Tabla de contenido

**Instrucciones para levantar la app 2**

**Endpoints 8**

**Arquitectura 9**

1. **Instrucciones para levantar la app**

**Prerequisitos.** Tener instalado java 8, Git, IDE Spring Tool Suite o simil

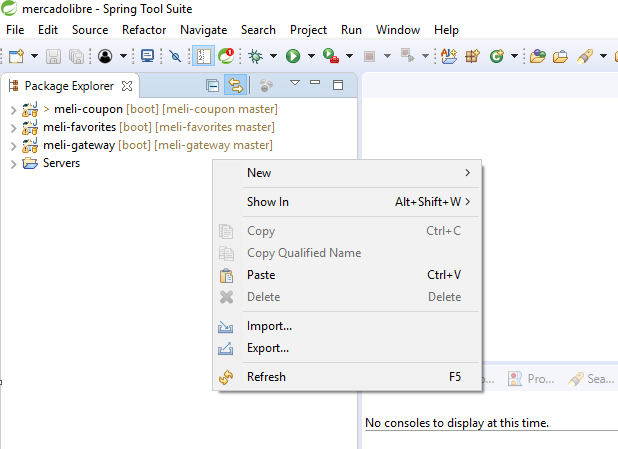
1. Cree un directorio y por medio de la consola de comandos descargue el repositorio ejecutando los siguientes comandos

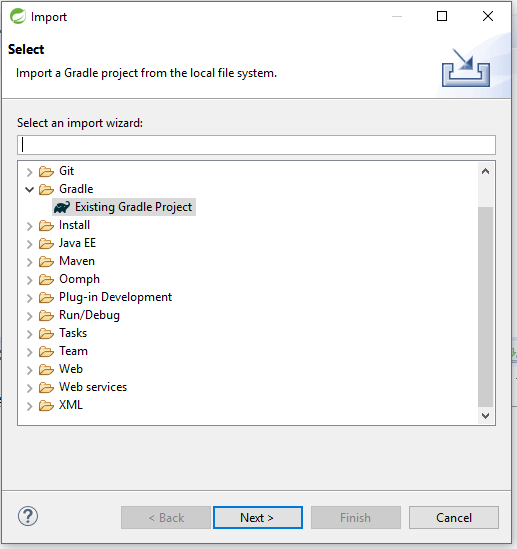
git clone https://github.com/germandiaz6191/meli-coupon.git

git clone https://github.com/germandiaz6191/meli-gateway.git

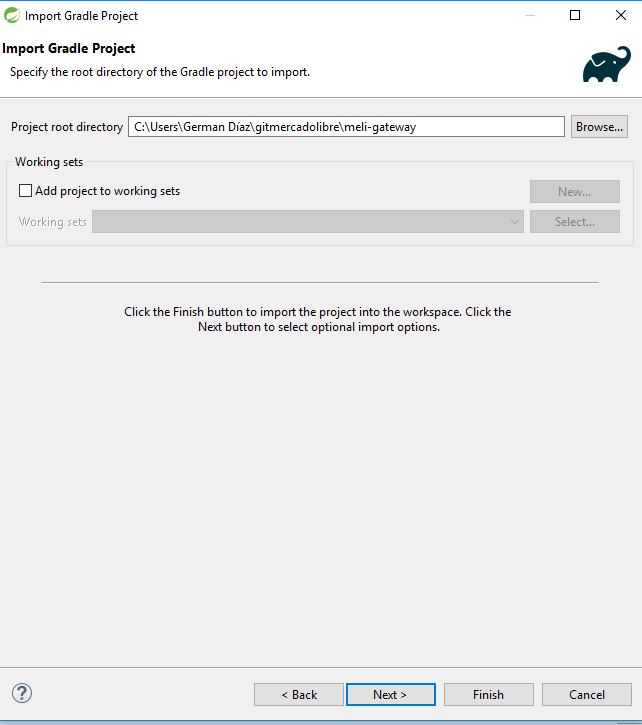
git clone https://github.com/germandiaz6191/meli-favorites.git

1. Por medio del ide que instaló importe los proyectos. El el IDE Spring Tools dar clic derecho en el área de explorador, seleccionar Import / Existing Project Gradle





Dar clic en Next / Next y seleccionamos la carpeta de un proyecto, en este caso meli-gateway



