

Mi primera Imagen en ECR.



# Amazon ECR

Amazon Elastic Container Registry (ECR) es un registro de contenedores de [Docker](#) completamente administrado que facilita a los desarrolladores las tareas de almacenamiento, administración e implementación de imágenes de contenedores de Docker.

Amazon ECR hospeda sus imágenes en una arquitectura escalable y de alta disponibilidad, lo que le permite implementar contenedores para sus aplicaciones con fiabilidad.

Vamos a ver como podemos subir una imagen de nuestra aplicación a Elastic Container Registry , a continuación se muestra el resumen de las tareas que realizaremos :

- *Crea una aplicación web simple.*
- *Crea una imagen Docker de la aplicación web.*
- *Enviar nuestra imagen Docker a AWS ECR.*

## 1.-Aplicación Web

Construiremos un aplicación web muy sencilla que consta de un archivo html.

## 2.-Crear imagen Docker de la Aplicación

Para poder crear nuestra imagen , necesitamos tener instalado Docker, utilizaremos la consola web de AWS y su herramienta integrada AWS CloudShell , que ya incluye Docker y la AWS CLI preinstalados, eliminando la necesidad de trabajar en tu máquina local.

Docker puede construir imágenes automáticamente, leyendo las instrucciones indicadas en un fichero Dockerfile. Se trata de un documento de texto que contiene todas las órdenes a las que un usuario dado puede llamar, desde la línea de comandos, para crear una imagen.

A continuación, se detalla la secuencia de comandos y pasos ajustados para usarlos directamente en AWS CloudShell.

Abrir AWS CloudShell: Inicia sesión en la consola de AWS y haz clic en el icono de CloudShell (un pequeño cuadro negro con el símbolo de una línea de comandos) en la barra de navegación superior.

Crear el directorio del proyecto:

```
mkdir demoNinjaDocker
```

```
cd demoNinjaDocker
```

Crear el archivo index.html: Usa un editor de texto como nano o vi para crear el archivo con el contenido de tu laboratorio, o usa el siguiente comando para hacerlo rápidamente:

```
cat > index.html <<EOF
<!DOCTYPE html>
<html>
<body style="background-color: rgb(210, 226, 250)">
<h1>Bienvenido al Taller Ninja</h1>
<p>Ninja Project BBVA</p>
<p>Application Version: V1</p>
</body>
</html>
EOF
```

Crear el Dockerfile: De manera similar, crea el Dockerfile con el contenido del laboratorio (que usa la imagen base de Nginx y copia el HTML).

```
cat > Dockerfile <<EOF
FROM nginx
COPY index.html /usr/share/nginx/html
EOF
```

Construir la imagen de Docker: En el directorio ~/demoNinjaDocker, ejecuta el comando docker build y etiqueta la imagen con el nombre que usarás.

```
docker build -t ninja-docker .
```

Nota: El comando original de tu laboratorio usa el nombre ninja-docker y el punto (.) indica que busque el Dockerfile en el directorio actual.

### 3.-Enviar nuestra imagen a AWS ECR

Antes de poder enviar nuestra imagen , necesitamos crear nuestro repositorio en ECR. Para eso nos dirigimos al panel de Elastic Container Registry y creamos nuestro repositorio.

ECR > Repositories > Create repository

## Create repository

### General settings

**Repository name**  
Provide a concise name. A developer should be able to identify the repository contents by the name.

818929933852.dkr.ecr.us-east-1.amazonaws.com/

10 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

**Tag immutability**  
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☐ Disabled

### Image scan settings

**Scan on push**  
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

☐ Disabled

Ahora tenemos nuestro repositorio para poder subir nuestra imagen. Pero antes de eso, necesitamos autenticar nuestra AWS CLI para enviar imágenes a nuestro repositorio. Para este caso necesitamos ingresar a nuestro repositorio y le damos click en View push commands y encontraremos los comandos necesarios para poder subir nuestra imagen.

