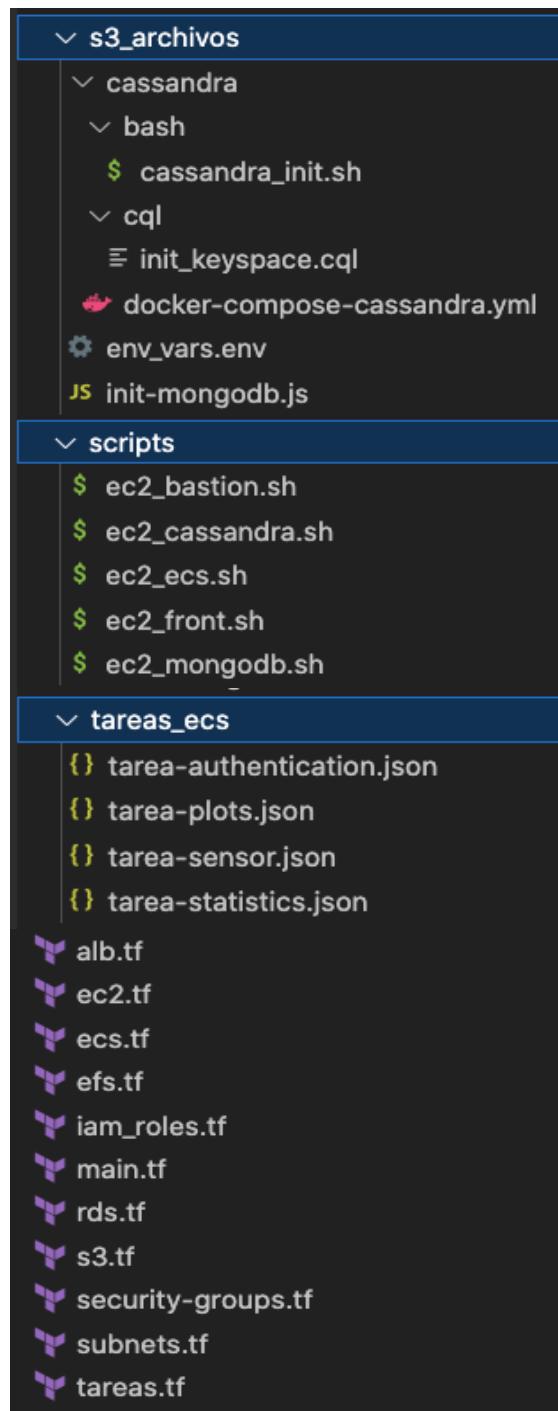


Ilustración 100 - Estructura Terraform

6.3.2. Detalles de implementación de los ficheros Terraform

La estructura de la infraestructura de Terraform que se desplegará será exactamente igual a la infraestructura desplegada en AWS anteriormente y utilizando los mismos scripts y ficheros.

Se presentan algunas diferencias mínimas entre ambas infraestructuras, como por ejemplo en los nombres de los servicios para poder identificarlos frente a los que están ya creados (tendrán un sufijo “-tf”). Con respecto a los scripts utilizados, algunos valores concretos serán variables de entorno a las que Terraform pasará el valor correspondiente cuando el servicio creado esté disponible a medida que se vaya construyendo la infraestructura. Aunque se explicará más adelante, el ejemplo más común es la utilización de una variable de entorno dentro de los scripts de datos de usuario, la cual obtendrá el valor de la dirección IP del sistema de ficheros EFS cuando Terraform levante este servicio puesto que no se puede conocer el valor de dicha IP hasta que esté creado.

Otro detalle que se podrá observar en este apartado es el uso de la dirección IP privada del sistema de fichero EFS en vez de su dirección DNS. Resulta que, al desplegar toda la infraestructura a la vez, la opción de resolución de nombres DNS en la configuración de la VPC tarda un tiempo en estar disponible. Como consecuencia, al realizar los montajes del EFS sobre las máquinas EC2 dentro de la carpeta “/efs” se reportaba un error al no poder detectar esta dirección.

- **Comprobación del despliegue con Terraform**

La comprobación del despliegue en funcionamiento de la infraestructura se ha realizado en el apartado [8. Pruebas](#) del documento, más concretamente en el [8.2.3. Despliegue con Terraform de los servicios AWS](#).

A continuación, se procede con la explicación del proyecto junto los recursos creados en los ficheros implementados y sus características más destacadas. Se recomienda leer las explicaciones al mismo tiempo que el código:

Carpeta “s3_archivos”

Archivos de configuración que se subirán al bucket S3 de la infraestructura.

Carpeta “scripts”

Fichero con los scripts de shell que se ejecutarán en los datos de usuario de las instancias EC2.

Carpeta “tareas_ecs”

Ficheros JSON con las configuraciones de ejecución que utilizarán las tareas ECS.

main.tf

Es el fichero principal se definirá la información del proveedor de cloud a utilizar en el proyecto junto con las credenciales de usuario para que Terraform pueda acceder a la cuenta de AWS. El conjunto de recursos o “resources” definirán la mayor parte de la estructura de red.

Bloques de configuración de Terraform:

- **Terraform**: bloque principal de terraform donde se indica el proveedor de cloud con el que se va a trabajar, en este caso “hashicorp/aws” y la versión del mismo que será la “4.0”.
- **provider "aws"**: bloque encargado de realizar la conexión con la cuenta de AWS en el que se especificará la región, clave de acceso y la clave secreta del usuario.

Recursos con los servicios AWS:

- **resource "aws_vpc" "vpc-tf"**: VPC de la infraestructura con la resolución de direcciones DNS activada.
- **resource "aws_internet_gateway" "igw-tf"**: Internet Gateway para permitir el acceso a Internet en la VPC.
- **resource "aws_eip" "ip-elastica-nat-tf"**: IP elástica que será asignada a la NAT Gateway.
- **resource "aws_nat_gateway" "nat-gateway-tf"**: NAT Gateway para permitir conexiones desde la VPC a Internet y rechazar las entrantes.
- **resource "aws_route_table" "tabla-rutas-internet-tf"**: tabla de enrutamiento utilizada por las subredes públicas que tendrá acceso a Internet a través del IGW.
- **resource "aws_route_table" "tabla-rutas-databases-tf"**: tabla de enrutamiento utilizada por las subredes privadas con ruta hacia Internet a través de la NAT Gateway.

- **resource "aws_vpc_endpoint" "s3-endpoint-tf"**: punto de enlace dentro de la VPC que comunicará los servicios definidos dentro de esta con los buckets S3.
- **resource "aws_route_table_association" "tabla-internet-subred-servicios"**: asociación de la tabla de enrutamiento pública con la subred pública de servicios que la utilizará.
- **resource "aws_route_table_association" "tabla-internet-subred-front"**: asociación de la tabla de enrutamiento pública con la subred pública del servicio Front de la aplicación que la utilizará.
- **resource "aws_route_table_association" "tabla-databases-subred-databases-1a"**: asociación de la tabla de enrutamiento privada con una de las subredes privadas que la utilizará.
- **resource "aws_route_table_association" "tabla-databases-subred-databases-1b"**: asociación de la tabla de enrutamiento a la segunda subred privada.
- **resource "aws_vpc_endpoint_route_table_association" "tabla-internet-s3-endpoint"**: asociación de la ruta que permitirá el tráfico hacia el “s3 endpoint” en la tabla de enrutamiento pública.
- **resource "aws_vpc_endpoint_route_table_association" "tabla-databases-s3-endpoint"**: asociación de la ruta que permitirá el tráfico hacia el “s3 endpoint” en la tabla de enrutamiento privada.

subnets.tf

En conjunto con el fichero “main.tf” desplegará toda la estructura de red de la infraestructura, más concretamente, se definirán los recursos de subredes.

- **resource "aws_subnet" "subred-servicios-1a-tf"**: subred pública destinada a desplegar las instancias EC2 Bastión y los microservicios de la aplicación: Plots, Sensor, Authentication y Statistics.
- **resource "aws_subnet" "subred-front-1b-tf"**: subred pública donde se desplegará la instancia EC2 con el servicio Front.
- **resource "aws_subnet" "subred-databases-1a-tf"**: subred privada donde estarán los servicios de bases de datos RDS de MySQL y PostgreSQL, junto con las instancias EC2 de Cassandra y MongoDB.

- **resource "aws_subnet" "subred-databases-1b-tf":** segunda red privada, que junto con la subred-databases-1a-tf formará un grupo de subredes necesaria para desplegar las RDS.

security-groups.tf

Definición de todos los grupos de seguridad que utilizarán el resto de servicios.

- **resource "aws_security_group" "servicios-sg-tf":** grupo de seguridad que utilizarán las instancias EC2 pertenecientes al clúster ECS y que permitirá el tráfico hacia los microservicios de la aplicación.
- **resource "aws_security_group" "bastion-sg-tf":** grupo de seguridad para permitir la conexión a la instancia EC2 Bastión.
- **resource "aws_security_group" "front-sg-tf":** grupo de seguridad para permitir el tráfico hacia la instancia EC2 con el servicio Front.
- **resource "aws_security_group" "databases-sg-tf":** grupo de seguridad para el tráfico hacia todos los servicios de bases de datos desplegados en los servicios RDS y EC2.
- **resource "aws_security_group" "efs-sg-tf":** grupo de seguridad que permite el acceso al sistema de ficheros EFS.
- **resource "aws_security_group" "alb-sg-tf":** grupo de seguridad que habilita el tráfico a través del balanceador de carga ALB hacia los microservicios y el servicio Front.

iam_roles.tf

Definición de los roles, políticas y perfiles IAM que utilizarán las instancias EC2 para obtener los permisos necesarios de al interactuar con los servicios S3 y ECS. Primero se crean los roles, después se les asignan políticas y finalmente se crean los perfiles IAM con estos roles para poder asignarse a las instancias EC2.

- **resource "aws_iam_role" "AmazonS3FullAccess-tf":** rol de tipo EC2 donde las futuras políticas asignadas al rol tendrán efecto sobre las instancias EC2.

- **Resource "aws_iam_role_policy_attachment" "asignacion-politica- AmazonS3FullAccess":** asigna al rol creado "AmazonS3FullAccess-tf" una política con permisos de acceso al servicio S3 mediante su identificador "arn" de AWS (arn:aws:iam::aws:policy/AmazonS3FullAccess).
- **resource "aws_iam_instance_profile" "perfil-AmazonS3FullAccess-tf":** perfil de instancia IAM con el rol creado que se asignará a las instancias EC2 durante su creación para poder acceder a los buckets del servicio S3.
- **resource "aws_iam_role" "ecsInstanceRole-tf":** rol de tipo EC2 donde las futuras políticas asignadas al rol tendrán efecto sobre las instancias EC2.
- **resource "aws_iam_role_policy_attachment" "asignacion-politica-ecsInstanceRole":** asigna al rol creado "ecsInstanceRole-tf" una política con los permisos para realizar operaciones ECS mediante su identificador "arn" de AWS (arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role).
- **resource "aws_iam_instance_profile" "perfil-ecsInstanceRole-tf":** perfil de instancia IAM con el rol creado que se asignará solo a las instancias EC2 que pertenecerán al clúster ECS.

s3.tf

Creación del bucket S3 y subida de todos los ficheros de configuración que utilizarán las instancias. Los ficheros que se subirán al S3 se encuentran en la carpeta "s3_archivos" del proyecto.

- **resource "aws_s3_bucket" "db-services-bucket-tf":** bucket S3 que almacenarán los ficheros de configuración de la infraestructura.
- **resource "aws_s3_object" "init-mongodb_fichero":** script inicial que utilizará la base de datos MongoDB.
- **resource "aws_s3_object" "env_vars_fichero":** fichero con las variables de entorno que utilizarán las instancias EC2.

Este recurso utilizará la función “templatefile”, a la que se le pasará la ruta del fichero “env_vars.env” destino que será modificado mediante las variables definidas con los valores del endpoint ALB, las direcciones host de las RDS y las IP privadas de EC2 de Cassandra y MongoDB. De esta forma el fichero tendrá los valores de las variables de entorno actualizados antes de subirse al bucket:

```
content = templatefile("s3_archivos/env_vars.env", {
    front_endpoint      = aws_lb.ALB-tf.dns_name
    postgres_host       = aws_db_instance.rds-sensor-postgres-tf.address
    mysql_host          = aws_db_instance.rds-authentication-mysql-tf.address
    mongo_host          = aws_instance.ec2-plots-mongo-tf.private_ip
    cassandra_host      = aws_instance.ec2-statistics-cassandra-tf.private_ip
})
```

Estos valores serán recogidos durante la construcción de la infraestructura a medida que los recursos se crean para ser enviados al fichero destino. Dentro del fichero “env_vars.env” se recogen los valores haciendo referencia a las variables como si fueran variables de entorno:

```
# Endpoint Front-End
FRONT_ENDPOINT=${front_endpoint}
# Servicio cliente dentro de Plots
SENSOR_CLIENT=${front_endpoint}
# Endpoints Bases de Datos
POSTGRES_HOST=${postgres_host}
MYSQL_HOST=${mysql_host}
MONGODB_HOST=${mongo_host}
CASSANDRA_HOST=${cassandra_host}
```

- **resource "aws_s3_object" "cassandra_carpeta"**: creación de la carpeta principal “cassandra/” dentro del bucket que contendrá los ficheros de configuración para la base de datos Cassandra. Es necesario crear a mano el directorio puesto que no se puede pasar directamente al recurso la carpeta completa junto con todos sus ficheros.
- **resource "aws_s3_object" "docker-compose-cassandra_fichero"**: subida del fichero “docker-compose-cassandra.yml” dentro de la carpeta “cassandra/”.

Utilizará la función “templatefile” para pasar al fichero “docker-compose-cassandra.yml” el valor de la IP privada del EFS a través de la variable definida:

```
content = templatefile("s3_archivos/cassandra/docker-compose-
cassandra.yml", {
    efs_ip = aws_efs_mount_target.efs-mt.ip_address
})
```

Dentro del fichero docker-compose se recibirá el valor en la variable \${efs_ip}:

```
volumes:  
  cassandra_volume:  
    driver_opts:  
      type: "nfs"  
      o:  
        "addr=${efs_ip},nfsvers=4.1,rsize=1048576,wszie=1048576,hard,timeo  
        =600,retrans=2,noresvport"  
      device:(":/cassandra-efs"
```

- **resource "aws_s3_object" "bash_carpeta"**: creación de la carpeta “bash/” dentro de la carpeta principal “cassandra/”.
- **resource "aws_s3_object" "cassandra_init_fichero"**: subida del script de configuración “cassandra_init.sh” dentro de la carpeta “cassandra/bash”.
- **resource "aws_s3_object" "cql_carpeta"**: creación de la carpeta “cql/” dentro de la carpeta principal “cassandra/”.
- **resource "aws_s3_object" "init_keyspace_fichero"**: subida del script “init_keyspace.cql” dentro de la carpeta “cassandra/cql”.

[*efs.tf*](#)

Sistema de ficheros que utilizarán las instancias EC2 para la persistencia de los datos de los contenedores y creación del destino de montaje que permitirá el acceso al mismo.

- **resource "aws_efs_file_system" "efs-tf"**: sistema de ficheros EFS de la infraestructura.
- **resource "aws_efs_mount_target" "efs-mt"**: destino de montaje del sistema de ficheros EFS. Se creará en la subred privada “subred-databases-1a-tf”.

[*ec2.tf*](#)

Conjunto de instancias EC2 encargadas de desplegar el servicio Front, las bases de datos de Cassandra y MongoDB junto con la instancia Bastión. Todas las EC2 podrán acceder al bucket S3 gracias al perfil IAM con permisos S3 creados anteriormente. Utilizarán en sus datos de usuario los scripts definidos en la carpeta “scripts” del proyecto.

- **resource "aws_instance" "ec2-bastion-tf"**: instancia EC2 creada en la subred pública “subred-servicios-1a-tf” destinada a la creación de carpetas dentro del EFS. Este recurso dependerá de la creación del recurso de destino de montaje puesto que utiliza la IP privada del EFS.

En los datos de usuario se utilizará la función “templatefile”, que pasará al script “**ec2_bastion.sh**” el valor de la IP privada del EFS a través de la variable definida.

```
user_data = templatefile("scripts/ec2_bastion.sh", {
    efs_ip= aws_efs_mount_target.efs-mt.ip_address
})
```

Dentro del script se recibirá el valor en la variable \${efs_ip}.

```
<resto del código>

# Montaje EFS en el path indicado /efs
mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,n
oresvport ${efs_ip}: / efs
# Montaje del EFS automatico al reiniciar la instancia
echo "${efs_ip}: / efs nfs4
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 0
0" >> /etc/fstab

<resto del código>
```

- **resource "aws_instance" "ec2-front-tf"**: instancia EC2 creada en la subred pública “subred-front-1b-tf” donde se desplegará el servicio Front. Podrá utilizar el script “**ec2_front.sh**” en los datos de usuario gracias a la función “file”:

```
user_data = file("scripts/ec2_front.sh")
```

Este recurso dependerá de la creación del recurso “**env_vars_fichero**” porque necesitará que el fichero esté disponible en el bucket S3 para utilizarlo en el script:

```
<resto del código>

aws s3 cp s3://db-services-bucket-tf/env_vars.env env_vars.env

<resto del código>

docker pull jesusbc/tfm-front-end:latest
# Ejecutar el contenedor
docker run --name tfm-front-end -d -p 4200:4200 --env-file
env_vars.env jesusbc/tfm-front-end:latest
```

- **resource "aws_instance" "ec2-plots-mongo-tf"**: instancia EC2 creada en la subred privada “subred-databases-1a-tf” para el despliegue del contenedor con el servicio MongoDB. Este recurso dependerá de la creación de los recursos de destino de montaje puesto, la instancia EC2 Bastión y el bucket S3.

En los datos de usuario se utilizará la función “templatefile”, que pasará al script “**ec2_mongodb.sh**” el valor de la IP privada del EFS a través de la variable definida:

```
user_data = templatefile("scripts/ec2_mongodb.sh", {
    efs_ip= aws_efs_mount_target.efs-mt.ip_address
})
```

Dentro del script se recibirá el valor en la variable \${efs_ip}:

```
<resto del código>

# Montaje EFS en el path indicado /efs
mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wszie=1048576,hard,timeo=600,retrans=2,no
resvport ${efs_ip}: / efs
# Montaje del EFS automatico al reiniciar la instancia
echo "${efs_ip}:/ /efs nfs4
nfsvers=4.1,rszie=1048576,wszie=1048576,hard,timeo=600,retrans=2 0
0" >> /etc/fstab

<resto del código>

# Volumen mongodb-volume creado
docker volume create \
    --driver local \
    --opt type=nfs \
    --opt
o=addr=${efs_ip},nfsvers=4.1,rszie=1048576,wszie=1048576,hard,timeo
=600,retrans=2,noresvport \
    --opt device=/mongo-efs \
    mongodb-volume
```

- **resource "aws_instance" "ec2-statistics-cassandra-tf"**: instancia EC2 creada en la subred privada “subred-databases-1a-tf” para el despliegue del contenedor con el servicio Cassandra. Este recurso dependerá de la creación de los recursos de destino de montaje puesto, la instancia EC2 Bastión y el bucket S3.

Podrá utilizar el script “**ec2_cassandra.sh**” en los datos de usuario gracias a la función “file”:

```
user_data = templatefile("scripts/ec2_cassandra.sh", {
    efs_ip= aws_efs_mount_target.efs-mt.ip_address
})
```

rds.tf

Creación de las bases de datos de MySQL y PostgreSQL y el grupo de subredes que utilizan.

- **resource "aws_db_subnet_group" "databases-grupo-subnet-tf":** grupo de subredes de la que forman parte la subred “subred-databases-1a-tf” y “subred-databases-1b-tf”.
- **resource "aws_db_instance" "rds-sensor-postgres-tf":** servicio de base de datos relacional RDS de tipo PostgreSQL.
- **resource "aws_db_instance" "rds-authentication-mysql-tf":** servicio de bases de datos relacional RDS de tipo MySQL.

ecs.tf

Definición de un clúster ECS, inicialmente vacío, y las dos instancias EC2 que pertenecerán a este clúster creadas a partir de la AMI de AWS optimizada para el uso de ECS con Docker y el agente ECS ya instalado. Las instancias serán configuradas para que ejecuten el agente ECS al iniciarse mediante los datos de usuario, pasándole el nombre del clúster creado para que se identifiquen como instancias de contenedor de ese clúster. Las instancias obtendrán permisos para realizar operaciones ECS gracias al perfil IAM creado. Para los datos de usuario se utilizarán los scripts de configuración definidos en la carpeta “scripts” del proyecto.

- **resource "aws_ecs_cluster" "cluster-app-tf":** clúster ECS de la infraestructura.
- **resource "aws_instance" "ec2-ecs-tf":** creación de las dos instancias EC2 del clúster en la subred pública “subred-servicios-1a-tf”. Gracias a la opción “count = 2” se podrán crear dos instancias idénticas dentro del mismo recurso.

En los datos de usuario se utiliza la función “templatefile” para pasar al script “**ec2_eks.sh**” el nombre del clúster para que pueda ser identificado.

```
user_data = templatefile("scripts/ec2_eks.sh", {
    cluster_name = aws_efs_mount_target.efs-mt.ip_address
})
```

Dentro del script se recibirá el nombre del clúster en la variable \${cluster_name}:

```
<resto del código>

echo ECS_CLUSTER=${cluster_name} >> /etc/ecs/ecs.config
cat /etc/ecs/ecs.config | grep "ECS_CLUSTER"
```

tareas.tf

Recursos de definición de tareas con la configuración de cada uno de los contenedores que se lanzarán sobre las instancias del clúster. Las tareas utilizarán los ficheros JSON con las configuraciones de lanzamiento de contenedores definidos en la carpeta “tareas_ecs”.

Durante la creación de los recursos se establecerá el rol de AWS que permite realizar operaciones ECS, el cual se identifica con su “arn” de AWS (arn:aws:iam::690031176274:role/ecsTaskExecutionRole).

- **resource "aws_ecs_task_definition" "tarea-plots-tf":** definición de tarea del servicio Plots.
- **resource "aws_ecs_task_definition" "tarea-sensor-tf":** definición de tarea del servicio Sensor.
- **resource "aws_ecs_task_definition" "tarea-statistics-tf":** definición de tarea del servicio Statistics.
- **resource "aws_ecs_task_definition" "tarea-authentication-tf":** definición de tarea del servicio Authentication.

alb.tf

Creación del balanceador de carga expuesto a Internet de tipo ALB, sus “listeners” y los grupos de destino que redirigirán el tráfico hacia las instancias EC2 al recibir las peticiones.

- **resource "aws_lb" "ALB-tf":** balanceador de carga de tipo aplicación o ALB. Tendrá efecto sobre las subredes públicas “subred-servicios-1a-tf” y “subred-front-1b-tf” donde se encuentran desplegados los servicios de la aplicación.
- **resource "aws_lb_listener" "front-listener-tf":** listener en el puerto 4200 que escuchará las peticiones hacia el Front.

