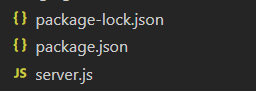
¿Creación del servidor de señalización y despliegue

1. Para configurar nuestro servidor, lo primero que debemos hacer, es crear un archivo.js para este caso se llamará server.js y además crearemos un package.json donde se agreguen las dependencias con el comando npm-init.

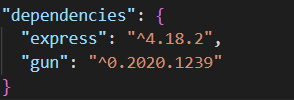


1. Las dependencias que vamos a instalar son las siguientes:

-npm install gun

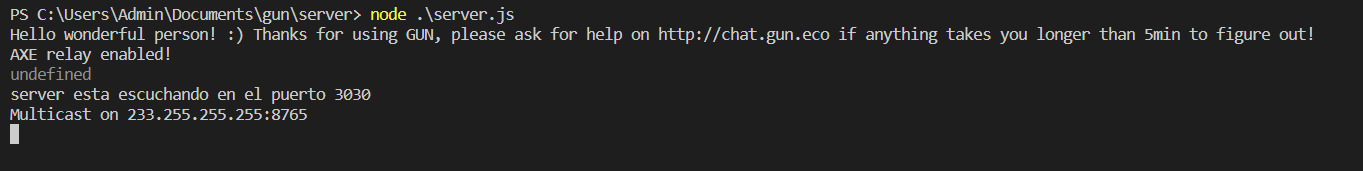
-npm install express

Una vez realizado esto, validamos que las dependencias se hayan agregado correctamente al package.json.



1. Configuración del servidor: En el archivo.js añada el siguiente código
2. import Express from 'express';
3. import Gun from 'gun';
4. const app = Express();
5. const port = 3030;
6. app.use(Gun.serve, function (req, res, next) {
7. next();
8. });
9. const server = app.listen(port, (event) => {
10. console.log(`server esta escuchando en el puerto ${port}`);
11. });
12. Gun({
13. web:server,
14. file: false, // desactiva la persistencia de archivos
15. radisk: false // desactiva la persistencia en disco
16. });

con el comando node archivo.js puede iniciar el servidor, el cual estará disponible en el puerto que haya seleccionado al momento de la creación, en este caso el 3030.



Una vez comprobado que todo funciona correctamente. Podemos proceder a desplegar este servidor en una instancia de EC2.

