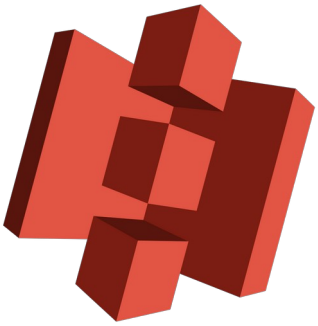
Almacenamiento de objetos de alto rendimiento con MinIO



Francisco Diz Ureña

2. Índice

3.Objetivos que se quieren conseguir y se han conseguido

3.1.Objetivos que se quieren conseguir

Los objetivos que se desean conseguir son:

- Conocer el concepto de almacenamiento de objetos.
- Diferencias entre las distintas opciones de almacenamiento actuales.
- Conocer los servicios similares de los distintos proveedores.
- Instalar y configurar servidor MinIO local.
- Instalación del cliente de MinIO.
- Instalación del cliente de AWS.
- Configuración para HA.
- Incorporación de MinIO con AWS S3.
- Servir una aplicación/pagina web estática obteniendo los recursos mediante MinIO.

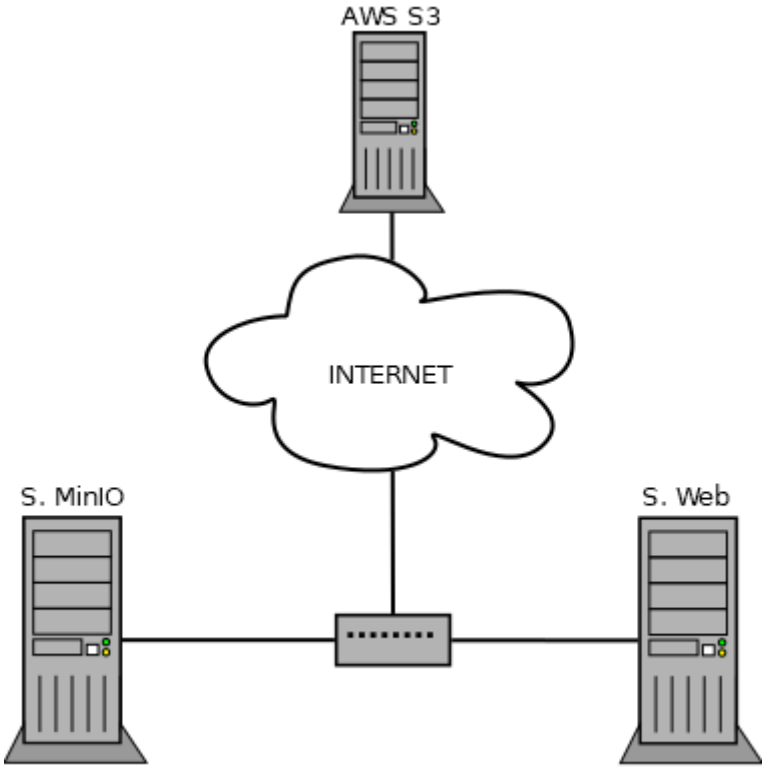
3.2.Objetivos que se han conseguido

Los objetivos que se han conseguido son:

- Conocer el concepto de almacenamiento de objetos.
- Diferencias entre las distintas opciones de almacenamiento actuales.
- Conocer los servicios similares de los distintos proveedores.
- Instalar y configurar servidor MinIO local.
- Instalación del cliente MinIO.
- Incorporación de MinIO con AWS S3.
- Servir una aplicación/pagina web estática obteniendo los recursos mediante MinIO.

4.Escenario necesario para la realización del proyecto

Para este proyecto se necesitará:



Este escenario tal y como podemos observar en la imagen constará de tres máquinas. En este caso tanto S.MinIO y S.Web serán máquinas Ubuntu 22.04 locales, y por otro lado el servicio S3 de Amazon Web Services.

Lo que se hará será lo siguiente:

1º Montaremos un servidor de almacenamiento MinIO local. A este servidor le pondremos un nombre el cual utilizaremos posteriormente.
2º Serviremos una aplicación web/página web cuyo contenido estático almacenaremos en nuestro servidor MinIO.
3º Tendremos que hacer las modificaciones oportunas en los ficheros .html para que tiren de las nuevas rutas del contenido estático.
4º A través del cliente de MinIO instalado en S.MinIO, subiremos parte del contenido estático a S3 de AWS.
5º Modificaremos de nuevo los ficheros .html de S.Web y veremos como se muestra todo el contenido estático de la aplicación correctamente.

5.Si es necesario los fundamentos teóricos y conceptos.

5.1.¿Qué es MinIO?

MinIO es un sistema de almacenamiento de objetos de código abierto diseñado para la nube. Proporciona una forma escalable y de alto rendimiento de almacenar y acceder a grandes volúmenes de datos no estructurados, como imágenes, vídeos, archivos de registro, copias de seguridad y otros tipos de archivos.

A diferencia de los sistemas tradicionales de almacenamiento en bloques o archivos, MinIO utiliza una arquitectura de almacenamiento distribuido basada en objetos. Esto significa que los datos se dividen en objetos individuales, cada uno con su propia identificación única, y se almacenan en múltiples servidores en un cluster. Esta distribución de datos mejora la disponibilidad y la tolerancia a fallos, así como el rendimiento de lectura y escritura.

MinIO es compatible con el protocolo S3 de Amazon Web Services (AWS), lo que permite a las aplicaciones y herramientas existentes que utilizan S3 integrarse fácilmente con MinIO. Esto hace que sea una opción popular para construir soluciones de almacenamiento en la nube privada o híbrida.

Además, ofrece características como la replicación de datos, la compresión, el cifrado y la gestión de políticas de acceso, lo que permite a los usuarios controlar y administrar sus datos de manera eficiente y segura.

Al ser un proyecto de código abierto, MinIO cuenta con una comunidad activa de desarrolladores y usuarios que contribuyen a su desarrollo y mejora continua. También se ofrece una versión empresarial llamada MinIO Enterprise, que proporciona funciones adicionales y soporte comercial para entornos empresariales.

5.2.¿Qué es el almacenamiento de objetos y en que consiste?

El almacenamiento de objetos es una forma de almacenar y gestionar datos no estructurados a gran escala. En lugar de organizar los datos en una jerarquía de carpetas como en el almacenamiento de archivos tradicional, el almacenamiento de objetos trata cada archivo o conjunto de datos como un objeto individual. Cada objeto se guarda en un sistema de almacenamiento y se le asigna una clave única para su identificación.

Este se utiliza ampliamente en entornos de almacenamiento en la nube debido a su escalabilidad, flexibilidad y eficiencia en el manejo de grandes volúmenes de datos. También se utiliza en sistemas de almacenamiento local y en aplicaciones empresariales para organizar y gestionar eficientemente archivos y datos.

Este tipo de almacenamiento es especialmente útil en entornos de análisis de datos y Big Data, donde se procesan grandes volúmenes de datos no estructurados. Los objetos individuales pueden representar registros de eventos, datos de sensores, registros de servidor y otros tipos de datos

utilizados en análisis y minería de datos. La capacidad de acceder y procesar estos datos a través de API facilita el análisis y la extracción de información valiosa.

5.3.Principales diferencias con las distintas opciones de almacenamiento.

El almacenamiento de objetos es un tipo de almacenamiento que se está utilizando cada vez más en diversos sectores y aplicaciones, ya que este ofrece una serie de ventajas frente a los otros tipos de almacenamiento.

Para un crecimiento exponencial de datos este tipo nos ofrece una solución eficiente para gestionar y escalar grandes volúmenes de datos no estructurados, que no se adaptan bien a los modelos tradicionales de almacenamiento basados en bloques o archivos.

Además, a diferencia del almacenamiento de archivos que utiliza nombres y rutas para acceder a los datos, en este el acceso a los datos requiere una API o una interfaz de usuario.

Ventajas frente a otros tipos de almacenamiento:

- Manejar grandes cantidades de datos no estructurados.
- Almacenamiento más eficiente al no existir una jerarquía rígida de archivos.
- Mejora las capacidades de búsqueda y análisis gracias a metadatos personalizables.
- Si es de pago el servicio como S3 de AWS, se paga únicamente por el uso que le demos.

Desventajas frente a otros tipos de almacenamiento:

- Es mucho menos conocido que otros tipos de almacenamiento como el de archivos o bloques.
- Hay que dedicar más tiempo a gestionar los metadatos.
- Una vez creados los objetos no podemos modificarlos y únicamente podemos volver a crearlos.
- No permite a sus usuarios bloquear archivos.

5.4.¿Qué es S3?

S3 o Simple Storage Service es un servicio de almacenamiento en la nube proporcionado por Amazon Web Services (AWS). Permite a los usuarios almacenar y recuperar grandes cantidades de datos de forma segura y escalable. Puede almacenar cualquier cantidad de datos y acceder a ellos en cualquier momento. Ofrece una interfaz simple para cargar y descargar archivos, y proporciona una URL única para cada objeto almacenado, lo que permite un fácil acceso a los datos a través de la web.

Algunas características que nos ofrece S3:

Escalabilidad: Almacenar una cantidad ilimitada de datos y puede manejar una gran cantidad de solicitudes simultáneas.

Durabilidad: Los datos se replican automáticamente en múltiples ubicaciones físicas, lo que garantiza una alta durabilidad y disponibilidad.

Seguridad: Proporciona opciones de seguridad robustas, como control de acceso basado en políticas, cifrado de datos en tránsito y en reposo, y registro de actividades para auditar el acceso a los datos.

Administración de datos: Ofrece diversas funciones para administrar los datos almacenados, como versionado de objetos, control de acceso a nivel de objeto y ciclo de vida de los datos.

Integración con otros servicios de AWS: Se integra con muchos otros servicios de AWS, como EC2, Lambda, CloudFront y Athena, lo que permite una fácil integración en arquitecturas y aplicaciones basadas en la nube.

5.5.Servicios similares de los distintos proveedores públicos.

En este proyecto hasta hora hemos hablado de MinIO que es un servidor de almacenamiento de objetos privado y también de S3 de AWS como público, pero aparte de estos existen una variedad de otros similares, tanto de carácter público como privado.

Servidores de almacenamiento de objetos públicos(en la nube):

- Google Cloud Storage o GCS
- Microsoft Azure Blob Storage
- IBM Cloud Object Storage

Servidores de almacenamiento de objetos privados(open source):

- OpenStack Swift
- Riak CS
- Apache Kafka

6.Descripción detalla de lo que se ha realizado.

6.1.Instalación del servidor MinIO.

La instalación del servidor la haremos a partir del binario que nos proporcionan en la web.

Descargamos desde la página oficial el binario:

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

Le asignamos permiso de ejecución al binario:

```
chmod +x minio
```

Una vez hecho esto tenemos que mover el binario a /usr/local/bin/:

```
mv minio /usr/local/bin/
```

EXPLICAR OTROS METODOS DE INSTALACION Y EN OTRO SISTEMA (rocky)

6.2.Configuración del servidor MinIO.

Ahora, para poder utilizarlo con facilidad, crearemos un servicio con systemd:

```
cat << EOF > /lib/systemd/system/minio.service
[Unit]
Description=MinIO
Documentation=https://docs.min.io
Wants=network-online.target
After=network-online.target
AssertFileIsExecutable=/usr/local/bin/minio
AssertFileNotEmpty=/etc/default/minio
[Service]
Type=notify
WorkingDirectory=/usr/local/
User=minio-user
Group=minio-user
ProtectProc=invisible
EnvironmentFile=/etc/default/minio
```

```
ExecStartPre=/bin/bash -c "if [ -z \"${MINIO_VOLUMES}\" ]; then
echo \"Variable MINIO_VOLUMES not set in>
ExecStart=/usr/local/bin/minio server $MINIO_OPTS $MINIO_VOLUMES
# Let systemd restart this service always
Restart=always
# Specifies the maximum file descriptor number that can be opened
by this process
LimitNOFILE=1048576
# Specifies the maximum number of threads this process can create
TasksMax=infinity
# Disable timeout logic and wait until process is stopped
TimeoutStopSec=infinity
SendSIGKILL=no
[Install]
WantedBy=multi-user.target
EOF
```

Ya creado podemos habilitarlo y arrancarlo con la siguiente orden:

```
systemctl enable --now minio.service
```

Con el servicio ya creado, pasaremos a configurar el servidor. Para ello crearemos un fichero con lo siguiente:

```
cat << EOF > /etc/default/minio
# Carpeta o volumen que usará el servidor MinIO para el
almacenaje.
MINIO_VOLUMES="/data"
# Configuración de puertos.
MINIO_OPTS="--address :9199 --console-address :9001"
# Usuario root del servidor.
MINIO_ROOT_USER=admin
# Contraseña del usuario root del servidor.
MINIO_ROOT_PASSWORD=password
```

```
# Nos permite reiniciar con el comando “mc admin service restart”
MINIO_CONFIG_ENV_FILE=/etc/default/minio
EOF
```

Una vez aplicada reiniciamos el servicio con:

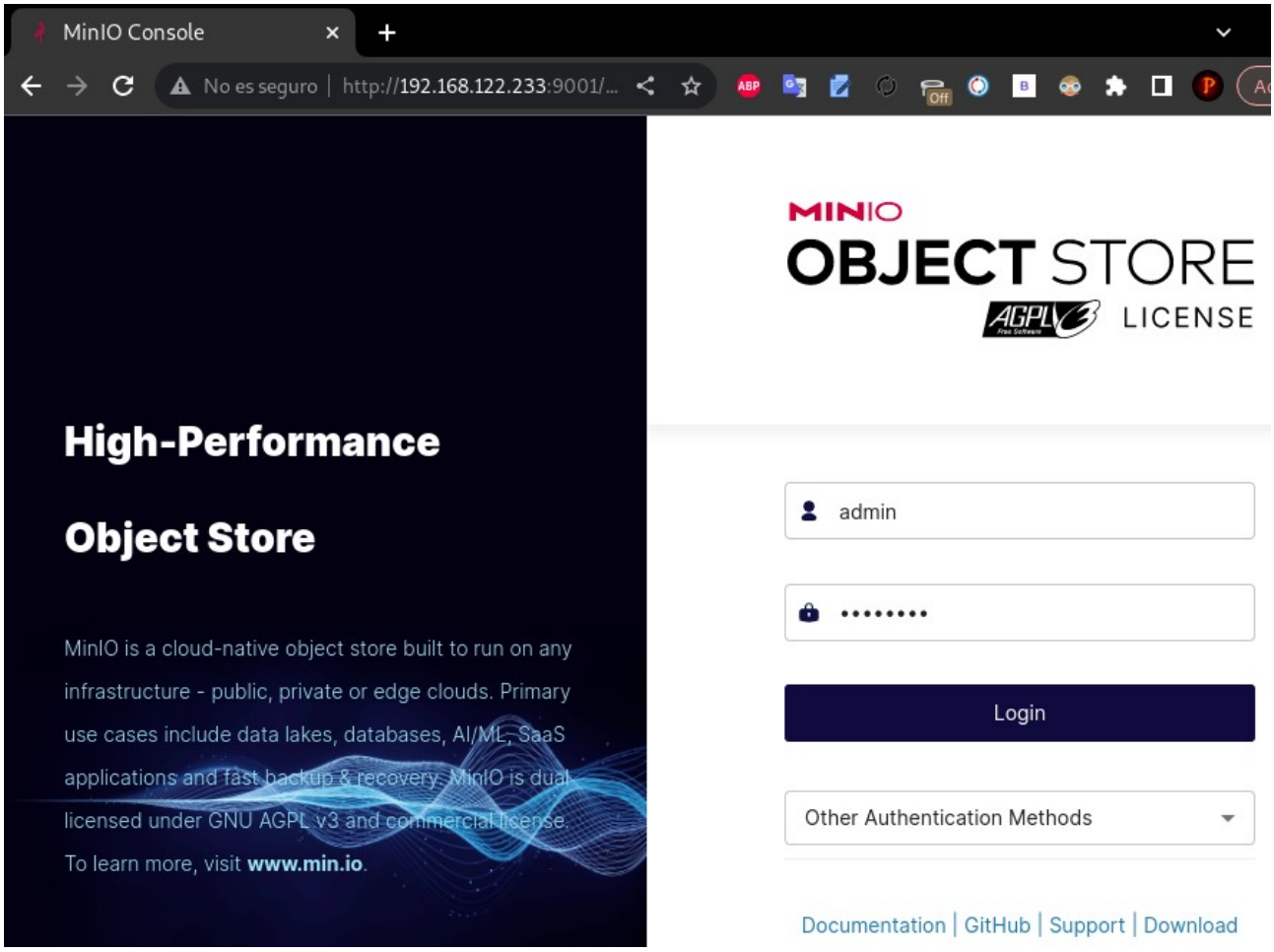
```
systemctl restart minio.service
```

Con todo esto realizado, podremos ver el estado del servicio ejecutando la orden:

```
systemctl status minio.service
```

Como podemos ver nos aparece el servicio levantado y nos indica la URL para entrar a la interfaz web.

Entramos y nos aparecerá una página web con un login. Recordamos que dicho usuario y contraseña son los que especificamos en el fichero de configuración.



Una vez introducido el login se abrirá el área de administración web del servidor.

6.3.Instalación del cliente MinIO.

La instalación del cliente, al igual que la del servidor, la haremos a partir del binario que nos proporcionan en la web.

