



# Tecnológico de Monterrey

Inteligencia artificial avanzada para la ciencia de datos 2

Gpo 501

## **Docentes**

Dr. Benjamín Valdés Aguirre

Ma. Eduardo Daniel Juárez Pineda

Dr. Ismael Solis Moreno

Dr. José Antonio Cantoral Ceballos

Dr. Carlos Alberto Dorantes Dosamantes

## **Integrantes**

Carlos Rodrigo Salguero Alcántara	A00833341
Diego Perdomo Salcedo	A01709150
Dafne Fernández Hernández	A01369230
José Emiliano Riosmena Castañón	A01704245
Luis Arturo Rendón Iñarritu	A01703572

Querétaro, Querétaro

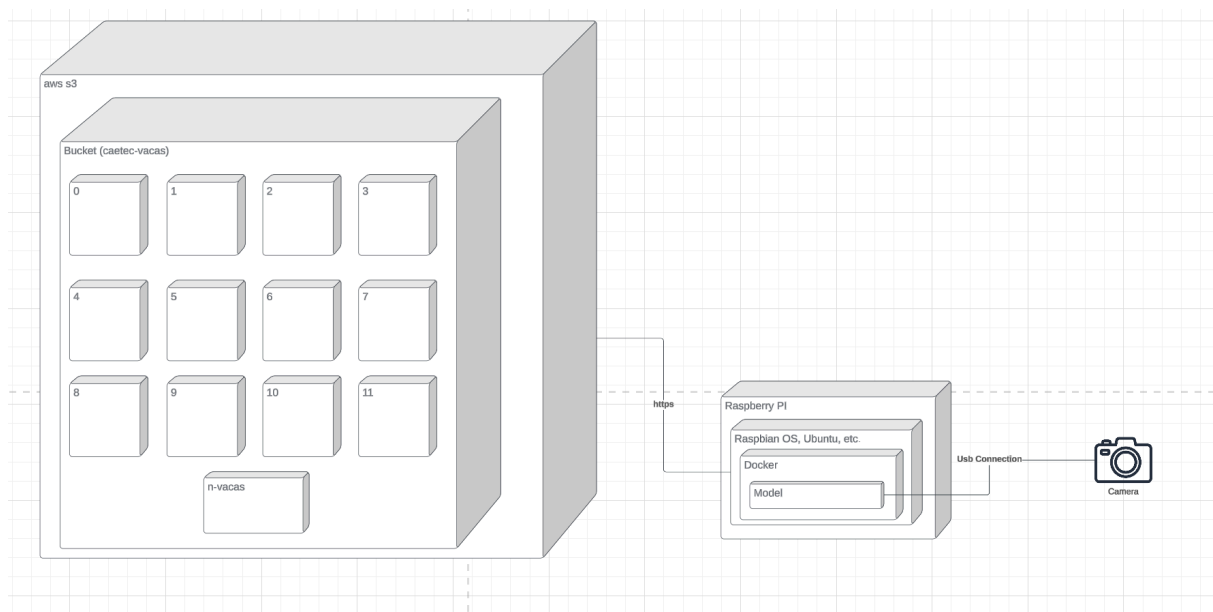
## 1.0 Descripción general del sistema

El sistema de detección de vacas es una aplicación que utiliza inteligencia artificial para contar vacas en tiempo real a través de una cámara web. Las imágenes capturadas se almacenan automáticamente en AWS S3, organizadas en carpetas según la cantidad de vacas detectadas.

### 1.1 Características principales

- Detección en tiempo real de vacas mediante YOLOv8
- Almacenamiento automático en AWS S3
- Procesamiento de imágenes cada cierta cantidad de segundos
- Organización automática por cantidad de vacas detectadas

### 1.2 Manual de despliegue



## 2.0 Requisitos previos

### 2.1 Hardware

- Cámara web compatible con Linux (conectada a /dev/video0)

- PC (Raspberry PI 4 por ejemplo, cualquier PC con estas características funciona)
  - Procesador ARM64 compatible con las dependencias de PyTorch
  - Mínimo 4GB de RAM recomendado

## 2.2 Software

- Docker Engine (versión 20.10.0 o superior)
- Docker Compose (versión 2.0.0 o superior)
- Acceso a Internet para la descarga de imágenes y modelos
- Distribución desarrollada a partir de Debian, como Ubuntu o Raspberry PI OS

## 2.3 Credenciales y accesos

- Cuenta de AWS con acceso programático
- Credenciales de AWS (Access Key ID y Secret Access Key)
- Bucket S3 previamente creado

## 3.0 Estructura del proyecto

```
proyecto/  
├── docker-compose.yml  
├── .env  
└── logs/
```

## 4.0 Configuración del entorno (En la Raspberry PI, o en la PC)

En los siguientes puntos se presentan los comandos para preparar el entorno donde se ejecutará la imagen de Docker la cual ya contiene el modelo preestablecido dentro.