- **resource "aws_lb_listener" "plots-listener-tf"**: listener en el puerto 8080 que escuchará las peticiones hacia el servicio Plots.
- **resource "aws_lb_listener" "sensor-listener-tf"**: listener en el puerto 4200 que escuchará las peticiones hacia el servicio Sensor.
- **resource "aws_lb_listener" "statistics-listener-tf"**: listener en el puerto 4200 que escuchará las peticiones hacia el servicio Statistics.
- **resource "aws_lb_listener" "authentication-listener-tf"**: listener en el puerto 4200 que escuchará las peticiones hacia el servicio Authentication.
- **resource "aws_lb_target_group" "front-target-tf"**: grupo de destino Front que redirigirá el tráfico hacia las instancias en el puerto 4200.
- **resource "aws_lb_target_group_attachment" "asignacion-destinos-front"**: asignación de la instancia EC2 Front creada como destino al grupo de destino Front.
- **resource "aws_lb_target_group" "plots-target-tf"**: grupo de destino Plots que redirigirá el tráfico hacia las instancias en el puerto 8080.
- **resource "aws_alb_target_group_attachment" "asignacion-destinos-plots"**: asignación de las instancias EC2 pertenecientes al clúster ECS como destinos al grupo de destino Plots.
- **resource "aws_lb_target_group" "sensor-target-tf"**: grupo de destino Sensor que redirigirá el tráfico hacia las instancias en el puerto 8081.
- **resource "aws_alb_target_group_attachment" "asignacion-destinos-sensor"**: asignación de las instancias EC2 pertenecientes al clúster ECS como destinos al grupo de destino Sensor.
- **resource "aws_lb_target_group" "statistics-target-tf"**: grupo de destino Statistics que redirigirá el tráfico hacia las instancias en el puerto 8082.
- **resource "aws_alb_target_group_attachment" "asignacion-destinos-statistics"**: asignación de las instancias EC2 pertenecientes al clúster ECS como destinos al grupo de destino Statistics.

- **resource "aws_lb_target_group" "authentication-target-tf"**: grupo de destino Authentication que redirigirá el tráfico hacia las instancias en el puerto 8999.
- **resource "aws_alb_target_group_attachment" "asignacion-destinos-authentication"**: asignación de las instancias EC2 pertenecientes al clúster ECS como destinos al grupo de destino Authentication.

7. Procedimiento para la ejecución de la aplicación local y de la infraestructura de Terraform

En este apartado se resumirá el proceso a seguir para poner en marcha tanto la aplicación de contenedores en local como los pasos para desplegar la infraestructura a través de Terraform:

7.1. Aplicación local

El primer paso será la empaquetación de los servicios de Spring Boot para generar sus correspondientes archivos ejecutables “.jar”. El servicio Front no necesita empaquetado.

El siguiente comando deberá ejecutarse en la raíz del proyecto de cada servicio:

```
./gradlew bootJar
```

Después se procede con la construcción de las imágenes a través del comando “docker build”. También se realizarán los siguientes comandos cada uno en su servicio correspondiente:

```
docker build -t jesusbc/sensor:latest .
docker build -t jesusbc/plots:latest .
docker build -t jesusbc/tfm-front-end:latest .
docker build -t jesusbc/statistics:latest .
docker build -t jesusbc/authentication:latest .
docker build -t jesusbc/cassandra:latest .
```

Otra opción es importar las imágenes utilizadas durante el desarrollo disponibles en el repositorio de Docker-Hub para ser importadas mediante el comando “docker pull”. Habrá que estar logueado en Docker para poder ejecutar el comando.

```
docker pull jesusbc/sensor:latest
docker pull jesusbc/plots:latest
docker pull jesusbc/tfm-front-end:latest
docker pull jesusbc/statistics:latest
docker pull jesusbc/authentication:latest
docker pull jesusbc/cassandra:latest
```

Una vez construidas las imágenes, se lanzarán todos los contenedores mediante el fichero Docker-Compose:

```
docker-compose -f docker-compose-base.yml up
```

El Front de la aplicación se desplegará en la dirección: **localhost:4200**

7.2. Infraestructura de Terraform

Antes de lanzar la infraestructura en Terraform, será necesario crear unas credenciales de AWS con los permisos necesarios de administrador. Para ello, se recomienda consultar el apartado [12.2.1. Creación de un usuario con credenciales AWS para Terraform](#) del documento. Una vez creado el usuario con credenciales AWS, estas se deberán especificar dentro del archivo "main.tf" del proyecto de Terraform, concretamente en su recurso "provisioner", proporcionando el "access_key" y "secret_key".

```
# AWS Provider
provider "aws" {
    region      = "us-east-1"
    access_key  = "AKIA2BKIQFJJIIU72X6P"
    secret_key  = "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"
}
```

Para crear una cuenta de AWS consultar el apartado [12.1. Creación de la cuenta de AWS](#) del documento.

Una vez que se han introducido las credenciales en el "provisioner", se procede con el despliegue de la infraestructura a través de los siguientes comandos:

1. **Terraform init**: inicializa Terraform para comenzar con el despliegue (Ilustración 101).

```
● jesusbarquerocuadrado@MacBook-Pro-de-Jesus Terraform AWS % terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.50.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Ilustración 101 - Terraform init

2. **Terraform validate:** comprueba el código para detectar errores de sintaxis, relaciones entre recursos inexistentes... Si no hay errores el resultado de la validación será exitoso (Ilustración 102).

```
● jesusbarquerocuadrado@MacBook-Pro-de-Jesus Terraform AWS % terraform validate
Success! The configuration is valid.

○ jesusbarquerocuadrado@MacBook-Pro-de-Jesus Terraform AWS %
```

Ilustración 102 - Terraform validate

3. **Terraform plan:** lee todos los ficheros Terraform (.tf) e indica todos los recursos que se van a crear o modificar. También validará el código e informa de errores (Ilustración 103).

```
+ resource "aws_vpc_endpoint_route_table_association" "tabla-databases-s3-endpoint" {
    + id          = (known after apply)
    + route_table_id = (known after apply)
    + vpc_endpoint_id = (known after apply)
}

# aws_vpc_endpoint_route_table_association.tabla-internet-s3-endpoint will be created
+ resource "aws_vpc_endpoint_route_table_association" "tabla-internet-s3-endpoint" {
    + id          = (known after apply)
    + route_table_id = (known after apply)
    + vpc_endpoint_id = (known after apply)
}

Plan: 74 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly
these actions if you run "terraform apply" now.
○ jesusbarquerocuadrado@MacBook-Pro-de-Jesus Terraform AWS %
```

Ilustración 103 - Terraform plan

4. **Terraform apply:** confirmará el plan de recursos que se creará a partir el “terraform plan”. Pedirá una confirmación previa antes de proceder a crearlos. Si ocurre algún error, se notificará e interrumpirá la ejecución del comando (Ilustración 104).

```
aws_db_instance.rds-authentication-mysql-tf: Still creating... [4m0s elapsed]
aws_db_instance.rds-authentication-mysql-tf: Creation complete after 4m6s [id=rds-authentication-mysql-tf]
aws_s3_object.env_vars_fichero: Creating...
aws_s3_object.env_vars_fichero: Creation complete after 1s [id=env_vars.env]
aws_instance.ec2-front-tf: Creating...
aws_instance.ec2-front-tf: Still creating... [10s elapsed]
aws_instance.ec2-front-tf: Still creating... [20s elapsed]
aws_instance.ec2-front-tf: Still creating... [30s elapsed]
aws_instance.ec2-front-tf: Creation complete after 34s [id=i-0d172183ef6c75d4b]
aws_lb_target_group_attachment.asignacion-destinos-front: Creating...
aws_lb_target_group_attachment.asignacion-destinos-front: Creation complete after 1s [id=arn:aws:elast
icloadbalancing:us-east-1:690031176274:targetgroup/front-target-tf/50759a4e9bd84922-202306121601177395
00000013]

Apply complete! Resources: 74 added, 0 changed, 0 destroyed.
○ jesusbarquerocuadrado@MacBook-Pro-de-Jesus Terraform AWS %
```

Ilustración 104 - Terraform apply

5. **Terraform destroy:** destruye todos los recursos creados. Puede utilizarse en el caso de que se quiera destruir toda la infraestructura para evitar el consumo de recursos de cara a levantarla más adelante o eliminar recursos creados a medias cuando se produce un error para solucionarlo y volver a lanzar todo de nuevo entre otros usos (Ilustración 105)

```
aws_db_subnet_group.databases-grupo-subnet-tf: Destroying... [id=databases-grupo-subnet-tf]
aws_security_group.databases-sg-tf: Destroying... [id=sg-0c477e771e92dcb2b]
aws_db_subnet_group.databases-grupo-subnet-tf: Destruction complete after 1s
aws_subnet.subred-databases-la-tf: Destroying... [id=subnet-0621d0458d4b48d0f]
aws_subnet.subred-databases-lb-tf: Destroying... [id=subnet-030c492f2d77888ef]
aws_subnet.subred-databases-la-tf: Destruction complete after 1s
aws_subnet.subred-databases-lb-tf: Destruction complete after 1s
aws_security_group.databases-sg-tf: Destruction complete after 2s
aws_security_group.bastion-sg-tf: Destroying... [id=sg-0558130baae3ea2da]
aws_security_group.bastion-sg-tf: Destruction complete after 1s
aws_vpc.vpc-tf: Destroying... [id=vpc-0ff8672d0ccc3cc11]
aws_vpc.vpc-tf: Destruction complete after 1s

Destroy complete! Resources: 74 destroyed.
```

Ilustración 105 - Terraform destroy

8. Pruebas

En este apartado se comprobará el funcionamiento de la aplicación una vez puesta en ejecución. Se aportará la información y las capturas necesarias para determinar que todos los microservicios funcionan correctamente y se conectan a sus correspondientes bases de datos. También se probarán las operaciones API REST para ver cuáles de ellas se pueden utilizar mediante la herramienta Postman (Postman, 2019).

8.1. Funcionalidad auxiliar de comprobación de estado de las API REST

De forma complementaria al desarrollo y refactorización de la aplicación, se ha implementado un método dentro de los microservicios que permite realizar una comprobación del estado de las API REST (Plots, Sensor, Authentication y Statisticis) a través de una herramienta proporcionada por Spring Boot llamada Actuator (Spring Boot, 2023). Esta funcionalidad se ha decidido explicar en esta sección de Pruebas ya que es un procedimiento que tiene que ver con la comprobación del funcionamiento de la aplicación. No obstante, como es también en cierta medida una refactorización del proyecto en cuanto a “añadir código nuevo” se refiere, **la instalación de esta herramienta Actuator se realizaría antes de empaquetar el código y transformarlo en imágenes.**

Las comprobaciones de estado es una práctica muy extendida en el desarrollo de aplicaciones puesto que es una manera de comprobar si sus servicios se encuentran o no en funcionamiento. Generalmente se implementan a través de peticiones HTTP de tipo GET hacia un endpoint de una API. La API responderá con un simple estado u otro tipo de información que determinará si el servicio se encuentra disponible o las razones por las que no lo está (Hombergs, 2020).

Esta funcionalidad se ha implementado de cara a su utilización en la parte de despliegue de la aplicación en AWS. Ha servido para establecer una URL dentro de las tareas del ALB “Application Load Balancer” para poder realizar estas comprobaciones de salud en las instancias EC2 destino. Entre los destinos, el ALB solo redirigirá el tráfico hacia los que estén saludables, por lo que son unas operaciones importantes.

La instalación de Actuator es muy sencilla, solo basta con añadir la siguiente dependencia dentro del fichero de Gradle “build.gradle”:

```
implementation("org.springframework.boot:spring-boot-starter-actuator")
```

Esto creará un endpoint dentro del servicio en la ruta <http://<host>:<puerto>/actuator/health> que devolverá el estado del servicio con un valor “UP” si se encuentra disponible o “DOWN” si no lo está.

8.2. Comprobación del despliegue

Se realizarán tres comprobaciones: la primera a nivel de despliegue de los contenedores en el equipo local, la segunda para comprobar el funcionamiento de los contenedores y servicios dentro de la infraestructura de AWS y la tercera para comprobar que se crean todos los servicios definidos en los ficheros Terraform.

8.2.1. Despliegue local de contenedores

Para comprobar que todos los contenedores se encuentran en ejecución en sus puertos correspondientes se utiliza el comando “**docker ps**” para determinar su estado (Ilustración 106):

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d88fa21858da	jesusbc/sensor:latest	"java -jar app.jar"	14 minutes ago	Up 14 minutes	0.0.0.0:8081->8081/tcp	sensor
1585a7119d99	jesusbc/plots:latest	"java -jar app.jar"	14 minutes ago	Up 14 minutes	0.0.0.0:8080->8080/tcp	plots
cc2f91e8c607	jesusbc/authentication:latest	"java -jar app.jar"	14 minutes ago	Up 14 minutes	0.0.0.0:8999->8999/tcp	authentication
64d74e8f78fa	jesusbc/statistics:latest	"java -jar app.jar"	14 minutes ago	Up 10 minutes	0.0.0.0:8082->8082/tcp	statistics
4965f97a1db4	jesusbc/tfm-front-end:latest	"/docker-env-js-to-s..."	14 minutes ago	Up 14 minutes	0.0.0.0:4200->4200/tcp	tfm-front-end
f13203e4b3dd	jesusbc/cassandra:latest	"/cassandra_init-ent..."	14 minutes ago	Up 14 minutes	7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp	cassandra
7a369bbcdcca	mongo:5.0.7	"docker-entrypoint.s..."	12 days ago	Up 14 minutes	0.0.0.0:27017->27017/tcp	mongodb
c7aa478ad123	mysql:5.7	"docker-entrypoint.s..."	2 months ago	Up 14 minutes	0.0.0.0:3306->3306/tcp, 33060/tcp	mysql
e5eb54a1e82b	postgres	"docker-entrypoint.s..."	2 months ago	Up 14 minutes	0.0.0.0:5432->5432/tcp	postgres

Ilustración 106 - Comprobación de contenedores en ejecución

Con el comando “**docker volume ls**” se comprueba la lista de volúmenes desplegados (Ilustración 107):

DRIVER	VOLUME NAME
local	3cdf818c0efec971e3b7abe112b6aadeaa7f33eee17844c65d38a
local	72439bf330e26bcfd6d72daae82d7a32ea5b0d61243170edee70
local	f33ed6f7391e5c69655a75980acac75e2b3ac6e44f4f688975dc6
local	tfm_cassandra_data
local	tfm_mongo_data
local	tfm_mysql_data
local	tfm_postgres_data

Ilustración 107 - Comprobación de volúmenes

A continuación, se mostrará la información de los “logs” de cada contenedor de la aplicación y se consultarán las bases de datos para comprobar que se han inicializado correctamente con las estructuras especificadas.

Para visualizar los logs en Docker, se utiliza el comando “docker logs” acompañado del id o nombre del contenedor:

```
docker logs <id_nombre_contenedor>
```

Para iniciar una conexión a través de la consola con bash en el contenedor, se utiliza el comando “docker exec -it” acompañado del id o nombre del contenedor seguido de la dirección /bin/bash:

```
docker exec -it <id_nombre_contenedor> /bin/bash
```

Servicio Front

Se ejecuta sin ningún error como puede observarse en los logs (Ilustración 108) y despliega el servicio en la dirección del navegador “localhost:4200” (Ilustración 109).

```
MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker logs 4965f97a1db4
localhost
> tfm-front-end@0.0.0 start
> ng serve --host 0.0.0.0 --disable-host-check

Warning: This is a simple server for use in testing or debugging Angular applications
locally. It hasn't been reviewed for security issues.

Binding this server to an open connection can result in compromising your application or
computer. Using a different host than the one passed to the "--host" flag might result in
websocket connection issues. You might need to use "--disableHostCheck" if that's the
case.
Warning: Running a server with --disable-host-check is a security risk. See https://medium.com/webpack/webpack-dev-server-middleware-security-issues
-1489d950874a for more information.
Warning: Entry point 'angular-highcharts' contains deep imports into '/app/node_modules/highcharts/highmaps', '/app/node_modules/highcharts/highstock'.
This is probably not a problem, but may cause the compilation of entry points to be out of order.
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
Compiling @angular/common/http : es2015 as esm2015
Compiling @angular/router : es2015 as esm2015
Compiling @angular/forms : es2015 as esm2015
Compiling @ng-bootstrap/ng-bootstrap : es2015 as esm2015
Compiling @asymmetrik/ngx-leaflet : module as esm5
Compiling angular-highcharts : es2015 as esm2015
Compiling angular-feather : es2015 as esm2015
Compiling angular-feather/icons : es2015 as esm2015
- Generating browser application bundles...
(node:24) [DEP0148] DeprecationWarning: Use of deprecated folder mapping "./" in the "exports" field module resolution of the package at /app/node_modules/postcss/package.json.
Update this package.json to use a subpath pattern like "./".
(Use 'node --trace-deprecation ...' to show where the warning was created)
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Size
vendor.js           | vendor        | 5.67 MB
styles.css          | styles        | 228.10 KB
scripts.js          | scripts       | 170.16 KB
main.js             | main          | 149.20 KB
polyfills.js        | polyfills     | 137.16 KB
runtime.js          | runtime       | 6.15 KB

Warning: /app/node_modules/angular-highcharts/__ivy_ngcc__/fesm2015/angular-highcharts.js depends on 'highcharts/highstock'. CommonJS or AMD dependencies can cause optimization bailouts.
For more info see: https://angular.io/guide/build#configuring-commonjs-dependencies

** Angular Live Development Server is listening on 0.0.0.0:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

Ilustración 108- Logs Front local

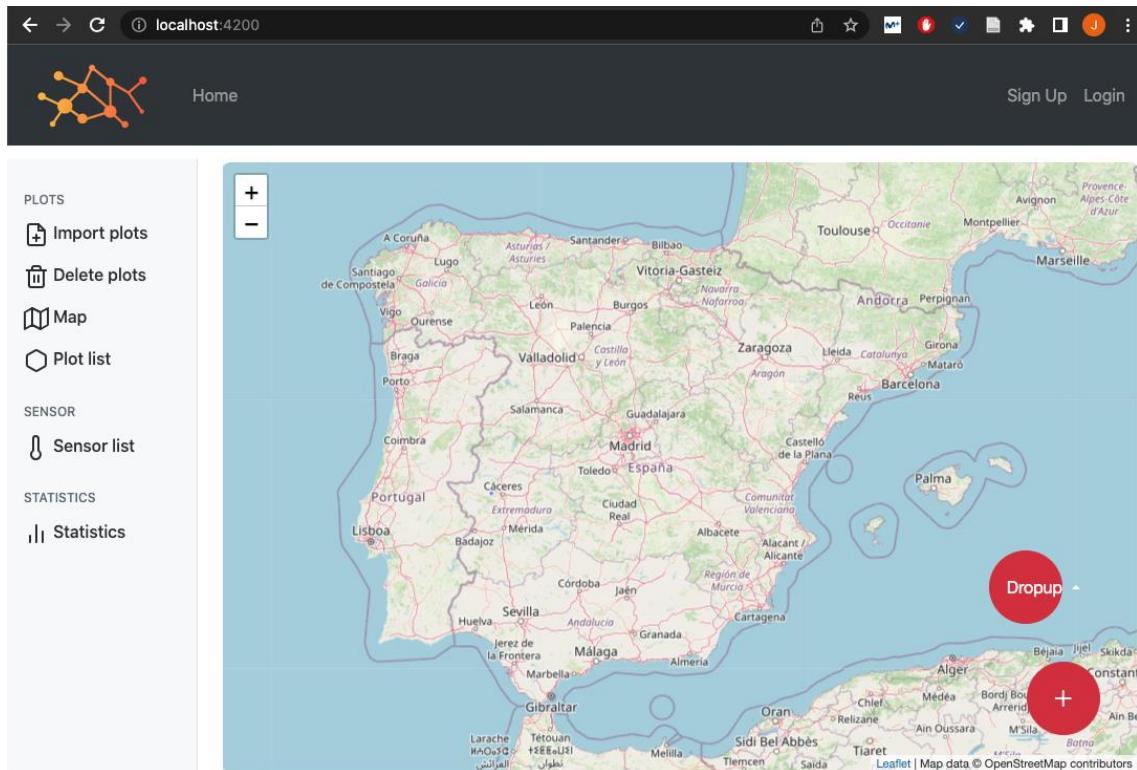


Ilustración 109 - Front en navegador

Servicio Plots

Se ejecuta sin errores en el puerto 8080 y se conecta al servicio contenedor mongo (Ilustración 110).

```

:: Spring Boot ::          (v2.6.2)

2023-06-12 01:00:29.184  INFO 1 --- [      main] c.t.m.plots.PlotsApplication        : Starting PlotsApplication using Java 11.0.19 on 1585a7119d99 with PID 1 (/app/p/app.jar started by root in /app)
2023-06-12 01:00:29.247  INFO 1 --- [      main] c.t.m.plots.PlotsApplication        : No active profile set, falling back to default profiles: default
2023-06-12 01:00:50.555  INFO 1 --- [      main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data MongoDB repositories in DEFAULT mode.
2023-06-12 01:00:53.711  INFO 1 --- [      main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 3107 ms. Found 1 MongoDB repository interfaces.
2023-06-12 01:00:59.681  INFO 1 --- [      main] o.s.cloud.context.scope.GenericScope   : BeanFactory id=40af2e8b-b893-3202-8150-c65854767457
2023-06-12 01:01:08.648  INFO 1 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8080 (http)
2023-06-12 01:01:08.780  INFO 1 --- [      main] o.apache.catalina.core.StandardService: Starting service [Tomcat]
2023-06-12 01:01:08.786  INFO 1 --- [      main] org.apache.catalina.core.StandardEngine: Starting Servlet engine: [Apache Tomcat/9.0.56]
2023-06-12 01:01:12.247  INFO 1 --- [      main] o.a.c.c.C.[Tomcat].[localhost].[]/: Initializing Spring embedded WebApplicationContext
2023-06-12 01:01:12.262  INFO 1 --- [      main] w.s.c.ServletWebServerApplicationContext: Root WebApplicationContext: initialization completed in 41570 ms
2023-06-12 01:01:23.682  INFO 1 --- [      main] org.mongodb.driver.cluster           : Cluster created with settings {hosts=[{host:27017}], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2023-06-12 01:01:29.444  INFO 1 --- [1'-->mongo:27017] org.mongodb.driver.connection: Opened connection [connectionId{localValue:1, serverValue:2}] to mongo:27017
2023-06-12 01:01:29.554  INFO 1 --- [1'-->mongo:27017] org.mongodb.driver.cluster: Monitor thread successfully connected to server with description ServerDescription{address:mongo:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=4157983247}
2023-06-12 01:01:29.553  INFO 1 --- [1'-->mongo:27017] org.mongodb.driver.connection: Opened connection [connectionId{localValue:2, serverValue:1}] to mongo:27017
2023-06-12 01:02:07.268  INFO 1 --- [      main] o.s.b.a.e.web.EndpointLinksResolver: Exposing 1 endpoint(s) beneath base path '/actuator'
2023-06-12 01:02:07.908  INFO 1 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http) with context path ''
2023-06-12 01:02:08.152  INFO 1 --- [      main] c.t.m.plots.PlotsApplication        : Started PlotsApplication in 117.211 seconds (JVM running for 132.8)
2023-06-12 02:41:44.629  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[]/: Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-06-12 02:41:44.634  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Initializing Servlet 'dispatcherServlet'
2023-06-12 02:41:44.699  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Completed initialization in 64 ms
2023-06-12 02:41:48.199  INFO 1 --- [nio-8080-exec-1] org.mongodb.driver.connection: Opened connection [connectionId{localValue:3, serverValue:3}] to mongo:27017
2023-06-12 02:41:48.200  INFO 1 --- [nio-8080-exec-1] org.mongodb.driver.connection: Opened connection [connectionId{localValue:4, serverValue:4}] to mongo:27017