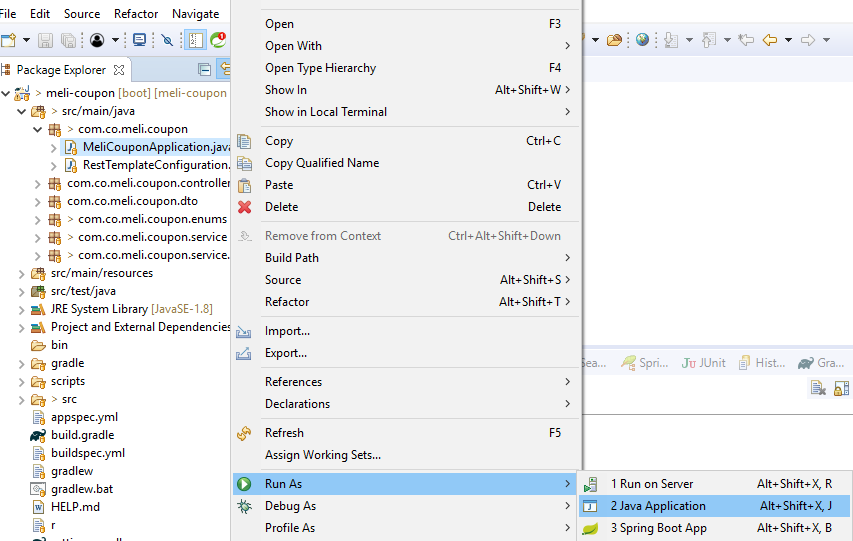
Dar clic en el botón Finish.

Realizar este paso para los demás proyectos.

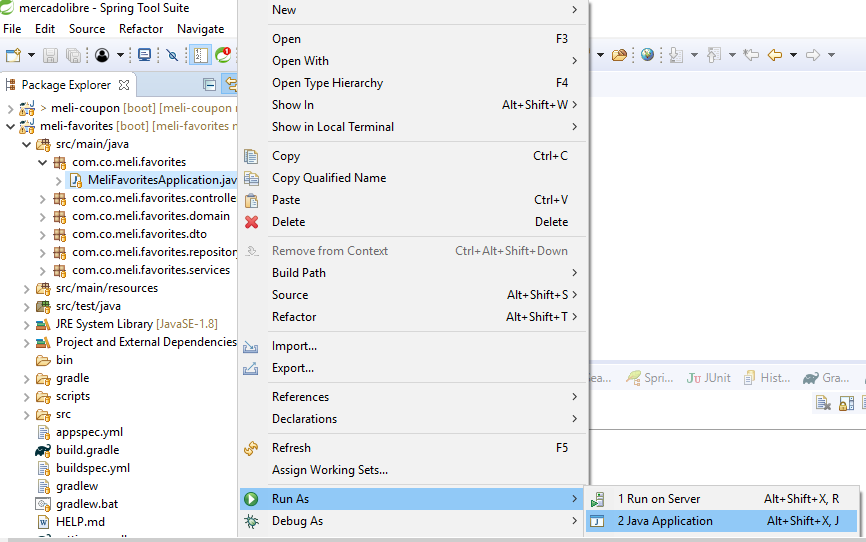
1. Una vez importados, por medio del explorador desplegamos cada uno de los proyectos hasta llegar a la clase Application, seleccionamos clic derecho Run As / Java Application

**Nota:** Para realizar este paso, los puertos 2020, 8080 y 8081 deben estar libres, de lo contrario cambiarlos en el archivo application-dev.yml

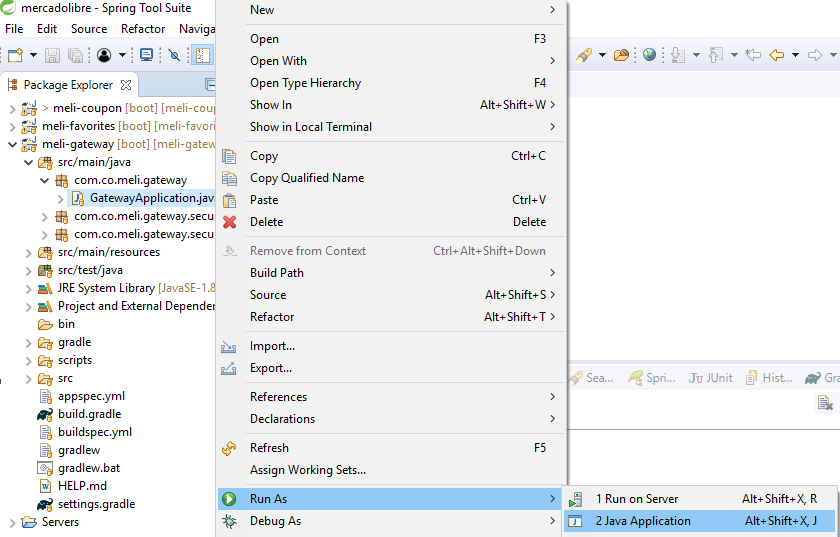
Proyecto Coupon.