El procesador de la computadora necesita ser de arquitectura ARM64 y con un sistema operativo basado en Linux, específicamente una distribución desarrollada a partir de Debian, como Ubuntu o Raspberry PI OS como ya se mencionó anteriormente.

## 4.1 Instalación de Docker Engine

Lo mejor es acceder a la documentación oficial para la instalación de Docker Engine:

- <https://docs.docker.com/engine/install/ubuntu/>

De igual manera se colocarán los comandos necesarios para la instalación aquí mismo, de encontrarse un error o de ser comandos obsoletos, favor de consultar la documentación oficial.

Primero ejecutaremos el siguiente comando para desinstalar cualquier paquete o dependencia previamente instalada en el sistema y prevenir conflictos con la nueva versión e instalación de Docker.

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc;  
do sudo apt-get remove $pkg; done
```

Ahora configuraremos el repositorio de apt para poder instalar Docker:

```
# Add Docker's official GPG key:
```

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

Una vez configurado el repositorio, es momento de instalar Docker con el siguiente comando:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin
```

Para poder ejecutar Docker sin conflictos hay que agregarlo a la lista de superusuarios:

```
sudo groupadd docker
```

```
sudo usermod -aG docker $USER
```

Y por último, para comprobar que Docker está instalado correctamente, se puede utilizar el siguiente comando:

```
sudo docker run hello-world
```

De estar todo en orden se descargará la imagen y se ejecutará mostrando un mensaje de confirmación.

## 4.2 Preparación de directorios

Ubicándonos en el lugar en donde correremos la imagen de Docker se creará un directorio con el nombre de logs/

```
mkdir -p logs
```

## 4.3 Archivo .env

Para pasarle a la imagen las credenciales de seguridad y darle acceso a la Bucket de S3 se necesita crear un archivo llamado .env con el siguiente contenido:

```
AWS_ACCESS_KEY_ID=tu_access_key_id
```

```
AWS_SECRET_ACCESS_KEY=tu_secret_access_key
```

`AWS_REGION=tu_region`

`S3_BUCKET_NAME=nombre_de_tu_bucket`

`CAMERA_SECONDS=el_tiempo_entre_fotos_en_segundos`

## 4.4 Docker Compose

Tenemos que agregar en la misma ubicación el archivo docker-compose.yml el cual está ubicado en el repositorio de GitHub, en el siguiente link:

- <https://github.com/salgue441/cow-project/blob/main/deployment/docker-compose.yml>

## 4.5 Descargar imagen de docker

La imagen se encuentra en “Docker Hub”, una plataforma donde los usuarios pueden subir sus imágenes para que sean utilizadas por el público. La imagen con el modelo corriendo se encuentra en el siguiente link:

- <https://hub.docker.com/repository/docker/sen00/auto-deploy-cow-project/general>

Por último tenemos que bajar la imagen a la Raspberry PI y necesitamos usar este comando en la misma:

`docker pull sen00/auto-deploy-cow-project`

## 5.0 AWS

Para el correcto funcionamiento del modelo es necesario generar una Bucket de S3 en el proveedor de servicios AWS.