```

Ilustración 110 - Logs Plots local

Base de datos MongoDB

Conexión establecida con el contenedor desplegado en el puerto 27017 y con la base de datos utilizando el usuario “admin” y contraseña “example”. En la base de datos de autentificación “admin” se encuentra el usuario autorizado para permitirle realizar consultas y operaciones. También está el usuario creado en la base de datos “test” con los permisos adecuados de lectura y escritura, y dentro de esta se encuentra la colección “layers” con el documento inicial de parcela insertado (Ilustración 111 y 112).

```

MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker exec -it 7a369bbcdcc /bin/bash
root@a369bbcdcc:/# mongo -u admin -p example
MongoDB shell version v5.0.7
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { : UUID("df8d4924-791e-4de2-8c8b-0fc63c0a4654") }
MongoDB server version: 5.0.7
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
The server generated these startup warnings when booting:
  2023-06-12T00:59:53.541+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
=====
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
=====
```

Ilustración 111 - Consulta mongo local (1/2)

```

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
> use admin
switched to db admin
> show users
{
    "_id" : "admin.admin",
    "userId" : UUID("b802b342-3c44-4c1e-b735-5c383d944b38"),
    "user" : "admin",
    "db" : "admin",
    "roles" : [
        {
            "role" : "root",
            "db" : "admin"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
> use test
switched to db test
> show users
{
    "_id" : "test.admin",
    "userId" : UUID("ba6cef0d-8c6d-47c2-925b-d88dbb4d8937"),
    "user" : "admin",
    "db" : "test",
    "roles" : [
        {
            "role" : "readWrite",
            "db" : "test"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
> show collections
layers
> db.layers.find()
{ "_id" : ObjectId("611d1ad5dfb24e64eb994bce"), "owner" : "pepe", "catalsticalReference" : "10022A010000380000WZ", "geoData" : { "type" : "FeatureCollection", "features" : [ { "type" : "Feature", "geometry" : { "type" : "Polygon", "coordinates" : [ [ [ -6.611499, 39.479508 ], [ -6.611499, 39.479508 ], [ -6.611397, 39.479586 ], [ -6.611327, 39.47952 ], [ -6.611263, 39.479538 ], [ -6.6112, 39.479557 ], [ -6.611136, 39.479562 ], [ -6.611083, 39.479553 ], [ -6.611013, 39.4795 ], [ -6.610954, 39.479473 ], [ -6.610813, 39.479339 ], [ -6.610701, 39.479254 ], [ -6.610619, 39.479187 ], [ -6.610594, 39 ] ] ] ] ] } }
```

Ilustración 112 - Consulta mongo local (2/2)

Servicio Sensor

Se ejecuta sin errores en el puerto 8081 y se conecta con el servicio contenedor postgres (Ilustración 113).

```

MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ./app.jar
. . .
:: Spring Boot ::      (v2.6.2)

2023-06-12 06:27:41.923  INFO 1 --- [main] c.t.m.sensor.SensorApplication      : Starting SensorApplication using Java 11.0.19 on d88fa21858da with PID 1
(/app/app.jar started by root in /app)
2023-06-12 06:27:42.103  INFO 1 --- [main] c.t.m.sensor.SensorApplication      : No active profile set, falling back to default profiles: default
2023-06-12 06:28:09.134  INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-06-12 06:28:09.872  INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 588 ms. Found 1 JPA repository interfaces.
2023-06-12 06:28:12.778  INFO 1 --- [main] o.s.cloud.context.scope.GenericScope   : BeanFactory id=30b4adb2-d7a9-307c-b287-103d9cea1807
2023-06-12 06:28:22.152  INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8081 (http)
2023-06-12 06:28:22.362  INFO 1 --- [main] o.apache.catalina.core.StandardService: Starting service [Tomcat]
2023-06-12 06:28:22.369  INFO 1 --- [main] org.apache.catalina.core.StandardEngine: Starting Servlet engine: [Apache Tomcat/9.0.56]
2023-06-12 06:28:23.393  INFO 1 --- [main] o.a.c.c.C.[Tomcat].localhost.:1/    : Initializing Spring embedded WebApplicationContext
2023-06-12 06:28:30.313  INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext: Root WebApplicationContext: initialization completed in 37865 ms
2023-06-12 06:28:30.313  INFO 1 --- [main] o.hibernate.jpa.internal.util.LogHelper: HHH000204: Processing PersistenceUnitInfo [name: default]
2023-06-12 06:28:31.880  INFO 1 --- [main] org.hibernate.Version:               HHH000412: Hibernate ORM core version 5.6.3.Final
2023-06-12 06:28:37.148  INFO 1 --- [main] o.hibernate.annotations.common.Version: HACN000001: Hibernate Commons Annotations {5.1.2.Final}
2023-06-12 06:28:40.219  INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource: HikariPool-1 - Starting...
2023-06-12 06:28:43.875  INFO 1 --- [main] com.zaxxer.hikari.HikariDataSource: HikariPool-1 - Start completed.
2023-06-12 06:28:44.844  INFO 1 --- [main] org.hibernate.dialect.Dialect:      HHH000400: Using dialect: org.hibernate.dialect.PostgreSQL95Dialect
2023-06-12 06:29:03.089  INFO 1 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-06-12 06:29:03.161  INFO 1 --- [main] j.LocalContainerEntityManagerFactoryBean: Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-06-12 06:29:13.806  WARN 1 --- [main] JpaBaseConfiguration$JpaWebConfiguration: spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-06-12 06:29:24.105  INFO 1 --- [main] o.s.b.a.e.web.EndpointLinksResolver: Exposing 1 endpoint(s) beneath base path '/actuator'
2023-06-12 06:29:24.965  INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8081 (http) with context path ''
2023-06-12 06:29:25.331  INFO 1 --- [main] c.t.m.sensor.SensorApplication      : Started SensorApplication in 126.922 seconds (JVM running for 148.426)
MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ 
```

Ilustración 113 - Logs Sensor local

Base de datos PostgreSQL

Conexión establecida con el contenedor desplegado en el puerto 5432 y la base de datos utilizando el usuario “postgres”. Cuando se accede a PostgreSQL desplegado en el “localhost” no pide contraseña, solo si se encuentra desplegado en un host remoto. La base de datos “sensors” se encuentra creada y dentro de esta la tabla “sensor” que almacenará los sensores (Ilustración 114).

```

[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker exec -it e5eb54a1e82b /bin/bash
[root@e5eb54a1e82b:~# psql -U postgres
psql (14.2 (Debian 14.2-1.pgdg110+1))
Type "help" for help.

[postgres=# \l
                                         List of databases
  Name   | Owner    | Encoding | Collate | Ctype    | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | en_US.utf8 | en_US.utf8 |
sensors  | postgres | UTF8    | en_US.utf8 | en_US.utf8 |
template0 | postgres | UTF8    | en_US.utf8 | en_US.utf8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
          |          |          |          |          | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
(4 rows)

[postgres=# \c sensors
You are now connected to database "sensors" as user "postgres".
[sensors=# \dt
             List of relations
 Schema | Name | Type | Owner
-----+-----+-----+-----+
 public | sensor | table | postgres
(1 row)
```

Ilustración 114 - Consulta postgres local

Servicio Authentication

Se ejecuta sin errores en el puerto 8999 y se conecta al servicio contenedor mysql (Ilustración 115).

Ilustración 115 - Logs Authentication local

Base de datos MySQL

Conexión establecida con el contenedor desplegado en el puerto 3306 y la base de datos utilizando el usuario “root”, la contraseña “password” y el host “localhost”. La base de datos “users” se encuentra creada y contiene la tabla “user” que almacenará los usuarios (Ilustración 116).

Cloud Computing y migración de una aplicación de riego a AWS

```
MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker exec -it c7aa478ad123 /bin/bash
bash-4.2# mysql -h localhost -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.7.41 MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database      |
+-----+
[] information_schema
[] mysql
[] performance_schema
[] sys
[] users
+-----+
5 rows in set (0.02 sec)

mysql> use users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

[Database changed]
mysql> show tables;
+-----+
| Tables_in_users |
+-----+
[] user
+-----+
1 row in set (0.00 sec)

mysql> describe user;
+-----+-----+-----+-----+-----+-----+
| Field    | Type          | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
[] id      | bigint(20)   | NO   | PRI | NULL    | auto_increment
[] password | varchar(255) | YES  |     | NULL    |
[] username | varchar(255) | YES  |     | NULL    |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)
```

Ilustración 116 - Consulta mysql local

Servicio Statistics

Se ejecuta sin errores en el puerto 8082 y se conecta al servicio contenedor cassandra (Ilustración 117).

Ilustración 117 - Logs Statistics local

Base de datos Cassandra

Conexión establecida con el contenedor desplegado en el puerto 9042 y la base de datos utilizando el usuario “cassandra” y contraseña “cassandra”. El keyspace “sensorkeyspace” se encuentra creado y contiene la tabla “statisticdata” que almacenará la información de las estadísticas (Ilustración 118 y 119).

```
MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ docker exec -it f13203e4b3dd /bin/bash
root@f13203e4b3dd:/# cqlsh -u cassandra -p cassandra

Warning: Using a password on the command line interface can be insecure.
Recommendation: use the credentials file to securely provide the password.

Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.0 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
[cassandra@cqlsh> describe keyspaces;

sensorkeyspace  system_auth      system_schema  system_views
system          system_distributed system_traces  system_virtual_schema
```

Ilustración 118 - Consulta cassandra local (1/2)

```
cassandra@cqlsh> use sensorkeyspace;
cassandra@cqlsh:sensorkeyspace> desc tables;

statisticdata

cassandra@cqlsh:sensorkeyspace> desc statisticdata;

CREATE TABLE sensorkeyspace.statisticdata (
    id text PRIMARY KEY,
    insertontime timestamp,
    sensorid text,
    type text,
    value text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
```

Ilustración 119 - Consulta cassandra local (2/2)

8.2.2. Despliegue de servicios en AWS

Durante la fase de desarrollo en AWS se han ido realizando las capturas necesarias a los servicios a medida que se iba construyendo la infraestructura siguiendo los distintos apartados. Se comprobará el funcionamiento de la aplicación partiendo de que toda la infraestructura se encuentra desplegada y en funcionamiento.

- **Bases de datos**

Se establecerá la conexión hacia las RDS privadas, las cuales no pueden ser accedidas desde fuera de la VPC excepto a través de la instancia EC2 Bastión pública configurada. Se conectará a ella de la siguiente forma (Ilustración 120):

Cloud Computing y migración de una aplicación de riego a AWS

```
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ssh -i "clave_maquinas_TFG.pem" ec2-user@ec2-44-212-59-92.compute-1.amazonaws.com
The authenticity of host 'ec2-44-212-59-92.compute-1.amazonaws.com (44.212.59.92)' can't be established.
ECDSA key fingerprint is SHA256:C1J+/vuzu7fyY90PoCxR9AwI/KR1udnGbW3IYMsDAHo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-212-59-92.compute-1.amazonaws.com' (ECDSA) to the list of known hosts.

A newer release of "Amazon Linux" is available.
 Version 2023.0.20230617:
 Version 2023.0.20230607:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   _ _#
  ~\_\_ #####_      Amazon Linux 2023
~~ \_\_#####\_
~~ \_\#\#\#
~~ \#/ _-- https://aws.amazon.com/linux/amazon-linux-2023
~~     \`-' '->
~~~ /
~~~ _/ _/
~~~ _/m/:
Last login: Mon Jun 12 10:04:54 2023 from 81.43.1.157
[ec2-user@ip-10-0-0-142 ~]$
```

Ilustración 120 - Conexión bastión AWS

RDS Postgres

La conexión a la base de datos “sensor-postgres” se establece utilizando el punto de enlace proporcionado por la RDS: **sensor-postgres.cvt9cc7dxazz.us-east-1.rds.amazonaws.com**. La base de datos “sensors” y su tabla “sensor” se encuentran creadas correctamente (Ilustración 121).

```
[ec2-user@ip-10-0-0-142 ~]$ psql -U postgres -h sensor-postgres.cvt9cc7dxazz.us-east-1.rds.amazonaws.com -p 5432
Password for user postgres:
psql (15.0, server 14.1)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.

postgres=> \l
                                         List of databases
   Name    | Owner     | Encoding | Collate | Ctype | ICU Locale | Locale Provider | Access privileges
---+-----+-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8    | en_US.UTF-8 | en_US.UTF-8 | | libc          |
rdsadmin | rdsadmin | UTF8    | en_US.UTF-8 | en_US.UTF-8 | | libc          | rdsadmin=CTc/rdsadm
in+
      |         |         |         |         |         |         |
in
sensors | postgres | UTF8    | en_US.UTF-8 | en_US.UTF-8 | | libc          |
template0 | rdsadmin | UTF8    | en_US.UTF-8 | en_US.UTF-8 | | libc          | =c/rdsadmin
+
      |         |         |         |         |         |         |
in
template1 | postgres | UTF8    | en_US.UTF-8 | en_US.UTF-8 | | libc          | =c/postgres
+
      |         |         |         |         |         |         |
es
(5 rows)

postgres=> \c sensors
psql (15.0, server 14.1)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
You are now connected to database "sensors" as user "postgres".
sensors=> \dt
                                         List of relations
 Schema |   Name   | Type  | Owner
---+-----+-----+-----+
public | sensor  | table | postgres
(1 row)
```

Ilustración 121 - Consulta postgres AWS

RDS MySQL

La conexión con la base de datos “authentication-mysql” se establece mediante el punto de enlace proporcionado por la RDS: **authentication-mysql.cvt9cc7dxazz.us-east-1.rds.amazonaws.com**. La base de datos “users” y su tabla “user” se encuentran creadas correctamente (Ilustración 122).

```
[ec2-user@ip-10-0-0-142 ~]$ mysql -h authentication-mysql.cvt9cc7dxazz.us-east-1.rds.amazonaws.com -u root -p
[Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 49
Server version: 5.7.41-log Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| innodb |
| mysql |
| performance_schema |
| sys |
| users |
+-----+
6 rows in set (0.01 sec)

mysql> use users
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_users |
+-----+
| user |
+-----+
1 row in set (0.00 sec)

mysql> describe user;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id    | bigint(20) | NO  | PRI | NULL    | auto_increment |
| password | varchar(255) | YES |     | NULL    |             |
| username | varchar(255) | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Ilustración 122 - Consulta mysql AWS

A continuación, se comprobará el despliegue de las EC2 con bases de datos. Son instancias privadas, por lo que también se utilizará el EC2 Bastión, pero con la particularidad de que se necesita la confirmación del par de claves. Se realizarán las conexiones SSH utilizando el método “**SSH Agent Forwarded**”. Para ello, primero se establecerá la conexión SSH desde el equipo local del desarrollador hacia la EC2 Bastión guardando dentro de esta el par de claves necesario para acceder a las instancias de la infraestructura. Una vez dentro del bastión, se podrá acceder al resto de máquinas de forma normal puesto que ya se cuenta con el par de claves necesario para ello (Ilustración 123). Este procedimiento se realiza a través de estos tres comandos:

```
ssh-add -K clave_maquinas_TFG.pem
ssh -A ec2-user@<IP_EC2>
ssh ec2-user@<IP_EC2>
```

```
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ssh-add -K clave_maquinas_TFG.pem
Identity added: clave_maquinas_TFG.pem (clave_maquinas_TFG.pem)
[MacBook-Pro-de-Jesus:~ jesusbarquerocuadrado$ ssh -A ec2-user@44.212.59.92

A newer release of "Amazon Linux" is available.
 Version 2023.0.20230517:
 Version 2023.0.20230607:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #_
      ~\_ #####_      Amazon Linux 2023
      ~\_#####\
      ~~ \###|
      ~~ \#/   https://aws.amazon.com/linux/amazon-linux-2023
      ~~ V~, '-->
      ~~ \_/
      ~~ \_/
      ~~ \_/
      _/m/'
```

Ilustración 123 - Conexión SSH bastión con par de claves

EC2 MongoDB

Desde el bastión se establece conexión con la instancia EC2 de MongoDB a partir de su IP privada: **10.0.2.168** (Ilustración 124).

```
[ec2-user@ip-10-0-0-142 ~]$ ssh ec2-user@10.0.2.168
A newer release of "Amazon Linux" is available.
 Version 2023.0.20230517:
 Version 2023.0.20230607:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #_
      ~\_ #####_      Amazon Linux 2023
      ~\_#####\
      ~~ \###|
      ~~ \#/   https://aws.amazon.com/linux/amazon-linux-2023
      ~~ V~, '-->
      ~~ \_/
      ~~ \_/
      ~~ \_/
      _/m/'
Last login: Mon Jun 12 08:18:26 2023 from 10.0.0.142
[ec2-user@ip-10-0-2-168 ~]$
```

Ilustración 124 - Conexión ec2 mongo

Tanto los usuarios, como la colección “layers” y su documento con la parcela inicial se encuentran creados correctamente (Ilustración 125 y 126).

```
[ec2-user@ip-10-0-2-168 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
e9cee0c6c18c        mongo:5.0.7       "docker-entrypoint.s..."   27 hours ago    Up 2 hours          0.0.0.0:27017->27017/tcp, :::27017->27017/tcp
7/tcp                mongod

[ec2-user@ip-10-0-2-168 ~]$ docker exec -it e9cee0c6c18c /bin/bash
[root@e9cee0c6c18c:~]# mongo -u admin -p example
MongoDB shell version v5.0.7
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("42aae0df-6d95-4d19-802a-a2f2a72532d2") }
MongoDB server version: 5.0.7
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
The server generated these startup warnings when booting:
2023-05-31T12:08:13.465+00:00: Soft riimits for open file descriptors too low
2023-05-31T12:08:13.465+00:00:          currentValue: 32768
2023-05-31T12:08:13.465+00:00:          recommendedMinimum: 64000
---
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

Ilustración 125 - Consulta mongo AWS (1/2)

```

[> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
-> use admin
switched to db admin
-> show users
{
    "_id" : "admin.admin",
    "userId" : UUID("9123849e-61f9-4cdd-8dc6-f9d5878988a2"),
    "user" : "admin",
    "db" : "admin",
    "roles" : [
        {
            "role" : "root",
            "db" : "admin"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
-> use test
switched to db test
-> show users
{
    "_id" : "test.admin",
    "userId" : UUID("e12cadd7-8480-4a86-b9ec-3801870885fc"),
    "user" : "admin",
    "db" : "test",
    "roles" : [
        {
            "role" : "readWrite",
            "db" : "test"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
-> use test
switched to db test
-> show users
{
    "_id" : "test.admin",
    "userId" : UUID("e12cadd7-8480-4a86-b9ec-3801870885fc"),
    "user" : "admin",
    "db" : "test",
    "roles" : [
        {
            "role" : "readWrite",
            "db" : "test"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
-> show collections
layers
-> db.layers.find()
{ "_id" : ObjectId("611d1ad5dfb24e64eb994cce"), "owner" : "pepe", "catalstralReference" : "10022A010000380000WZ", "geoData" : { "type" : "FeatureCollection", "features" : [ { "type" : "Feature", "geometry" : { "type" : "Polygon", "coordinates" : [ [ [ -6.611499, 39.479588 ], [ -6.611499, 39.479588 ], [ -6.611484, 39.479585 ], [ -6.611397, 39.479586 ], [ -6.611327, 39.479582 ], [ -6.611263, 39.479538 ], [ -6.6112, 39.479557 ], [ -6.611136, 39.479562 ], [ -6.611083, 39.479553 ], [ -6.610183, 39.479527 ], [ -6.610954, 39.479473 ], [ -6.610813, 39.479339 ], [ -6.610781, 39.479254 ], [ -6.610619, 39.479187 ], [ -6.610594, 39.479149 ], [ -6.61104, 39.478937 ], [ -6.611493, 39.479257 ] ] ] }, "properties" : { } } ] }, "class" : "com.tfm.microservices.plots.repository.Plot" }
-> █

```

Ilustración 126 - Consulta mongo AWS (2/2)

Mediante el comando “docker volumen ls” se comprueba que ha creado el volumen Docker para la persistencia de datos (Ilustración 127):

```

[le2-user@ip-10-0-2-168 ~]$ docker volume ls
DRIVER      VOLUME NAME
local      dffe172c12d5548537b85419eaa3d14aec50eb048a27de9d66404fc1fae3e98a
local      mongodb-volume
[le2-user@ip-10-0-2-168 ~]$ █

```

Ilustración 127 - Mongo volumen

Para confirmar que se han guardado los datos en el EFS, al volver a conectarse a la instancia bastión se puede ver que dentro de la carpeta “mongo-efs” los datos se encuentran almacenados correctamente (Ilustración 128).

```
[ec2-user@ip-10-0-2-168 ~]$ exit
logout
Connection to 10.0.2.168 closed.
[ec2-user@ip-10-0-0-142 ~]$ cd /
[ec2-user@ip-10-0-0-142 /]$ ls
bin boot dev efs etc home lib lib64 local media mnt opt proc root run sbin srv sys tmp usr var
[ec2-user@ip-10-0-0-142 /]$ cd efs/
[ec2-user@ip-10-0-0-142 efs]$ ls
cassandra-efs mongo-efs
[ec2-user@ip-10-0-0-142 efs]$ ls mongo-efs/
WiredTiger           collection-2--3105313988360081996.wt   index-6--3105313988360081996.wt
WiredTiger.lock       collection-4--3105313988360081996.wt   index-8--3105313988360081996.wt
WiredTiger.turtle     collection-7--3105313988360081996.wt   index-9--3105313988360081996.wt
WiredTiger.wt         diagnostic.data
WiredTigerHS.wt      index-1--3105313988360081996.wt
_mdb_catalog.wt       index-11--3105313988360081996.wt
collection-0--3105313988360081996.wt   index-3--3105313988360081996.wt
collection-10--3105313988360081996.wt  index-5--3105313988360081996.wt
[ec2-user@ip-10-0-0-142 efs]$ ■
```

Ilustración 128 - Mongo EFS

EC2 Cassandra

Desde el bastión se establece conexión con la instancia EC2 de MongoDB a partir de su IP privada: **10.0.2.34** (Ilustración 129).