Dockerizacion del servidor

1. Creamos el dockerfile y el .dockerignore

Contenido del docker file

FROM node:12

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

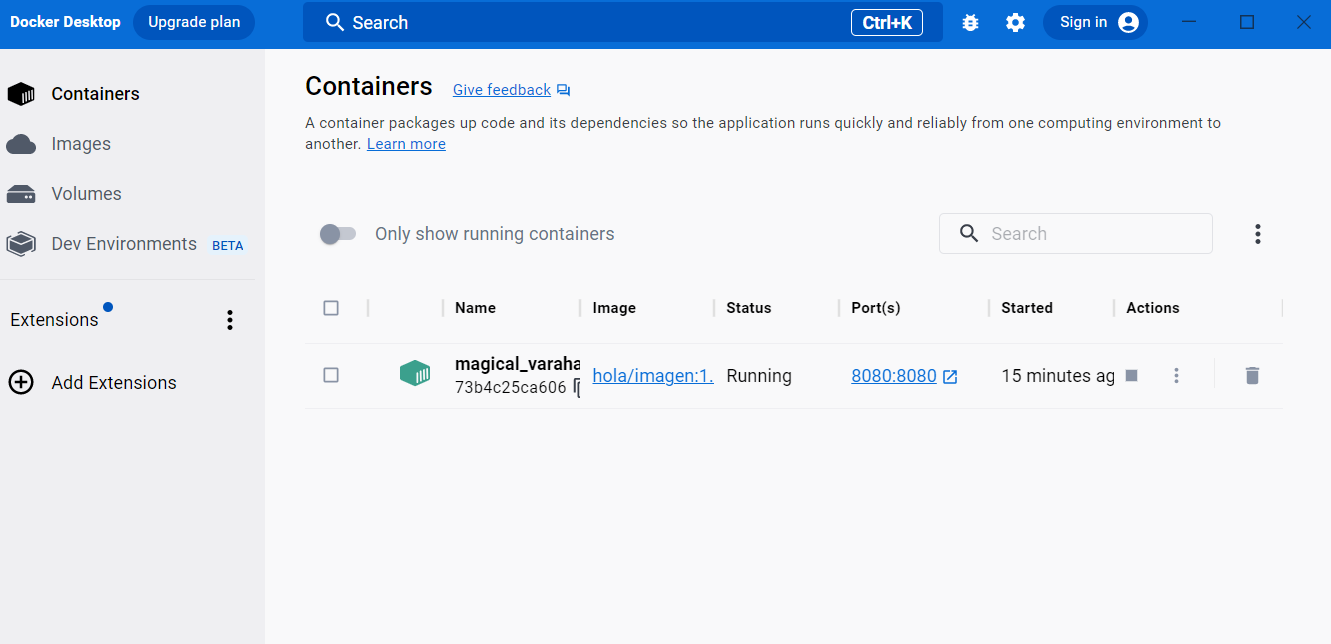
ENV PORT=3030

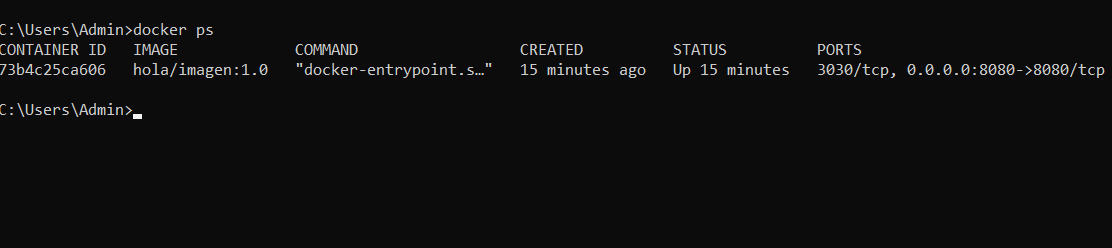
EXPOSE 3030

CMD ["npm","start"]

Este Dockerfile define la creación de una imagen de Docker basada en la versión 12 de Node.js. Establece el directorio de trabajo como "/app" y copia los archivos de paquete y de configuración de NPM en ese directorio. Luego, ejecuta el comando "npm install" para instalar las dependencias necesarias. Después de copiar todos los archivos del contexto de construcción en el directorio de trabajo, establece la variable de entorno "PORT" en 3030 y expone el puerto 3030 para que pueda ser accedido desde el host de Docker. Finalmente, el contenedor se inicia con el comando "npm start".

1. Lo siguiente sera generar la imagen, para eso nos ayudaremos del comando docker build -t [image-name] .
2. Una vez construida la imagen podemos ejecutarla con el comando docker run -p 8080:8080 [image-name]
3. podemos ver los contenedores en ejecución podemos valernos de la aplicacion Dockerdesktop o del comando docker





Despliegue de contenedor haciendo uso de aws Fargate

Amazon Fargate es un servicio de orquestación de contenedores de AWS (Amazon Web Services) que permite ejecutar contenedores sin tener que administrar los servidores subyacentes. Con Fargate, puede centrarse en desarrollar y ejecutar sus aplicaciones en contenedores, sin tener que preocuparse por la gestión de la infraestructura de servidores subyacente.

Fargate elimina la necesidad de administrar servidores EC2 o Kubernetes para ejecutar sus contenedores. En su lugar, proporciona una plataforma de contenedores totalmente gestionada en la que puede ejecutar sus aplicaciones de manera fácil y sencilla.

Fargate es compatible con la mayoría de las herramientas y tecnologías de contenedores, lo que significa que puede utilizar su mismo flujo de trabajo y tecnologías actuales para ejecutar sus aplicaciones en contenedores en Fargate. Además, Fargate es compatible con otros servicios de AWS, lo que significa que puede integrar fácilmente sus aplicaciones en contenedores con otras soluciones de AWS.

En resumen, Fargate es una opción fácil y conveniente para ejecutar aplicaciones en contenedores en AWS, sin tener que administrar la infraestructura subyacente.