Descargamos desde la página oficial el binario:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

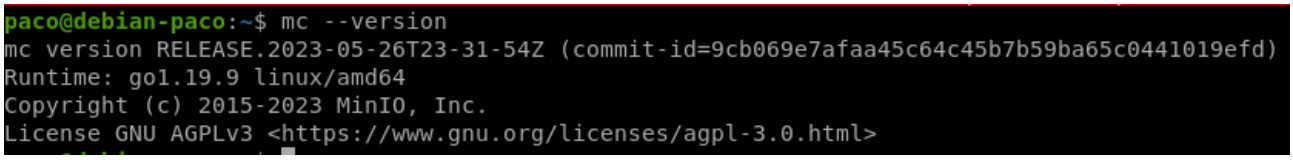
Le asignamos permiso de ejecución al binario:

```
chmod +x mc
```

Una vez hecho esto tenemos que mover el binario a /usr/local/bin/:

```
mv minio /usr/local/bin/
```

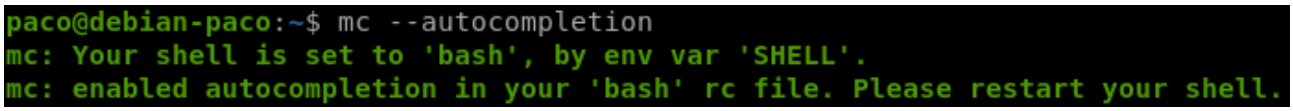
Como podemos ver en la imagen, se instaló correctamente.



6.4.Configuración del cliente MinIO.

Para que su uso nos sea más ameno podemos ejecutar el siguiente comando que activará el autocompletado del mismo:

```
mc --autocompletion
```



Una vez hecho esto ejecutaremos nuestra bash de nuevo:

```
cd $HOME && . .bashrc
```

Y ya tendremos habilitada el autocompletado.

Una vez instalado y habiendolo ejecutado una vez, por ejemplo --help se nos creará en el directorio HOME del usuario una carpeta oculta llamada .mc. En esta carpeta se guardará toda la configuración del cliente referente a ese usuario del sistema.

```
paco@debian-paco:~$ ls -la | grep "mc"
drwx------  5 paco paco         66 may 29 21:14 .mc
```

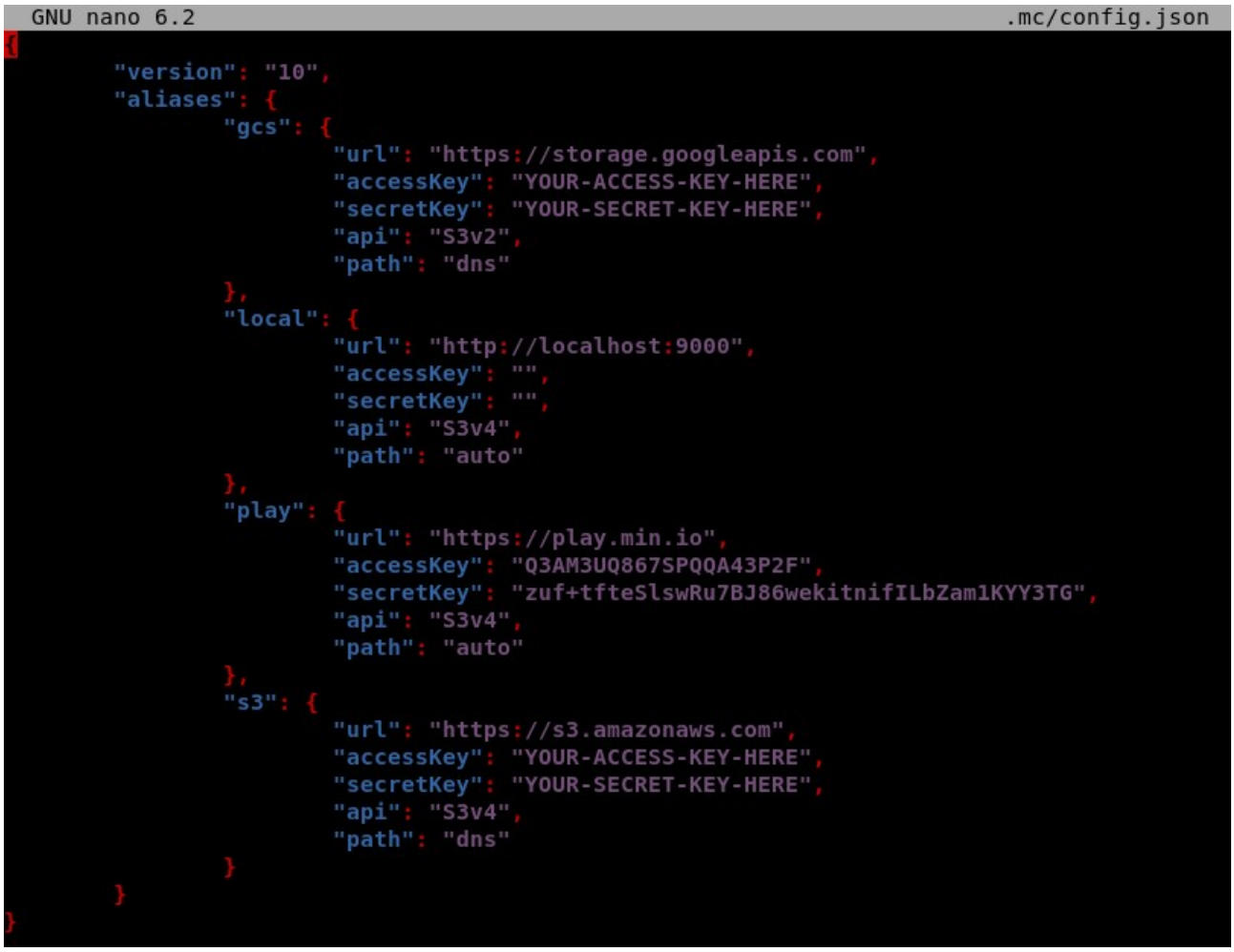
Este directorio tiene la siguiente estructura:

```
paco@debian-paco:~$ tree .mc/
.mc/
├── certs
│   └── CAs
├── config.json
├── session
└── share
    ├── downloads.json
    └── uploads.json

4 directories, 3 files
```

En la carpeta **certs/CAs/** se guardaran los certificados de las entidades certificadoras en el caso de usar una conexión cifrada.
En el fichero **config.json** se almacena la configuración global del cliente MinIO, incluyendo los alias de host, credenciales de acceso, ajustes de seguridad y otras opciones de configuración.
En la carpeta **session** se almacenan los datos de sesión generados cuando se autentica con un servidor MinIO. Estos datos de sesión incluyen tokens de acceso y otras información relacionada con la autenticación.
En la carpeta **share** podemos encontrar dos ficheros .json, **downloads.json** y **uploads.json**, los cuales son utilizados para guardar la información sobre las descargas y cargas de archivos respectivamente. Además, este directorio se utilizada para compartir archivos y carpetas con otros usuarios o equipos.

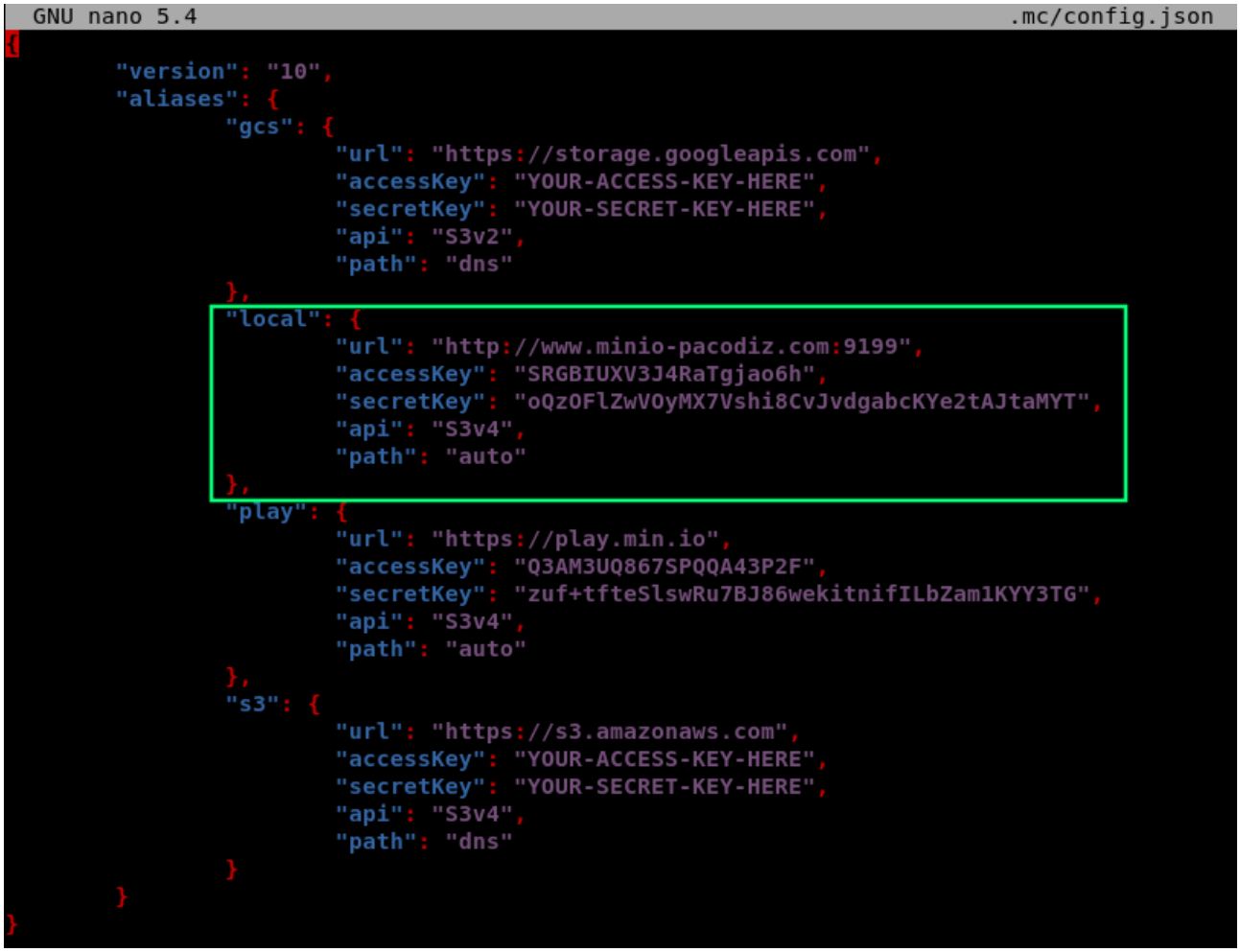
El fichero principal y por lo cual el más importante del cliente de MinIO es el config.json, mencionado anteriormente. Como comentamos este fichero contendrá todas las conexiones con los servidores, las cuales configuraremos para poder conectar a nuestro servidor minio y gestionarlo desde este.

Este es el contenido del fichero:



Podemos ver como ya viene una pequeña plantilla describiendo como sería una conexión a GCS de google, a S3 de Amazon o simplemente una local.

Para conectarnos a nuestro servidor local tendremos que modificar la “url” poniendo la de nuestro servidor y añadir una clave de acceso y una clave secreta las cuales tenemos que crear en el servidor(veremos posteriormente como crearlas):



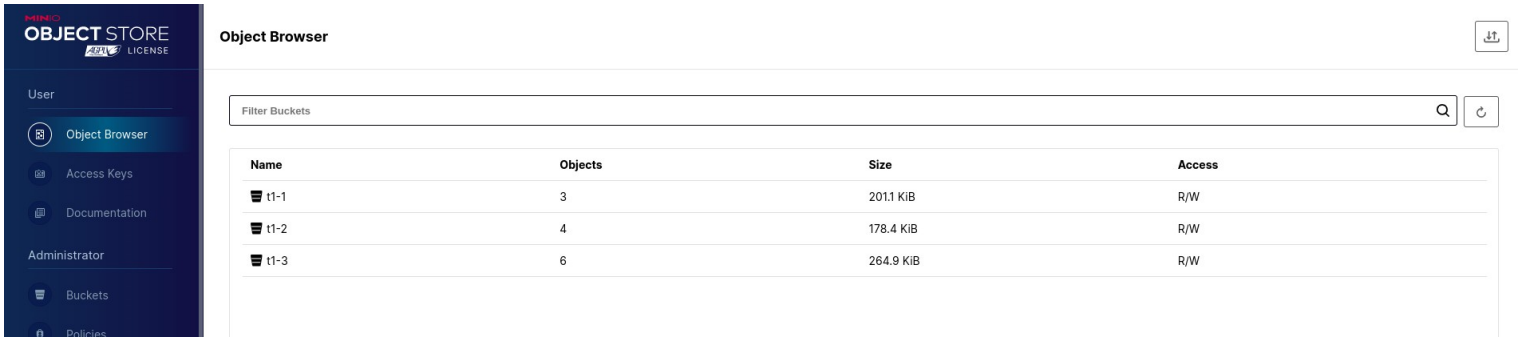
Con esto configurado ya podremos hacer uso de nuestro servidor desde el cliente MinIO.

6.5.Administración mediante interfaz web.

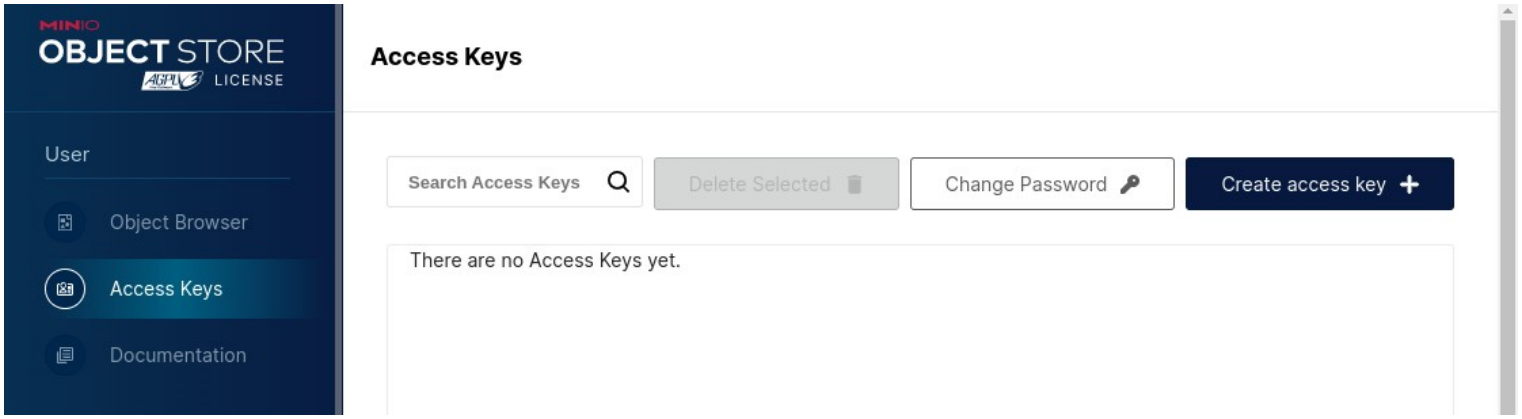
La interfaz web nos permitirá administrar nuestro servidor de almacenamiento de objetos de una manera más fácil e intuitiva.

6.5.1.Object Browser

Como podemos ver nada más entrar nos sitúa en el apartado “User” > “Object Browser”. En este nos aparecerán todos los buckets que tenga nuestro servidor con posibilidad de buscar entre ellos y ver una pequeña información de cada uno.



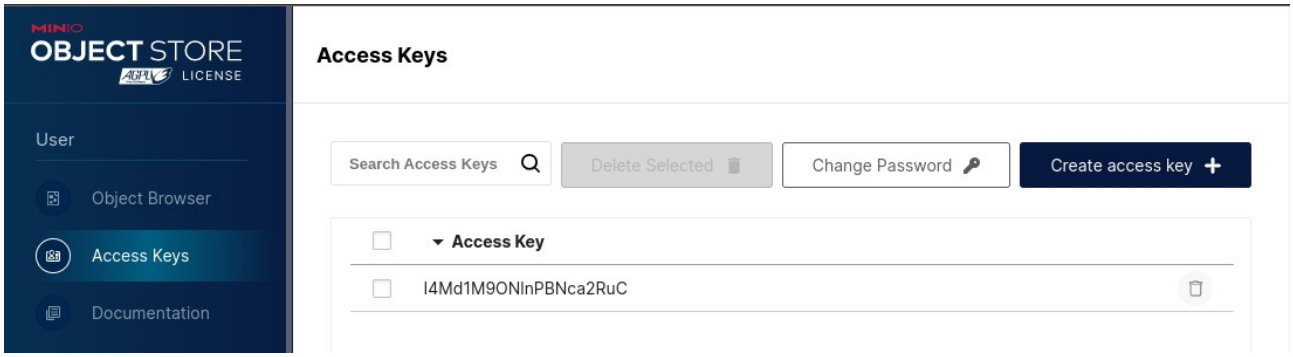
En el siguiente apartado “Access Keys” podemos crear claves de acceso para los determinados clientes que quieran realizar operaciones sobre nuestro servidor. Estas claves servirán para identificarse en este.