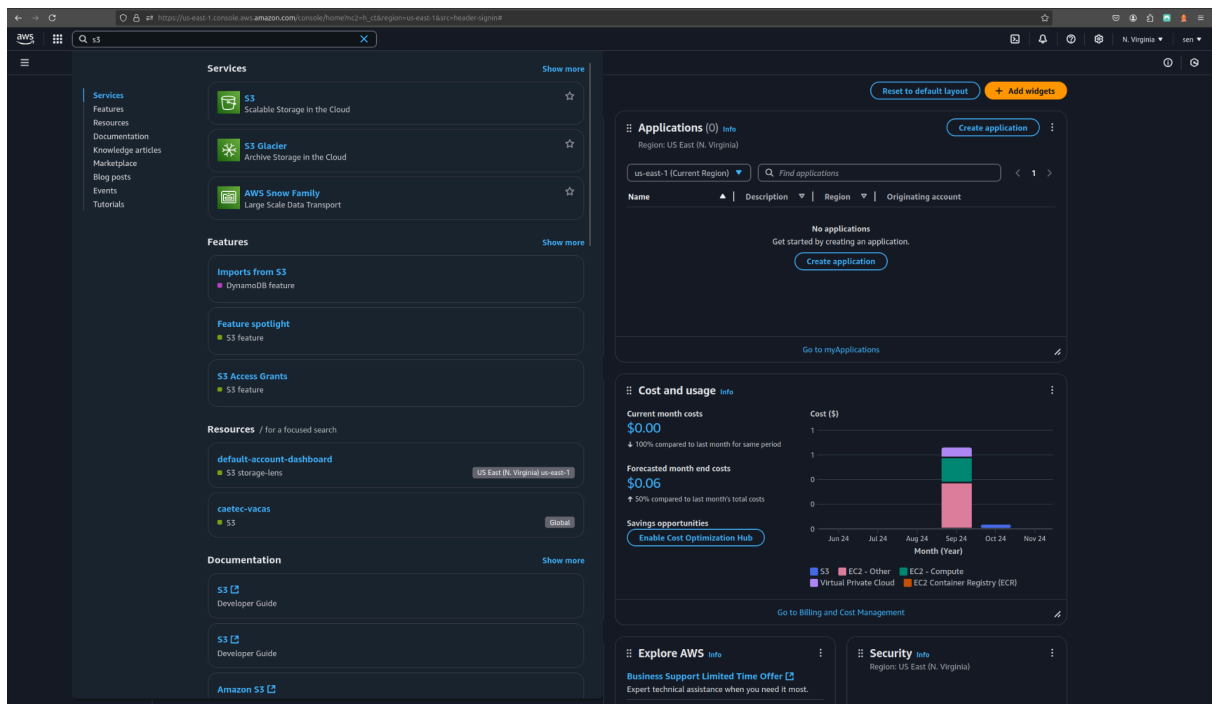
### 5.1 Creación de cuenta

Para poder acceder a los servicios de AWS primero se necesita crear una cuenta en el sistema mediante la siguiente liga:

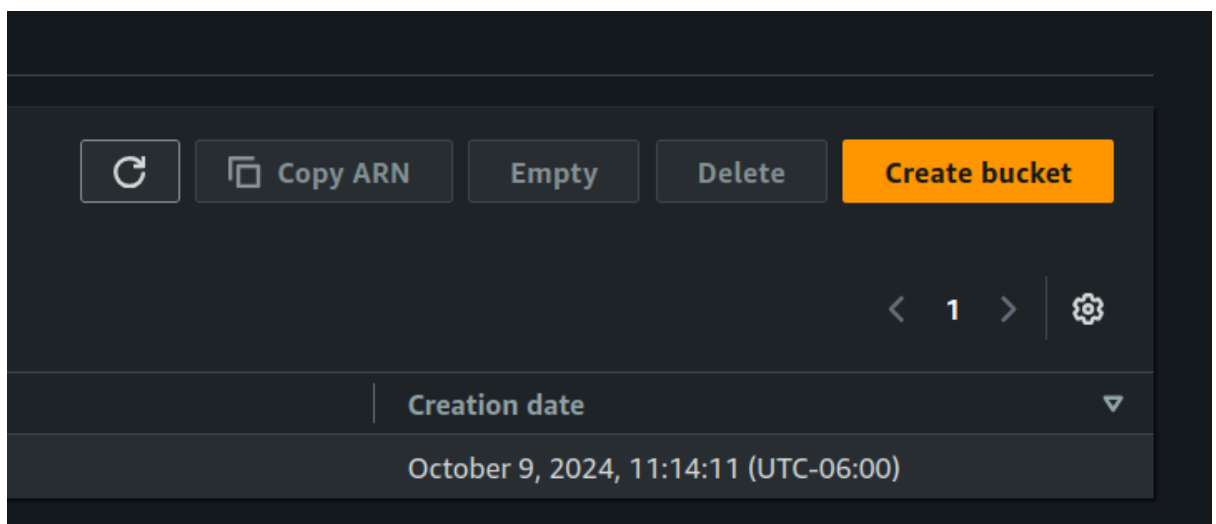
- <https://aws.amazon.com/>

## 5.1 Creación de Bucket S3

Para crear la Bucket en donde se almacenarán las imágenes procesadas, primero se necesita buscar el servicio correspondiente en la barra de búsqueda superior en la consola:



Ahora seleccionamos el botón de “Create Bucket”



En la configuración general dejamos “General purpose” y nombramos la Bucket como deseemos.