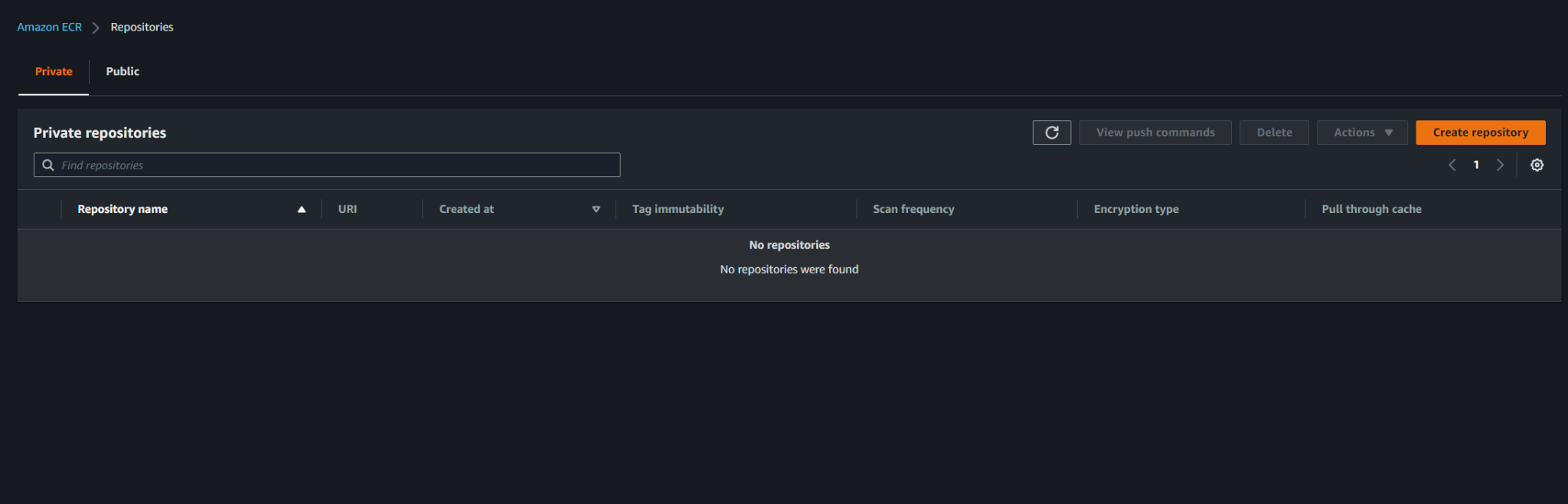
AWS Fargate Tutorial | AWS Tutorial For Beginners | AWS Certification Training | Edureka

<https://www.youtube.com/watch?v=fmFlAWtKnGA>

AWS Fargate tutorial - Running a Docker container with a Python Flask app

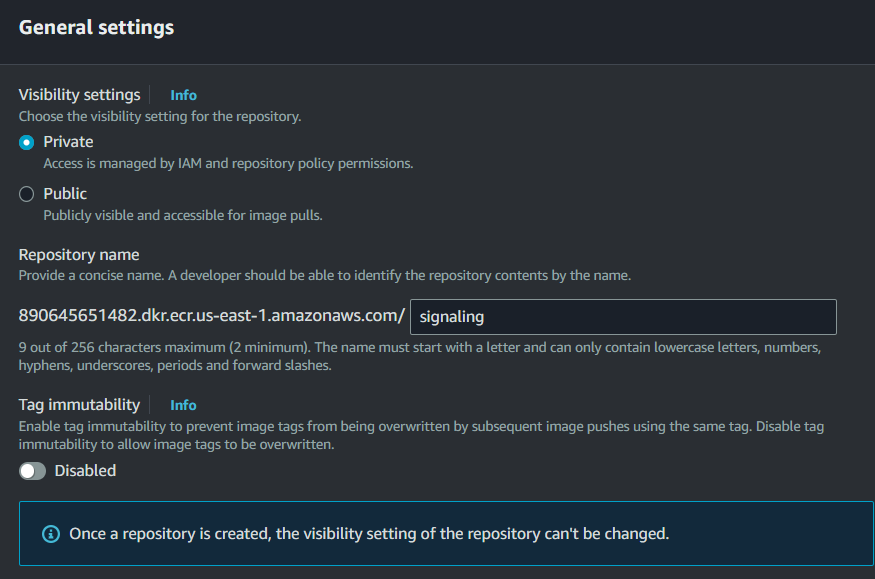
<https://www.youtube.com/watch?v=-Vsuzi4OByY>