6.5.2.Access Keys

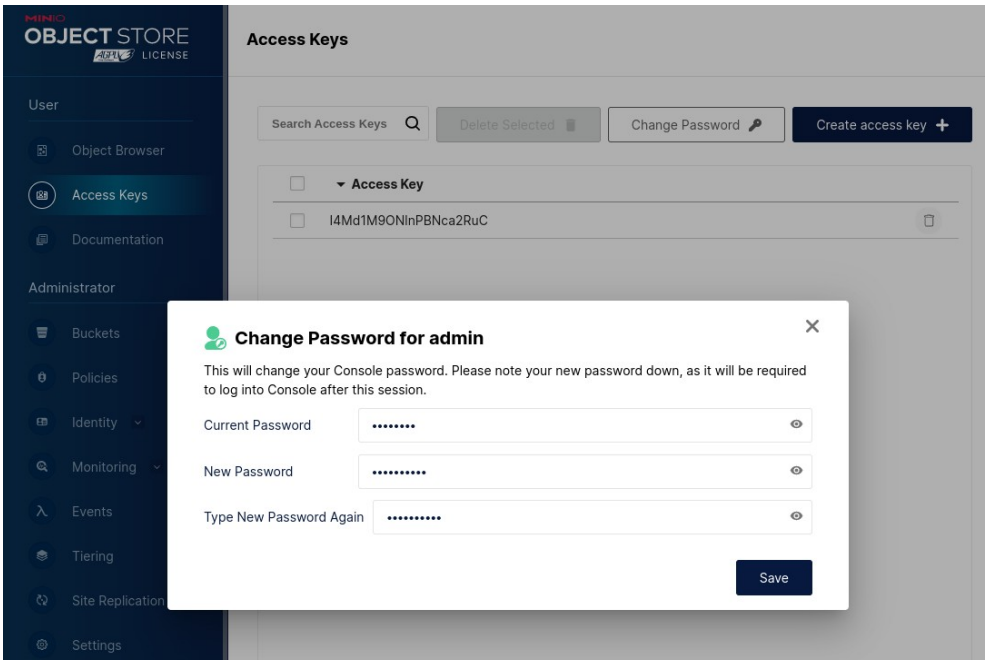
Para crear una únicamente tendremos que dar en “Create access key” > “Create” si queremos la que nos genero o eliminar los datos y crear una personalizada.

Es muy importante anotar la “Secret key” ya que solo la tendremos visible en el proceso de creación y si la olvidamos la clave quedará inutilizable y tendremos que generar otra.



Si nos pasa esto podemos también eliminarla seleccionándola y dando en “Delete Selected”.

En este apartado también podemos cambiar la contraseña de nuestro usuario administrador “admin” para obtener una mayor seguridad. Para ello haremos clic en “Change Password”, nos aparecerá un menú donde tendremos que escribir la antigua y la nueva dos veces.



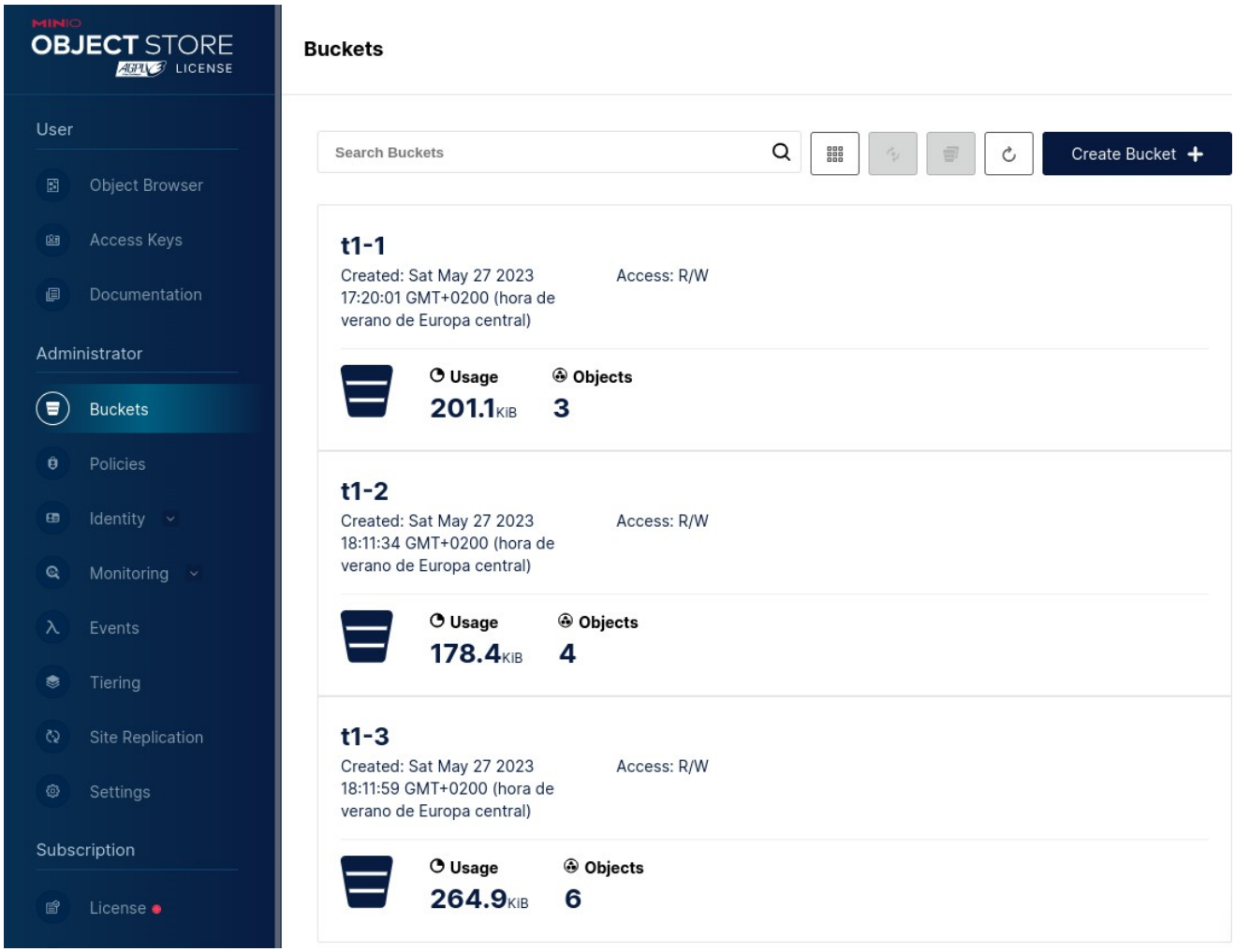
En el último apartado de “User”,”Documentation”, si clicamos en nos llevará a la página oficial de la documentación de MinIO para la opción de instalación que hayamos escogido.

https://min.io/docs/minio/linux/index.html?ref=con

Pasamos a la categoria de “Administrador” en el cual tendremos todas las opciones que nos ofrece MinIO para administrar nuestro servidor desde la interfaz web.

6.5.3.Buckets

Comenzaremos con el apartado “Bucket”. En este apartado podremos ver toda la información sobre estos. Además podremos crear, buscar, eliminar o modificar cualquier elemento referente a nuestros buckets, así como sus objetos.



Para crear un bucket únicamente tendremos que dar en “Create Bucket”. Seguidamente nos aparecerá una nueva ventana en la que tendremos que poner el nombre del bucket en “Bucket Name” y habilitar las características que veamos oportunas.

Tenemos tres características principales a la hora de crear un bucket que son:

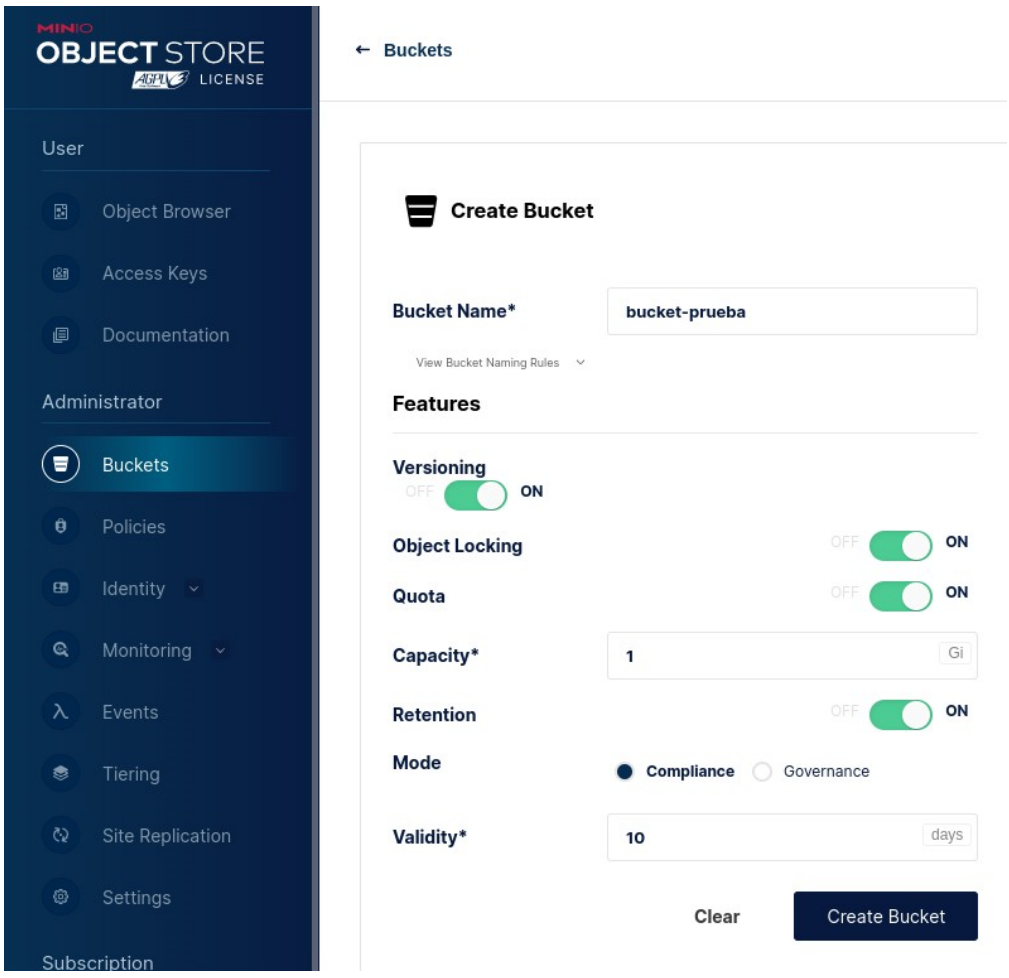
- Versioning: Si lo activamos tendremos disponible un control de versiones que nos permitirá tener múltiples versiones del mismo objeto bajo la misma clave. Al activarlo se mostrará otra nueva característica llamada “Retention”

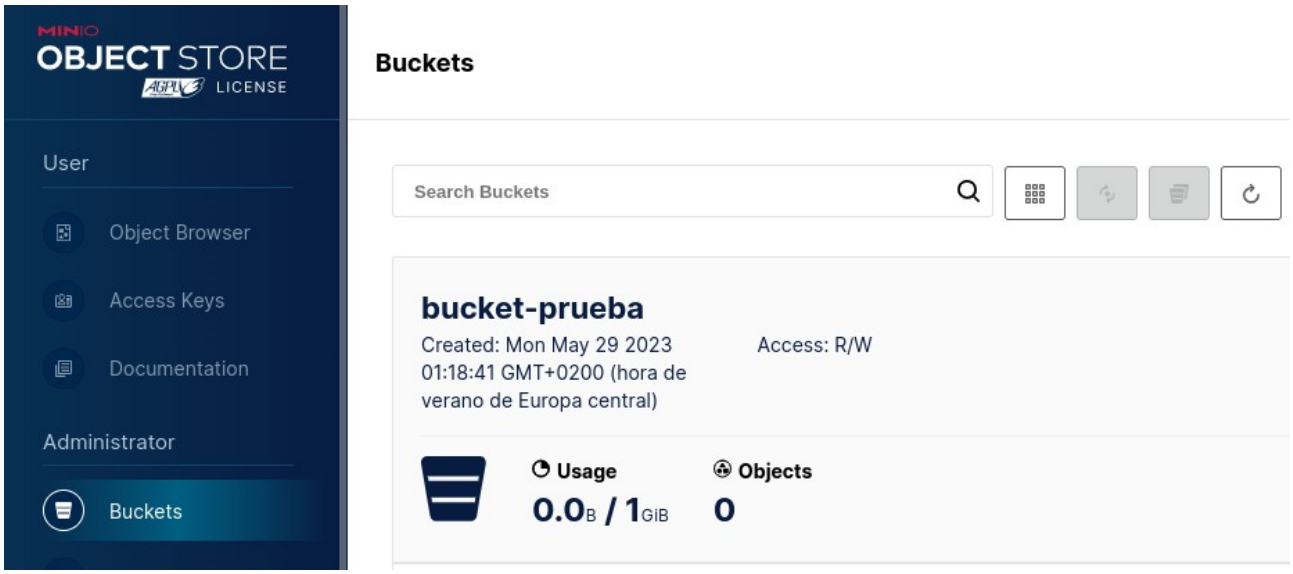
- Object Locking: Si activamos el bloqueo de objetos evitaremos que se puedan eliminar objetos de este bucket. Esto será necesario para respaldar la retención y solo podemos activarlo en el proceso de creación.

- Quota: Si lo activamos nos permitirá establecer un límite de espacio para el bucket.

- Retention: Como vimos justo arriba, al activar el control de versiones nos aparecerá esta nueva característica la cual nos permite establecer una serie de reglas para evitar la eliminación de objetos durante un periodo de tiempo que configuremos. Al activar esta opción se activará automáticamente “Object Locking”.

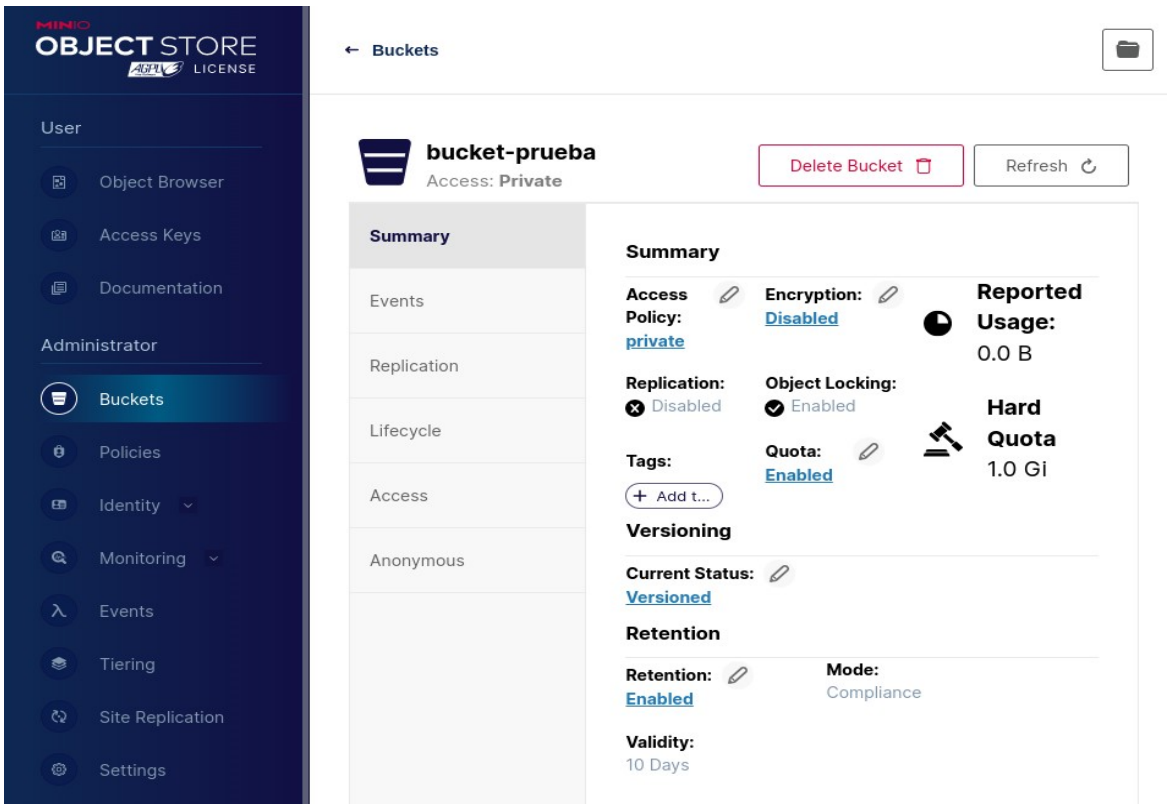
Por último damos en “Create bucket” y como podemos ver se creo correctamente.