**General configuration**

AWS Region  
US East (N. Virginia) us-east-1

Bucket type [Info](#)

☒ **General purpose**  
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory**  
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

nombre-test

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.

**Choose bucket**

Format: s3://bucket/prefix

Toda la configuración adicional no la necesitamos cambiar en nada y al final le damos “Create Bucket”. El valor que pusiste en “Bucket name” en este caso “nombre-test” es el que es necesario en el archivo .env mencionado en el punto 5.1.

**Bucket Key**  
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ Disable

☒ Enable

► **Advanced settings**

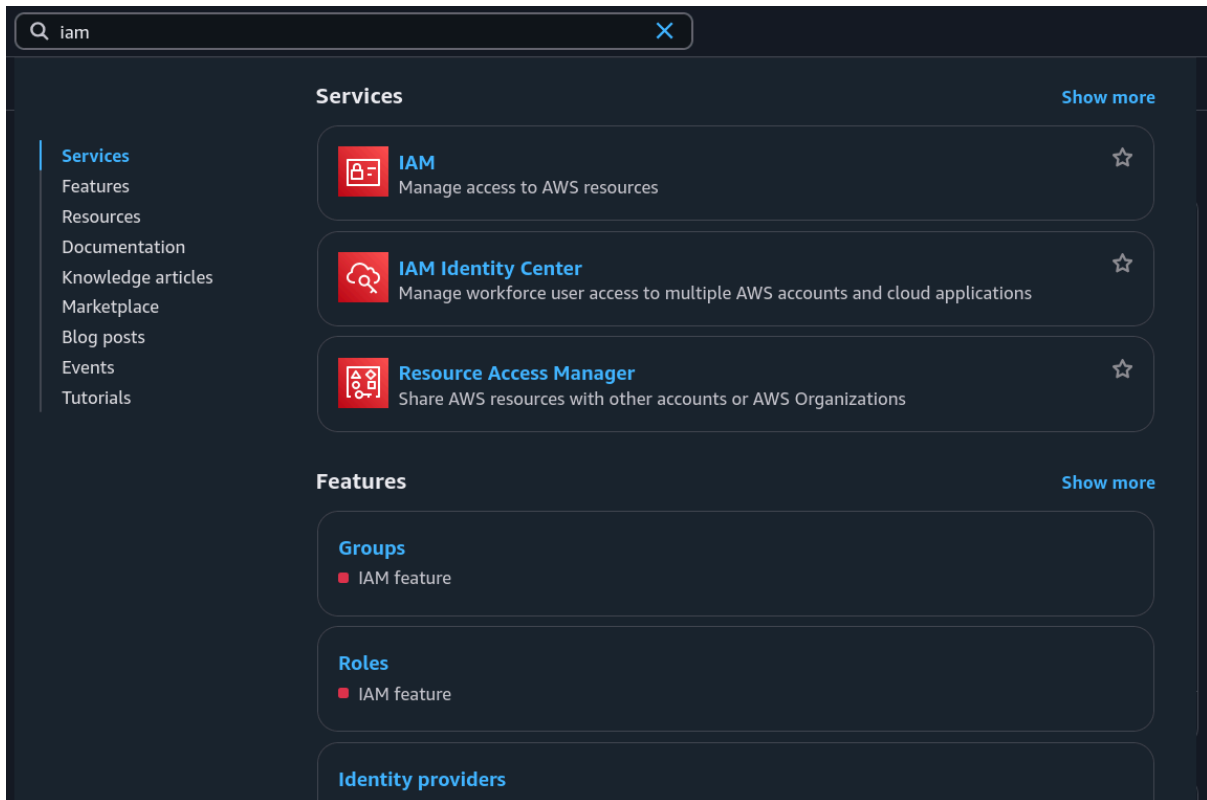
**After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.**