1. Debemos subir nuestro docker container a un repositorio de ECR

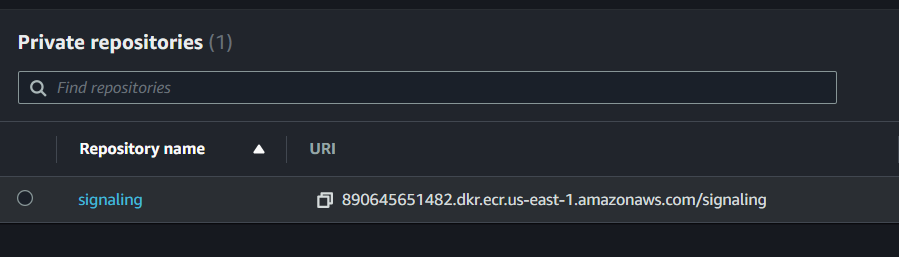


Un repositorio en ECR es un lugar donde se pueden almacenar y distribuir imágenes de contenedor. Cada repositorio puede contener múltiples imágenes, y los usuarios pueden acceder a ellas y utilizarlas en sus aplicaciones de contenedor.

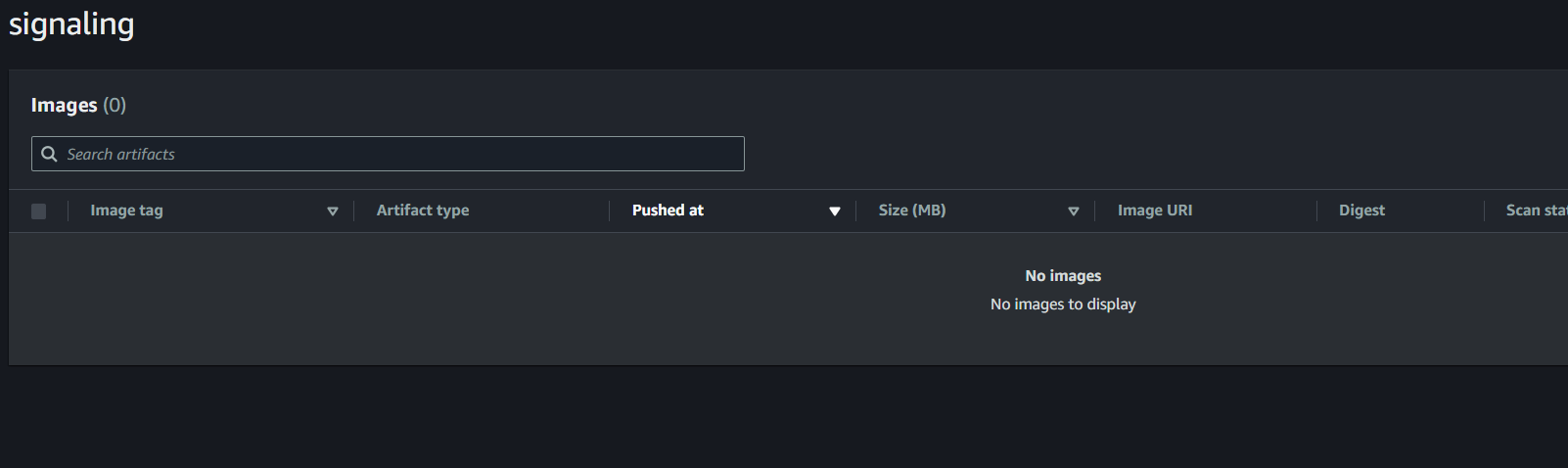
La creación es muy sencilla. Seleccionamos que queremos un repositorio privado y adicional a esto debemos definir el nombre del repositorio.