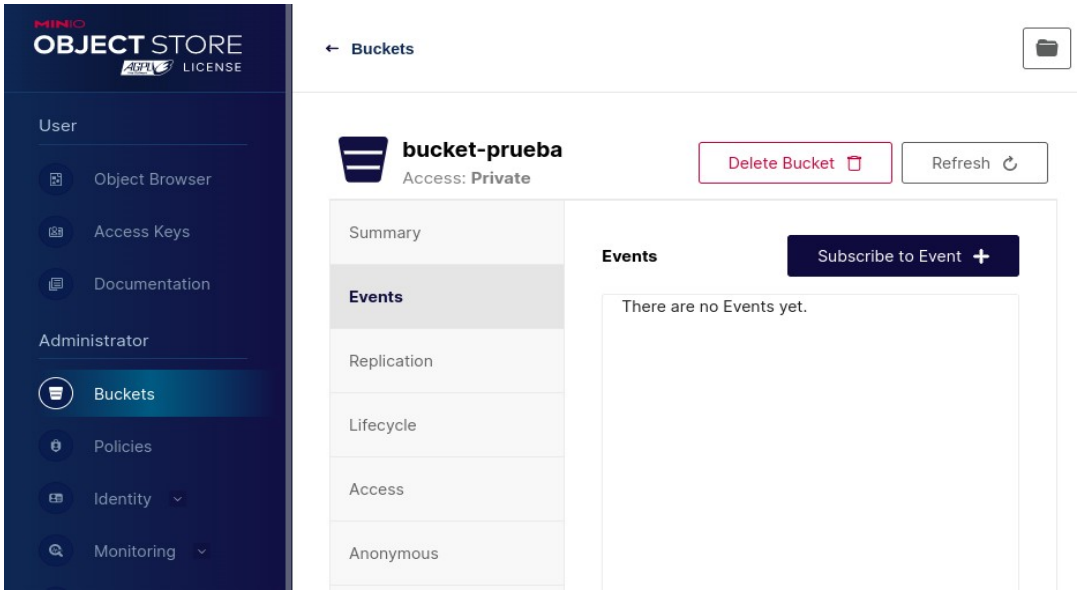
Si hacemos clic encima del nombre podremos ver la información de este y acceder a las distintas opciones.

Como podemos ver en la siguiente imagen se nos muestra todo tipo de información sobre nuestro nuevo bucket, desde la configuración que le hicimos a la hora de crearlo como la quota, versionado y el bloqueo de objetos activado hasta política de acceso predeterminada en privado, información de uso, etiquetas y si se encuentra replicado o no.



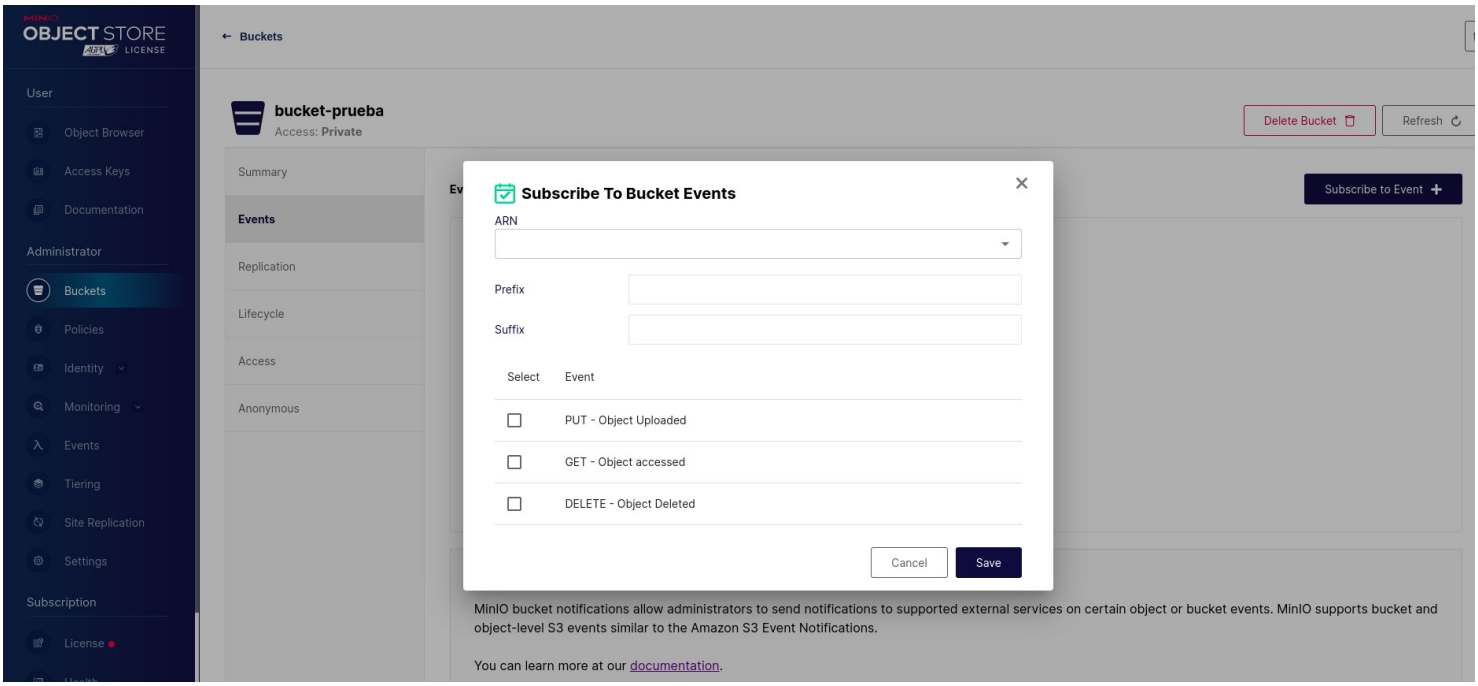
Además de esto podemos configurar adicionalmente a nuestro bucket eventos, replicado, ciclo de vida, control de acceso y control de acceso anónimo.

En el apartado de “Events” podremos crear eventos que lancen notificaciones al administrador cuando ocurra una determinada acción configurada.

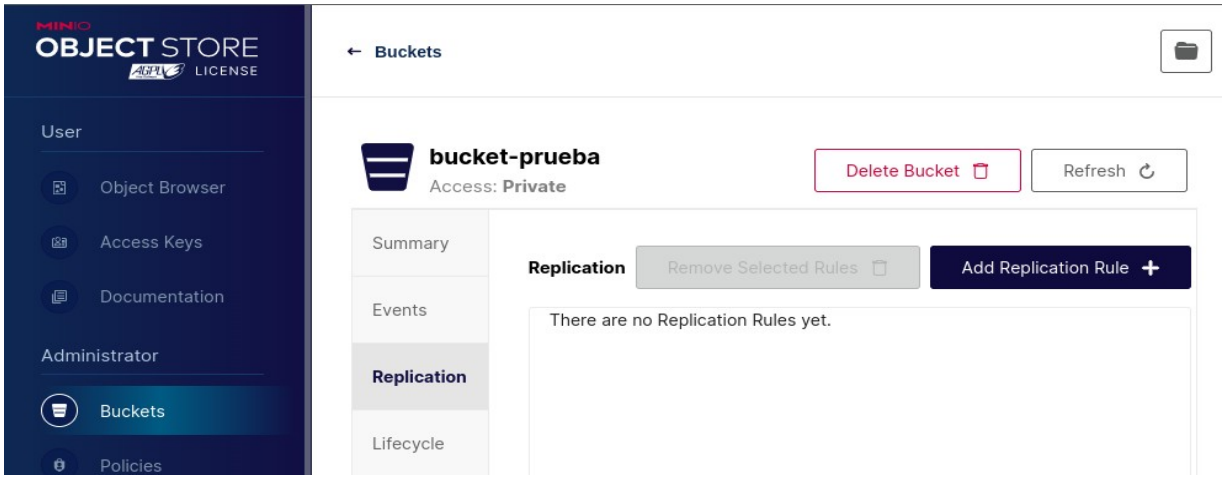


MinIO nos permite seleccionar una gran variedad de destinos para la publicación de notificaciones como Apache Kafka, mysql, redis, y siendo además compatible con las notificaciones de eventos de Amazon S3.

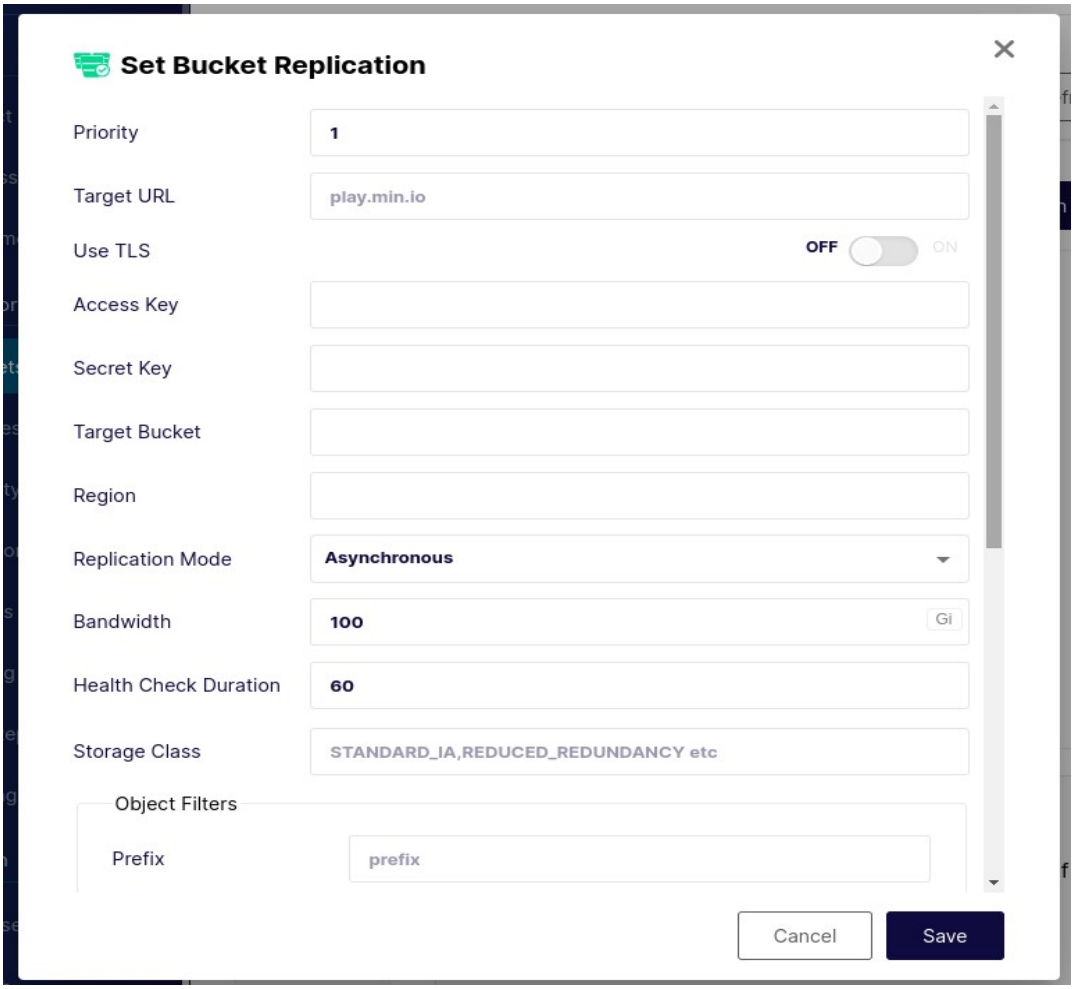
Para su configuración podemos visitar:
https://min.io/docs/minio/linux/administration/monitoring/bucket-notifications.html?ref=con



MinIO nos permite añadir reglas para replicar el bucket en otros servidores o nodos. Esto lo podemos configurar en el apartado de “Replication”.

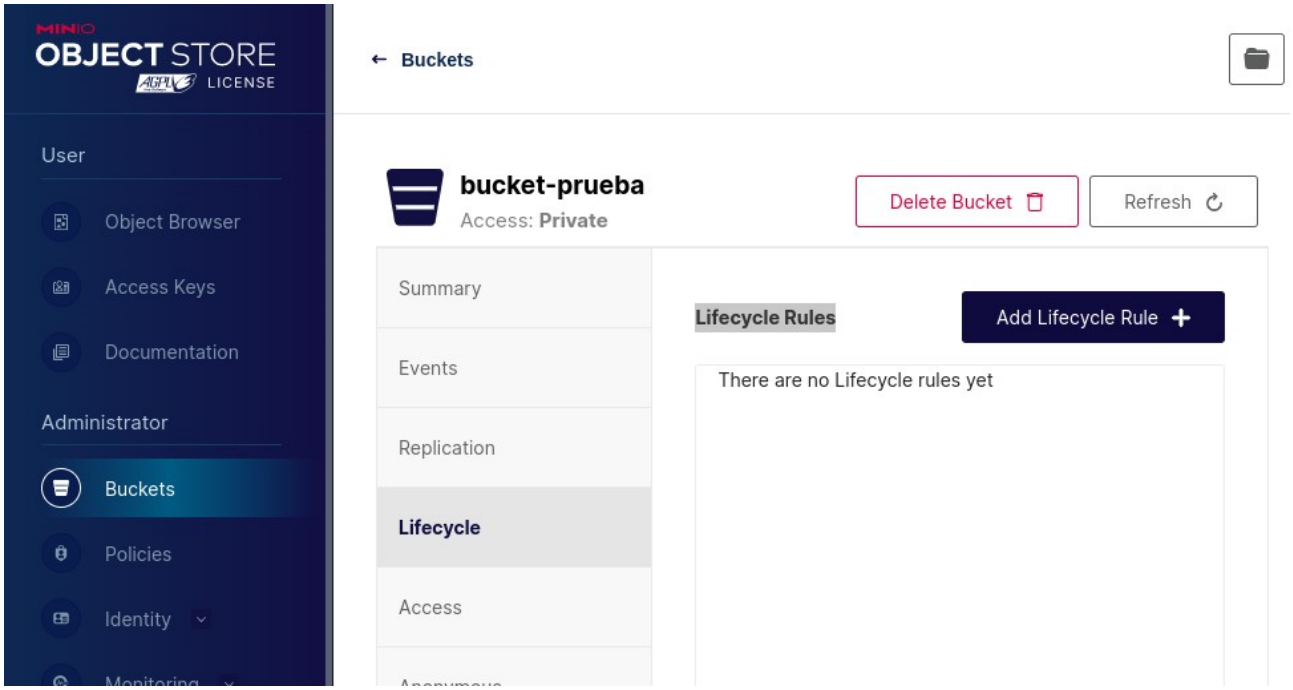


Podemos configurarlo dando en “Add Replication Rule”.

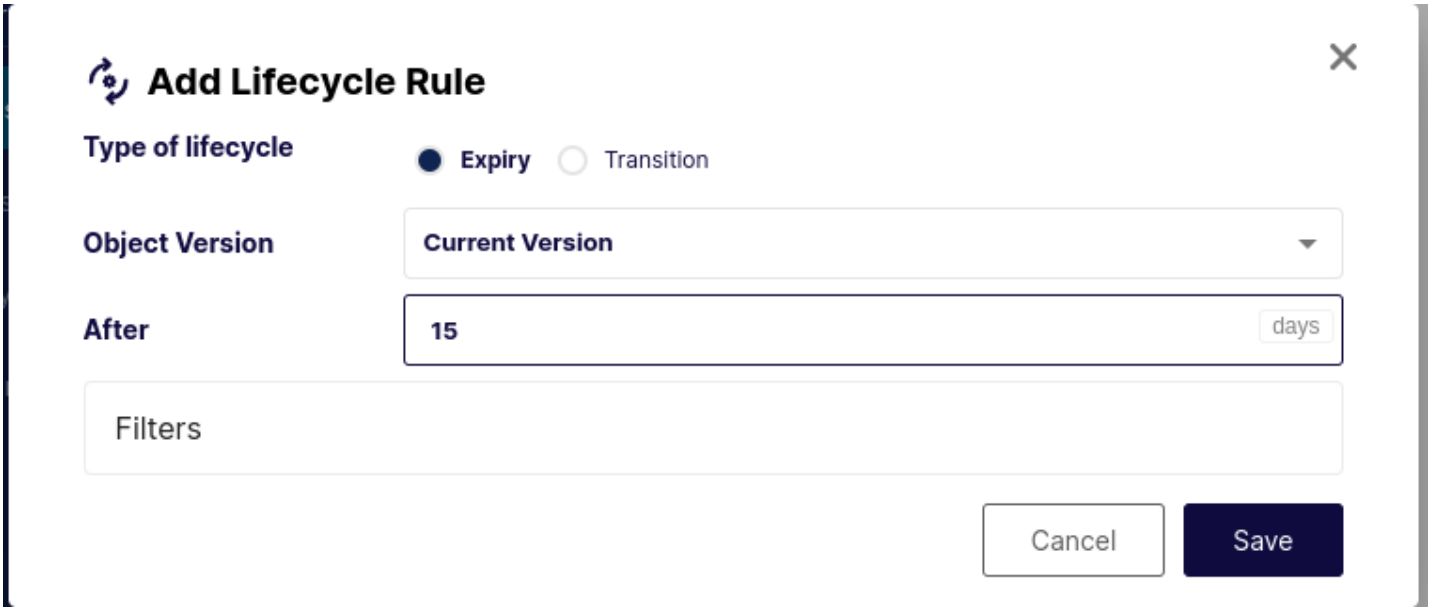


Podemos consultar la documentación oficial en:
https://min.io/docs/minio/linux/administration/bucket-replication.html?ref=con

Podemos establecerle al bucket un ciclo de vida determinado el cual configuremos en el apartado “Lifecycle”:

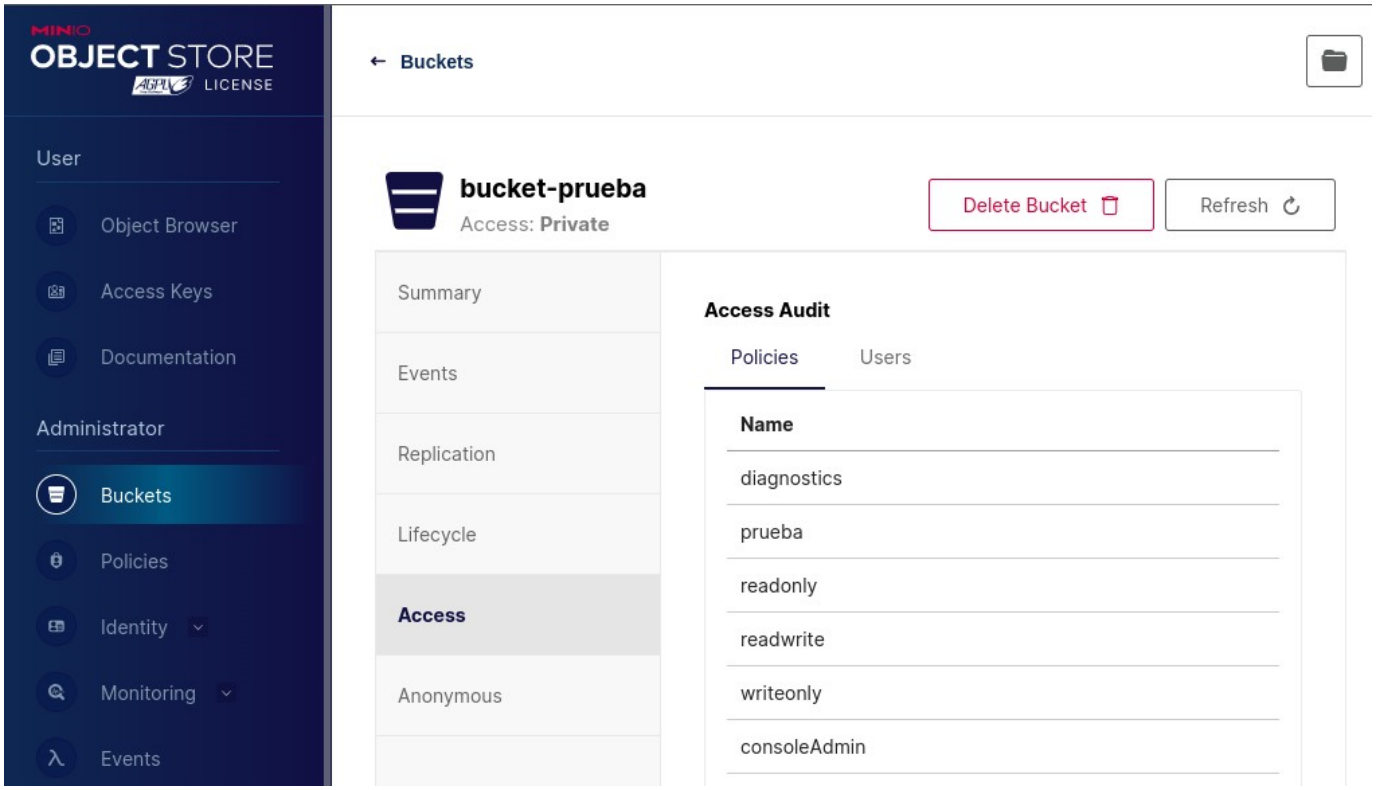


Nos permite configurarle el número de días que consideremos oportuno a una versión concreta del bucket:



Desde el mismo menú de este apartado podemos modificar esta regla y deshabilitarla y habilitarla a necesidad.

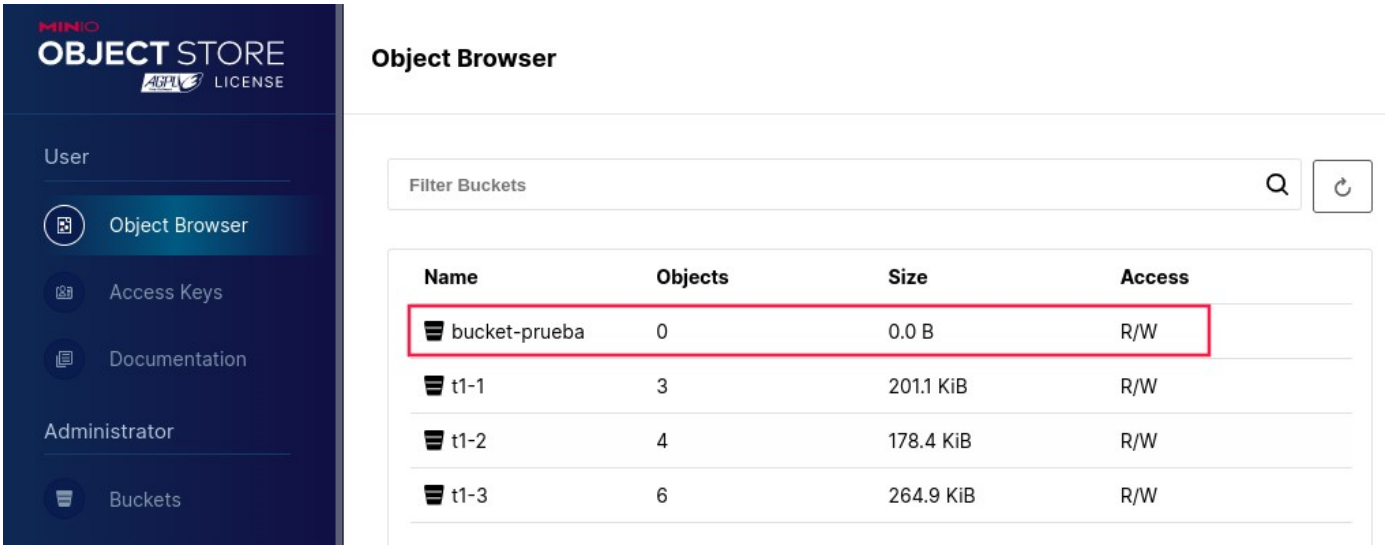
En el apartado “Acess” podemos ver las políticas creadas en nuestro sistema. Posteriormente veremos como crearlas y consultar su información.



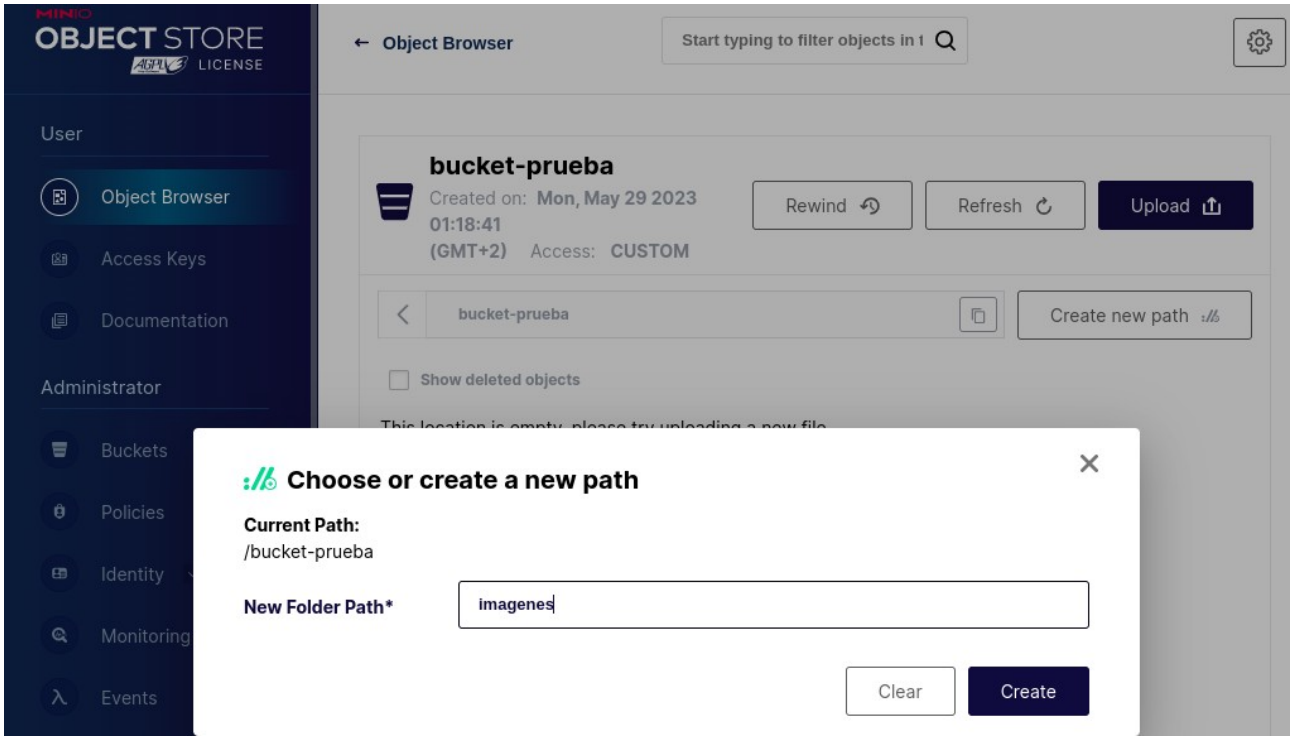
En el último apartado, “Anonymous” nos permite crear reglas para acceso anónimo al bucket mediante prefijos que los usuarios no autenticados pueden usar para leer o escribir objetos.

Con la configuración de la siguiente imagen podemos dar acceso a usuarios sin autentificación a lectura de todo el contenido del bucket.

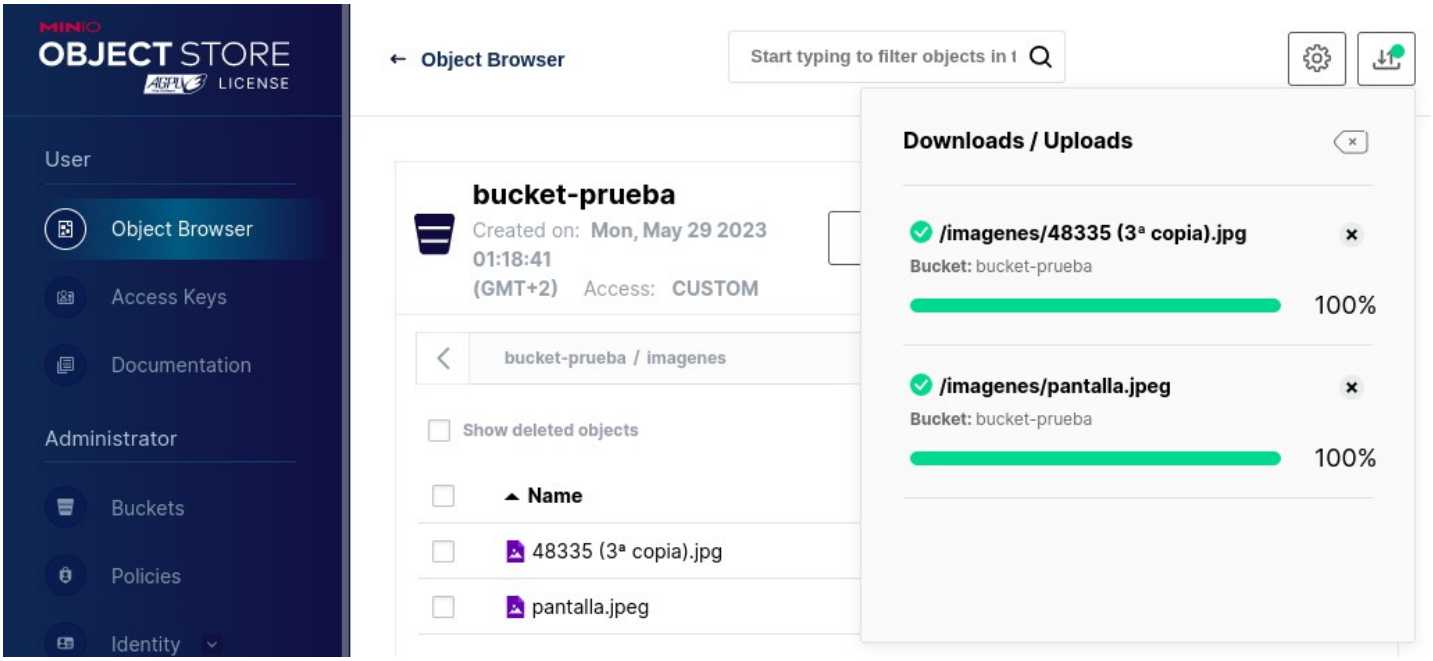
Con esto ya terminado vamos a ver como podemos gestionar la subida y bajada de objetos además de otras muchas opciones que nos ofrece. Para ello tendremos que situarnos en el apartado “User” > “Object Browser” y seleccionaremos el nuevo bucket.



Dentro del bucket podremos crear nuevos directorio en los que alojar objetos que contengan una relación entre ellos y así organizar un poco el contenido. Para crear esto únicamente tendremos que dar en “Create new path” y ponerle un nombre.

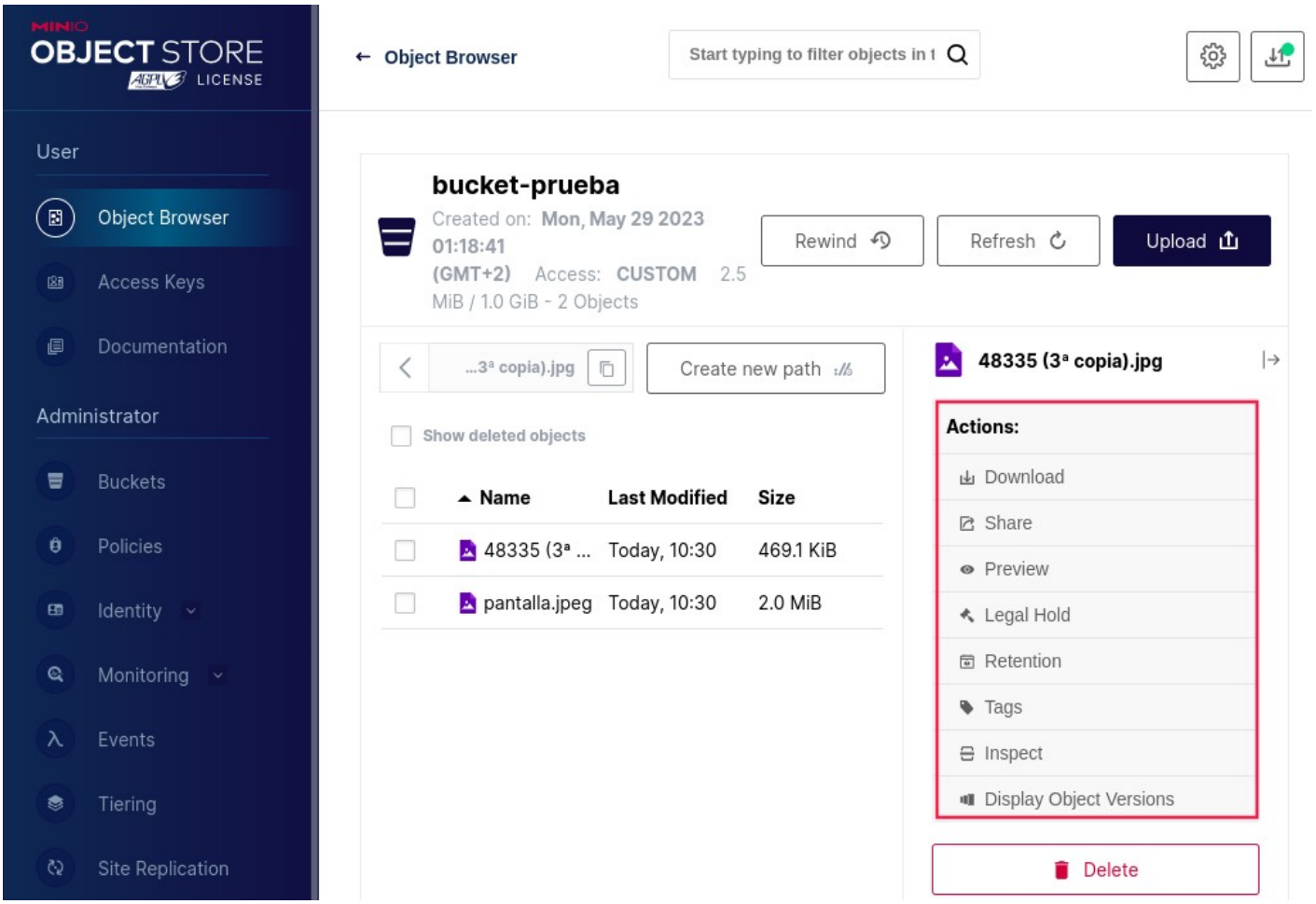


Automáticamente se meterá dentro de esta nueva ruta la cual desaparecerá sino tiene contenido. Para que esto no ocurra subiremos un par de imágenes en esta ruta para ver el proceso de subida de archivos. El proceso es muy sencillo, únicamente tendremos que dar en “Upload” y elegir si lo que queremos subir es un fichero solo o una carpeta. Se abrirá nuestro explorador de archivos local y seleccionaremos el archivo o carpeta para completar la subida.