Cancel **Create bucket**

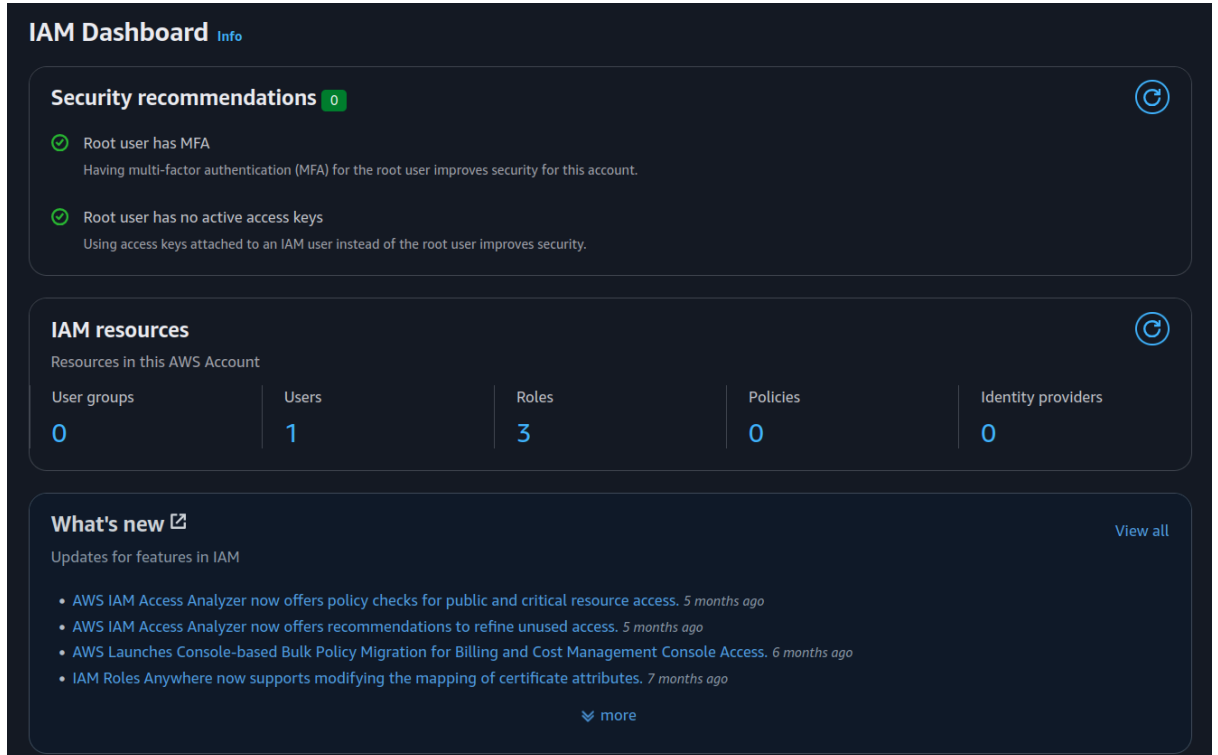
## 5.2 Creación de cuenta para acceso externo a la Bucket

Esta vez en la barra superior buscaremos iam para acceder a las cuentas y sus permisos.

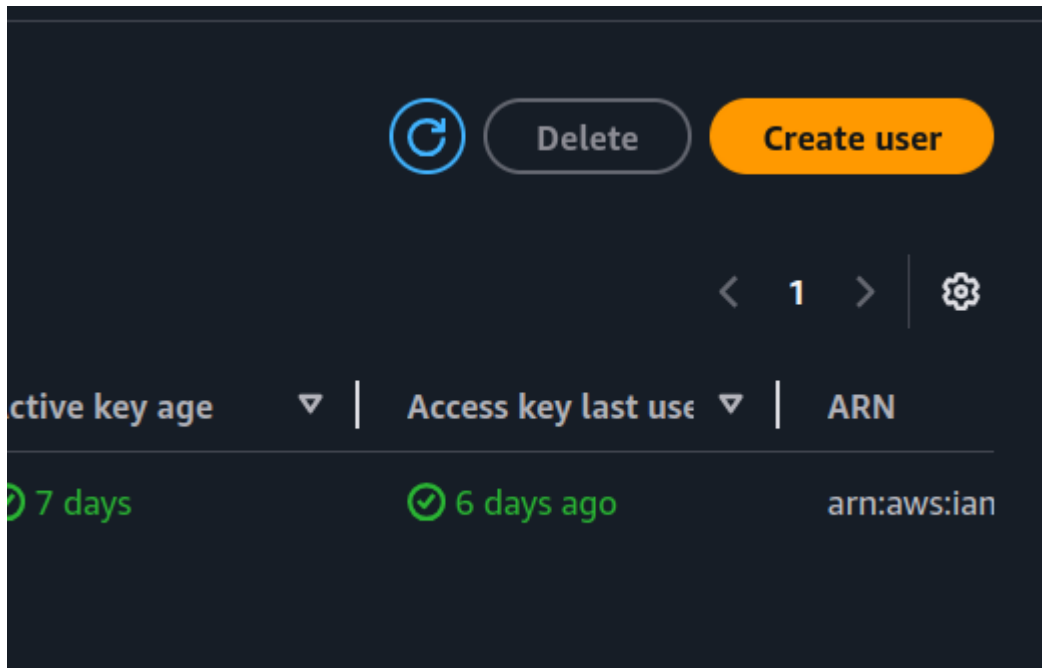




Seleccionamos “Users”.