```
[ec2-user@ip-10-0-0-142 ~]$ ssh ec2-user@10.0.2.34
A newer release of "Amazon Linux" is available.
 Version 2023.0.20230517:
 Version 2023.0.20230607:
Run "/usr/bin/dnf check-release-update" for full release and version update info
' _#
~\_\_ #####          Amazon Linux 2023
~~ \_\#\#\#\_
~~ \#\#\#
~~ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~ V~' '-->
~~~ /
~~~ /_/
~~~ /_/
~/m/
Last login: Mon Jun 12 08:39:12 2023 from 10.0.0.142
[ec2-user@ip-10-0-2-34 ~]$
```

Ilustración 129 - Conexión ec2 cassandra

El keyspace “sensorkeyspace” y su tabla “statisticdata” se encuentran creadas correctamente (Ilustración 130).

```
[ec2-user@ip-10-0-2-34 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
306838d3ed97        cassandra:4.1.0   "docker-entrypoint.s..."  2 hours ago        Up 2 hours        7000-7001/tcp, 7199/tcp, 9160/tcp
, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp    cassandra_db
[ec2-user@ip-10-0-2-34 ~]$ docker exec -it 306838d3ed97 /bin/bash
[root@306838d3ed97:~]# cqlsh -u cassandra -p cassandra
Warning: Using a password on the command line interface can be insecure.
Recommendation: use the credentials file to securely provide the password.

Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.0 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
[cassandra@cqlsh]> DESCRIBE keyspaces;

sensorkeyspace  system_auth          system_schema  system_views
system          system_distributed    system_traces  system_virtual_schema

[cassandra@cqlsh]> USE sensorkeyspace;
[cassandra@cqlsh:sensorkeyspace]> desc tables;

statisticdata

[cassandra@cqlsh:sensorkeyspace]> desc statisticdata;

CREATE TABLE sensorkeyspace.statisticdata (
    id text PRIMARY KEY,
    insertontime timestamp,
    sensorid text,
    type text,
    value text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32',
    'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
```

Ilustración 130 - Consulta cassandra AWS

Mediante el comando “docker volumen ls” se comprueba que ha creado el volumen Docker para la persistencia de datos junto con EFS (Ilustración 131):

```
[ec2-user@ip-10-0-2-34 ~]$ docker volume ls
DRIVER      VOLUME NAME
local        43156dbe6df8129e6493b6fa6677527f9cc87d13b821d7908dc580af692e54a0
local        cassandra_cassandra_volume
[ec2-user@ip-10-0-2-34 ~]$
```

Ilustración 131 - Cassandra volumen

Para confirmar que se han guardado los datos en el EFS, al volver a conectarse a la instancia bastión se puede ver que dentro de la carpeta “cassandra-efs” los datos se encuentran almacenados correctamente (Ilustración 132).

```
[ec2-user@ip-10-0-0-142 ~]$ cd /
[ec2-user@ip-10-0-0-142 /]$ ls
bin  boot  dev  efs  etc  home  lib  lib64  local  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[ec2-user@ip-10-0-0-142 /]$ cd efs/
[ec2-user@ip-10-0-0-142 efs]$ ls
cassandra-efs  mongo-efs
[ec2-user@ip-10-0-0-142 efs]$ ls cassandra-efs/
commitlog  data  hints  saved_caches
[ec2-user@ip-10-0-0-142 efs]$
```

Ilustración 132 - EFS Cassandra

- **Instancia EC2 Front**

EC2 Front-End

Aunque esta instancia sea pública, su grupo de seguridad está configurado para solo aceptar conexiones SSH desde la instancia Bastión para añadir una capa más de seguridad. Se establece la conexión desde el bastión a su IP privada: **10.0.1.16**.

El contenedor se encuentra en funcionamiento y sus logs muestran que se ha ejecutado correctamente (Ilustración 133 y 134):

```
[ec2-user@ip-10-0-1-16 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
a52d001b55df      jesusbc/tfm-front-end:latest   "/docker-env-js-to-s..."   28 minutes ago    Up 28 minutes     0.0.0.0
:4200->4200/tcp, :::4200->4200/tcp   tfm-front-end
[ec2-user@ip-10-0-1-16 ~]$ docker logs a52d001b55df
ALB-principal-1848667940.us-east-1.elb.amazonaws.com

> tfm-front-end@0.0.0 start
> ng serve --host 0.0.0.0 --disable-host-check

Warning: This is a simple server for use in testing or debugging Angular applications
locally. It hasn't been reviewed for security issues.

Binding this server to an open connection can result in compromising your application or
computer. Using a different host than the one passed to the "--host" flag might result in
websocket connection issues. You might need to use "--disableHostCheck" if that's the
case.
Warning: Running a server with --disable-host-check is a security risk. See https://medium.com/webpack/webpack-
dev-server-middleware-security-issues-1489d950874a for more information.
Warning: Entry point 'angular-highcharts' contains deep imports into '/app/node_modules/highcharts/highmaps', '.
/app/node_modules/highcharts/highstock'. This is probably not a problem, but may cause the compilation of entry
points to be out of order.
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
Compiling @angular/common/http : es2015 as esm2015
Compiling @angular/router : es2015 as esm2015
Compiling @angular/forms : es2015 as esm2015
Compiling @ng-bootstrap/ng-bootstrap : es2015 as esm2015
```

Ilustración 133 – Docker ps y logs Front (1/2)

```
** Angular Live Development Server is listening on 0.0.0.0:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
- Generating browser application bundles...
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Size
styles.css          | styles | 228.10 kB

5 unchanged chunks

Build at: 2023-05-31T12:09:33.954Z - Hash: 394f1d6520d2de657160 - Time: 2849ms

✓ Compiled successfully.
[ec2-user@ip-10-0-1-16 ~]$
```

Ilustración 134 – Docker ps y logs Front (2/2)

Se dirige correctamente a la página principal al abrir el navegador en la URL “<http://alb-principal-1848667940.us-east-1.elb.amazonaws.com:4200/>”. No se utiliza la dirección “localhost” puesto que el ALB se encarga de redirigir las peticiones del Front al resto de servicios (Ilustración 135).

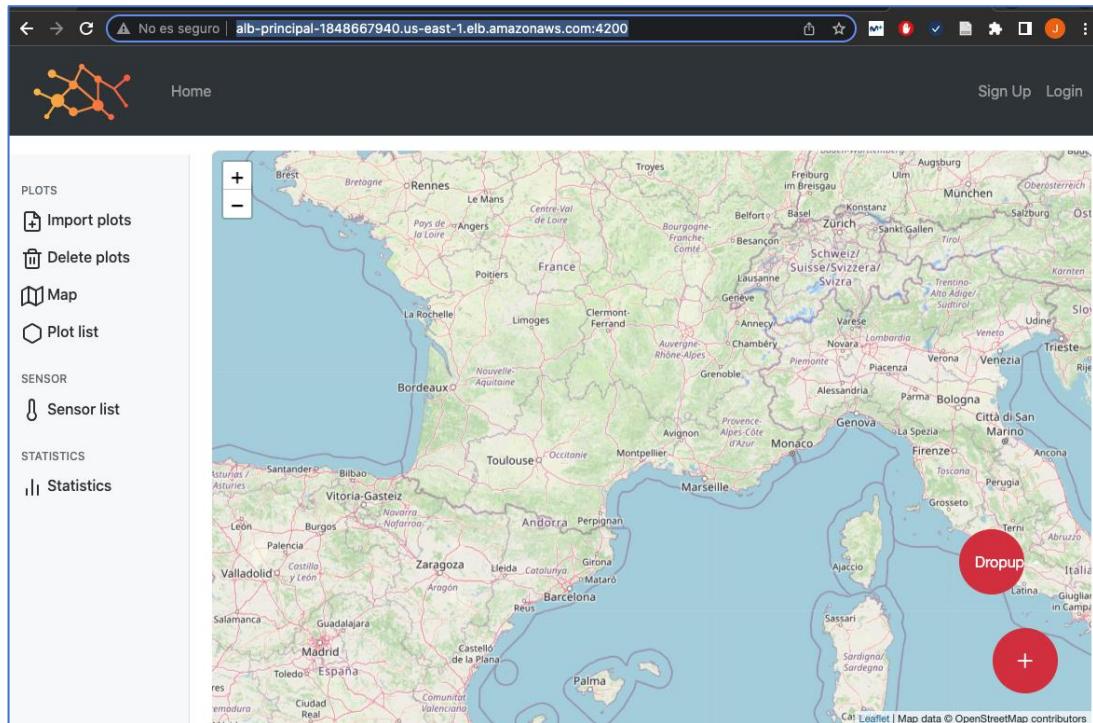


Ilustración 135 - Front AWS navegador

- Clúster ECS y comprobación del funcionamiento de las tareas con contenedores**

Todas las tareas se encuentran lanzadas sobre las instancias contenedoras EC2 del clúster. Se han repartido automáticamente situando tres tareas dentro de la instancia “i-0a1...” y una tarea en “i-0c81...” (Ilustración 136).

Servicios	Tareas	Instancias de ECS	Métricas	Tareas programadas	Etiquetas	Proveedores de capacidad		
Una instancia de Amazon ECS es una instancia externa registrada mediante ECS Anywhere o una instancia de Amazon EC2.								
Para registrar una instancia externa, seleccione Registrar instancias externas y siga los pasos. Más información								
Para registrar una instancia de Amazon EC2, puede utilizar la consola de Amazon EC2. Más información								
Registrar instancias externas	Acciones	Última actualización realizada a las mayo 31, p. m. 3:37:03 p. m. (hace 0 minutos)						
Estado: ALL ACTIVE DRAINING		< 1-2 > Tamaño de la página 50						
Filtrar por atributos (haga clic o pulse la flecha hacia abajo para ver las opciones de filtro)								
Insta...	Instancia de E...	Zona de di...	Instancia ...	Agente cone...	Estado	Recuento d...	CPU dispon...	Memoria d...
1f293...	i-0c81cdb358b...	us-east-1a	false	true	ACTIVE	1	1024	670
23c94...	i-0a1f60acf77b...	us-east-1a	false	true	ACTIVE	3	1024	70

Ilustración 136 - Panel instancias ECS

En el panel de tareas se puede ver que todas se encuentran en estado de ejecución (Ilustración 137):

The screenshot shows the AWS ECS Tasks console. The top navigation bar has tabs: Servicios, Tareas (selected), Instancias de ECS, Métricas, Tareas programadas, Etiquetas, and Proveedores de capacidad. Below the tabs are buttons: Ejecutar nueva tarea (blue), Detener, Detener todo, and Acciones. A status message says "Última actualización realizada a las mayo 31, p. m. 4:44:25 p. m. (hace 0 minutos)". There are refresh and help icons. The main area shows a table of tasks:

	Tarea	Definición de tarea ...	Instancia de ...	Último esta...	Estado ...	In...	I...	G...	Tipo d...	V...
<input type="checkbox"/>	5195dffaa2b3...	tarea-authentication:1	23c94817ceeb...	RUNNING	RUNNING	p....		fa...	EC2	--
<input type="checkbox"/>	94960dfe80...	tarea-sensor:7	23c94817ceeb...	RUNNING	RUNNING	p....		fa...	EC2	--
<input type="checkbox"/>	b44ff9872f1...	tarea-statistics:1	23c94817ceeb...	RUNNING	RUNNING	p....		fa...	EC2	--
<input type="checkbox"/>	e842708081...	tarea-plots:3	1f293cc490b7...	RUNNING	RUNNING	p....		fa...	EC2	--

Ilustración 137 - Panel tareas ECS

A continuación, se conectará a las instancias ECS a través del bastión para comprobar que los contenedores funcionan. Junto a los contenedores de tareas, se estará ejecutando también el agente de ECS:

Primera instancia: i-0c81cdb358b1bee23

Su dirección privada a la que se conecta el bastión es **10.0.0.45** y contiene ejecutándose solo la tarea con el servicio plots (Ilustración 138):

```
[ec2-user@ip-10-0-0-45 ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
ebde491405ec        jesusbc/plots      "java -jar app.jar"   19 minutes ago    Up 19 minutes       0.0.0.0:8080->8080
/tcp, :::8080->8080/tcp   ecs-tarea-plots-3-plots-dac8f482c1d99cc78201
bf42a876fd92        amazon/amazon-ecs-agent:latest  "/agent"
                      ecs-agent
[ec2-user@ip-10-0-0-45 ~]$
```

Ilustración 138 - Docker ps instancia ECS 1

Tarea Plots

El contenedor se encuentra en funcionamiento y sus logs muestran que se ha ejecutado correctamente (Ilustración 139):

```
[ec2-user@ip-10-0-0-45 ~]$ docker logs ebde49140sec
[ec2-user@ip-10-0-0-45 ~]$ [REDACTED]
:: Spring Boot ::
(v2.6.2)

2023-05-31 13:36:02.007 INFO 1 --- [           main] c.t.m.plots.PLOTSApplication      : Starting PLOTSApplication using Java 11.0.19 on ebde49140sec
with PID 1 (/app/app.jar started by root in /app)
2023-05-31 13:36:02.015 INFO 1 --- [           main] c.t.m.plots.PLOTSApplication      : No active profile set, falling back to default profiles: def
ault
2023-05-31 13:36:04.658 INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data MongoDB repositories in DEFAULT mo
de.
2023-05-31 13:36:04.783 INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 115 ms. Found 1
MongoDB repository interfaces.
2023-05-31 13:36:05.249 INFO 1 --- [           main] o.s.cloud.context.scope.GenericScope   : BeanFactory id=40af2e8b-b893-3202-8150-c65854767457
2023-05-31 13:36:06.300 INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-05-31 13:36:06.337 INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-05-31 13:36:06.338 INFO 1 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.56]
2023-05-31 13:36:06.526 INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2023-05-31 13:36:06.530 INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4371
ms
2023-05-31 13:36:07.824 INFO 1 --- [           main] org.mongodb.driver.cluster          : Cluster created with settings {hosts=[10.0.2.168:27017], mod
e=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2023-05-31 13:36:08.040 INFO 1 --- [0.0.2.168:27017] org.mongodb.driver.connection : Opened connection [connectionId{localValue:2, serverValue:1}]
to 10.0.2.168:27017
2023-05-31 13:36:08.040 INFO 1 --- [0.0.2.168:27017] org.mongodb.driver.connection : Opened connection [connectionId{localValue:1, serverValue:2}]
to 10.0.2.168:27017
2023-05-31 13:36:08.053 INFO 1 --- [0.0.2.168:27017] org.mongodb.driver.cluster          : Monitor thread successfully connected to server with descrip
tion ServerDescription{address=10.0.2.168:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13, maxDocumentSize=16777216, lo
gicalSessionTimeoutMinutes=30, roundTripTimeNanos=538733126}
2023-05-31 13:36:10.512 INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver   : Exposing 1 endpoint(s) beneath base path '/actuator'
2023-05-31 13:36:10.598 INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-05-31 13:36:10.637 INFO 1 --- [           main] c.t.m.plots.PLOTSApplication        : Started PLOTSApplication in 10.015 seconds (JVM running for
11.324)
2023-05-31 13:36:29.548 INFO 1 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-31 13:36:29.548 INFO 1 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet    : Initializing Servlet 'dispatcherServlet'
2023-05-31 13:36:29.552 INFO 1 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet    : Completed initialization in 2 ms
2023-05-31 13:36:30.041 INFO 1 --- [nio-8080-exec-1] org.mongodb.driver.connection       : Opened connection [connectionId{localValue:4, serverValue:3}]
to 10.0.2.168:27017
2023-05-31 13:36:30.041 INFO 1 --- [nio-8080-exec-2] org.mongodb.driver.connection       : Opened connection [connectionId{localValue:3, serverValue:4}]
[ec2-user@ip-10-0-0-45 ~]$
```

Ilustración 139 - Plot logs AWS

Segunda instancia: i-0a1f60acf77b46271

Su dirección privada a la que se conecta el bastión es **10.0.0.50** y contiene ejecutándose las tareas con los servicios: sensor, authentication y statistics. Los contenedores se encuentran en funcionamiento y sus logs muestran que se ha ejecutado correctamente (Ilustración 140-144):

```
[ec2-user@ip-10-0-0-50 ~]$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED             STATUS              PORTS
18198d6d361a   jesusbc/sensor     "java -jar app.jar"   16 minutes ago   Up 16 minutes   0.0.0
.e:8081->8081/tcp, :::8081->8081/tcp   ecs-tarea-sensor-7-sensor-86d5c3dabfb6f494fb01
e5fc8c77d32f   jesusbc/authentication "java -jar app.jar"   About an hour ago   Up About an hour   0.0.0
.e:8999->8999/tcp, :::8999->8999/tcp   ecs-tarea-authentication-1-authentication-fee0e782d2cff8b2ca01
5e07d603cac8   jesusbc/statistics  "java -jar app.jar"   About an hour ago   Up About an hour   0.0.0
.e:8082->8082/tcp, :::8082->8082/tcp   ecs-tarea-statistics-1-statistics-8ef8fff4f2d0bae4f101
856e6181le50   amazon/amazon-ecs-agent:latest  "/agent"           3 hours ago      Up 3 hours (healthy)
                                         ecs-agent
[ec2-user@ip-10-0-0-50 ~]$
```

Ilustración 140 - Docker ps instancia ECS 2

Tarea Sensor

```
[ec2-user@ip-10-0-0-50 ~]$ docker logs 18198d6d361a
.
.
.
:: Spring Boot ::          (v2.6.2)

2023-05-31 13:34:54.132  INFO 1 --- [           main] c.t.m.sensor.SensorApplication      : Starting SensorApplication using Java 11.0.19 on 18198d6
d361a with PID 1 (/app/app.jar started by root in /app)
2023-05-31 13:34:54.142  INFO 1 --- [           main] c.t.m.sensor.SensorApplication      : No active profile set, falling back to default profiles: default
2023-05-31 13:34:57.033  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2023-05-31 13:34:57.180  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 126 ms. Found 1 JPA repository interfaces.
2023-05-31 13:34:57.686  INFO 1 --- [           main] o.s.cloud.context.scope.GenericScope   : BeanFactory id=30b4adb2-d7a9-307c-b287-103d9cea1807
2023-05-31 13:34:59.201  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8081 (http)
2023-05-31 13:34:59.056  INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-05-31 13:34:59.057  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.56]
2023-05-31 13:34:59.221  INFO 1 --- [           main] o.a.c.c.C.[localhost].[]            : Initializing Spring embedded WebApplicationContext
2023-05-31 13:34:59.226  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 4935 ms
2023-05-31 13:35:00.804  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-05-31 13:35:00.986  INFO 1 --- [           main] org.hibernate.Version                 : HHH000412: Hibernate ORM core version 5.6.3.Final
2023-05-31 13:35:01.557  INFO 1 --- [           main] o.hibernate.annotations.common.Version : HCANN00001: Hibernate Commons Annotations (5.1.2.Final)
2023-05-31 13:35:01.874  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-05-31 13:35:03.084  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-05-31 13:35:03.131  INFO 1 --- [           main] org.hibernate.dialect.Dialect       : HHH000400: Using dialect: org.hibernate.dialect.PostgresQL5Dialect
2023-05-31 13:35:04.920  INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator  : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-05-31 13:35:04.946  INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-05-31 13:35:06.029  WARN 1 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-05-31 13:35:07.241  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver  : Exposing 1 endpoint(s) beneath base path '/actuator'
2023-05-31 13:35:07.349  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8081 (http) with context path ''
2023-05-31 13:35:07.392  INFO 1 --- [           main] c.t.m.sensor.SensorApplication        : Started SensorApplication in 14.774 seconds (JVM running for 16.458)
2023-05-31 13:35:09.696  INFO 1 --- [nio-8081-exec-1] o.a.c.c.C.[localhost].[]            : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-31 13:35:09.697  INFO 1 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet     : Initializing Servlet 'dispatcherServlet'
2023-05-31 13:35:09.700  INFO 1 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet     : Completed initialization in 3 ms
[ec2-user@ip-10-0-0-50 ~]$
```

Ilustración 141 - Logs sensor AWS

Tarea Authentication

```
[ec2-user@ip-10-0-0-50 ~]$ docker logs e5fc8c77d32f
.
.
.
:: Spring Boot ::          (v2.4.5)

2023-05-31 12:44:25.863  INFO 1 --- [           main] c.t.m.a.AuthenticationApplication    : Starting AuthenticationApplication using Java 11.0.19 on e5fc8c77d32f with PID 1 (/app/app.jar started by root in /app)
2023-05-31 12:44:25.873  INFO 1 --- [           main] c.t.m.a.AuthenticationApplication    : No active profile set, falling back to default profiles: default
2023-05-31 12:44:29.545  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-05-31 12:44:29.678  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 118 ms. Found 1 JPA repository interfaces.
2023-05-31 12:44:30.634  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler' of type [org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2023-05-31 12:44:30.645  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'methodSecurityMetadataSource' of type [org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2023-05-31 12:44:31.648  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8999 (http)
2023-05-31 12:44:31.691  INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-05-31 12:44:31.692  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.45]
2023-05-31 12:44:31.830  INFO 1 --- [           main] o.a.c.c.C.[localhost].[]            : Initializing Spring embedded WebApplicationContext
2023-05-31 12:44:31.834  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 5794 ms
2023-05-31 12:44:33.592  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name:
```

Ilustración 142 - Logs Auth AWS (1/2)

Cloud Computing y migración de una aplicación de riego a AWS

```

2023-05-31 12:44:31.834 INFO 1 --- [      main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 5794 ms
2023-05-31 12:44:33.592 INFO 1 --- [      main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-05-31 12:44:33.783 INFO 1 --- [      main] org.hibernate.Version                  : HHH000412: Hibernate ORM core version 5.4.30.Final
2023-05-31 12:44:34.256 INFO 1 --- [      main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2023-05-31 12:44:34.683 INFO 1 --- [      main] com.zaxxer.hikari.HikariDataSource   : HikariPool-1 - Starting...
2023-05-31 12:44:35.645 INFO 1 --- [      main] com.zaxxer.hikari.HikariDataSource   : HikariPool-1 - Start completed.
2023-05-31 12:44:35.782 INFO 1 --- [      main] org.hibernate.dialect.Dialect       : HHH000400: Using dialect: org.hibernate.dialect.MySQL57Dialect
2023-05-31 12:44:37.728 INFO 1 --- [      main] o.h.e.t.j.p.i.JtaPlatformInitiator  : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-05-31 12:44:37.758 INFO 1 --- [      main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-05-31 12:44:37.879 WARN 1 --- [      main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-05-31 12:44:39.298 INFO 1 --- [      main] o.s.s.web.DefaultSecurityFilterChain  : Will secure any request with [org.springframework.web.context.request.async.WebAsyncManagerIntegrationFilter@22d9ca63, org.springframework.security.web.context.SecurityContextPersistenceFilter@314a31b0, org.springframework.security.web.header.HeaderWriterFilter@41ef1ea2, org.springframework.security.web.authentication.logout.LogoutFilter@7f323b3a, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@75cf0de5, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@484149eb, org.springframework.security.web.session.SessionManagementFilter@104dc1a2, org.springframework.security.web.access.ExceptionTranslationFilter@1cfc336fd, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@45b32dfe]
2023-05-31 12:44:39.789 INFO 1 --- [      main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2023-05-31 12:44:41.117 INFO 1 --- [      main] o.s.b.a.e.web.EndpointLinksResolver   : Exposing 2 endpoint(s) beneath base path '/actuator'
2023-05-31 12:44:41.256 INFO 1 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8999 (http) with context path ''
2023-05-31 12:44:41.257 INFO 1 --- [      main] o.l.springboot.grpc.GRpcServerRunner  : Starting gRPC Server ...
2023-05-31 12:44:41.348 INFO 1 --- [      main] o.l.springboot.grpc.GRpcServerRunner  : 'com.tfm.microservices.authentication.TykDispatcher' service has been registered.
2023-05-31 12:44:41.683 INFO 1 --- [      main] o.l.springboot.grpc.GRpcServerRunner  : gRPC Server started, listening on port 6565.
2023-05-31 12:44:41.687 INFO 1 --- [      main] c.t.m.a.AuthenticationApplication    : Started AuthenticationApplication in 17.29 seconds (JVM running for 18.836)
2023-05-31 12:45:03.565 INFO 1 --- [nio-8999-exec-2] o.a.c.c.C.[Tomcat].[localhost].[]/  : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-31 12:45:03.566 INFO 1 --- [nio-8999-exec-2] o.s.web.servlet.DispatcherServlet     : Initializing Servlet 'dispatcherServlet'
2023-05-31 12:45:03.567 INFO 1 --- [nio-8999-exec-2] o.s.web.servlet.DispatcherServlet     : Completed initialization in 1 ms
[ec2-user@ip-10-0-0-50 ~]$ 