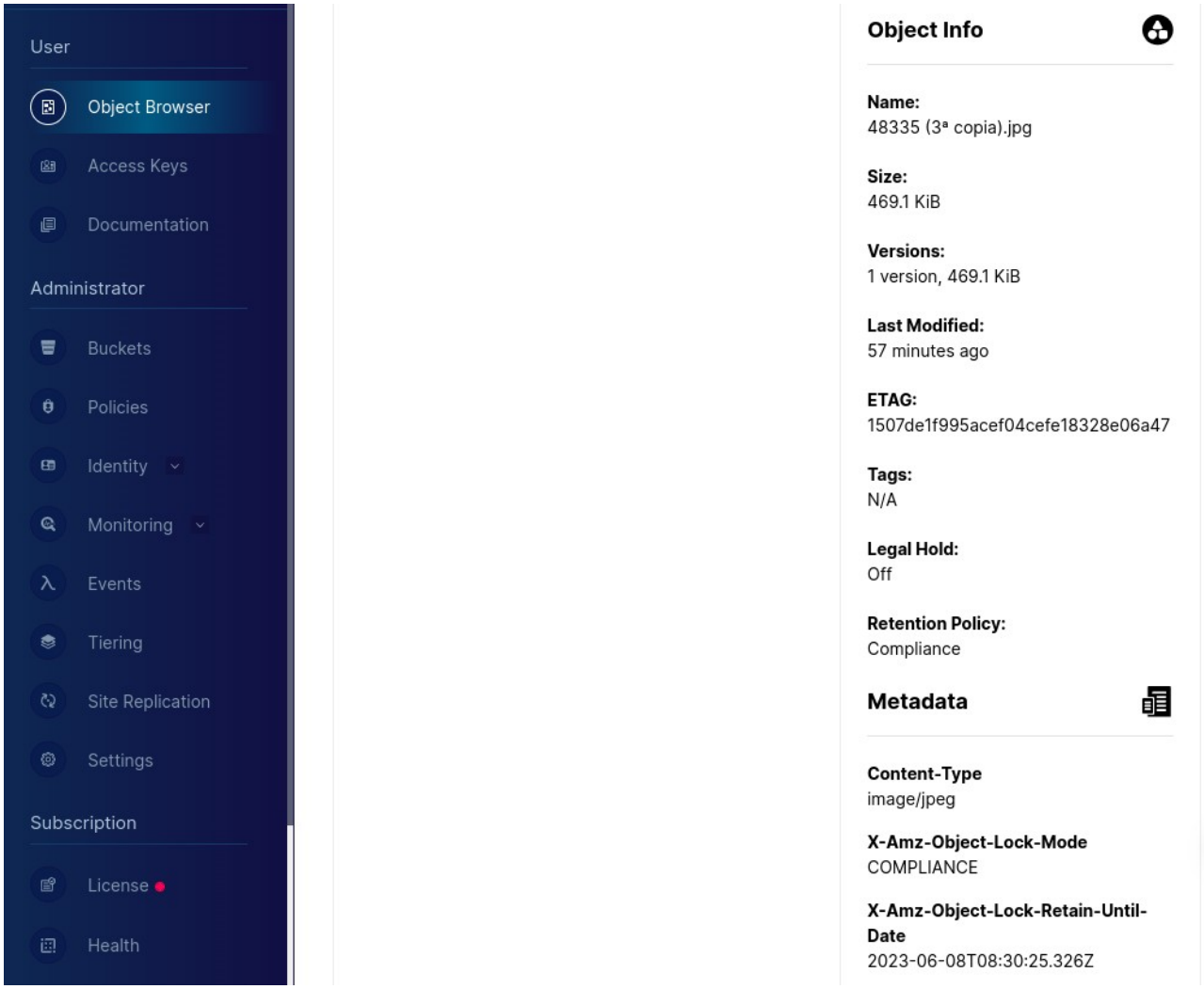


Como podemos ver se nos abrirá una pequeña ventana en la esquina superior derecha la cual nos informará del estado de la subida.

Una vez subidos los ficheros podremos ver las distintas acciones que podemos ejecutar como por ejemplo descargarlos, compartirlos, previsualizarlos, asignarle etiquetas, etc.



También justo debajo podremos consultar cierta información sobre el objeto como el nombre, tamaño, versión del archivo, ultima modificación, etiquetas asignadas, etc.

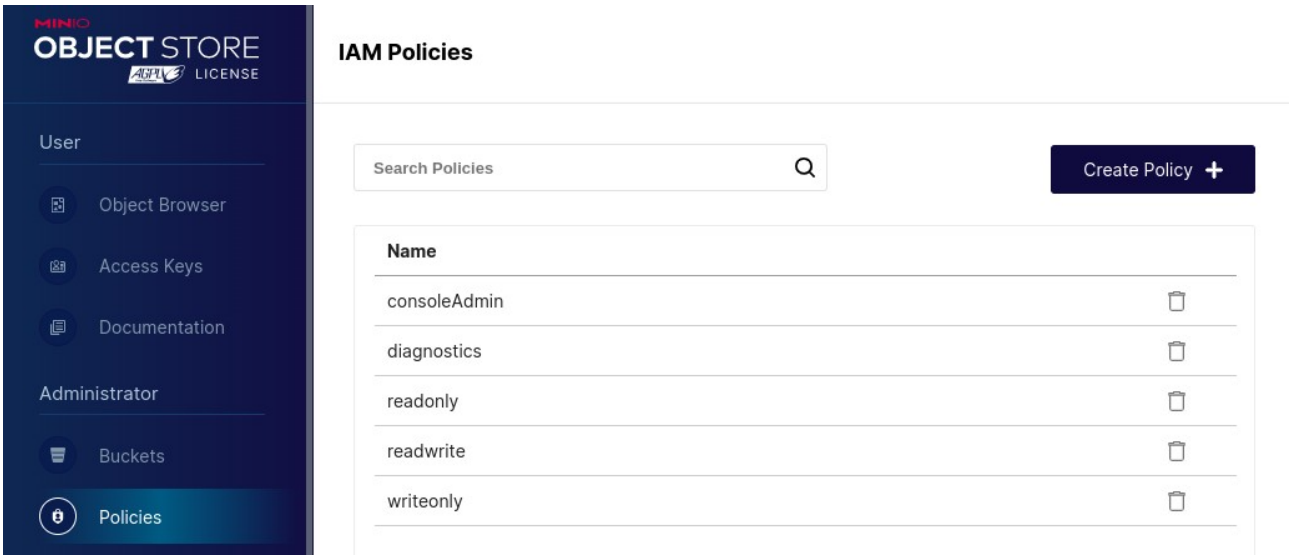


6.5.4.Policies

El siguiente apartado sería el de “Policies” en el cual podremos ver, crear, modificar y eliminar políticas. Estas definen las acciones y los recursos autorizados a los que tiene acceso un usuario autenticado. Cada política describe una o más acciones que un usuario, un grupo de usuarios o una clave de acceso pueden realizar o las condiciones que deben cumplir.

Las políticas son archivos de texto con formato JSON compatibles con la sintaxis, la estructura y el comportamiento de las políticas de Amazon AWS Identity and Access Management.

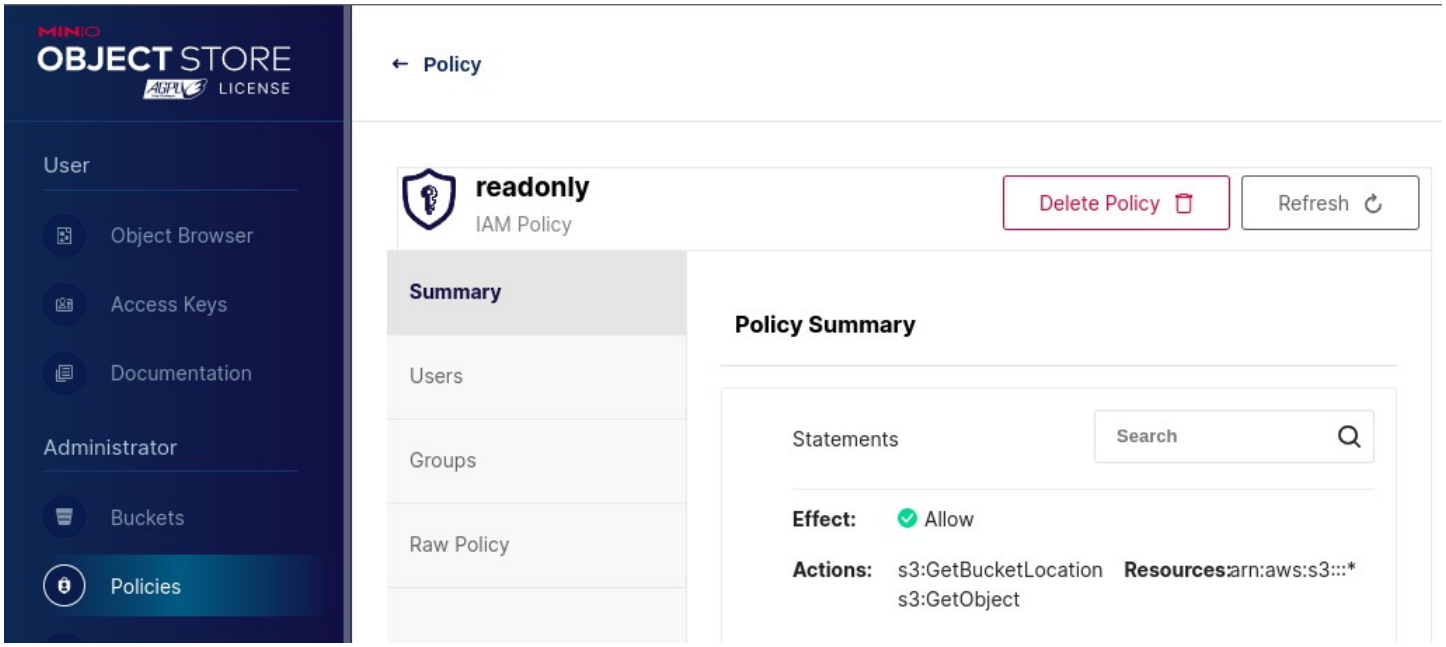
Estas son las políticas que vienen por defecto al instalar MinIO:



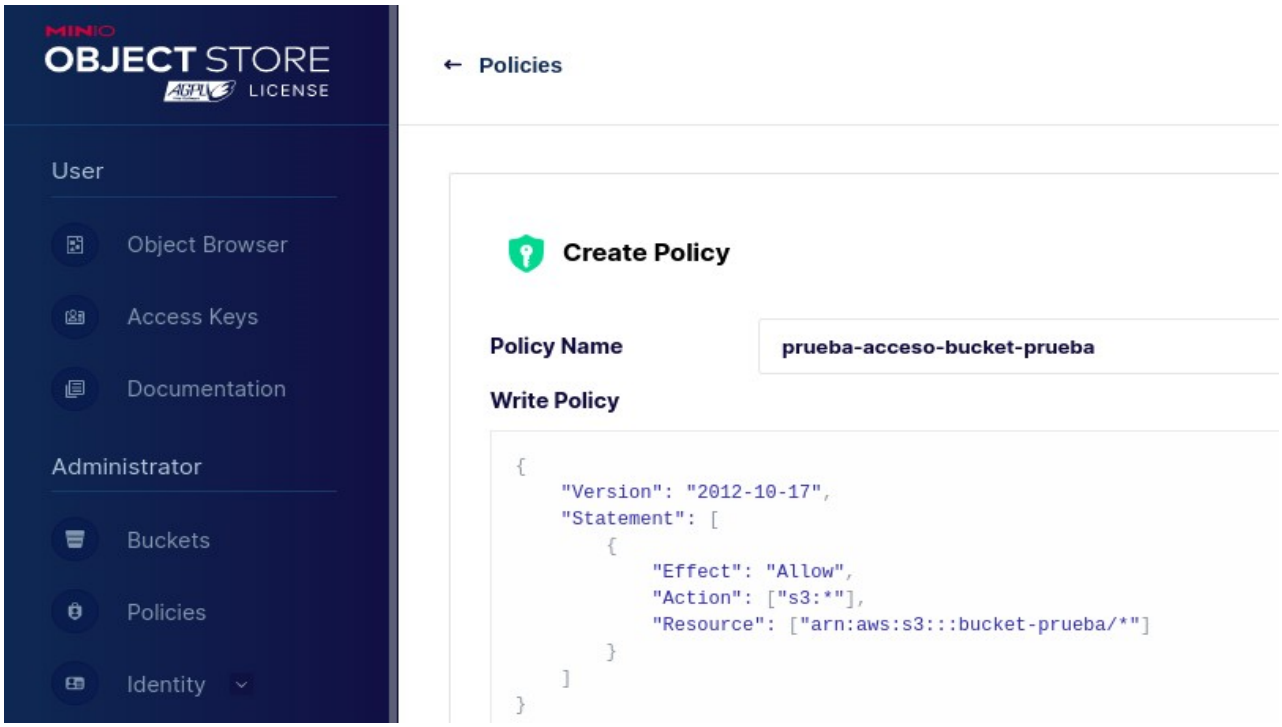
Podemos clicar sobre ellas y ver varias vistas:

- Summary: Nos muestra un breve resumen de la política.
- Users: Nos muestra todos los usuarios que tienen asignada esa política.
- Groups: Nos muestra todos los grupos de usuarios que tienen asignada esa política.
- Raw Policy: Nos muestra la política en cuestión sin procesar en formato JSON.

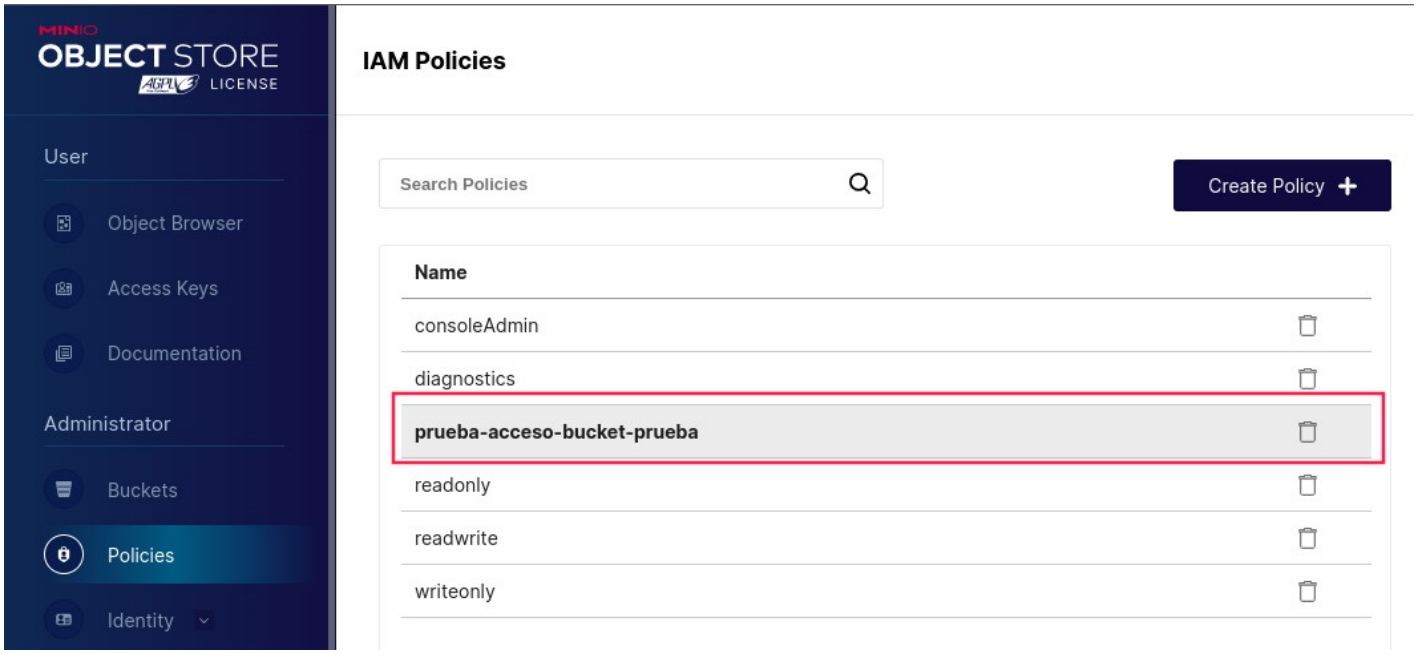
Vamos a coger como ejemplo la política de readonly la cual otorga permisos de solo lectura en cualquier objeto en la implementación de MinIO. Equivale a las acciones “s3:GetBucketLocation” y “s3:GetObject”

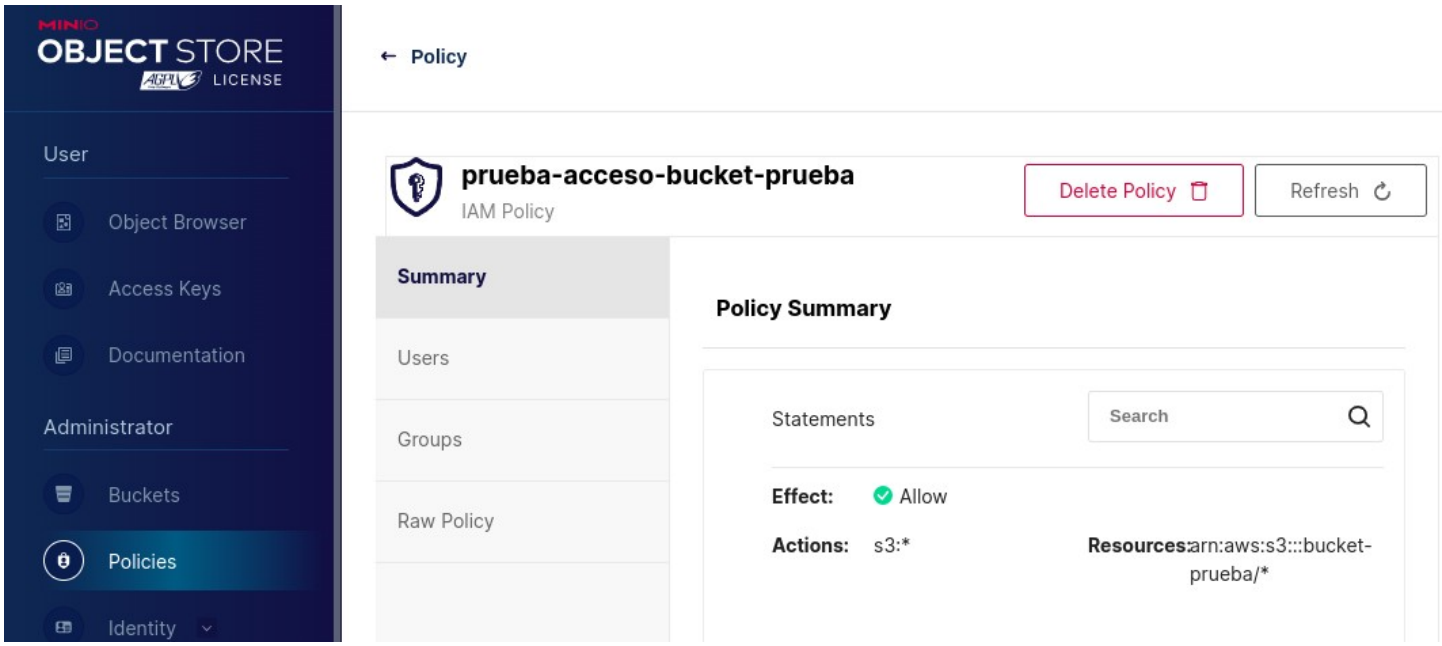


Volviendo al menú del apartado principal veremos como podremos también crear nuevas políticas dando en “Create Policy”.
Crearemos una nueva política llamada “prueba-acceso-bucket-prueba” la cual permitirá todo tipo de acciones únicamente sobre el bucket “bucket-prueba” y todos los objetos que tenga este al usuario o grupos de usuarios a la que se le asigne.