Proyecto Favorites

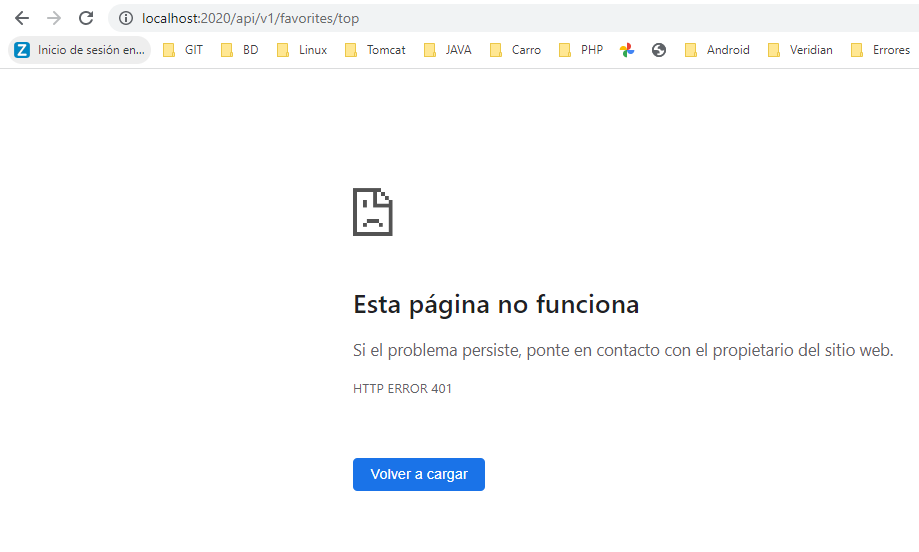


Proyecto Gateway



1. Al ingresar al endpoint por el navegador, debemos visualizar un error 401 de no autorizado

<http://localhost:2020/api/v1/favorites/top>

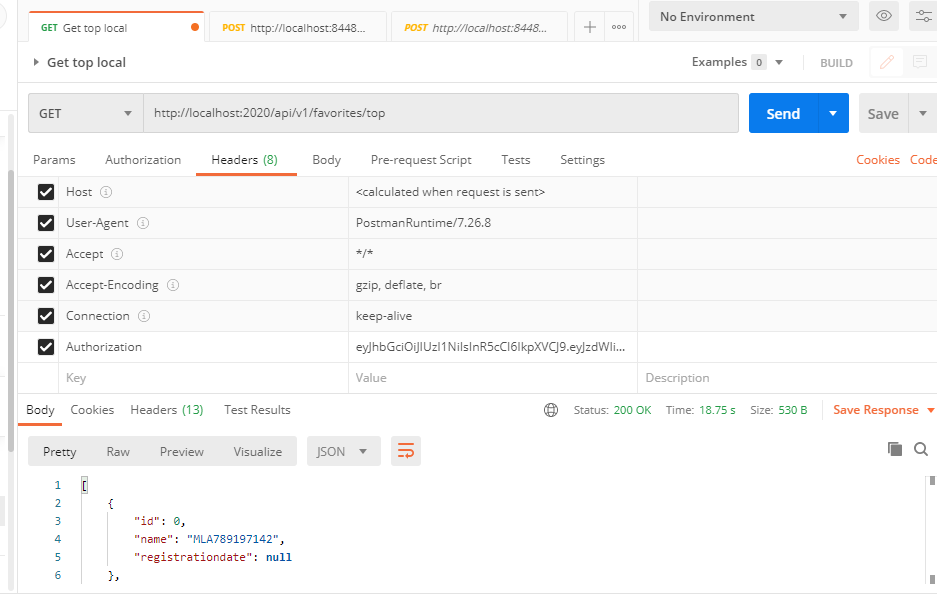


1. Para consumir la api ingresamos a postman, crear un nuevo Request y poner el endpoint con el tipo de petición, en la pestaña Headers agregar un nuevo atributo llamado Authorization y pegar el token del ambiente QA. La siguiente imagen presenta el consumo del servicio top de la api favorites.