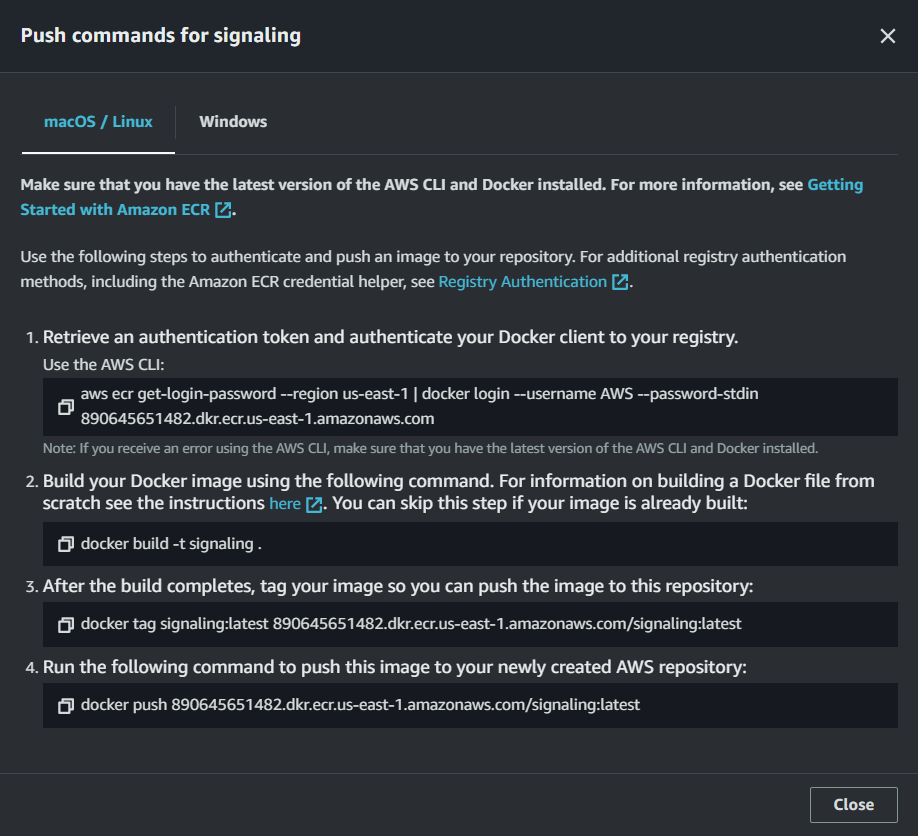
Una vez esto, podemos proceder a crear el repositorio, el cual se nos mostrará en nuestro listado de repositorios creados.



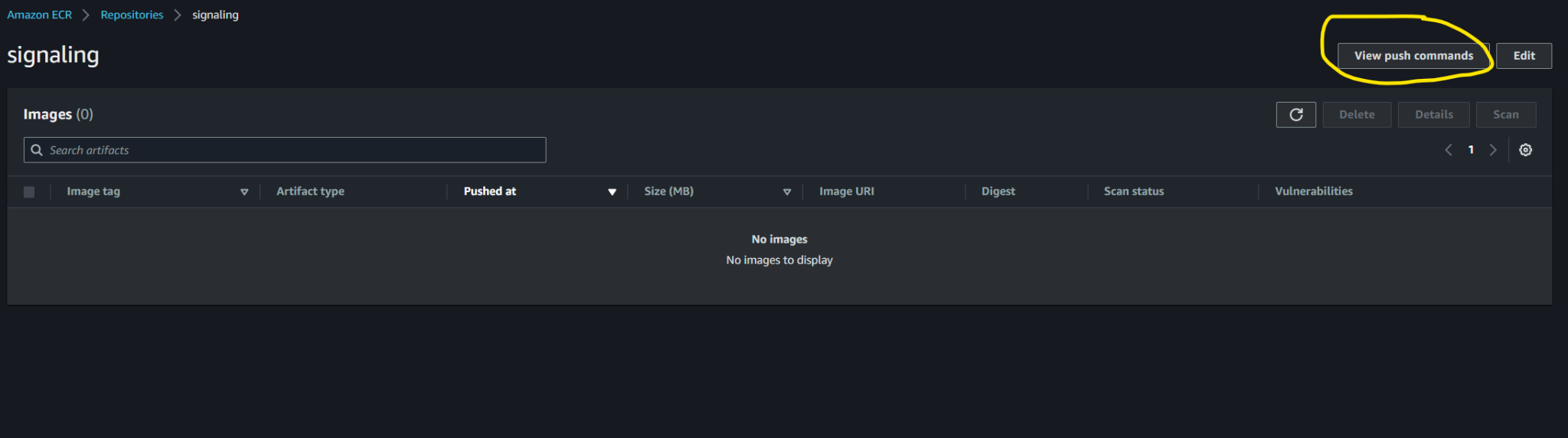
Podemos observar al acceder al repositorio que no hay ninguna imagen en este aun



1. Vamos a subir nuestro docker image al repositorio. Para esto no valemos de los comandos que se nos indica en la propia consola del repositorio. (Debemos tener el aws cli instalado)

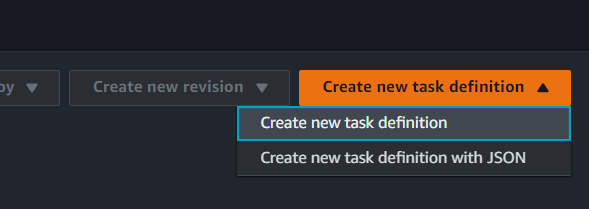


Podemos acceder a estos ingresando al repositorio desde el listado mostrado previamente y en la parte superior derecha está la opción para verlos.