```

Ilustración 143 - Logs Auth AWS (2/2)

Tarea Statistics

```

[ec2-user@ip-10-0-0-50 ~]$ docker logs 5e07d603cac8
[      main] c.t.m.statistics.StatisticsApplication : Starting StatisticsApplication using Java 11.0.19 on 5e07d603cac8 with PID 1 (/app/app.jar started by root in /app)
2023-05-31 12:43:38.625 INFO 1 --- [      main] c.t.m.statistics.StatisticsApplication : No active profile set, falling back to default profiles: default
2023-05-31 12:43:40.946 INFO 1 --- [      main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Reactive Cassandra repositories in DEFAULT mode.
2023-05-31 12:43:41.021 INFO 1 --- [      main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 62 ms. Found 0 reactive Cassandra repository interfaces.
2023-05-31 12:43:41.030 INFO 1 --- [      main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Cassandra repositories in DEFAULT mode.
2023-05-31 12:43:41.081 INFO 1 --- [      main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 58 ms. Found 1 Cassandra repository interfaces.
2023-05-31 12:43:42.476 INFO 1 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8082 (http)
2023-05-31 12:43:42.537 INFO 1 --- [      main] o.apache.catalina.core.StandardService: Starting service [Tomcat]
2023-05-31 12:43:42.550 INFO 1 --- [      main] org.apache.catalina.core.StandardEngine: Starting Servlet engine: [Apache Tomcat/9.0.56]
2023-05-31 12:43:42.802 INFO 1 --- [      main] o.a.c.c.C.[Tomcat].[localhost].[]/  : Initializing Spring embedded WebApplicationContext
2023-05-31 12:43:42.802 INFO 1 --- [      main] w.s.c.ServletWebServerApplicationContext: Root WebApplicationContext: initialization completed in 4049 ms
2023-05-31 12:43:44.704 INFO 1 --- [      main] c.d.o.d.i.core.DefaultMavenCoordinates: DataStax Java driver for Apache Cassandra(R) (com.datastax.oss:java-driver-core) version 4.13.0
2023-05-31 12:43:45.775 INFO 1 --- [s0-admin-0] c.d.o.s.driver.internal.core.time.Clock: Using native clock for microsecond precision
2023-05-31 12:43:46.180 WARN 1 --- [s0-io-0] c.d.o.d.a.c.a.PlainTextAuthProviderBase: [] /10.0.2.34:9042 did not send an authentication challenge; This is suspicious because the driver expects authentication
2023-05-31 12:43:46.455 WARN 1 --- [s0-io-1] c.d.o.d.a.c.a.PlainTextAuthProviderBase: [] /10.0.2.34:9042 did not send an authentication challenge; This is suspicious because the driver expects authentication
2023-05-31 12:43:50.621 INFO 1 --- [      main] o.s.b.a.e.web.EndpointLinksResolver   : Exposing 1 endpoint(s) beneath base path '/actuator'
2023-05-31 12:43:50.718 INFO 1 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8082 (http) with context path ''
2023-05-31 12:43:50.719 INFO 1 --- [      main] o.s.m.s.b.SimpleBrokerMessageHandler: Starting...
2023-05-31 12:43:50.719 INFO 1 --- [      main] o.s.m.s.b.SimpleBrokerMessageHandler: BrokerAvailabilityEvent[available=true, SimpleBrokerMessageHandler [org.springframework.messaging.simp.broker.DefaultSubscriptionRegistry@21c7208d]]
2023-05-31 12:43:50.721 INFO 1 --- [      main] o.s.m.s.b.SimpleBrokerMessageHandler: Started.
2023-05-31 12:43:50.746 INFO 1 --- [      main] c.t.m.statistics.StatisticsApplication: Started StatisticsApplication in 13.366 seconds (JVM running for 14.533)
2023-05-31 12:44:19.017 INFO 1 --- [nio-8082-exec-2] o.a.c.c.C.[Tomcat].[localhost].[]/  : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-31 12:44:19.017 INFO 1 --- [nio-8082-exec-2] o.s.web.servlet.DispatcherServlet     : Initializing Servlet 'dispatcherServlet'
2023-05-31 12:44:19.026 INFO 1 --- [nio-8082-exec-2] o.s.web.servlet.DispatcherServlet     : Completed initialization in 8 ms
2023-05-31 12:44:48.827 INFO 1 --- [MessageBroker-1] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[0 current WS(0)-HttpStream(0)-HttpPoll(0), 0 total, 0 closed abnormally (0 connect failure, 0 send limit, 0 transport error)], stompSubProtocol[processed CONNECT(0)-CONNECTED(0)-DISCONNECT(0)], stompBrokerRelay(null), inboundChannel[pool size = 0, active threads = 0, queued tasks = 0], outboundChannel[pool size = 0, active threads = 0, queued tasks = 0, completed tasks = 0], sockJsScheduler[pool size = 1, active threads = 1, queued tasks = 0, completed tasks = 0]
[ec2-user@ip-10-0-0-50 ~]$ 

```

Ilustración 144 - Logs Stats AWS

- **ALB y comprobación de salud de los servicios**

Por último, se comprueba que el balanceador de carga ALB puede alcanzar los destinos (instancias) con los servicios desplegados. Esta tarea la realizan los grupos de destino a través de su petición de comprobación del estado. Si un destino se encuentra en estado saludable, el ALB redirigirá la petición escuchada en el “listener”, en función del puerto, a través del grupo de destino hacia el destino correspondiente.

Destino Front

El destino saludable se corresponde con la instancia EC2 donde se encuentra desplegado el servicio Front. No se deberían haber incluido en el grupo de destino las instancias del clúster ECS puesto que en ellas no se ejecuta el Front, pero sirve como ejemplo para ver que el único destino válido es la EC2 del Front (Ilustración 145).

Registered targets (3)						
<input type="button" value="Filter resources by property or value"/> <input type="button" value="C"/> <input type="button" value="Deregister"/> <input type="button" value="Register targets"/> < 1 > 						
	Instance ID	Name	Port	Zone	Health ...	Health status details
<input type="checkbox"/>	i-0de1f716...	ec2-front	4200	us-east-1b	healthy	
<input type="checkbox"/>	i-0a1f60acf...	2 ECS Instan...	4200	us-east-1a	unhealthy... Request timed out	
<input type="checkbox"/>	i-0c81cdb35...	1 ECS Instan...	4200	us-east-1a	unhealthy... Request timed out	

Ilustración 145 - Destino front

Destino Plots

El destino saludable se corresponde con la instancia EC2 donde se encuentra desplegado el servicio Plots (Ilustración 146).

Registered targets (2)						
<input type="button" value="Filter resources by property or value"/> <input type="button" value="C"/> <input type="button" value="Deregister"/> <input type="button" value="Register targets"/> < 1 > 						
	Instance ID	Name	Port	Zone	Health s...	Health status details
<input type="checkbox"/>	i-0a1f60acf...	2 ECS Instan...	8080	us-east-1a	unhealthy	Health checks failed
<input type="checkbox"/>	i-0c81cdb35...	1 ECS Instan...	8080	us-east-1a	healthy	

Ilustración 146 - Destino plots

Destino Sensor

El destino saludable se corresponde con la instancia EC2 donde se encuentra desplegado el servicio Sensor (Ilustración 147).

Registered targets (2)						
<input type="text"/> Filter resources by property or value						
	Instance ID	Name	Port	Zone	Health s...	Health status de...
<input type="checkbox"/>	i-0a1f60acf...	2 ECS Instan...	8081	us-east-1a	healthy	
<input type="checkbox"/>	i-0c81cdb35...	1 ECS Instan...	8081	us-east-1a	unhealthy	Health checks fail...

Ilustración 147 - Destino sensor

Destino Authentication

El destino del servicio Authentication no se encuentra saludable. No se debe a que el servicio en sí no se encuentre operativo (se ha comprobado anteriormente que se encuentra desplegado correctamente), sino que se debe a un problema de autorización (status: 401) que genera la implementación de este servicio (Ilustración 148).

Registered targets (2)						
<input type="text"/> Filter resources by property or value						
	Instance ID	Name	Port	Zone	Health ...	Health status details
<input type="checkbox"/>	i-0a1f60acf...	2 ECS Instan...	8999	us-east-1a	unhealthy	Health checks failed with these codes: [401]
<input type="checkbox"/>	i-0c81cdb35...	1 ECS Instan...	8999	us-east-1a	unhealthy	Health checks failed

Ilustración 148 - Destino Authentication

Destino Statistics

El destino saludable se corresponde con la instancia EC2 donde se encuentra desplegado el servicio Statistics (Ilustración 149).

Instance ID	Name	Port	Zone	Health ...	Health status details
i-0c81cdb358b1b...	1 ECS Instance...	8082	us-east-1a	☒ unhealthy...	Health checks failed
i-0af60acf77b46...	2 ECS Instance...	8082	us-east-1a	☑ healthy	

Ilustración 149 - Destino statistics

8.2.3. Despliegue con Terraform de los servicios AWS

El proyecto de Terraform proporcionado desplegará y configurará la infraestructura de la misma manera en la que se ha explicado durante el desarrollo. Este apartado se centrará en verificar que todos los servicios creados en los ficheros Terraform se encuentren disponibles en AWS (Ilustración 150-200).

Una vez lanzada la infraestructura, los siguientes recursos se crearán en AWS:

VPC

ID de la VPC	Estado	Nombres de host de DNS	Resolución de DNS
vpc-Off8672d0ccc3cc11	☒ Available	Habilitado	Habilitado

Tenencia	Conjunto de opciones de DHCP	Tabla de enrutamiento principal	ACL de red principal
Default	dopt-004e64d0b5e69221a	rtb-0856db645286f2702	acl-07ca115aa7bd1ee34

VPC predeterminada	CIDR IPv4	Grupo IPv6	CIDR IPv6 (grupo de bordes de red)
No	10.1.0.0/16	-	-

Métricas de uso de direcciones de red	Grupos de reglas del firewall de DNS de Route 53 Resolver	ID de propietario
Desactivado	-	690031176274

Ilustración 150 - VPC tf

Internet Gateway

Name	ID de gateway de Internet	Estado	ID de la VPC
igw-tf	igw-03b5d6222b3b5029c	Attached	vpc-Off8672d0ccc3cc11 vpc-tf

Detalles

ID de gateway de Internet igw-03b5d6222b3b5029c	Estado Attached	ID de la VPC vpc-Off8672d0ccc3cc11 vpc-tf	Propietario 690031176274
--	---	--	-----------------------------

Ilustración 151 - IGW tf

Subredes

Name	ID de subred	Estado	VPC	CIDR IPv4
subred-servicios-1a-tf	subnet-02f91e2...	Available	vpc-Off8672d0ccc3cc11 vpc-tf	10.1.10.0/24
subred-front-1b-tf	subnet-0e15c3...	Available	vpc-Off8672d0ccc3cc11 vpc-tf	10.1.20.0/24
subred-databases-1b-tf	subnet-030c49...	Available	vpc-Off8672d0ccc3cc11 vpc-tf	10.1.40.0/24
subred-databases-1a-tf	subnet-0621d0...	Available	vpc-Off8672d0ccc3cc11 vpc-tf	10.1.30.0/24

Ilustración 152 - Subredes tf

IP Elástica

The screenshot shows the AWS Elastic IP console with the following details:

Name	Dirección IPv4 asignada	Tipo	ID de asignación
ip-elastica-nat-tf	52.203.123.24	IP pública	eipalloc-0dd34660a5797a689

Resumen

Dirección IPv4 asignada 52.203.123.24	Tipo IP pública	ID de asignación eipalloc-0dd34660a5797a689	Registro DNS inverso -
ID de asociación eipassoc-0bsec2ab4dcdbda05	Ámbito VPC	ID de la instancia asociada -	Dirección IP privada 10.1.20.46
ID de la interfaz de red eni-09a2f915aa1dc82b8	ID de la cuenta del propietario de la interfaz de red 690031176274	DNS público ec2-52-203-123-24.compute-1.amazonaws.com	ID de la gateway NAT nat-033f1a92078b519a4 (nat-gateway-tf)
Grupo de direcciones Amazon	Grupo fronterizo de red us-east-1		

Ilustración 153 - IP Elástica tf

NAT Gateway

The screenshot shows the AWS NAT Gateways console with the following details:

Name	ID de gateway NAT	Tipo de conexión	Estado	Mensaje de estado
nat-gateway-tf	nat-033f1a92078b519a4	Public	Available	-

Detalles

ID de gateway NAT nat-033f1a92078b519a4	Tipo de conectividad Public	Estado Available	Mensaje de estado -
ARN de puerta de enlace NAT arn:aws:ec2:us-east-1:690031176274:natgateway/nat-033f1a92078b519a4	Dirección IPv4 principal 52.203.123.24	Dirección IPv4 privada principal 10.1.20.46	ID de interfaz de red principal eni-09a2f915aa1dc82b8
VPC vpc-0ff8672d0ccc3cc11 / vpc-tf	Subred subnet-0e15c3d7d0e1fc11b / subred-front-1b-tf	Creado lunes, 12 de junio de 2023, 17:56:40 CEST	Eliminado -

Ilustración 154 - NAT tf

Tablas de enrutamiento

Name	ID de tabla de enrutamiento	Asociaciones de subredes	Asociaciones de borde
checked	rtb-0113c3f196189f024	2 subredes	-
<input type="checkbox"/>	rtb-06dc8c0c5d4876686	2 subredes	-
<input type="checkbox"/>	rtb-0856db645286f2702	-	-

Ilustración 155 - Tabla 1 tf

Destino	Destino	Estado	Propagada
pl-63a5400a	vpce-0c508a40fea604763	Activado	No
0.0.0.0/0	igw-03b5d6222b3b5029c	Activado	No
10.1.0.0/16	local	Activado	No

Ilustración 156 - Tabla 2 tf

Destino	Destino	Estado	Propagada
pl-63a5400a	vpce-0c508a40fea604763	Activado	No
0.0.0.0/0	nat-033f1a92078b519a4	Activado	No
10.1.0.0/16	local	Activado	No

Ilustración 157 - Tabla 3 tf

Asociaciones de tablas de rutas a subredes

Name	ID de subred	CIDR IPv4	CIDR IPv6
subred-servicios-1a-tf	subnet-02f91e2468446f16a	10.1.10.0/24	-
subred-front-1b-tf	subnet-0e15c3d7d0e1fc11b	10.1.20.0/24	-

Ilustración 158 - Asociación 1 tf

Cloud Computing y migración de una aplicación de riego a AWS

Asociaciones de subredes explícitas (2)

Name	ID de subred	CIDR IPv4	CIDR IPv6
subred-databases-1b-tf	subnet-030c492f2d7788ef	10.1.40.0/24	-
subred-databases-1a-tf	subnet-0621d0458d4b48d0f	10.1.30.0/24	-

Ilustración 159 - Asociación 2 tf

Punto de enlace de VPC

Puntos de enlace (1/1)

Name	ID de punto de enlace	ID de la VPC	Nombre del servicio
s3-endpoint-tf	vpce-0c508a40fea604763	vpc-Off8672d0ccc3cc11 vpc-tf	com.amazonaws.us-east-1.s3

vpce-0c508a40fea604763 / s3-endpoint-tf

Detalles

ID de punto de enlace vpce-0c508a40fea604763	Estado Disponible	Hora de creación lunes, 12 de junio de 2023, 17:56:29 CEST	Tipo de punto de enlace Gateway
ID de la VPC vpc-Off8672d0ccc3cc11 (vpc-tf)	Mensaje de estado -	Nombre del servicio com.amazonaws.us-east-1.s3	Nombres de DNS privados habilitados No

Ilustración 160 - Punto enlace S3 tf

Grupos de seguridad

Grupos de seguridad (6)

Name	ID del grupo de segu...	Nombre del grup...	ID de la VPC	Descripción
bastion-sg-tf	sg-0558130baae3ea2da	bastion-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico
databases-sg-tf	sg-0c477e771e92dcb2b	databases-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico
alb-sg-tf	sg-0ab1f542d322038fc	alb-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico
front-sg-tf	sg-0a0814e650b5272f5	front-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico
servicios-sg-tf	sg-0fe386af3c2b0d306	servicios-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico
efs-sg-tf	sg-09e8f168584ee6c37	efs-sg-tf	vpc-Off8672d0ccc3cc11	SG para el tráfico

Ilustración 161 – Grupo seguridad 1 tf

Cloud Computing y migración de una aplicación de riego a AWS

sg-0558130baae3ea2da - bastion-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (1/1)

Reglas de entrada (1/1)								
<input type="checkbox"/> Name		ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen	
<input checked="" type="checkbox"/>		-	sgr-004827449a09...	IPv4	SSH	TCP	22	0.0.0.0/0

Ilustración 162 - Grupo seguridad 2 tf

sg-0c477e771e92dcb2b - databases-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (5)

Reglas de entrada (5)								
<input type="checkbox"/> Name		ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen	
<input type="checkbox"/>		-	sgr-064d966628f1...	-	SSH	TCP	22	sg-0558130baae3ea2
<input type="checkbox"/>		-	sgr-010dfaadd4d7cf...	IPv4	TCP perso...	TCP	27017	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0390153b0def...	IPv4	MySQL/A...	TCP	3306	0.0.0.0/0
<input type="checkbox"/>		-	sgr-02a506755578...	IPv4	PostgreSQL	TCP	5432	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0d2ad4d4a38d...	IPv4	TCP perso...	TCP	9042	0.0.0.0/0

Ilustración 163 - Grupo seguridad 3 tf

sg-0ab1f542d322038fc - alb-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (5)

Reglas de entrada (5)								
<input type="checkbox"/> Name		ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen	
<input type="checkbox"/>		-	sgr-06ba42004fbe...	IPv4	TCP perso...	TCP	4200	0.0.0.0/0
<input type="checkbox"/>		-	sgr-067f1a4f589e1...	IPv4	TCP perso...	TCP	8080	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0f97b376fe832...	IPv4	TCP perso...	TCP	8081	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0074cb022406...	IPv4	TCP perso...	TCP	8082	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0bcbe9593f76...	IPv4	TCP perso...	TCP	8999	0.0.0.0/0

Ilustración 164 - Grupo seguridad 4 tf

sg-0a0814e650b5272f5 - front-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (2)

Reglas de entrada (2)								
<input type="checkbox"/> Name		ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen	
<input type="checkbox"/>		-	sgr-0d996117e43a...	IPv4	TCP perso...	TCP	4200	0.0.0.0/0
<input type="checkbox"/>		-	sgr-0ef778c5f8d86...	-	SSH	TCP	22	sg-0558130baae3ea2

Ilustración 165 - Grupo seguridad 5 tf

Cloud Computing y migración de una aplicación de riego a AWS

sg-0fe386af3c2b0d306 - servicios-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (6)

	Name	ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen
<input type="checkbox"/>	-	sgr-0ace9e805572c...	-	SSH	TCP	22	sg-0558130baae3ea2
<input type="checkbox"/>	-	sgr-0b428728ca4f...	IPv4	HTTPS	TCP	443	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0a0b1a77881a...	IPv4	TCP perso...	TCP	8080	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0d9fb030def72...	IPv4	TCP perso...	TCP	8081	0.0.0.0/0
<input type="checkbox"/>	-	sgr-036c13043000...	IPv4	TCP perso...	TCP	8082	0.0.0.0/0
<input type="checkbox"/>	-	sgr-0d3d50f74e67...	IPv4	TCP perso...	TCP	8999	0.0.0.0/0

Ilustración 166 - Grupo seguridad 6 tf

sg-09e8f168584ee6c37 - efs-sg-tf

Detalles | **Reglas de entrada** | Reglas de salida | Etiquetas

Reglas de entrada (1/1)

	Name	ID de la regla de...	Versió...	Tipo	Protoc...	Interva...	Origen
<input checked="" type="checkbox"/>	-	sgr-0bbce4f5b791...	IPv4	NFS	TCP	2049	0.0.0.0/0

Ilustración 167 - Grupo seguridad 7 tf

Amazon S3 Bucket

db-services-bucket-tf Info

Objetos | Propiedades | Permisos | Métricas | Administración | Puntos de acceso

Objetos (3)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Acciones ▾ | Crear carpeta | **Cargar**

Buscar objetos por prefijo

	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	init-mongodb.js	js	12 Jun 2023 5:56:19 PM CEST	3.3 KB	Estándar
<input type="checkbox"/>	env_vars.env	env	12 Jun 2023 6:00:43 PM CEST	408.0 B	Estándar
<input type="checkbox"/>	cassandra/	Carpetas	-	-	-

Ilustración 168 - Bucket S3 1 tf

Cloud Computing y migración de una aplicación de riego a AWS

Amazon S3 > Buckets > db-services-bucket-tf > cassandra/

cassandra/

Objetos Propiedades

Objetos (3)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Copiar URI de S3 Copiar URL Descargar Abrir Eliminar Acciones ▾

Crear carpeta Cargar

Buscar objetos por prefijo

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
bash/	Carpeta	-	-	-
cql/	Carpeta	-	-	-
docker-compose-cassandra.yml	yml	12 Jun 2023 5:57:58 PM CEST	607.0 B	Estándar

Ilustración 169 - Bucket S3 2 tf

Amazon S3 > Buckets > db-services-bucket-tf > cassandra/ > bash/

bash/

Objetos Propiedades

Objetos (1)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Copiar URI de S3 Copiar URL Descargar Abrir Eliminar Acciones ▾

Crear carpeta Cargar

Buscar objetos por prefijo

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
cassandra_init.sh	sh	12 Jun 2023 5:56:19 PM CEST	557.0 B	Estándar

Ilustración 170 - Bucket S3 3 tf

Amazon S3 > Buckets > db-services-bucket-tf > cassandra/ > cql/

cql/

Objetos Propiedades

Objetos (1)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Copiar URI de S3 Copiar URL Descargar Abrir Eliminar Acciones ▾

Crear carpeta Cargar

Buscar objetos por prefijo

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
init_keyspace.cql	cql	12 Jun 2023 5:56:19 PM CEST	132.0 B	Estándar

Ilustración 171 - Bucket S3 4 tf

Rol IAM

The screenshot shows the 'AmazonS3FullAccess-tf' role details page. At the top, there's a summary section with basic information like creation date (June 12, 2023) and last activity (26 minutes ago). Below this are tabs for 'Permisos' (Permissions), 'Relaciones de confianza' (Trust relationships), 'Etiquetas' (Tags), 'Access Advisor', and 'Revocar las sesiones' (Revoke sessions). The 'Permisos' tab is selected, showing a table of associated policies. One policy is listed: 'AmazonS3FullAccess' (Administrated by AWS, provides full access to all buckets via S3). There are buttons for 'Simular' (Simulate), 'Eliminar' (Delete), and 'Añadir permisos' (Add permissions).