Como podemos ver se creo correctamente.

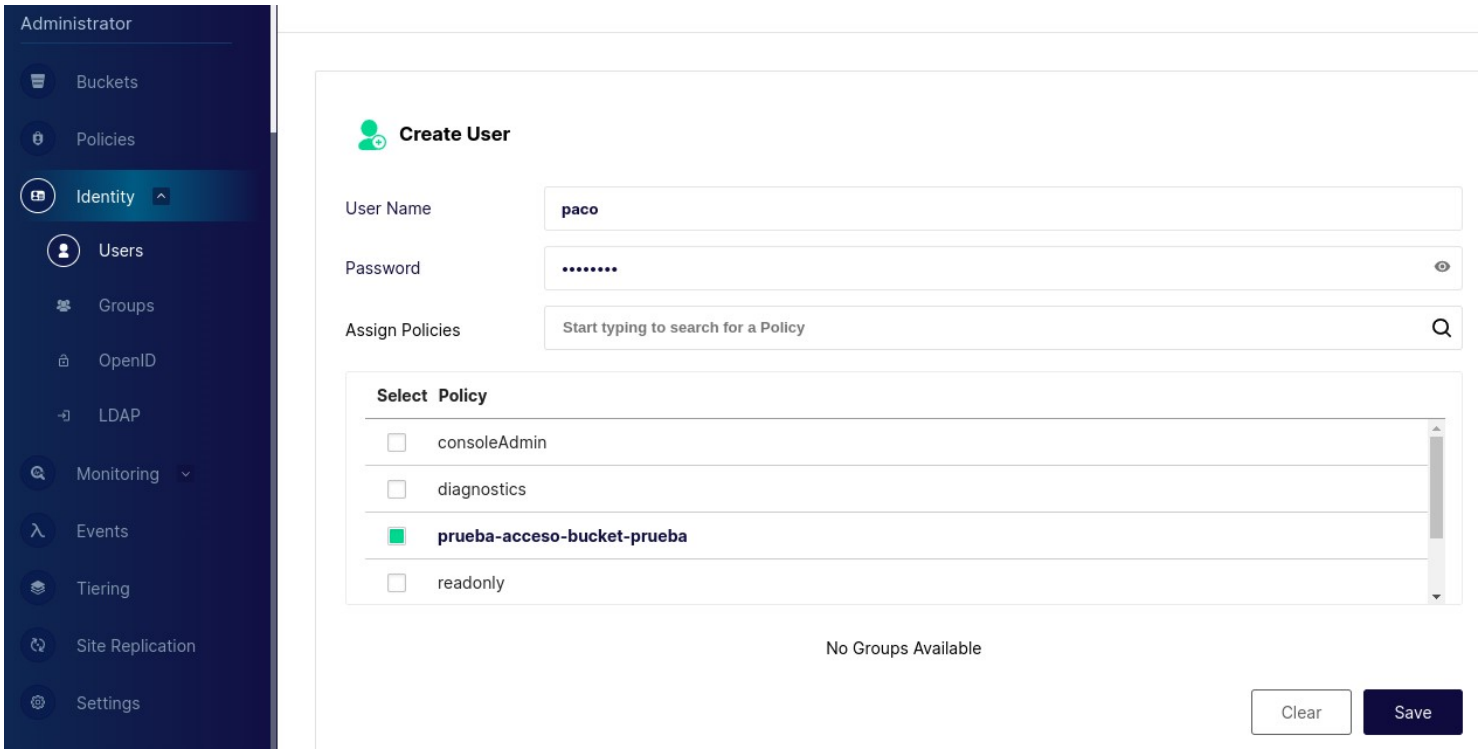




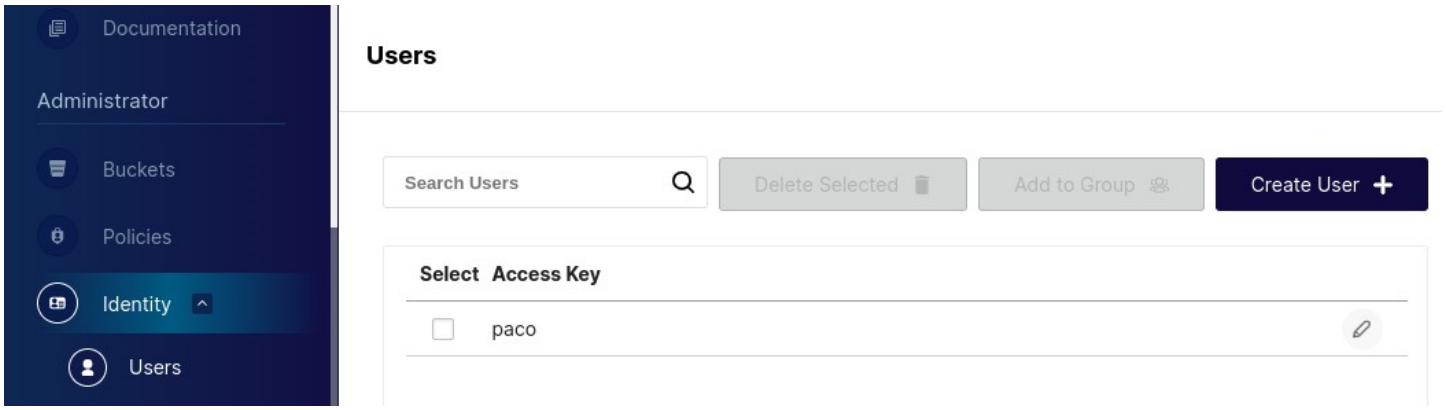
6.5.5.Identity

Este apartado nos permitirá gestionar usuarios, grupos, acceso por openID o acceso a través de un servicio LDAP.

Comenzamos con la creación de usuarios para ello nos situaremos en la vista “Identity” > “Users”. MinIO viene sin ningún usuario creado por lo que vamos a crear uno. Para ello daremos clic en “Create User” y se nos abrirá un nuevo menú en el que rellenaremos los datos del usuario. A este usuario le asignaremos la política que creamos en el apartado anterior.

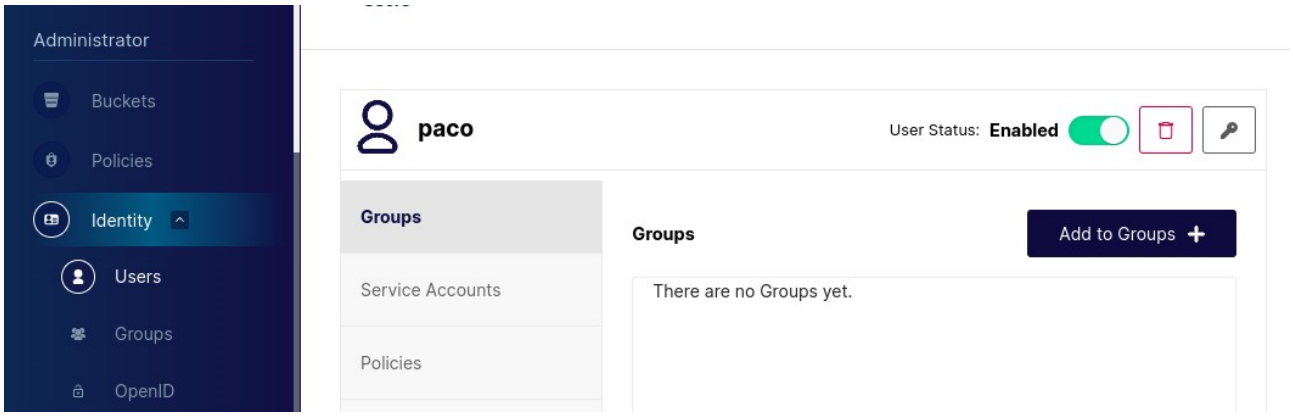


Guardamos dando en “Save” y podremos ver como se creo correctamente en la ventana anterior:

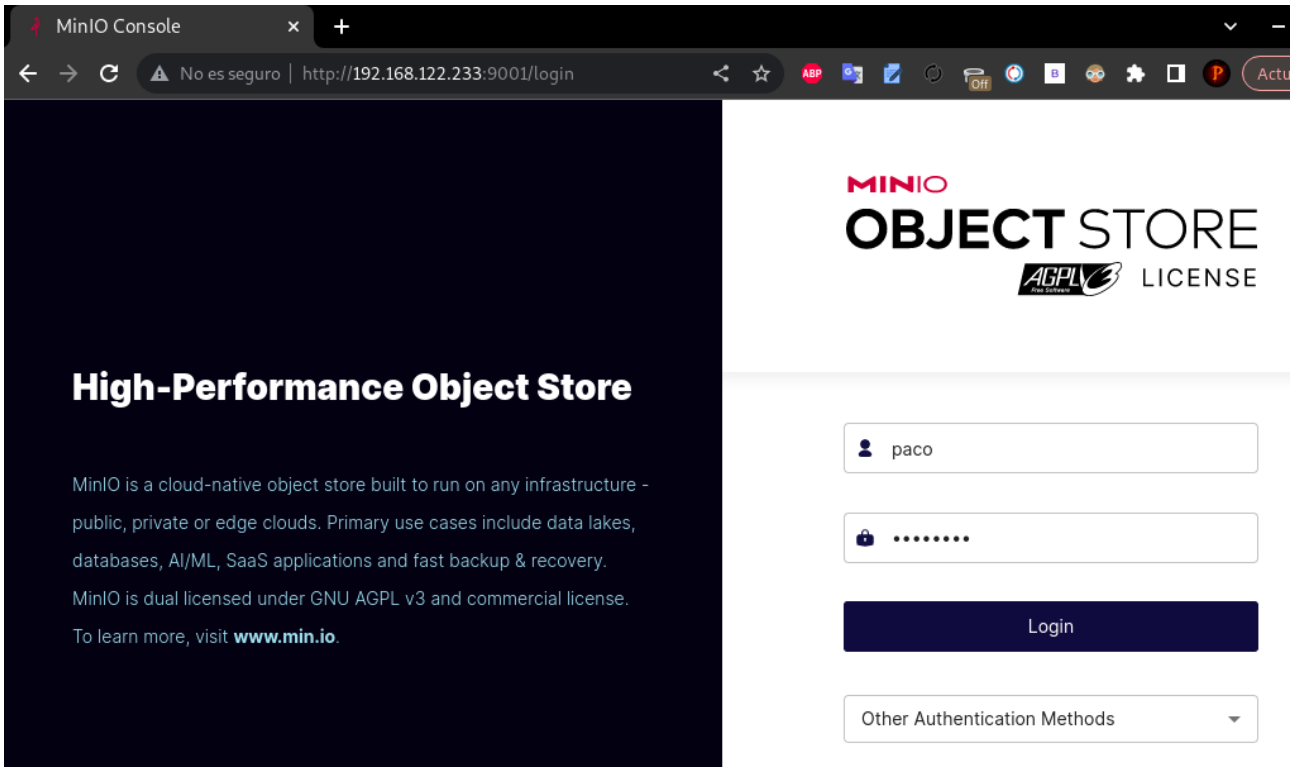


Desde este menú seleccionado dicho usuario tendremos acceso a la eliminación de este y también la posibilidad de añadirlo a un grupo de usuarios.

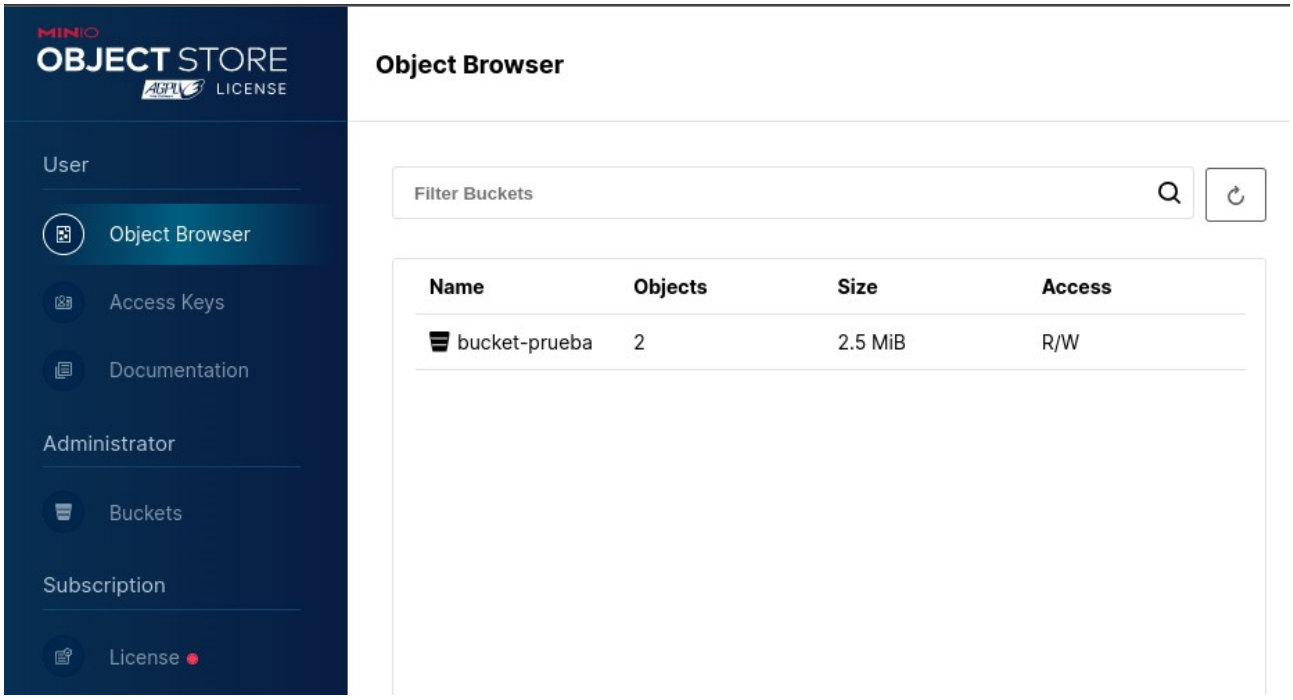
Si pulsamos sobre este, entraremos en un nuevo menú el cual nos permitirá gestionar los grupos a los que pertenece el usuario, clave de acceso para el usuario y políticas de acceso.



Ahora lo que haremos será cerrar sesión con el usuario administrador e iniciar con el nuevo usuario que creamos “paco” para observar si se aplicó correctamente la política que creamos y ver que permisos tiene dicho usuario.

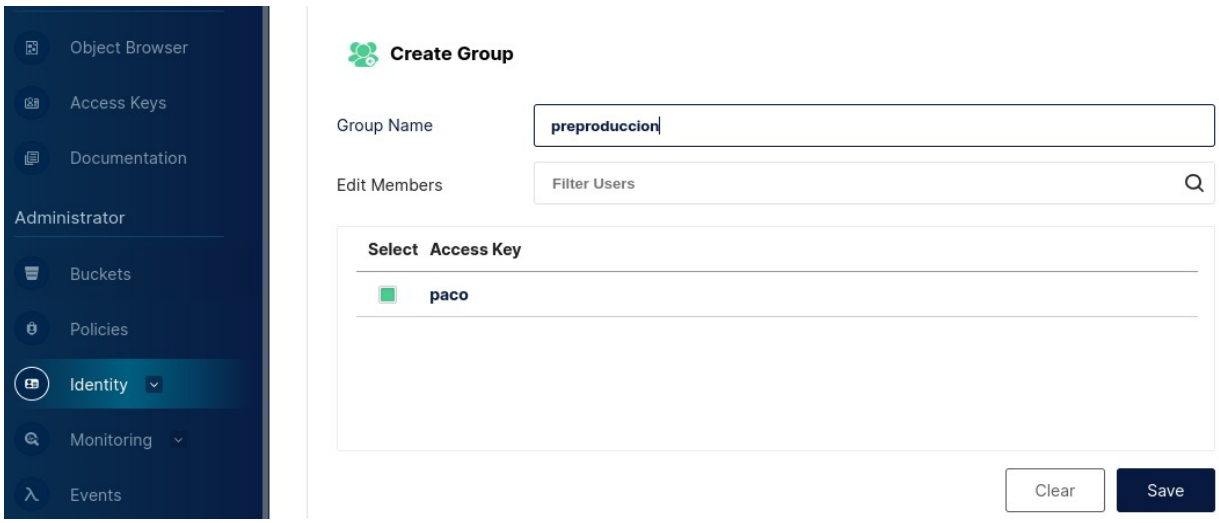


Una vez dentro podemos comprobar como únicamente podemos entrar al bucket “bucket-prueba” y ni siquiera vemos los otros.