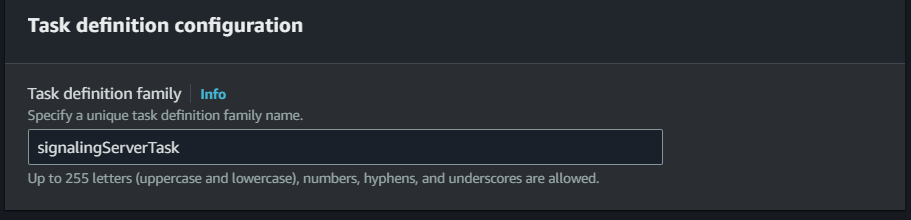


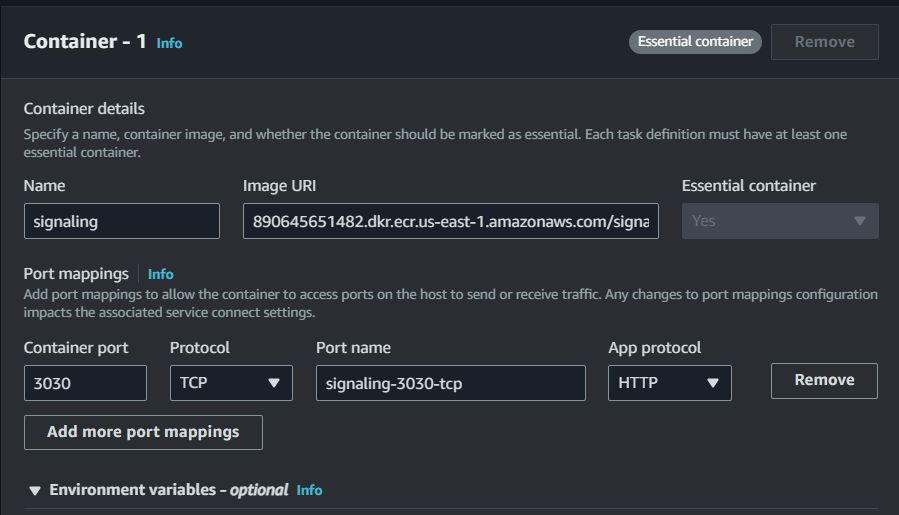
1. Creación de la task definition desde ECS. un Task Definition (Definición de tarea) es un archivo JSON que describe cómo ejecutar un conjunto de contenedores como una sola tarea en ECS. Una tarea es la unidad básica de trabajo en ECS, y puede estar compuesta por uno o varios contenedores que se ejecutan juntos en el mismo host EC2 o instancia Fargate.

ECS nos da la opción de crear la definición desde un formulario o desde un JSON directamente, para este ejemplo vamos a hacerlo desde el formulario.