Ilustración 172 - Rol S3 tf

The screenshot shows the 'ecsInstanceRole-tf' role details page. It has a similar structure to the previous one, with a summary section, tabs for 'Permisos', 'Relaciones de confianza', 'Etiquetas', 'Access Advisor', and 'Revocar las sesiones'. The 'Permisos' tab is selected, showing a table of associated policies. One policy is listed: 'AmazonEC2ContainerServiceforEC2Role' (Administrated by AWS, default policy for the Amazon EC2 R). There are buttons for 'Simular', 'Eliminar', and 'Añadir permisos'.

Ilustración 173 - Rol ECS tf

Elastic File System (EFS)

Amazon EFS > Sistemas de archivos > fs-0e723c82e71bbb7c8

efs-tf (fs-0e723c82e71bbb7c8)

General Editar

Modo de rendimiento	Copias de seguridad automáticas
Uso general	⊖ Desactivado
Modo de desempeño	Cifrado
Transmisión por ráfagas	f5be0bcc-1c11-486c-9f81-f6332cb9a94f (aws/elasticfilesystem)
Administración del ciclo de vida	Estado del sistema de archivos
Transición al acceso poco frecuente: Ninguno	⊕ Disponible
Transición fuera del acceso poco frecuente: Ninguno	Nombre de DNS
Zona de disponibilidad	fs-0e723c82e71bbb7c8.efs.us-east-1.amazonaws.com
Estándar	

Tamaño medido | Monitoreo | Etiquetas | Política del sistema de archivos | Puntos de acceso | **Red** | Replicación

Tamaño medido

Tamaño total
302.82 MIB

Ilustración 174 - EFS tf

Destino de montaje EFS

Zona de disponibilidad	ID del destino de montaje	ID de la subred	Estado de destino de montaje	Dirección IP	ID de la interfaz de red	Grupos de seguridad
Estándar	fsmt-0eb4ca7f4263a7e70	subnet-0621d0458d4b48d0f	⊕ Disponible	10.1.30.14	eni-089862e025af4051c	sg-09e8f168584ee6c37 (efs-sg-tf)

Ilustración 175 - Destino montaje tf

Instancias EC2

Instancias (1/17) Información								
		Conectar		Estado de la instancia ▾		Acciones ▾		Lanzar instancias
<input type="text"/> Buscar instancia por atributo o etiqueta (case-sensitive) <input type="button" value="tf"/> <input type="button" value="X"/> <input type="button" value="Quitar los filtros"/>								
	Name	ID de la instancia	Estado de la i...	Tipo de ...	Comprobación ...	Estado de la ...	Zona de dispon...	
<input type="checkbox"/>	ec2-bastion-tf	i-08c98efb946acba1c	En ejecución	t2.micro	2/2 comprobacion	Sin alarmas	+	us-east-1a
<input type="checkbox"/>	ec2-ecs-1-tf	i-058d6d0d62e451253	En ejecución	t2.micro	2/2 comprobacion	Sin alarmas	+	us-east-1a
<input type="checkbox"/>	ec2-front-tf	i-0d172183ef6c75d4b	En ejecución	t2.small	2/2 comprobacion	Sin alarmas	+	us-east-1b
<input type="checkbox"/>	ec2-plots-mongo-tf	i-0d02ada9f8e90bfcf	En ejecución	t2.micro	2/2 comprobacion	Sin alarmas	+	us-east-1a
<input type="checkbox"/>	ec2-statistics-cassandra-tf	i-0675f75605bd9af0	En ejecución	t2.medium	2/2 comprobacion	Sin alarmas	+	us-east-1a
<input type="checkbox"/>	ec2-ecs-2-tf	i-04b069db06979c5bc	En ejecución	t2.micro	2/2 comprobacion	Sin alarmas	+	us-east-1a

Ilustración 176 - Instancias EC2 tf

Instancia: i-08c98efb946acba1c (ec2-bastion-tf)						
Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la instancia i-08c98efb946acba1c (ec2-bastion-tf)	Dirección IPv4 pública 54.208.163.45 dirección abierta	Direcciones IPv4 privadas 10.1.10.145	Dirección IPv6 -	Estado de la Instancia En ejecución	DNS de IPv4 pública ec2-54-208-163-45.compute-1.amazonaws.com dirección abierta	
Tipo de nombre de anfitrión Nombre de IP: ip-10-1-10-145.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-1-10-145.ec2.internal					
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.micro	Direcciones IP elásticas -				
Dirección IP asignada automáticamente 54.208.163.45 [IP pública]	ID de VPC vpc-Off8672d0ccc3cc11 (vpc-tf)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.				
Rol de IAM AmazonS3FullAccess-tf	ID de subred subnet-02f91e2468446f16a (subred-servicios-1a-tf)	Más información				
			Nombre del grupo de Auto Scaling -			

Ilustración 177 - EC2 1 tf

Instancia: i-058d6d0d62e451253 (ec2-ecs-1-tf)						
Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la instancia i-058d6d0d62e451253 (ec2-ecs-1-tf)	Dirección IPv4 pública 54.174.203.244 dirección abierta	Direcciones IPv4 privadas 10.1.10.93	Dirección IPv6 -	Estado de la Instancia En ejecución	DNS de IPv4 pública ec2-54-174-203-244.compute-1.amazonaws.com dirección abierta	
Tipo de nombre de anfitrión Nombre de IP: ip-10-1-10-93.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-1-10-93.ec2.internal					
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.micro	Direcciones IP elásticas -				
Dirección IP asignada automáticamente 54.174.203.244 [IP pública]	ID de VPC vpc-Off8672d0ccc3cc11 (vpc-tf)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.				
Rol de IAM ecsInstanceRole-tf	ID de subred subnet-02f91e2468446f16a (subred-servicios-1a-tf)	Más información				
			Nombre del grupo de Auto Scaling -			

Ilustración 178 - EC2 2 tf

Cloud Computing y migración de una aplicación de riego a AWS

Instancia: i-0d172183ef6c75d4b (ec2-front-tf)		
Detalles	Seguridad	Redes
▼ Resumen de instancia Información		
ID de la instancia i-0d172183ef6c75d4b (ec2-front-tf)	Dirección IPv4 pública 3.95.61.240 dirección abierta	Direcciones IPv4 privadas 10.1.20.21
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública ec2-3-95-61-240.compute-1.amazonaws.com dirección abierta
Tipo de nombre de anfitrión Nombre de IP: ip-10-1-20-21.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-1-20-21.ec2.internal	
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.small	Direcciones IP elásticas -
Dirección IP asignada automáticamente 3.95.61.240 [IP pública]	ID de VPC vpc-0ff8672d0ccc3cc11 (vpc-tf)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.
Rol de IAM AmazonS3FullAccess-tf	ID de subred subnet-0e15c3d7d0e1fc11b (subred-front-1b-tf)	Más información
		Nombre del grupo de Auto Scaling -

Ilustración 179 - EC2 3 tf

Instancia: i-0d02ada9f8e90bfcf (ec2-plots-mongo-tf)		
Detalles	Seguridad	Redes
▼ Resumen de instancia Información		
ID de la instancia i-0d02ada9f8e90bfcf (ec2-plots-mongo-tf)	Dirección IPv4 pública -	Direcciones IPv4 privadas 10.1.30.253
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública -
Tipo de nombre de anfitrión Nombre de IP: ip-10-1-30-253.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-1-30-253.ec2.internal	
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.micro	Direcciones IP elásticas -
Dirección IP asignada automáticamente -	ID de VPC vpc-0ff8672d0ccc3cc11 (vpc-tf)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.
Rol de IAM AmazonS3FullAccess-tf	ID de subred subnet-0621d0458d4b48d0f (subred-databases-1a-tf)	Más información
		Nombre del grupo de Auto Scaling -

Ilustración 180 - EC2 4 tf

Instancia: i-0675f75605bd9af0 (ec2-statistics-cassandra-tf)		
Detalles	Seguridad	Redes
▼ Resumen de instancia Información		
ID de la instancia i-0675f75605bd9af0 (ec2-statistics-cassandra-tf)	Dirección IPv4 pública -	Direcciones IPv4 privadas 10.1.30.59
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública -
Tipo de nombre de anfitrión Nombre de IP: ip-10-1-30-59.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-10-1-30-59.ec2.internal	
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.medium	Direcciones IP elásticas -
Dirección IP asignada automáticamente -	ID de VPC vpc-0ff8672d0ccc3cc11 (vpc-tf)	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.
Rol de IAM AmazonS3FullAccess-tf	ID de subred subnet-0621d0458d4b48d0f (subred-databases-1a-tf)	Más información
		Nombre del grupo de Auto Scaling -

Ilustración 181 - EC2 5 tf

Cloud Computing y migración de una aplicación de riego a AWS

Instancia: i-04b069db06979c5bc (ec2-ecs-2-tf)

Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la Instancia i-04b069db06979c5bc (ec2-ecs-2-tf)	Dirección IPv4 pública 34.203.244.244 dirección abierta	Estado de la instancia En ejecución	Direcciones IPv4 privadas 10.1.10.72			
Dirección IPv6 -	Nombre DNS de IP privada (solo IPv4) ip-10-1-10-72.ec2.internal	Tipo de instancia t2.micro	DNS de IPv4 pública ec2-34-203-244-244.compute-1.amazonaws.com dirección abierta			
Tipo de nombre de anfitrón Nombre de IP: ip-10-1-10-72.ec2.internal	ID de VPC vpc-Off8672d0ccc3cc11 (vpc-tf)	ID de subred subnet-02f91e2468446f16a (subnet-servicios-1a-tf)	Direcciones IP elásticas -			
Responder al nombre DNS de recurso privado -			Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.			
Dirección IP asignada automáticamente 34.203.244.244 [IP pública]			Más información			
Rol de IAM ecsInstanceRole-tf			Nombre del grupo de Auto Scaling -			

Ilustración 182 - EC2 6 tf

RDS

RDS > Databases > rds-authentication-mysql-tf

rds-authentication-mysql-tf

				Modificar	Acciones ▾
Resumen					
Identificador de base de datos rds-authentication-mysql-tf	CPU <div style="width: 1.96%;">1.96%</div>	Estado Disponible	Clase db.t3.micro		
Rol Instancia	Actividad actual <div style="width: 0%;">0 Conexiones</div>	Motor MySQL Community	Región y AZ us-east-1a		
Conectividad y seguridad Supervisión Registros y eventos Configuración Mantenimiento y copias de seguridad Etiquetas					
Conectividad y seguridad					
Punto de enlace y puerto	Redes	Seguridad			
Punto de enlace rds-authentication-mysql-tf.cvt9cc7dxazz.us-east-1.rds.amazonaws.com	Zona de disponibilidad us-east-1a	Grupos de seguridad de la VPC databases-sg-tf (sg-0c477e771e92dcb2b)			
Puerto 3306	VPC vpc-tf (vpc-Off8672d0ccc3cc11)	Activo			
	Grupo de subredes databases-grupo-subnet-tf	Accesible públicamente No			
	Subredes subnet-030c492f2d77888ef subnet-0621d0458d4b48d0f	Entidad de certificación Información rds-ca-2019			

Ilustración 183 - RDS mysql tf

Ilustración 184 - RDS postgres tf

Grupo de subredes

Ilustración 185 - Grupo subred tf

Clúster ECS

ARN del clúster: arn:aws:ecs:us-east-1:690031176274:cluster/cluster-app-tf
Estado: ACTIVE
Instancias de contenedor registradas: 2
Recuento de tareas pendientes: 0 Fargate, 0 EC2, 0 External
Recuento de tareas en ejecución: 0 Fargate, 0 EC2, 0 External
Recuento de servicios activos: 0 Fargate, 0 EC2, 0 External
Recuento de servicios de vaciado: 0 Fargate, 0 EC2, 0 External

	Servicios	Tareas	Instancias de ECS	Métricas	Tareas programadas	Etiquetas	Proveedores de capacidad		
Una instancia de Amazon ECS es una instancia externa registrada mediante ECS Anywhere o una instancia de Amazon EC2. Para registrar una instancia externa, seleccione Registrar instancias externas y siga los pasos. Más información Para registrar una instancia de Amazon EC2, puede utilizar la consola de Amazon EC2. Más información									
Registrar instancias externas		Acciones Última actualización realizada a las junio 12, p. m. 7:16:01 p. m. (hace 0 minutos)							
Estado: ALL ACTIVE DRAINING < 1-2 > Tamaño de la página 50									
Filtrar por atributos (haga clic o pulse la flecha hacia abajo para ver las opciones de filtro)									
	Instancia de cont...	Instancia de...	Zona de dis...	Instanc...	Agente conect...	Estado	Recuent...	CPU disp...	Memoria disp...
<input type="checkbox"/>	4121489f23e047e...	i-058d6d0d6...	us-east-1a	false	true	ACTIVE	0	1024	970
<input type="checkbox"/>	5e2d0c0e7e084e7...	i-04b069db0...	us-east-1a	false	true	ACTIVE	0	1024	970

Ilustración 186 - ECS tf

Tareas

Estado: **ACTIVE** INACTIVE 4 seleccionadas < 1-8 > Tamaño de la página 50

	Definición de tarea	Estado de revisión más reciente
<input type="checkbox"/>	tarea-authentication	ACTIVE
<input checked="" type="checkbox"/>	tarea-authentication-tf	ACTIVE
<input type="checkbox"/>	tarea-plots	ACTIVE
<input checked="" type="checkbox"/>	tarea-plots-tf	ACTIVE
<input type="checkbox"/>	tarea-sensor	ACTIVE
<input checked="" type="checkbox"/>	tarea-sensor-tf	ACTIVE
<input type="checkbox"/>	tarea-statistics	ACTIVE
<input checked="" type="checkbox"/>	tarea-statistics-tf	ACTIVE

Ilustración 187 - Tareas tf

Balancedor de carga ALB

Details			
Load balancer type Application	Status Active	VPC vpc-0ff8672d0ccc3cc11	IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-02f91e2468446f16a us-east-1a (use1-az2) subnet-0e15c3d7d0e1fc11b us-east-1b (use1-az4)	Date created June 12, 2023, 17:56 (UTC+02:00)
Load balancer ARN <code>arn:aws:elasticloadbalancing:us-east-1:690031176274:loadbalancer/app/ALB-tf/626d7ae4914db17e</code>	DNS name <code>ALB-tf-1281770203.us-east-1.elb.amazonaws.com (A Record)</code>		

Ilustración 188 - ALB tf

Listeners ALB

Listeners (5)					
A listener checks for connection requests on its port and protocol. Traffic received by the listener is routed according to its rules.					
<input type="checkbox"/>	Protocol:Port	Default action	Rules	ARN	Security policy
<input type="checkbox"/>	HTTP:4200	Forward to target group • <code>front-target-tf</code> : 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable
<input type="checkbox"/>	HTTP:8080	Forward to target group • <code>plots-target-tf</code> : 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable
<input type="checkbox"/>	HTTP:8081	Forward to target group • <code>sensor-target-tf</code> : 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable
<input type="checkbox"/>	HTTP:8082	Forward to target group • <code>statistics-target-tf</code> : 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable
<input type="checkbox"/>	HTTP:8999	Forward to target group • <code>authentication-target-tf</code> : 1 (100%) • Group-level stickiness: Off	1 rule	<input type="checkbox"/> ARN	Not applicable

Ilustración 189 - Listeners ALB tf

Grupos de destino ALB

Target groups (5) Info								
		Actions			Create target group			
<input type="text"/> Find resources by attribute or tag								
<input checked="" type="checkbox"/>	tf	X	Clear filters					
	Name	ARN	Port	Protocol	Target type			
<input type="checkbox"/>	front-target-tf	arn:aws:elasticloadbalanci...	4200	HTTP	Instance	Edit		
<input type="checkbox"/>	plots-target-tf	arn:aws:elasticloadbalanci...	8080	HTTP	Instance	Edit		
<input type="checkbox"/>	sensor-target-tf	arn:aws:elasticloadbalanci...	8081	HTTP	Instance	Edit		
<input type="checkbox"/>	statistics-target-tf	arn:aws:elasticloadbalanci...	8082	HTTP	Instance	Edit		
<input type="checkbox"/>	authentication-target-tf	arn:aws:elasticloadbalanci...	8999	HTTP	Instance	Edit		

Ilustración 190 - Grupos destino tf

Target group: front-target-tf										
Details		Targets		Monitoring		Health checks				
Edit										
Registered targets (1)										
<input type="button"/> C Deregister Register targets										
<input type="text"/> Filter resources by property or value										
<input type="checkbox"/> Instance ID ▾ Name ▾ Port ▾ Zone ▾ Health s... ▾ Health status details										
<input type="checkbox"/> i-0d172183... ec2-front-tf 4200 us-east-1b healthy										

Ilustración 191 - Grupo 1 tf (1/2)

Target group: front-target-tf										
Details		Targets		Monitoring		Health checks				
Edit										
Health check settings										
<table border="1"> <tr> <td>Protocol HTTP</td> <td>Path /</td> <td>Port 4200</td> <td>Healthy threshold 2 consecutive health check successes</td> </tr> </table>							Protocol HTTP	Path /	Port 4200	Healthy threshold 2 consecutive health check successes
Protocol HTTP	Path /	Port 4200	Healthy threshold 2 consecutive health check successes							

Ilustración 192 - Grupo 1 tf (2/2)

Target group: plots-target-tf										
Details		Targets		Monitoring		Health checks				
Edit										
Registered targets (2)										
<input type="button"/> C Deregister Register targets										
<input type="text"/> Filter resources by property or value										
<input type="checkbox"/> Instance ID ▾ Name ▾ Port ▾ Zone ▾ Health s... ▾ Health status details										
<input type="checkbox"/> i-058d6d0d... ec2-ecs-1-tf 8080 us-east-1a ☒ unhealthy Health checks failed										
<input type="checkbox"/> i-04b069db... ec2-ecs-2-tf 8080 us-east-1a ☒ unhealthy Health checks failed										

Ilustración 193 - Grupo 2 tf (1/2)

Cloud Computing y migración de una aplicación de riego a AWS

Target group: plots-target-tf

Health check settings

Protocol HTTP	Path /actuator/health	Port 8080	Healthy threshold 2 consecutive health check successes
------------------	--------------------------	--------------	---

Ilustración 194 - Grupo 2 tf (2/2)

Target group: sensor-target-tf

Registered targets (2)

<input type="checkbox"/> Instance ID	Name	Port	Zone	Health s...	Health status details
i-058d6d0d...	ec2-ecs-1-tf	8081	us-east-1a	☒ unhealthy	Health checks failed
i-04b069db...	ec2-ecs-2-tf	8081	us-east-1a	☒ unhealthy	Health checks failed

Ilustración 195 - Grupo 3 tf (1/2)

Target group: sensor-target-tf

Health check settings

Protocol HTTP	Path /actuator/health	Port 8081	Healthy threshold 2 consecutive health check successes
------------------	--------------------------	--------------	---

Ilustración 196 - Grupo 3 tf (2/2)

Target group: statistics-target-tf

Registered targets (2)

<input type="checkbox"/> Instance ID	Name	Port	Zone	Health s...	Health status details
i-04b069db...	ec2-ecs-2-tf	8082	us-east-1a	☒ unhealthy	Health checks failed
i-058d6d0d...	ec2-ecs-1-tf	8082	us-east-1a	☒ unhealthy	Health checks failed

Ilustración 197 - Grupo 4 tf (1/2)

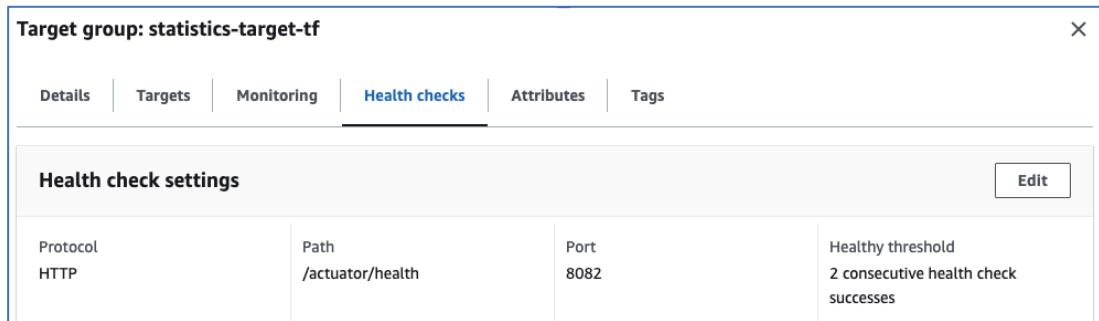


Ilustración 198 - Grupo 4 tf (2/2)

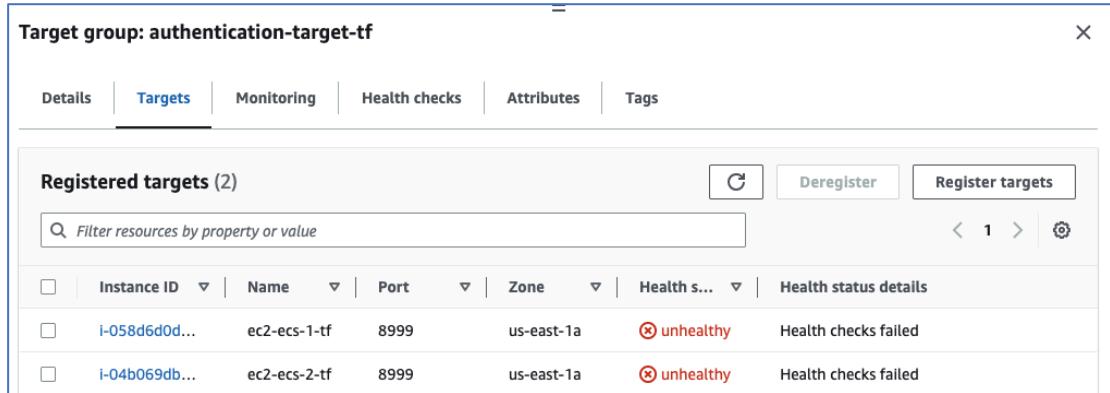


Ilustración 199 - Grupo 5 tf (1/2)

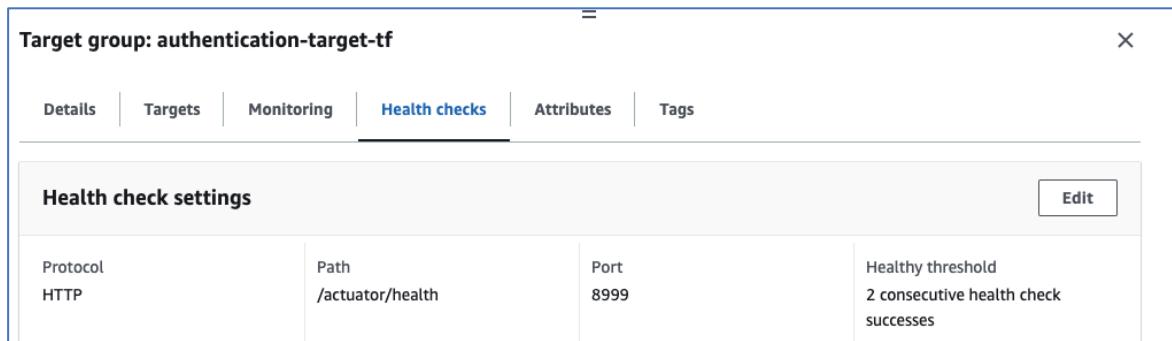


Ilustración 200 - Grupo 5 tf (2/2)

8.3. Comprobación de la funcionalidad de las APIs desplegadas

En este apartado se probarán los servicios API REST de los microservicios de la aplicación para determinar cuáles son utilizables:

8.3.1. API del servicio Sensor

Su mapeado corresponde con: <http://<host>:8081/sensor>

addSensor

Operación POST que inserta un sensor especificado en el cuerpo de la petición utilizando la URL por defecto "/sensor" (Ilustración 201).

```
addSensor (@RequestBody SensorRequest sensorRequest)
```

- <http://localhost:8081/sensor>

The screenshot shows a Postman interface with a POST request to `http://localhost:8081/sensor`. The request body is a JSON object:

```

1
2   ...
3     "owner": "jesus",
4     "catastralReference": "10022A010000380000WZ",
5     "latitude": 10.12,
6     "longitude": 10.5,
7     "type": "precipitaciones"

```

The response status is 204 No Content, time 234 ms, size 201 B.