ECR > Repositories > demo-ninja

## demo-ninja

[View push commands](#)

**Images (0)**

Image tag	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
No images						
No images to display						

### Push commands for demo-ninja

methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

1. Retrieve an authentication token and authenticate your Docker client to your registry.  
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t demo-ninja .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag demo-ninja:latest 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
```

4. Run the following command to push this image to your newly created AWS repository:

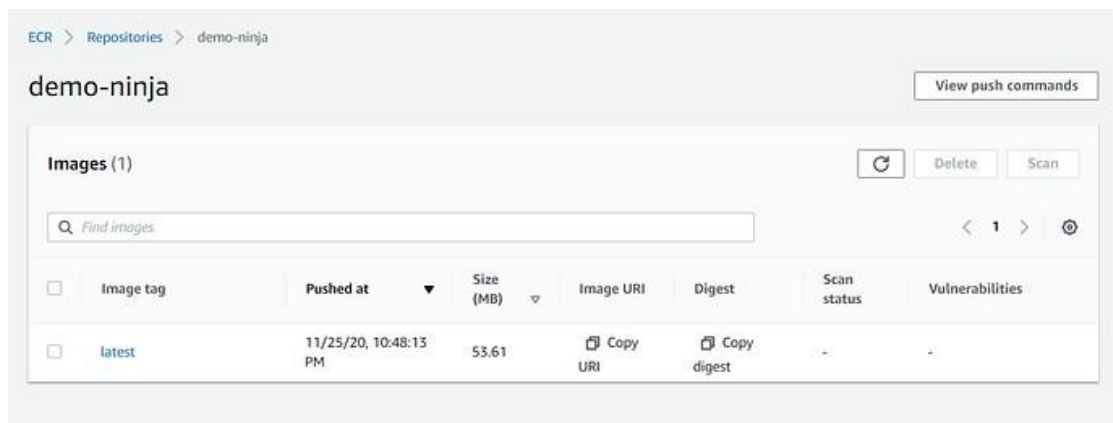
```
docker push 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
```

Close

Ejecutamos los comandos 1,3,4 en el orden indicado para poder subir nuestra imagen a nuestro repositorio.

```
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> aws ecr get-login-password
--region us-east-1 | docker login --username AWS --password-stdin 818929933852.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker tag demo-ninja:lates
t 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
Error response from daemon: No such image: demo-ninja:latest
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker tag ninja-docker:lat
est 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker push 818929933852.dk
r.ecr.us-east-1.amazonaws.com/demo-ninja:latest
The push refers to repository [818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja]
57e471e082e3: Pushed
7e914612e366: Pushed
f790aed835ee: Pushed
850c2400ea4d: Pushed
7ccabd267c9f: Pushing [=====>] 33.12MB/63.65MB
f5600c6330da: Pushing [=====>] 32.55MB/69.24MB
```

```
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> aws ecr get-login-password
--region us-east-1 | docker login --username AWS --password-stdin 818929933852.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker tag demo-ninja:lates
t 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
Error response from daemon: No such image: demo-ninja:latest
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker tag ninja-docker:lat
est 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
PS E:\Material Certificacion\Certificacion AWS\03.-Developer - DVA\Laboratorio-Docker\demoNinjaDocker> docker push 818929933852.dk
r.ecr.us-east-1.amazonaws.com/demo-ninja:latest
The push refers to repository [818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja]
57e471e082e3: Pushed
7e914612e366: Pushed
f790aed835ee: Pushed
850c2400ea4d: Pushed
7ccabd267c9f: Pushed
f5600c6330da: Pushed
latest: digest: sha256:13a1d70077a2c7b8a3be565f2ff6588f965e5c7700a347a8232d891de013d8c8 size: 1569
```



Al final de ejecutar todos los comandos indicados nuestra imagen debe de reflejarse en nuestro repositorio para poder ser utilizado, adicionalmente comentarte que en nuestro repositorio podemos tener mas de una versión de nuestra imagen.