Y una vez dentro de los usuarios seleccionamos “Create User”.



El nombre del usuario puede ser el que se desee y “Provide user access...” se dejará desmarcado. Ahora se seleccionará “Next”.

A screenshot of the 'Specify user details' form in the AWS IAM console. The form has a section titled 'User details' with a 'User name' input field. Below the input field, there is a note: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)'. There is a checkbox labeled 'Provide user access to the AWS Management Console - optional' with a sub-note: 'If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.' At the bottom right, there are 'Cancel' and 'Next' buttons. A blue information box at the bottom states: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more'.

Ahora seleccionaremos “Attach policies directly”.

### Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

#### Permissions options

☐ Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.







☐ Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly  
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

#### Permissions policies (1280)

Filter by Type  
All types

< 1 2 3 4 5 6 7 ... 64 >

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	 <a href="#">AccessAnalyzerServiceRolePolicy</a>	AWS managed	0
<input type="checkbox"/>	 <a href="#">AdministratorAccess</a>	AWS managed - job function	0
<input type="checkbox"/>	 <a href="#">AdministratorAccess-Amplify</a>	AWS managed	0
<input type="checkbox"/>	 <a href="#">AdministratorAccess-AWSElasticBeans...</a>	AWS managed	0
<input type="checkbox"/>	 <a href="#">AlexaForBusinessDeviceSetup</a>	AWS managed	0
<input type="checkbox"/>	 <a href="#">AlexaForBusinessFullAccess</a>	AWS managed	0

Y seleccionamos el permiso “AmazonS3FullAccess”.

## Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

### Permissions options

☐ Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly  
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

### Permissions policies (1/1280)

Choose one or more policies to attach to your new user.

Filter by Type
All types
14 matches

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS managed	0
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS managed	1
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRol...	AWS managed	0
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS managed	0
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS managed	0
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS managed	0
<input type="checkbox"/>	AWSBackupServiceRolePolicyForS3Ba...	AWS managed	0
<input type="checkbox"/>	AWSBackupServiceRolePolicyForS3Res...	AWS managed	0
<input type="checkbox"/>	AWSQuickSetupSSMDeploymentS3Bu...	AWS managed	0
<input type="checkbox"/>	AWSSES3OnOutpostsServiceRolePolicy	AWS managed	0
<input type="checkbox"/>	IVSRecordToS3	AWS managed	0
<input type="checkbox"/>	QuickSightAccessForS3StorageManag...	AWS managed	0
<input type="checkbox"/>	S3StorageLensServiceRolePolicy	AWS managed	0
<input type="checkbox"/>	S3UnlockBucketPolicy	AWS managed	0

► Set permissions boundary - optional

Seleccionamos en la parte inferior “Next” y en la siguiente ventana seleccionamos “Create User”.

## Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

### User details

User name

user-test

Console password type

None

Require password reset

No

### Permissions summary

Name	Type	Used as
AmazonS3FullAccess	AWS managed	Permissions policy

### Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

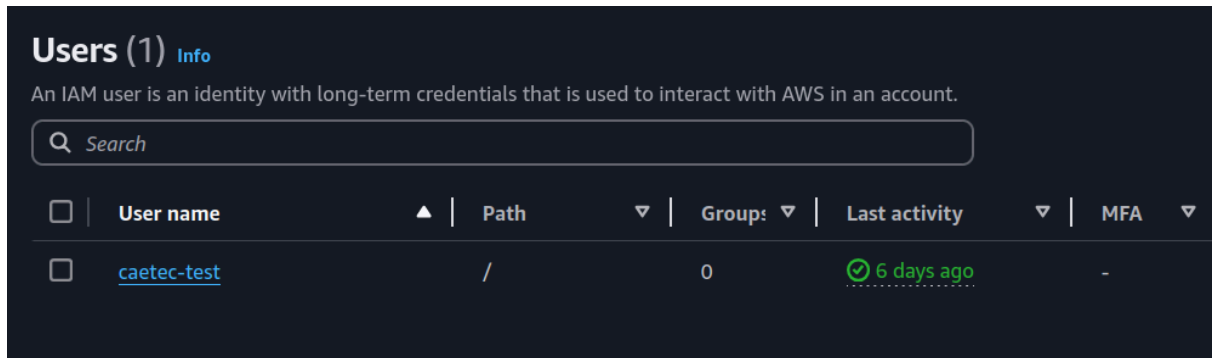
[Add new tag](#)

You can add up to 50 more tags.