**El token de autorización para QA**

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ik1lcmNhZG8gTGlicmUiLCJhdXRob3JpdGllcyI6WyJST0xFX01FTEkiXSwiaWF0Ijo2NTE2ODM5MDIyLCJwcm9maWxlIjoiZGV2In0.mQlj5u2zaLeDT54iC1QF2T9WL4AgZkJttjJrQYHdvSs

**Para productivo se tiene un token diferente.**



Al consumir el servicio por postman

1. **Endpoints**

* Get. Servicio Top

Retorna el top 5 de los items favoritos

* http://ec2-18-212-70-183.compute-1.amazonaws.com:8080/api/v1/favorites/top
* Post. Servicio coupon

Se envían los items para redimir el máximo del valor de cupón

* http://ec2-18-212-70-183.compute-1.amazonaws.com:8080/api/v1/coupon/redeem
* Post: Servicio save Favorito

En productivo este servicio se encuentra en una red privada, por lo que se consume cuando se realiza una petición de /api/v1/coupon/redeem

* http://ec2-18-212-70-183.compute-1.amazonaws.com:8080/api/v1/coupon/redeem

Para más detalle revisar la documentación en:

https://www.postman.com/dark-satellite-408918/workspace/mercado-libre/request/8617671-50046468-8619-437e-9e0b-f4aec2da850a

1. **Arquitectura**
   1. **Configuración Base de Datos:**

Se una BD en la nube, de pruebas y productivo para centralizar la información y configuraciones

* 1. **Proxy**

Se implementó una capa de proxy para manejar un primer nivel de seguridad hacia las peticiones entrantes de internet.

* 1. **EC2**

Para la implementación de la solución se crearon dos instancias auto escalables que permiten desplegar las apis Favorites y Coupon. Estos microservicios se comunican por puerto http ya que están en una subred segura que no se expone a internet.

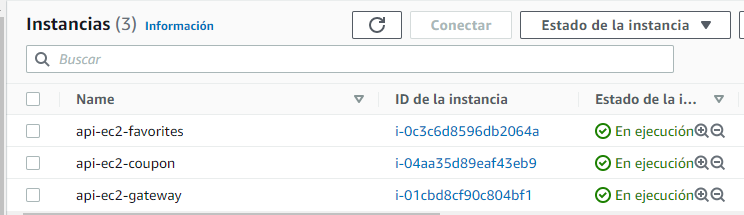
**Coupon**: Se realizó la implementación de un microservicios para el cupón, este se encarga de consultar la api de Mercado Libre <https://api.mercadolibre.com/items?ids=> para obtener el precio de cada item del cupón y consumir la api de **Favorites** para registrar los items que se rediman.

Como respuesta de la api Coupon obtenemos los items con el total gastado maximizado.

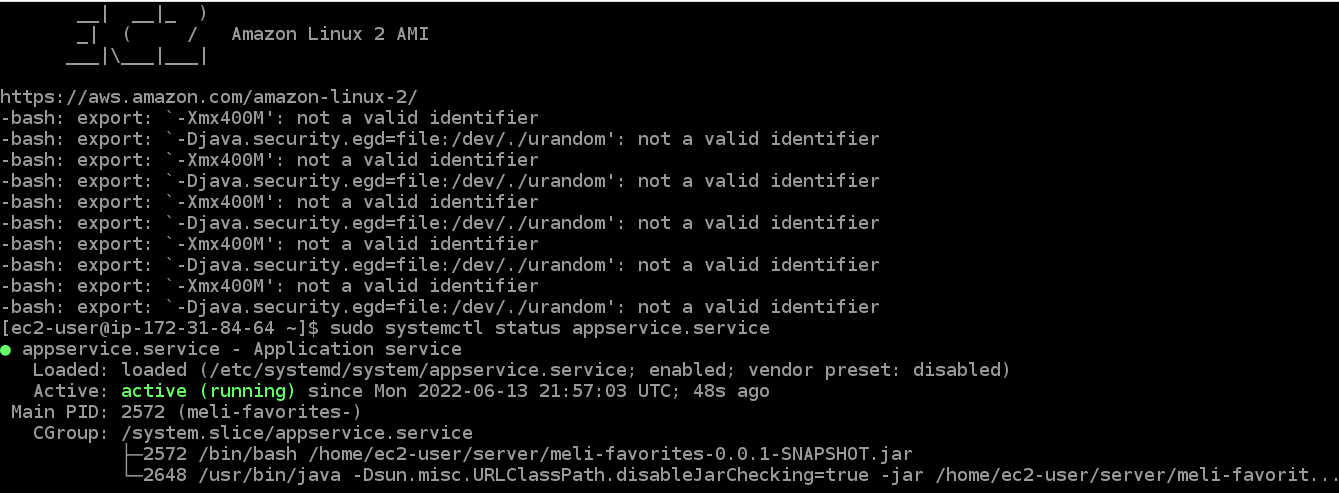
Servicios expuestos:

Post:

**Favorites**: Esta api tiene como objetivo obtener el top 5 de items favoritos redimidos, también cuenta con un servicio el cual recibe los items redimidos y los guarda en BD para posteriormente calcular el top.



Las instancias están desplegadas como servicio para iniciar automáticamente las aplicaciones cuando exista un reinicio.



Se limitó la memoria de las instancias para el buen funcionamiento del Garbage Collector y evitar el colapso y desborde cuando alcance el límite de las instancias.

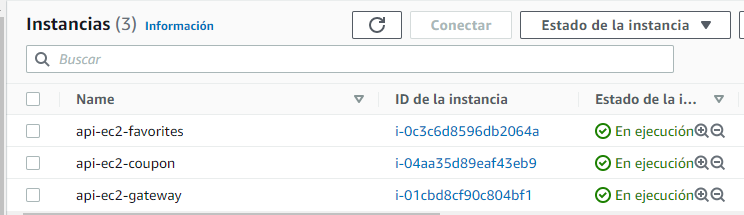


* 1. **Data Base**

Se utilizó una base de datos Postgres la cual se puede **clusterizar** para cumplir con la demanda de peticiones que se realice.

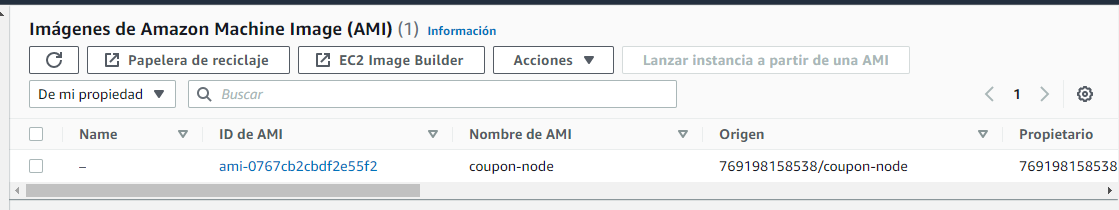
* 1. **Api Gateway**

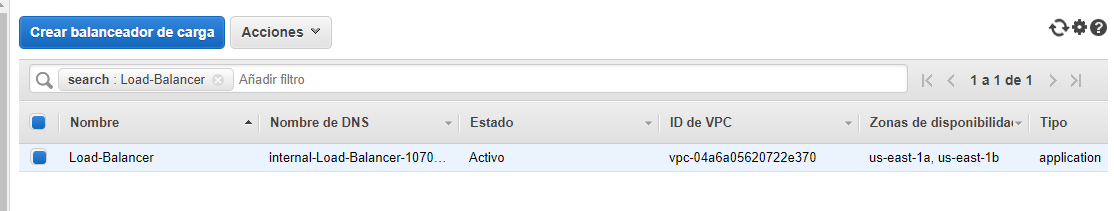
Se desplegó un Api Gateway el cual permite securizar los servicios expuestos por medio de un token, permitiendo validar que las peticiones que lleguen sean válidas. Está expuesto hacia internet, el cual permite recibir peticiones por medio de protocolo seguro https y redirecciona las peticiones hacia el balanceador de carga por protocolo no seguro http