Ilustración 201 - sensor addSensor

El resultado de la inserción del sensor ha sido exitoso.

getSensorList

Operación GET que devuelve todos los sensores por propietario y por referencia catastral de forma opcional (Ilustración 202 y 203).

```
getSensorList (@RequestParam String owner, @RequestParam(required = false) String catastralReference)
```

- <http://localhost:8081/sensor?owner=pepe>

The screenshot shows a Postman interface with a GET request to `http://localhost:8081/sensor?owner=pepe`. The query parameter `owner` is set to `pepe`. The response is a JSON array of sensor objects:

```

1 [
2   {
3     "sensorId": "b05c0778-6fca-11ec-90d6-0242ac120003",
4     "catastralReference": "10022A010000380000WZ",
5     "latitude": 12.12,
6     "longitude": 14.5,
7     "type": "temperature"
8   }

```

The response status is 200 OK, time 27 ms, size 405 B.

Ilustración 202 - sensor getSensorList 1

Se han recuperado los sensores por propietario exitosamente.

- <http://localhost:8081/sensor?owner=pepe&catastralReference=10022A010000380000WZ>

KEY	VALUE	DESCRIPTION	Bulk Edit
owner	pepe		
catalstralReference	10022A010000380000WZ		

```

1 [
2   {
3     "sensorId": "b05c0778-6fca-11ec-90d6-0242ac120003",
4     "catalstralReference": "10022A010000380000WZ",
5     "latitude": 12.12,
6     "longitude": 14.5,
7     "type": "temperature"
8   }
]
  
```

All Logs Clear 200

Ilustración 203 - sensor getSensorList 2

Se han recuperado los sensores por propietario y referencia catastral exitosamente.

getSensorBySensorId

Operación GET que devuelve el sensor con el atributo “id” especificado en la URL mediante la variable “/{sensorId}” (Ilustración 204).

```
getSensorBySensorId(@PathVariable String sensorId)
```

- <http://localhost:8081/sensor/b05c0778-6fca-11ec-90d6-0242ac120003>

KEY	VALUE	DESCRIPTION	Bulk Edit
sensorId	b05c0778-6fca-11ec-90d6-0242ac120003		
catalstralReference	10022A010000380000WZ		
latitude	12.12		
longitude	14.5		
type	temperature		

All Logs Clear 200

Ilustración 204 - sensor getSensorBySensorId

Se ha recuperado el sensor por su “id” exitosamente.

8.3.2. API del servicio Plots

Su mapeado corresponde con: **http://<host>:8080/plots**

getPlotsByOwner

Operación GET que devuelve las parcelas del propietario especificado en la URL mediante la variable “/owner/{owner}”. Solo funcionará con el propietario “pepe” ya que el código se encuentra implementado para crear las parcelas y operar con ese valor (Ilustración 205).

```
getPlotsByOwner(@PathVariable String owner)
```

- <http://localhost:8080/plots/owner/pepe>

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/plots/owner/pepe
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** None
- Body:** PRETTY (selected), Raw, Preview, Visualize, JSON (selected), `1 2 3 4 5 6 7 8 9 10 11 { "id": "611d1ad5dfb24e64eb994bce", "catastralReference": "10022AA010000380000WZ", "geoData": { "type": "FeatureCollection", "features": [{ "type": "Feature", "geometry": { "type": "Polygon",`
- Headers:** Status: 200 OK, Time: 102 ms, Size: 953 B, Save Response
- Console:** GET http://localhost:8080/plots/owner/pepe

Ilustración 205 - plots getPlotsByOwner

Se han recuperado las parcelas por propietario exitosamente.

getPlotsByCatastralReference

Operación GET que devuelve la parcela a partir de su referencia catastral especificada en la URL mediante: “/{catastralReference}” (Ilustración 206).

```
getPlotsByCatastralReference(@PathVariable String catastralReference)
```

- <http://localhost:8080/plots/10022A010000380000WZ>

```

1
2 "id": "611d1ad5dfb24e64eb994bce",
3 "cadastralReference": "10022A010000380000WZ",
4 "geoData": [
5     "type": "FeatureCollection",
6     "features": [
7         {
8             "type": "Feature",
9             "geometry": {
10                 "type": "Polygon",
11             }
12         }
13     ]
14 ]
  
```

Ilustración 206 - plots getPlotsByCadastralReference

Se ha recuperado la parcela por su referencia catastral exitosamente.

addPlotByReference

Operación POST que asigna a la parcela especificada el propietario “pepe”. Se especificará en la URL la referencia catastral de la parcela mediante la variable: “/reference/{reference}” (Ilustración 207).

```
addPlotByReference (@PathVariable String reference)
```

- <http://localhost:8080/plots/reference/8839103TK7484S0001GK>

KEY	VALUE	DESCRIPTION	Bulk Edit
1			

Ilustración 207 - plots addPlotByReference

Se ha añadido la parcela por su referencia catastral exitosamente.

La referencia deberá existir y estar registrada en la “Sede Electrónica del Catastro”. Se ha obtenido la referencia catastral en este [enlace del catastro](#).

checkOwnerRequest

Operación POST que procesa un objeto de tipo “CheckOwnerRequest” especificado en el cuerpo de la petición con los valores de un propietario y una referencia catastral. Verifica si el nombre del propietario es el mismo que el propietario de la parcela (Ilustración 208).

```
checkOwnerRequest (@RequestBody CheckOwnerRequest checkOwnerRequest)
```

- <http://localhost:8080/plots/check>

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/plots/check`. The request body is a JSON object:

```
1
2   ...
3     "owner": "pepe",
4     "catalsticalReference": "10022A010000380000WZ"
```

The response status is 204 No Content.

Ilustración 208 - plots checkOwnerRequest

El objeto especificado en el cuerpo de la petición verifica que “pepe” se corresponde como propietario de la parcela identificada por la referencia catastral “10022A010000380000WZ”.

8.3.3. API del servicio Statistics

Su mapeado corresponde con: <http://<host>:8082/statistics>

Este método no funciona correctamente y la API no posee suficientes operaciones para poder probarla. En el Front, esta funcionalidad para mostrar características es emulada mostrando gráficos predeterminados.

Esta es su única operación, la cual trabaja con un websocket desconocido que levanta en algún momento de su ejecución (Ilustración 209):

```
@MessageMapping("/statistics/{sensorId}")
public void send(@DestinationVariable String sensorId) throws Exception {
    scheduler.scheduleAtFixedRate(() -> {
        template.convertAndSend( destination: "/topic/statistics/" + sensorId, s
    }, initialDelay: 0, period: 5, SECONDS);
}
```

Ilustración 209 - statistics operación

8.3.4. API del servicio Authentication

Su mapeado corresponde con: **http://<host>:8999/auth**

createAuthenticationToken

Operación POST que crea un token de autentificación para un usuario especificado en el cuerpo de la petición en forma de objeto “JwtRequest”. Utilizará la URL: “/register” (Ilustración 210).

```
createAuthenticationToken (@RequestBody JwtRequest authenticationRequest)
```

- <http://localhost:8999/auth/register>

The screenshot shows a Postman interface with a POST request to `http://localhost:8999/auth/register`. The request body is set to `JSON` and contains the following JSON:

```
1 {
2   ...
3   "username": "juanjo",
4   ...
5   "password": "Juanjojuanjo123"
6 }
```

The response tab shows a status of `401 Unauthorized` with a timestamp of `2023-05-28T12:28:37.935+00:00`. The error message is `"error": "Unauthorized"`.

Ilustración 210 - authentication *createAuthenticationToken*

El resultado no ha sido exitoso por un problema de identificación. El servicio funciona, pero la operación no se ha ejecutado con éxito debido a la codificación de algún componente de la aplicación de Authentication.

RequestBody

Operación POST que registrar a un usuario especificado en el cuerpo de la petición. Utilizará la URL “/register” (Ilustración 211)

```
saveUser (@RequestBody UserRequest user)
```

- <http://localhost:8999/auth/authenticate>

The screenshot shows a Postman interface with a POST request to <http://localhost:8999/auth/authenticate>. The request body is set to 'JSON' and contains the following JSON:

```
1 {
2     "username": "jesus",
3     "password": "Jesusjesus123"
4 }
```

The response status is 401 Unauthorized, with a timestamp of 2023-05-28T12:27:36.049+00:00, and the message "Unauthorized".

Ilustración 211 - authentication RequestBody

El resultado no ha sido exitoso por un problema de identificación. Además, esta operación depende del token generado por la primera operación “createAuthenticationToken” poder ser ejecutada con éxito.

9. Lecciones aprendidas y errores

Este apartado se centrará en exponer los aprendizajes y errores cometidos durante la elaboración del proyecto. Primero se tratarán los aspectos positivos y negativos de la experiencia de trabajo a nivel organizativo, y después se realizará una lista con algunas recomendaciones para la migración y despliegue de aplicaciones basadas en microservicios según la experiencia adquirida durante la fase de desarrollo del proyecto.

En primer lugar, este proyecto de fin de grado ha supuesto un reto a niveles de gestión, planificación y desarrollo del mismo. Ha sido una prueba de madurez en el que se han puesto en práctica gran parte de los conocimientos adquiridos en el grado, pero con un nivel de complejidad superior debido al tamaño del proyecto. Se ha mejorado en la realización de tareas como la planificación a la hora de trabajar, la forma de elaboración de las documentaciones, aprender en profundidad nuevas tecnologías y a trabajar en “sprints” de tiempo, estableciendo objetivos a cumplir de cara a las revisiones con el tutor del proyecto. Algunos errores cometidos durante este proceso han sido no utilizar debidamente herramientas como “Trello” para establecer un seguimiento de tareas eficaz a lo largo del tiempo y no ser constante con la elaboración de la documentación, dando lugar a picos de trabajo bastante intensos en intervalos de tiempo muy cortos. Por último, se ha aprendido a trabajar con código y proyectos software ajenos puesto que, la aplicación inicial utilizada como base para la realización del proyecto, no disponía de documentación y su desarrollo no estaba completado.

Para finalizar, estas son algunas recomendaciones o buenas prácticas aprendidas trabajando con aplicaciones web basadas en microservicios utilizando contenedores Docker y su despliegue tanto en un entorno local como en una infraestructura en la nube de AWS:

- Es necesario entender cómo interactúan entre sí los microservicios y en qué ficheros se configuran sus conexiones y propiedades. Por ejemplo, los proyectos de Spring Boot (framework que utilizan los servicios API REST del proyecto) utilizan un fichero “application.properties” donde se definen las URL de conexión con las bases de datos, entre otras opciones de configuración como el puerto de despliegue del servicio. Muchos de los errores producidos durante la ejecución de la aplicación son provocados por una mala gestión de las conexiones entre los microservicios.
- Si una aplicación se ejecuta tanto en un entorno local como en un entorno en la nube, es necesario gestionar las direcciones de conexión

de los servicios. Cuando la aplicación despliega todos sus microservicios dentro de una misma máquina, no utilizará las mismas direcciones o “hostnames” que si estuvieran desplegados en diversas máquinas dentro de una infraestructura en la nube. La solución implementada durante la fase de desarrollo para solucionar este problema es la utilización de ficheros “.env” con una lista de variables de entorno para guardar el valor de las direcciones que utilizará la aplicación. De esta forma se utilizarán dos ficheros, uno para cada tipo de entorno.

- Una de las ventajas de trabajar con microservicios es que cada uno realiza su función de forma independiente, por lo que las modificaciones realizadas sobre alguno de ellos no afectarán de forma conjunta a los demás servicios. Esto permite refactorizar la aplicación progresivamente sin tener que realizar modificaciones de código que afecten a toda la aplicación al mismo tiempo.
- Para la monitorización de los servicios contenedores, Docker permite visualizar los ficheros “logs” generados tras el lanzamiento de los contenedores para comprobar su despliegue. Estos ficheros son muy útiles para ver los detalles cuando se producen errores o simplemente para confirmar que el servicio se ejecuta correctamente. De cara a monitorizar la infraestructura de AWS, las instancias de máquinas virtuales EC2 también poseen un sistema de visualización de “logs” conocido como registro de sistema para comprobar el arranque de la máquina.

10. Conclusiones

Se ha logrado profundizar en el mundo de la computación en la nube mediante el despliegue de una infraestructura dentro del proveedor de AWS para dar solución a un escenario de migración, donde se buscaba migrar una aplicación web basada en microservicios desde un entorno local hacia un entorno en la nube. La infraestructura podrá utilizarse para desplegar otras aplicaciones con una estructura similar de microservicios.

Antes de realizar el proceso de migración, se ha refactorizado correctamente la aplicación web inicial para desplegar los microservicios a través de contenedores, pudiendo ser desplegados tanto de forma local como en la infraestructura desarrollada. Finalmente, la infraestructura podrá ser desplegada de forma automatizada a través de ficheros Terraform.

Este proceso de migración representa una situación real a la que se enfrentan hoy en día las empresas tecnológicas dedicadas al desarrollo de aplicaciones, donde se ha adquirido unos conocimientos a cerca de tecnologías actuales que se podrán utilizar en el mundo laboral.

11. Trabajos futuros

Algunas de las ampliaciones que podrían realizar sobre el proyecto son:

- Incorporación de un sistema de recogida de datos en tiempo real a través de los sensores. Esta parte se descartó al inicio por la complejidad añadida que provocaba, pero aumentaría la funcionalidad de la aplicación transformándose en un sistema complejo de control de riego.
- Aplicación de procesos CI/CD para la automatización de la integración y despliegue continuo de cara a seguir desarrollando la aplicación.
- Probar el despliegue de la infraestructura construida con otros tipos de aplicaciones basadas en microservicios para comprobar la adaptabilidad y robustez de la implementación de cara a refinar y construir una infraestructura profesional.
- Implementación de un sistema para importar y exportar los datos de los servicios de la aplicación de forma directa entre los entornos locales y en la nube.

12. Manual de instalación y de usuario

El proyecto se ha realizado en un equipo con un sistema operativo Mac, más concretamente macOS Catalina (versión 10.15.7).

Entorno de trabajo utilizado en el equipo:

- **Java 11** (versión 11.0.13)
- **Gradle** (versión 7.4.2)
- **Node** (versión 16.14)
- **Npm** (versión 8.5.0)
- **Docker** (versión 16.14)
- **Terraform** (1.4.6)
- **Git** (versión 2.33.0)
- **IDE IntelliJ IDEA** (2022.1)

12.1. Creación de la cuenta de AWS

Para comenzar con la creación de la cuenta habrá que dirigirse a la página principal de AWS y abrir la consola de administración (Ilustración 212): <https://aws.amazon.com/es/>

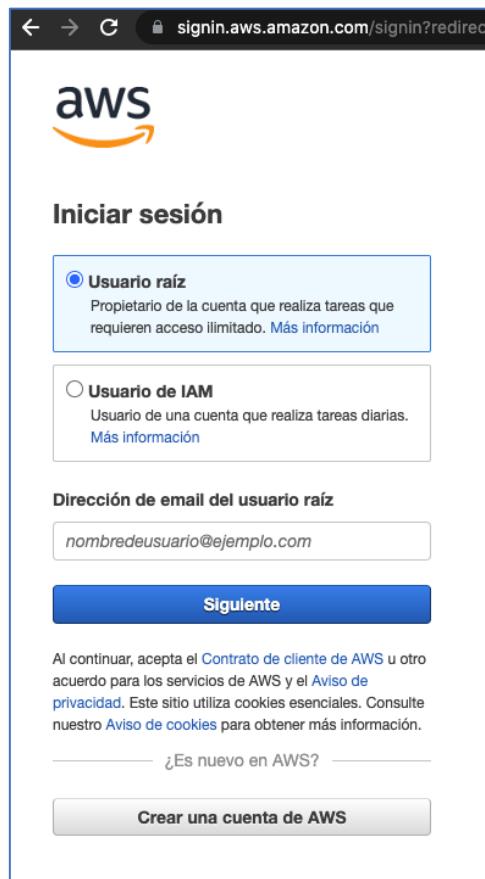


Ilustración 212 - Inicio registro aws

Se seleccionará la opción de crear una cuenta nueva y se seguirán los siguientes pasos (Ilustración 213):



Ilustración 213 - Registro en aws

1. Introducir el correo electrónico y el código de verificación enviado.
2. Crear una contraseña válida segura.
3. Asegurar la información de contacto donde se proporcionará el nombre completo, teléfono, país...
4. Agregar un método de pago. Para la utilización de la cuenta de AWS se necesita proporcionar la información de la tarjeta de crédito. Solo se cobrará un euro inicial por la creación, no se realizará ningún cobro más. Además, trabajando con la capa gratuita no se gastará dinero hasta superar un límite de uso.
5. Verificación del número de teléfono.
6. Escoger un plan de soporte. Se escogerá el soporte básico gratis.

Cualquier duda que pueda surgir, consultar el tutorial sobre creación de cuentas oficial de AWS: <https://repost.aws/es/knowledge-center/create-and-activate-aws-account>

12.2. Instalación Terraform

En la página principal de instalación se seleccionará el método compatible con el equipo de trabajo (Ilustración 214):

<https://developer.hashicorp.com/terraform/downloads>

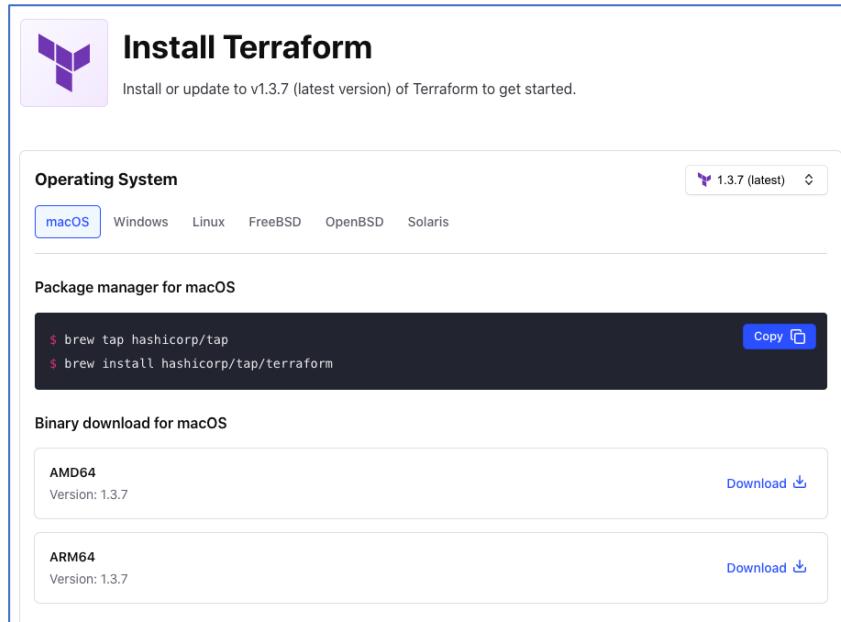


Ilustración 214 - Instalación Terraform

En este caso, se ha seleccionado la opción de instalación mediante el gestor de paquetes “brew install” en macOS.

Una vez instalado, se podrá consultar la versión mediante el comando “terraform -v” (Ilustración 215).

A screenshot of a macOS terminal window. The title bar says "jesusbarquerocuadrado — -bash — 95x40". The command entered is "terraform -v". The output shows "Terraform v1.3.7" and "on darwin_amd64".

Ilustración 215 - Terraform version

12.2.1. Creación de un usuario con credenciales AWS para Terraform

Una vez dentro de la cuenta de la consola de AWS, habrá que dirigirse al servicio de IAM donde se gestionan los usuarios y permisos (Ilustración 216).

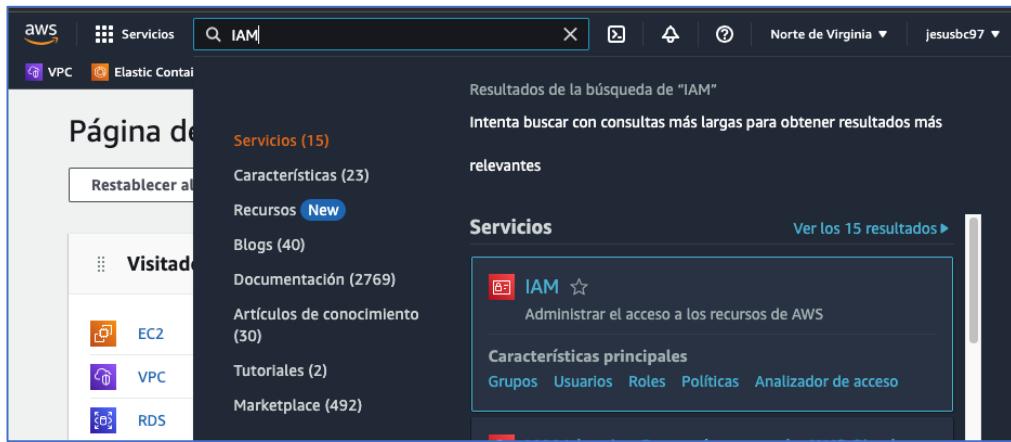


Ilustración 216 - IAM aws

Dentro del servicio IAM, navegar hasta la pestaña de “Usuarios” y comenzar la creación del usuario (Ilustración 217).