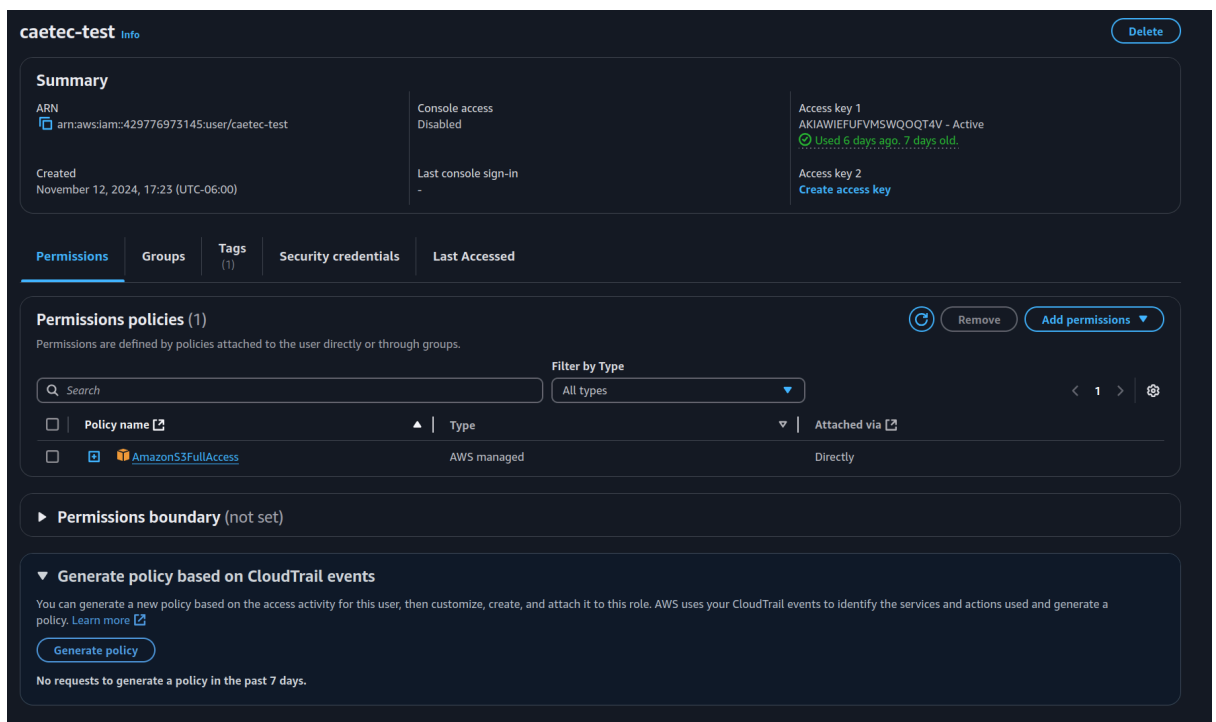
[Cancel](#)
[Previous](#)
[Create user](#)

## 5.3 Creación de claves de acceso

Volvemos a buscar en la parte superior iam y accedemos a los usuarios de nuevo. Ahora seleccionamos el nombre del usuario recién creado.



En la nueva ventana se puede ver la información general del usuario y el permiso de S3 que se le añadió durante su creación. En esta ventana seleccionamos el apartado de “Security Credentials”.



Más abajo en “Access Keys” seleccionamos “Create access key”.

**Access keys (1)** [Create access key](#)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

<b>AKIAWIEFUFVMSWQQQT4V</b>	<b>Status</b> Active
<b>Description</b> caetec docker image test	<b>Created</b> 7 days ago
<b>Last used</b> 6 days ago	<b>Last used service</b> s3
<b>Last used region</b> us-east-1	

[Actions](#)

En la nueva ventana seleccionamos “Local code” y marcamos “I understand the above...”.

### Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

**Use case**

- ☐ Command Line Interface (CLI)  
You plan to use this access key to enable the AWS CLI to access your AWS account.
- ☒ Local code  
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- ☐ Application running on an AWS compute service  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- ☐ Third-party service  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- ☐ Application running outside AWS  
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- ☐ Other  
Your use case is not listed here.