* 1. **Load Balancing**

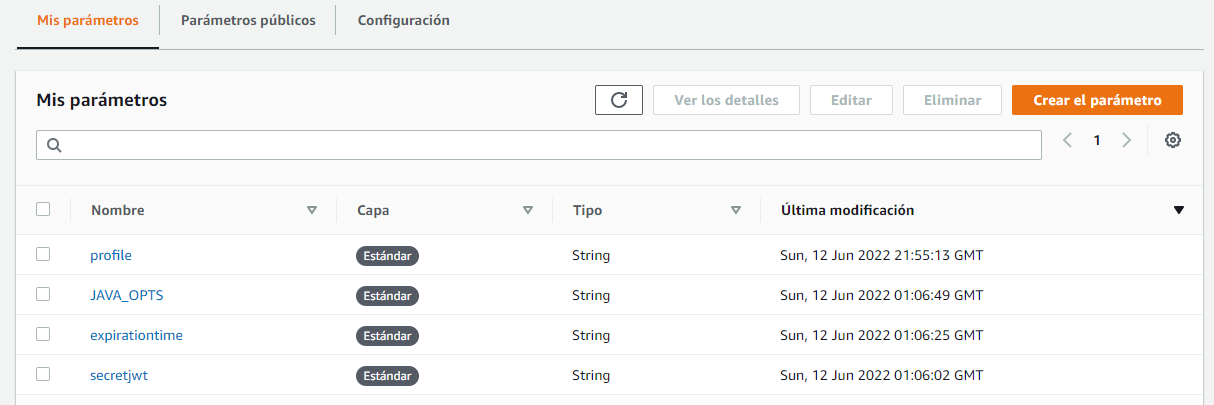
El componente de balanceo permite enviar las peticiones a las instancias con menos carga, permitiendo que estas no colapsen por la cantidad de procesamiento. Se creó una imagen de los componentes que se requieren autoescalar para en caso de llegar muchas peticiones se escale y estas se procesen. Una vez ya no exista tanto tráfico, estas instancias de imágenes se eliminan automáticamente.





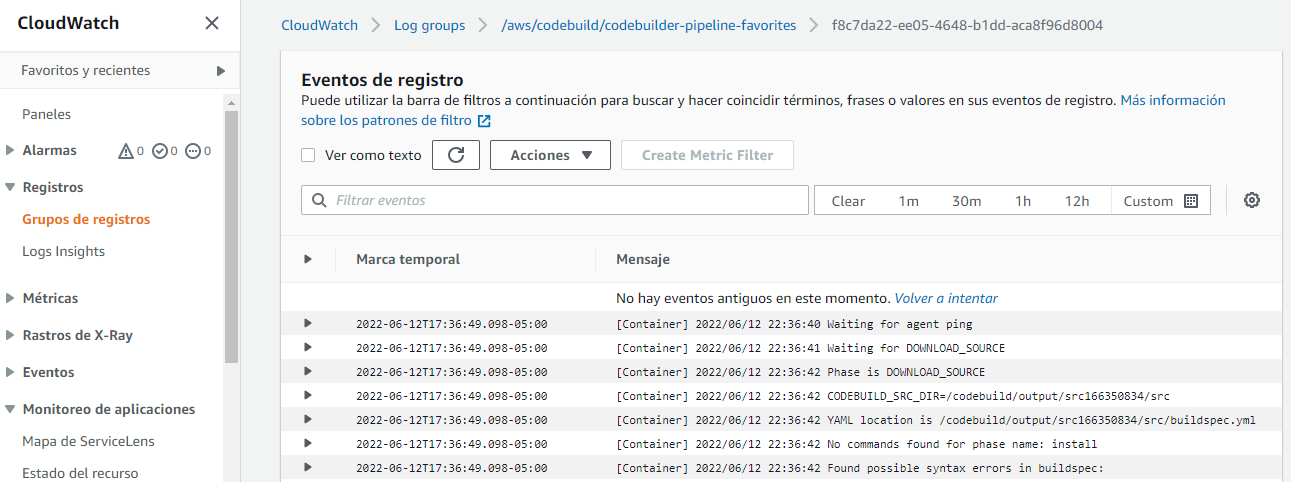
* 1. **System Manager**

Los parámetros que necesita cada api sensibles o no sensibles se registran por medio de este componente, centralizando la información y evitando despliegues por cambios en properties, este componente se utiliza para guardar información de BD encriptando y cualquier otro dato que necesite utilizar los jars o soluciones desplegadas.



* 1. **Cloud Watch**

Por medio de esta herramienta centralizamos los logs de las instancias o componentes de AWS, de este modo cualquier error generado tanto a nivel de despliegue de pipeline como de instancia se registra en este.

****

* 1. **DevOps**

El despliegue de los componentes se realiza de forma automática al aceptar un pull request en el repositorio de GitHub, esta rama está protegida y solo pueden aceptar o rechazar los Pull Request los usuarios que cuenten con los permisos. Para esto se utilizaron las herramientas de CodePipeline de AWS como son S3, CodeCommit, CodeBuild y CodeDeploy