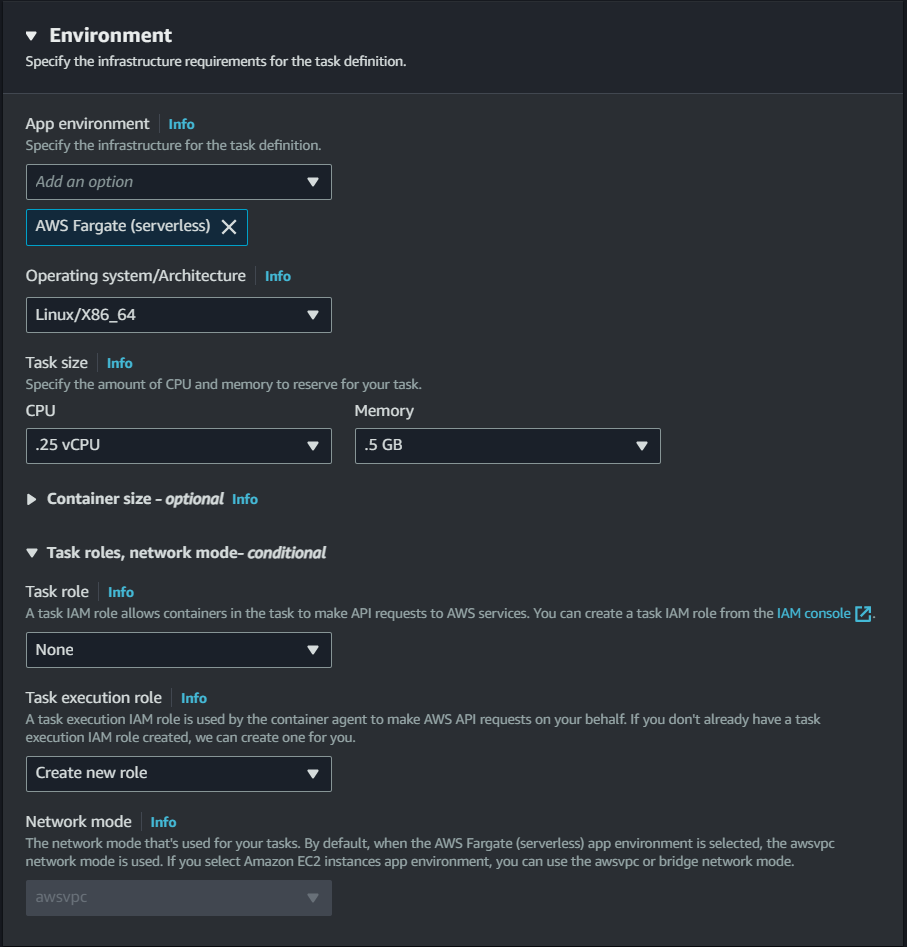
Una vez seleccionemos la opción, debemos llenar los campos asignando un nombre a la tarea y agregando nuestro contenedor alojado en ECR (ahí podremos ver el URI de nuestra imagen).

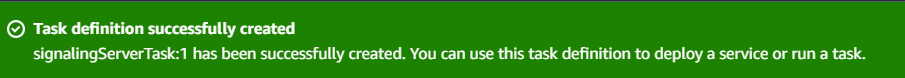


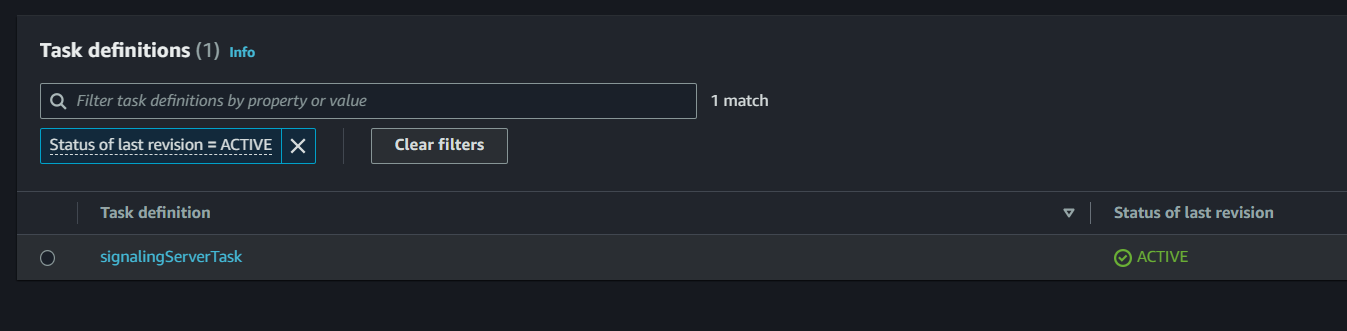


Asegurarse de definir la url de la imagen y no la del repositorio y damos en continuar.

Para la configuración del environment seleccionamos las características mínimas de CPU y memoria. Dejemos el resto por defecto, damos en continuar para ver el resumen y procedemos con la creación.