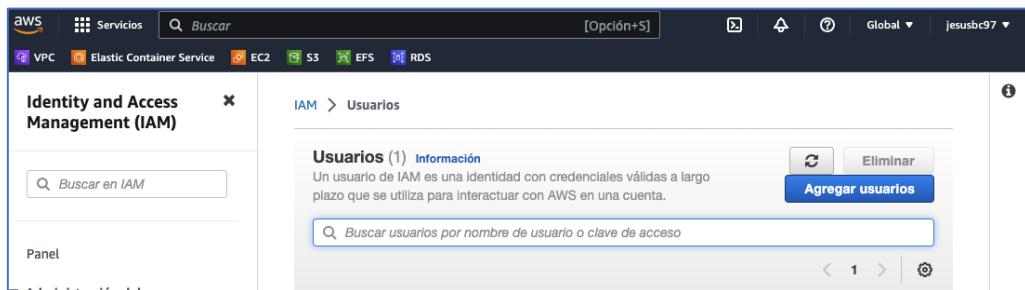


Ilustración 217 - Usuario

Se dará nombre al usuario, en este caso será “terraform” para una mejor identificación, y se le dará un tipo de acceso mediante claves acceso (Ilustración 218).

 A screenshot of the "Add user" wizard, step 1: Set user details. The title is "Add user". It shows a progress bar with step 1 highlighted. The "User name*" field is filled with "terraform". Below it is a link "+ Add another user". The next section, "Select AWS access type", asks how users will access AWS. It offers two options: "Access key - Programmatic access" (selected) and "Password - AWS Management Console access". Descriptions for each option are provided.

Ilustración 218 - Nombre usuario y clave

A continuación, se seleccionará la opción de adjuntar una política directamente al usuario para darle los permisos necesarios de administrador. De esta forma el usuario podrá utilizar la cuenta de AWS y realizar todo tipo

de operaciones. La política que habrá que escoger es “AdministratorAccess” (Ilustración 219).

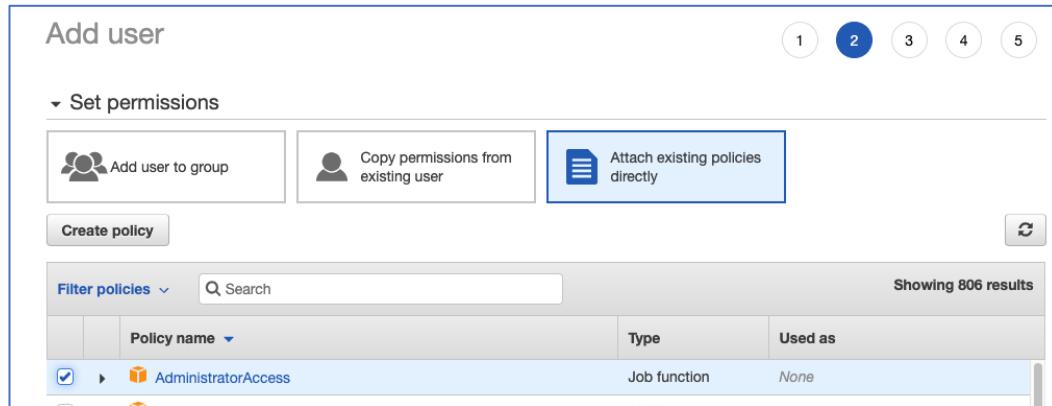


Ilustración 219 - Permisos administrador

Se seguirán los demás pasos hasta finalizar la creación del usuario. Ahora habrá que guardar la información de clave de acceso proporcionada, tanto el ID como la contraseña para poder utilizarlos en los ficheros Terraform (Ilustración 220).

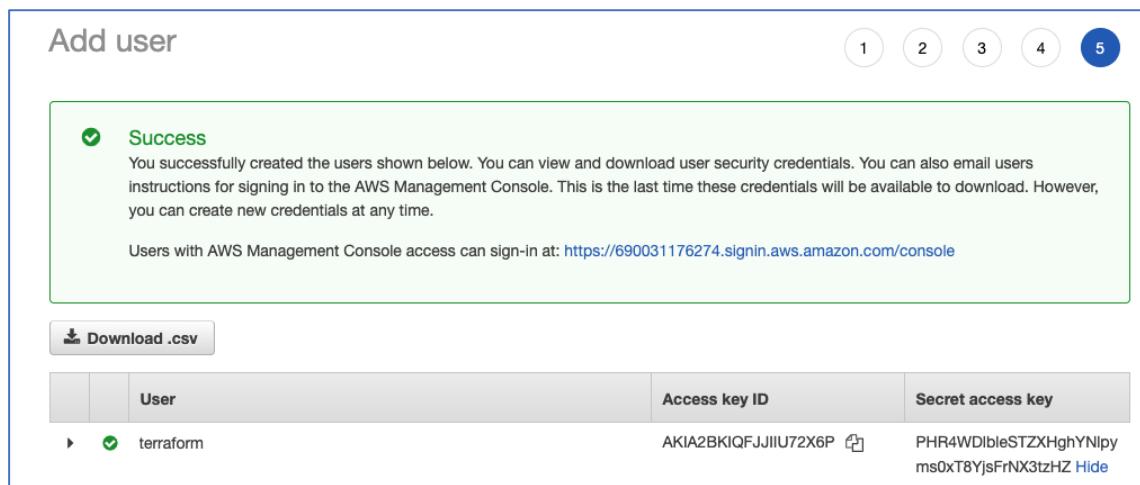


Ilustración 220 - Credenciales Terraform AWS

12.3. Instalación de Docker

Simplemente habrá que dirigirse a la página principal de Docker y seleccionar el instalador adecuado (Ilustración 221).

<https://www.docker.com/get-started/>

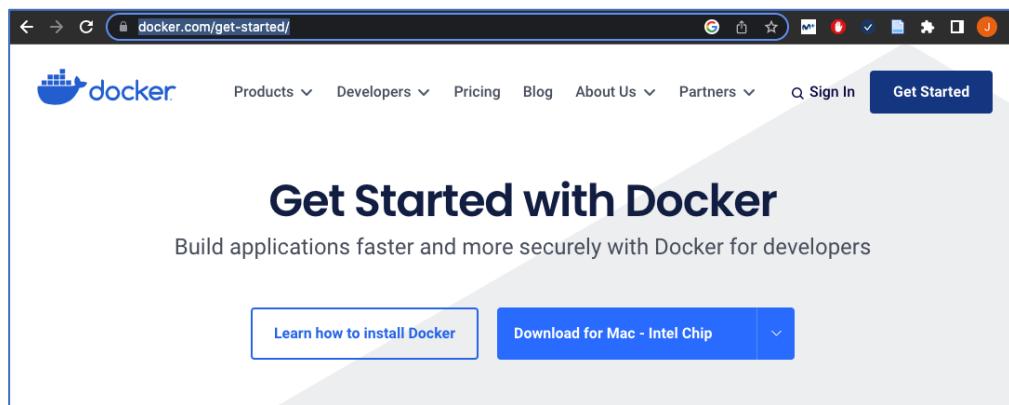


Ilustración 221 - Instalación Docker

12.4. Creación de cuenta en DockerHub

En la página de registro, especificar el nombre usuario, email y contraseña. Después solo habrá que verificar la cuenta a través del correo electrónico (Ilustración 222).

<https://hub.docker.com/signup>

A screenshot of the DockerHub sign-up page at https://hub.docker.com/signup. The page has a light gray header with the Docker logo. The main content is a white form titled 'Create a Docker Account.' It includes fields for 'Username', 'Email', and 'Password' (with a visibility icon). There are two checkboxes: one for receiving updates and another for agreeing to the terms and conditions. Below these is a reCAPTCHA field with the text 'No soy un robot' and a 'Sign Up' button.

Ilustración 222 - Cuenta Docker-Hub

13. Bibliografía

- AMAZON WEB SERVICES, INC., 2012a. Bases de datos no relacionales | Bases de datos de gráficos | AWS. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://aws.amazon.com/es/nosql/>.
- AMAZON WEB SERVICES, INC., 2012b. What is Amazon EC2? - Amazon Elastic Compute Cloud. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>.
- AMAZON WEB SERVICES, INC., 2012c. What is Amazon S3? - Amazon Simple Storage Service. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/AmazonS3/latest/userguide>Welcome.html>.
- AMAZON WEB SERVICES, INC., 2013. Precio de los servicios de la nube | AWS. [en línea]. [consulta: 9 junio 2023]. Disponible en: <https://aws.amazon.com/es/pricing/>.
- AMAZON WEB SERVICES, INC., 2014a. AWS service endpoints - AWS General Reference. [en línea]. [consulta: 25 junio 2023]. Disponible en: <https://docs.aws.amazon.com/general/latest/gr/rande.html>.
- AMAZON WEB SERVICES, INC., 2014b. Infraestructura global. [en línea]. [consulta: 9 junio 2023]. Disponible en: <https://aws.amazon.com/es/about-aws/global-infrastructure/>.
- AMAZON WEB SERVICES, INC., 2014c. What is Amazon Relational Database Service (Amazon RDS)? - Amazon Relational Database Service. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide>Welcome.html>.
- AMAZON WEB SERVICES, INC., 2014d. What is IAM? - AWS Identity and Access Management. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>.
- AMAZON WEB SERVICES, INC., 2015. What is Amazon Elastic File System? - Amazon Elastic File System. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html>.
- AMAZON WEB SERVICES, INC., 2016. What is an Application Load Balancer? - Elastic Load Balancing. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html>.
- AMAZON WEB SERVICES, INC., 2021. Información general sobre Amazon Web Services - Documento técnico de AWS. [en línea], Disponible en: https://docs.aws.amazon.com/es_es/whitepapers/latest/aws-overview/aws-overview.pdf.

AMAZON WEB SERVICES, INC., 2022a. ¿Qué es la virtualización? - Explicación de la virtualización de la computación en la nube - AWS. [en línea]. [consulta: 21 marzo 2023]. Disponible en: <https://aws.amazon.com/es/what-is/virtualization/>.

AMAZON WEB SERVICES, INC., 2022b. What is Amazon VPC? - Amazon Virtual Private Cloud. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>.

AMAZON WEB SERVICES, INC., 2023a. AWS | Cloud Computing - Servicios de informática en la nube. *Amazon Web Services, Inc.* [en línea]. [consulta: 23 junio 2023]. Disponible en: <https://aws.amazon.com/es/>.

AMAZON WEB SERVICES, INC., 2023b. What is Amazon Elastic Container Service? - Amazon Elastic Container Service. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide>Welcome.html>.

ANGULAR, 2015. Angular. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://angular.io/>.

APACHE CASSANDRA, 2009. Apache Cassandra | Apache Cassandra Documentation. [en línea]. [consulta: 11 junio 2023]. Disponible en: https://cassandra.apache.org/_cassandra-basics.html.

ATLASSIAN, 2022. Infraestructura como código (IaC). [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://www.atlassian.com/es/microservices/cloud-computing/infrastructure-as-code>.

AULA DE SOFTWARE LIBRE DE LA UCO, 2022. Crear imágenes propias - Taller de Docker. [en línea]. [consulta: 8 junio 2023]. Disponible en: <https://aulasoftwarelibre.github.io/taller-de-docker/dockerfile/>.

AVI, 2019. Docker Architecture y sus componentes para principiantes. Geekflare [en línea]. [consulta: 7 junio 2023]. Disponible en: <https://geekflare.com/es/docker-architecture/>.

AYUDALEY, 2020. Qué es PostgreSQL y sus principales ventajas. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://ayudaleyprotecciondatos.es/bases-de-datos/que-es-postgresql-ventajas/>.

BABENKO, A., 2021. Conceptos clave. [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://www.terraform-best-practices.com/v/es/key-concepts>.

DOCKER, 2021. What is a Container? | Docker. [en línea]. [consulta: 28 marzo 2023]. Disponible en: <https://www.docker.com/resources/what-container/>.

- DOCKER, 2022. Docker: Accelerated, Containerized Application Development. [en línea]. [consulta: 28 marzo 2023]. Disponible en: <https://www.docker.com/>.
- DOCKER, 2023. Docker Compose overview. *Docker Documentation* [en línea]. [consulta: 8 junio 2023]. Disponible en: <https://docs.docker.com/compose/>.
- DOCKER HUB, 2014. Docker Hub Container Image Library | App Containerization. [en línea]. [consulta: 28 marzo 2023]. Disponible en: <https://hub.docker.com/>.
- EDER, M., 2016. Hypervisor- vs. Container-based Virtualization. [en línea], [consulta: 23 marzo 2023]. DOI [10.2313/NET-2016-07-1_01](https://doi.org/10.2313/NET-2016-07-1_01). Disponible en: http://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_01.pdf.
- FREECODECAMP.ORG, 2021. ¿Qué es NPM? [en línea]. [consulta: 10 junio 2023]. Disponible en: <https://www.freecodecamp.org/espanol/news/que-es-npm/>.
- GARTNER, 2022. Magic Quadrant para servicios de infraestructura y plataforma en la nube. [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://www.gartner.com/technology/media-products/reprints/AWS/1-2AOZQARI-ESL.html>.
- GONÇALVES, M.J., 2021. ¿Qué es Angular y para qué sirve? *Blog de Hiberus Tecnología* [en línea]. [consulta: 10 junio 2023]. Disponible en: <https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/>.
- GOOGLE CLOUD, 2022. ¿Qué es una base de datos relacional (RDBMS)? | Google Cloud. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://cloud.google.com/learn/what-is-a-relational-database?hl=es-419>.
- HERNÁNDEZ, U., 2018. Qué es TypeScript. *CódigoFacilito* [en línea]. [consulta: 10 junio 2023]. Disponible en: <https://codigofacilito.com/articulos/typescript>.
- HOMBERGS, T., 2020. Health Checks with Spring Boot. [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://reflectoring.io/spring-boot-health-check/>.
- IBM, 2021a. ¿Qué es Java Spring Boot? | IBM. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://www.ibm.com/es-es/topics/java-spring-boot>.
- IBM, 2021b. ¿Qué es la virtualización? | IBM. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://www.ibm.com/es-es/topics/virtualization>.
- IBM, 2023. ¿Qué es Terraform? | IBM. [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://www.ibm.com/es-es/topics/terraform>.
- IZAGUIRRE TAPIA, H.E., 2022. Diseño de una aplicación web para la migración de máquinas virtuales construidas en Amazon Web Services hacia Oracle Cloud para SINERGIK S.A.C. En: Accepted: 2023-04-19T14:17:00Z,

Repository Institucional - UCV [en línea], [consulta: 22 junio 2023]. Disponible en: <https://repositorio.ucv.edu.pe/handle/20.500.12692/111888>.

JFROG, 2021. A Beginner's Guide to Understanding and Building Docker Images. JFrog [en línea]. [consulta: 7 junio 2023]. Disponible en: <https://jfrog.com/devops-tools/article/understanding-and-building-docker-images/>.

JOSÉ MANUEL M., 2019. Availability Zones - Zonas de Disponibilidad (AZ). [en línea]. [consulta: 9 junio 2023]. Disponible en: <https://pc-solucion.es/terminos/availability-zones-zonas-de-disponibilidad-az/>.

KUMAR, Rakesh; CHARU, Shilpi. An importance of using virtualization technology in cloud computing. Global Journal of Computers & Technology, 2015, vol. 1, no 2. Disponible en: https://scholar.google.es/scholar?hl=es&as_sdt=0%2C5&as_ylo=2015&as_yhi=2023&q=What+is+hardware+virtualization&btnG=#:~:text>An%20importance%20of%20using%20virtualization%20technology%20in%20cloud%20computing

MARINESCU, D.C., 2013. *Cloud computing: theory and practice* [en línea]. Boston: Elsevier/Morgan Kaufmann, Morgan Kaufmann is an imprint of Elsevier. ISBN 978-0-12-404627-6. Disponible en: <https://eclass.uoa.gr/modules/document/file.php/D416/CloudComputingTheoryAndPractice.pdf>. QA76.585 .M37 2013

MASON, R., 2013. NoSQL and Big Data connectors for Mule. *MuleSoft Blog* [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://blogs.mulesoft.com/dev-guides/how-to-tutorials/nosql-and-big-data-connectors-for-mule/>.

MEDRANO, A., 2022. Amazon Web Services. Servicios, redes, seguridad. AWS CLI y Cloudshell. - Inteligencia Artificial y Big Data. [en línea]. [consulta: 9 junio 2023]. Disponible en: <https://aitor-medrano.github.io/bigdata2122/apuntes/nube02aws.html>.

MELL, P. y GRANCE, T., 2011. The NIST Definition of Cloud Computing. [en línea], DOI <https://doi.org/10.6028/nist.sp.800-145>. Disponible en: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf>.

MICROSOFT AZURE, 2019. ¿Qué es Kubernetes? | Microsoft Azure. [en línea]. [consulta: 28 marzo 2023]. Disponible en: <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-kubernetes/>.

MICROSOFT AZURE, 2020. ¿Qué es un contenedor? | Microsoft Azure. [en línea]. [consulta: 28 marzo 2023]. Disponible en: <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-container>.

- MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA, 2009. Sede Electrónica del Catastro - Inicio. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://www.sedecatastro.gob.es/>.
- MONGODB, 2022. MongoDB: The Developer Data Platform | MongoDB. [en línea]. [consulta: 25 junio 2023]. Disponible en: <https://www.mongodb.com/>.
- MUÑOZ, J.D., 2016. Primeros pasos con Docker. *PLEDIN 3.0* [en línea]. [consulta: 7 junio 2023]. Disponible en: <https://www.josedomingo.org/pledin/2016/02/primeros-pasos-con-docker/>.
- MYSQL, 2011. MySQL :: MySQL Downloads. [en línea]. [consulta: 25 junio 2023]. Disponible en: <https://www.mysql.com/downloads/>.
- NAZIR, R., AHMED, Z., AHMAD, Z., SHAIKH, N.N., LAGHARI, A.A. y KUMAR, K., 2020. Cloud Computing Applications: A Review. *EAI Endorsed Transactions on Cloud Systems*, vol. 6, no. 17, ISSN 2410-6895. DOI [10.4108/eai.22-5-2020.164667](https://doi.org/10.4108/eai.22-5-2020.164667).
- NPM, 2023. dotenv. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://www.npmjs.com/package/dotenv>.
- PARAMIO, C., 2011. Una introducción a MongoDB. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://www.genbeta.com/desarrollo/una-introduccion-a-mongodb>.
- PATEL, H.B. y KANSARA, N., 2021a. *Cloud Computing Deployment Models: A Comparative Study* [en línea]. SSRN Scholarly Paper. 23 marzo 2021. Rochester, NY: s.n. [consulta: 8 junio 2023]. 3832832. Disponible en: <https://papers.ssrn.com/abstract=3832832>.
- PATEL, H.B. y KANSARA, N., 2021b. *Cloud Computing Deployment Models: A Comparative Study* [en línea]. SSRN Scholarly Paper. 23 marzo 2021. Rochester, NY: s.n. [consulta: 8 junio 2023]. 3832832. Disponible en: <https://papers.ssrn.com/abstract=3832832>.
- POSTGRESQL, 2008. PostgreSQL: Downloads. [en línea]. [consulta: 25 junio 2023]. Disponible en: <https://www.postgresql.org/download/>.
- POSTMAN, 2019. Postman API Platform | Sign Up for Free. [en línea]. [consulta: 12 junio 2023]. Disponible en: <https://www.postman.com>.
- PROJEKT CLOUD COMPUTING, UNIVERSIDAD TECNOLÓGICA DE BRESLAVIA, 2016a. Hypervisors. [en línea]. [consulta: 25 marzo 2023]. Disponible en: <https://oze.pwr.edu.pl/kursy/introcloud/content/start/K-04-03.html>.
- PROJEKT CLOUD COMPUTING, UNIVERSIDAD TECNOLÓGICA DE BRESLAVIA, 2016b. What is virtualization? [en línea]. [consulta: 22 marzo

2023]. Disponible en: <https://oze.pwr.edu.pl/kursy/introcloud/content/start/K-04-01.html>.

PROJEKT CLOUD COMPUTING, UNIVERSIDAD TECNOLÓGICA DE BRESLAVIA, 2018. Cloud Computing - Introduction Conclusion. [en línea]. [consulta: 8 junio 2023]. Disponible en: <https://oze.pwr.edu.pl/kursy/introcloud/content/start/K-01-04.html>.

RED HAT, INC, 2023. ¿Qué es y para qué sirve la virtualización? [en línea]. [consulta: 21 marzo 2023]. Disponible en: <https://www.redhat.com/es/topics/virtualization/what-is-virtualization>.

SALESFORCE, 2017. ¿Qué es Cloud Computing? [en línea]. [consulta: 22 junio 2023]. Disponible en: <https://www.salesforce.com/mx/cloud-computing/>.

SHUKUR, H., ZEEBAREE, S., ZEBARI, R., ZEEBAREE, D., AHMED, O. y SALIH, A., 2020. Cloud Computing Virtualization of Resources Allocation for Distributed Systems. *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, ISSN 2708-0757. DOI [10.38094/jastt1331](https://doi.org/10.38094/jastt1331).

SPRING BOOT, 2014. Getting Started | Accessing Data with MongoDB. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://spring.io/guides/gs/accessing-data-mongodb/>.

SPRING BOOT, 2021. Spring Boot. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://spring.io/projects/spring-boot>.

SPRING BOOT, 2023. Spring Boot Actuator | Baeldung. [en línea]. [consulta: 26 junio 2023]. Disponible en: <https://www.baeldung.com/spring-boot-actuators>.

SPRING INITIALIZR, 2020. Spring Initializr. [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://start.spring.io>.

TAVBULATOVA, Z.K., ZHIGALOV, K., KUZNETSOVA, S.Y. y PATRUSOVA, A.M., 2020. Types of cloud deployment. *Journal of Physics: Conference Series*, vol. 1582, no. 1, ISSN 1742-6596. DOI [10.1088/1742-6596/1582/1/012085](https://doi.org/10.1088/1742-6596/1582/1/012085).

TERRAFORM BY HASHICORP, 2023. Terraform by HashiCorp. [en línea]. [consulta: 23 junio 2023]. Disponible en: <https://www.terraform.io/>.

UDEMY, 2022. Amazon AWS al completo. [en línea]. [consulta: 26 junio 2023]. Disponible en: <https://www.udemy.com/course/amazon-aws-al-completo/>.

VERGARA, A., 2016. JPA vs Hibertate ¿cuál es la diferencia? *Tech blog for developers | Facilcloud* [en línea]. [consulta: 11 junio 2023]. Disponible en: <https://www.facilcloud.com/noticias/jpa-vs-hibernate/>.

VMWARE, INC., 2020. ¿Qué es la recuperación ante desastres? | Glosario de VMware | ES. [en línea]. [consulta: 24 junio 2023]. Disponible en: <https://www.vmware.com/es/topics/glossary/content/disaster-recovery.html>.