**Alternative recommended**  
Use an Integrated Development Environment (IDE) which supports the AWS Toolkit enabling authentication through IAM Identity Center. [Learn more](#)

**Confirmation**  
☒ I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

Ponemos cualquier descripción que nos ayude a identificar la clave de acceso en la consola y seleccionamos “Create access key”.

**Set description tag - optional** [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

**Description tag value**  
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @



[Cancel](#) [Previous](#) [Create access key](#)

En la última ventana nos presentan el “Access key” y el “Secret access key”.

et access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

**Retrieve access keys** [Info](#)

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
 AKIAWIEFUFVMVQY3CM63	 ***** <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

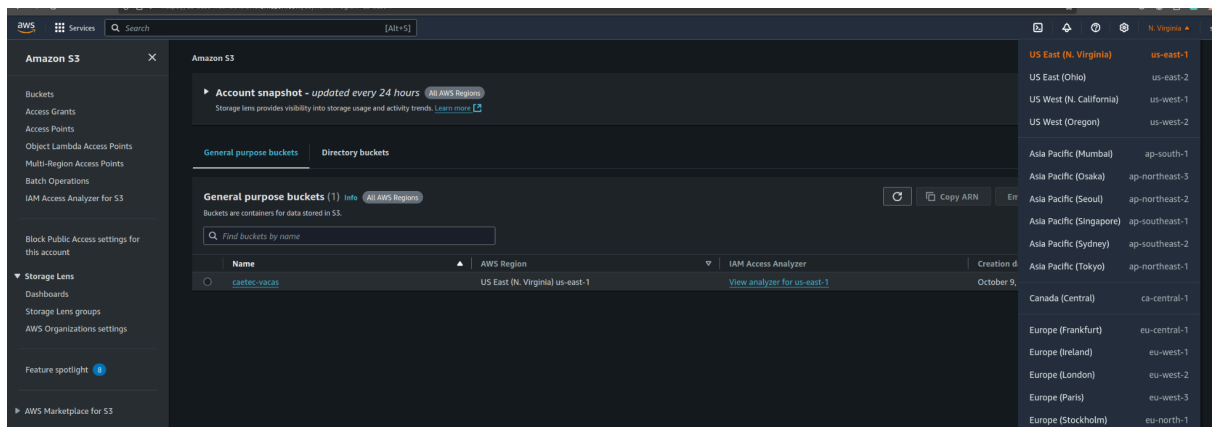
For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

Estos valores deberán ser guardados, ya que el “Secret access key no podrá ser consultado una vez que se seleccione “Done”. Estos valores son los que van en el .env creado en el punto 4.3.

## 5.4 Región

AWS permite el despliegue de infraestructura en distintas regiones o zonas del planeta, para ver la región en la que se encuentra la Bucket, en el menú s3 al que habíamos accedido en el paso 5.1, se puede revisar la esquina superior derecha.



En este caso, la región sería “us-east-1”. Este valor también es necesario en el .env mencionado en el punto 5.1.

US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1



## 6.0 Ejecución (En la Raspberry PI, o en la PC)

Una vez realizados los pasos anteriores, nos ubicamos en el índice del proyecto, o la carpeta principal del mismo. En el caso de la estructura de proyecto presentada antes, este sería en la carpeta proyecto/ y una vez ahí ejecutaremos el comando

- `docker composer up -d`

### 6.1 Verificar estado del contenedor

Para poder comprobar el estado del contenedor y del modelo en ejecución se pueden ejecutar los siguientes dos comandos:

- `docker ps`
- `docker logs cow-detector`

### 6.2 Verificar logs del contenedor

Para revisar los logs del contenedor en cuestión se puede utilizar el siguiente comando:

- `docker logs -f cow-detector`

### 6.3 Revisar AWS

Una vez que el contenedor y el modelo se ejecuten sin problemas, las imágenes serán clasificadas y subidas a la Bucket S3 en AWS de manera automática en carpetas de 0 a n, siendo n el número de vacas encontradas en la imagen.

Se puede acceder al menú de S3 en la consola de AWS, en la cual se podrán ver las carpetas y las imágenes subidas, con las bounding boxes correspondientes.

### 6.4 Detener contenedor y modelo

Para detener el contenedor de Docker solo se necesita ubicar en el índice del proyecto, en el mismo lugar donde se encuentra el docker-compose.yml mencionado en el paso 4.4. Una vez ahí solo se necesita usar el comando:

- `docker compose down`