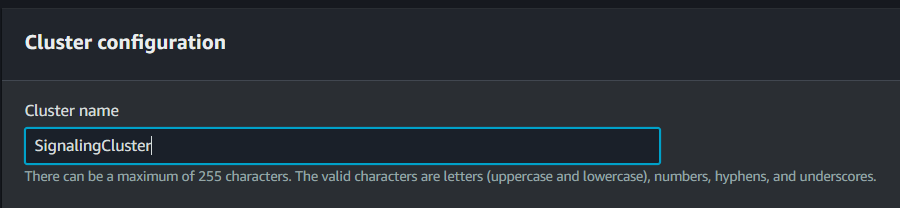


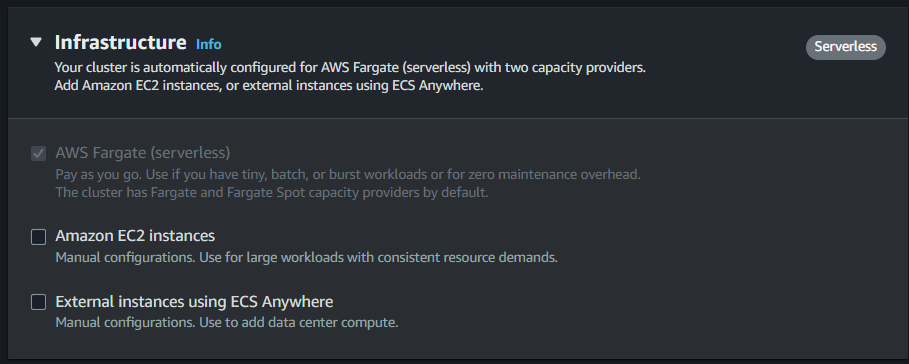


1. Creación del cluster.

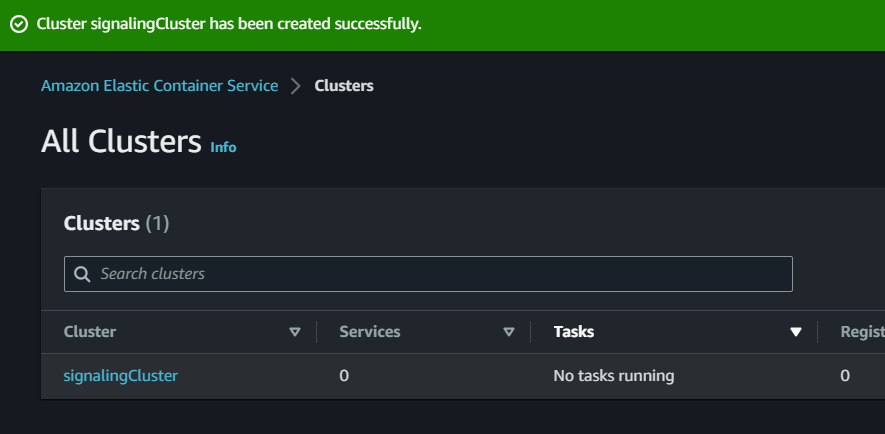
Un clúster de Amazon ECS agrupa tareas y servicios, y permite una capacidad compartida y configuraciones comunes. Todas sus tareas, servicios y capacidad deben pertenecer a un clúster.

Este paso es muy sencillo, solo será asignarle un nombre y asegurarnos que esté configurado para fargate.

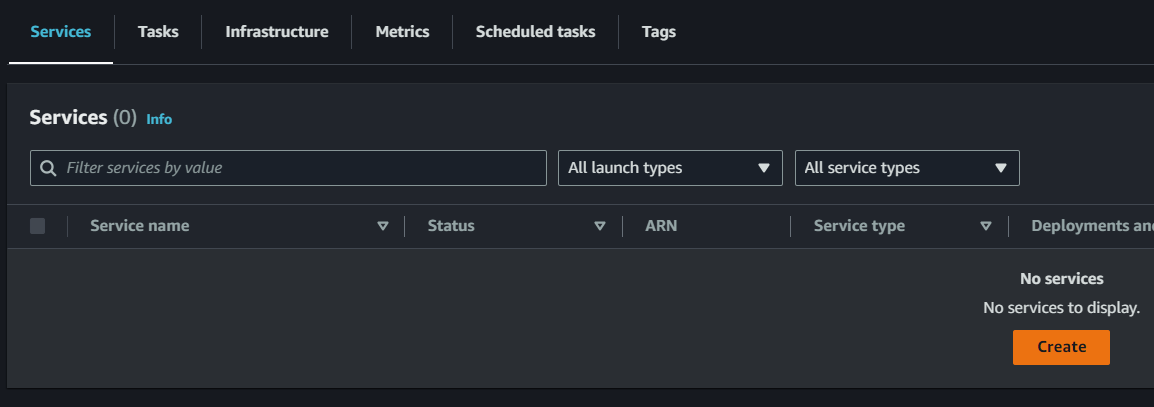




Las demás configuraciones las dejamos por defecto y creamos el cluster.



1. Creación del service. Para esto accedemos al cluster que acabamos de crear y en la sección de service, le damos crear uno nuevo.



En la sección de deployment configuration, seleccionamos la definición de tarea que definimos previamente y le asignamos un nombre a nuestro service.