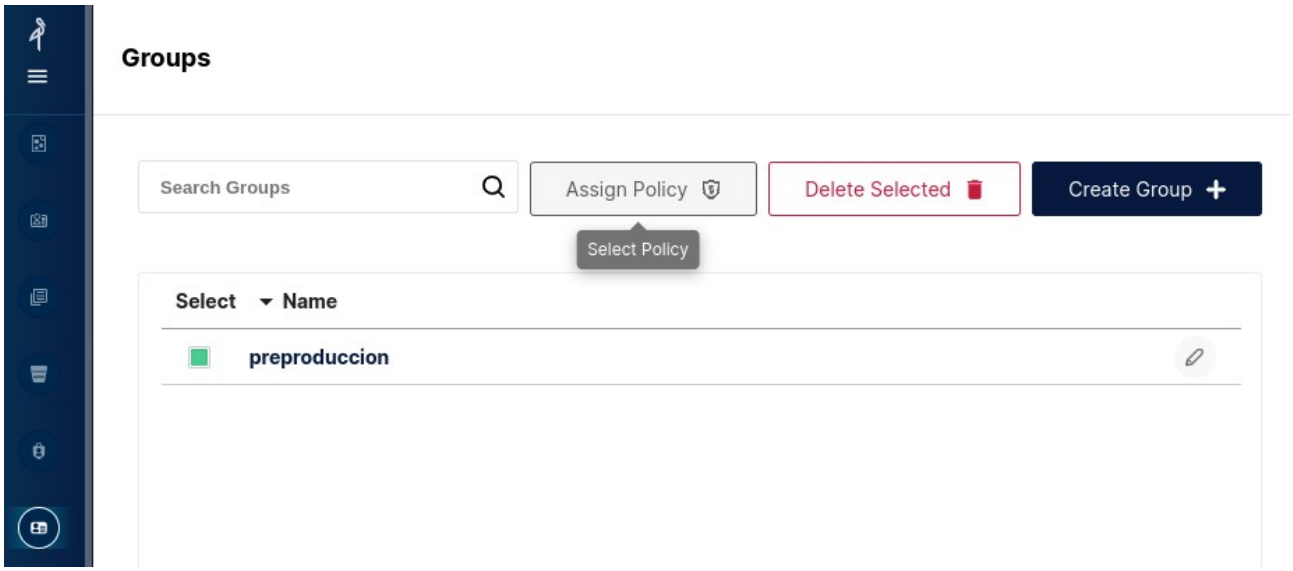


También podemos observar que el usuario al no estar en la política consoleAdmin únicamente puede acceder al buscador, menú de gestión de claves de acceso, documentación e información de sus buckets.

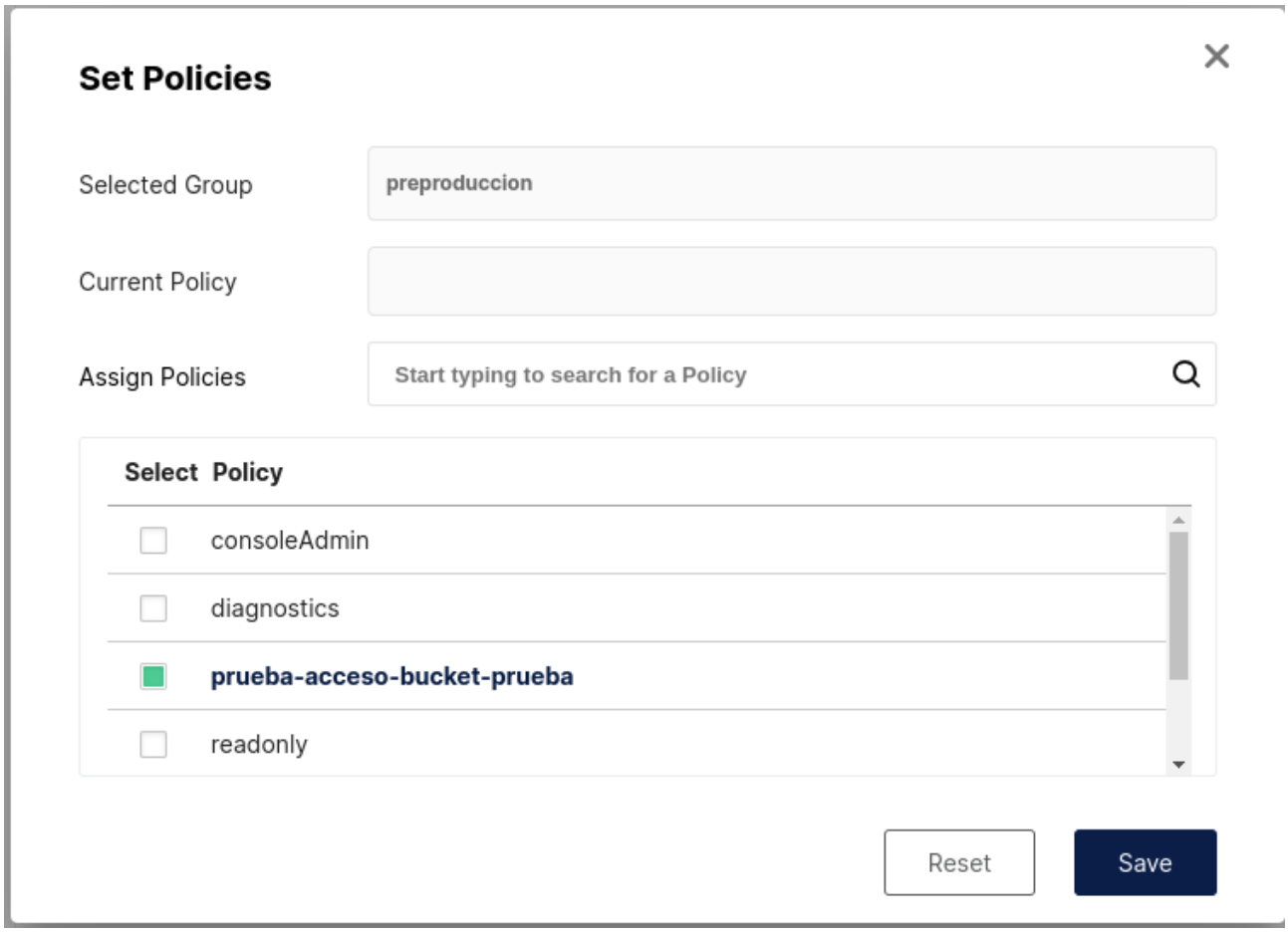
Seguimos con la creación de grupos de usuarios los cuales nos permitirán el proceso de asignación de ciertos permisos a determinados usuarios. Para ello nos dirigiremos al apartado “Identity” > “Groups”. Para crear un grupo daremos en “Create Group” y se nos abrirá un menú en el cual pondremos un nombre al grupo y seleccionaremos a los usuarios que lo formarán. En mi caso únicamente he creado al usuario “paco”, por lo cual únicamente nos aparecerá él.



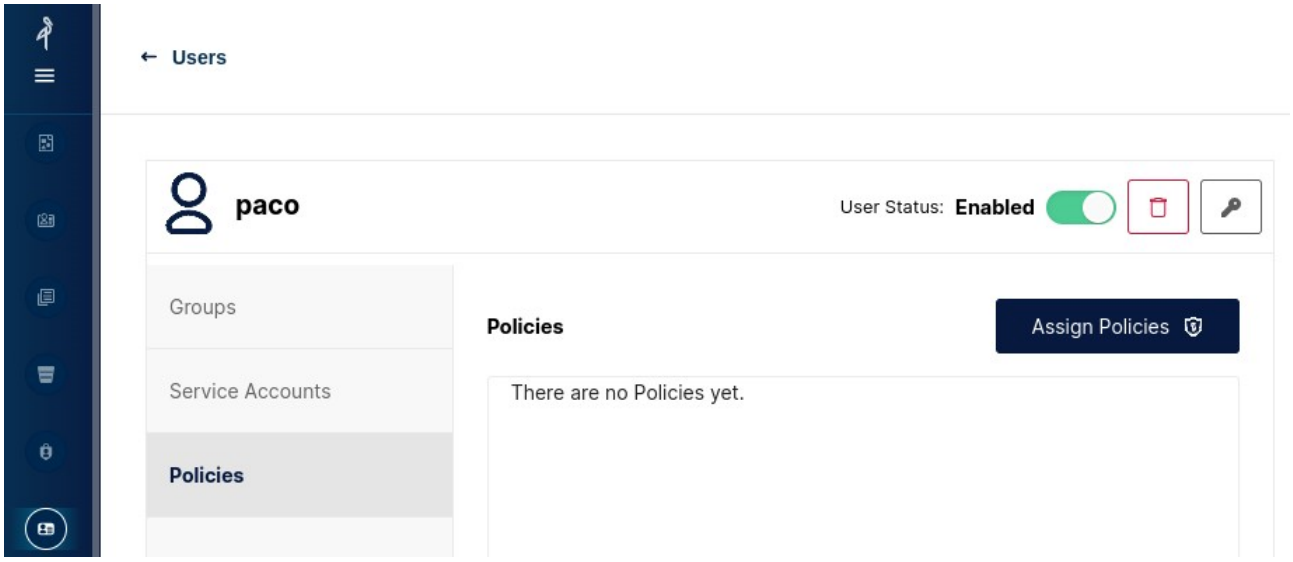
Ahora seleccionaremos el grupo que acabamos de crear para asignarle una política.



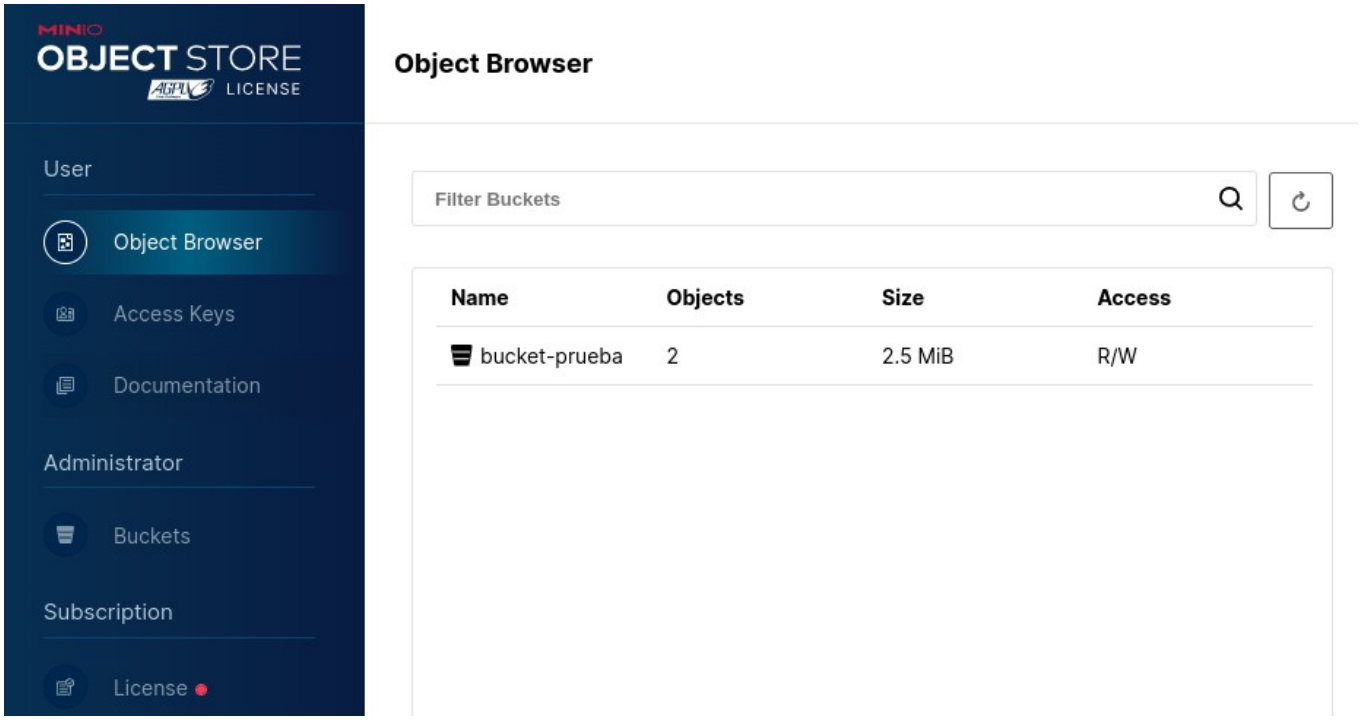
Le asignamos la que creamos anteriormente que tiene asignado el usuario “paco”.



Y ahora para ver si su funcionamiento es correcto desasignamos esta misma política al usuario “paco” para que la obtenga a nivel de grupo.

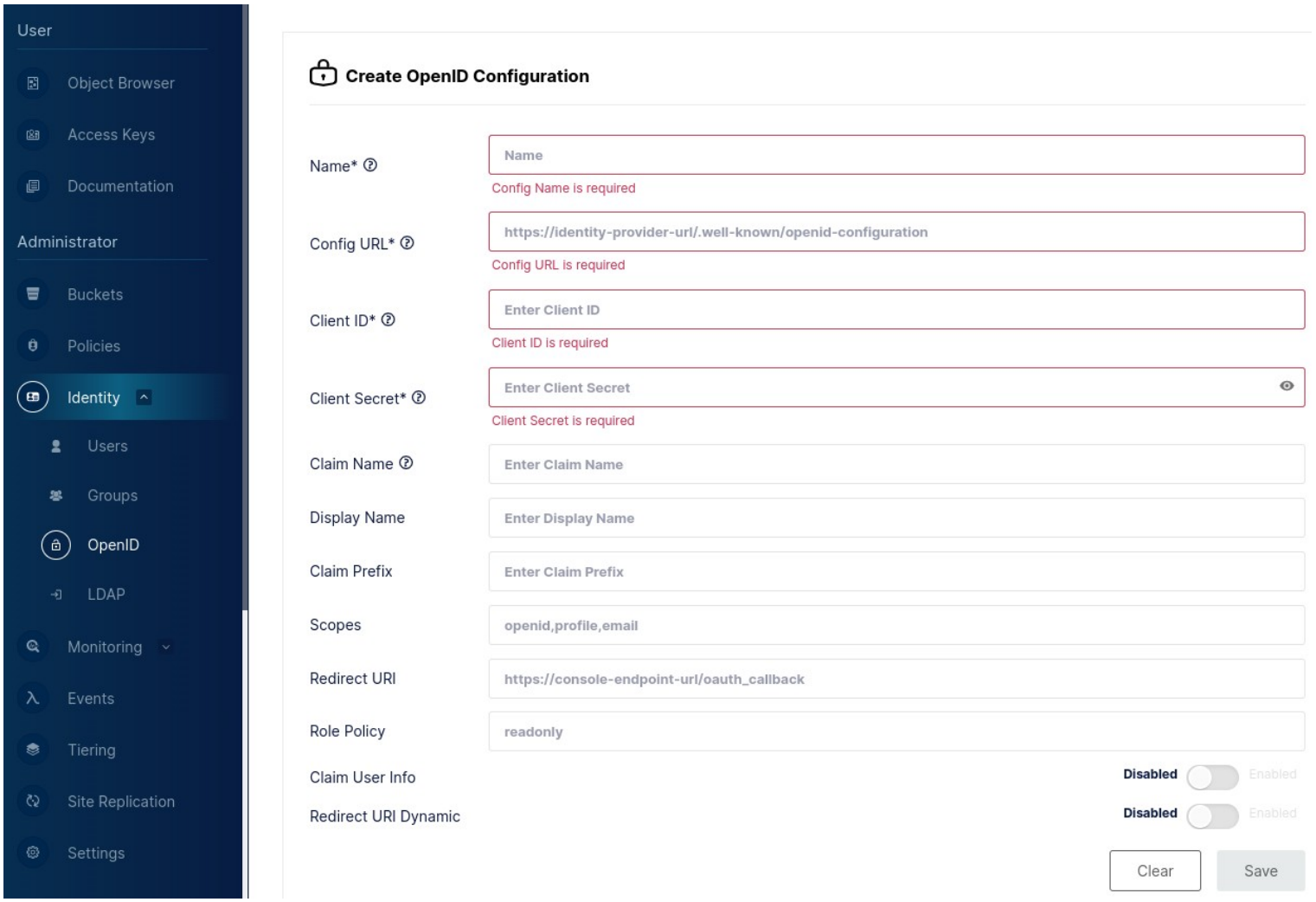


Y si nos metemos de nuevo con el usuario veremos como podemos seguir teniendo acceso a lo mismo que anteriormente vimos:

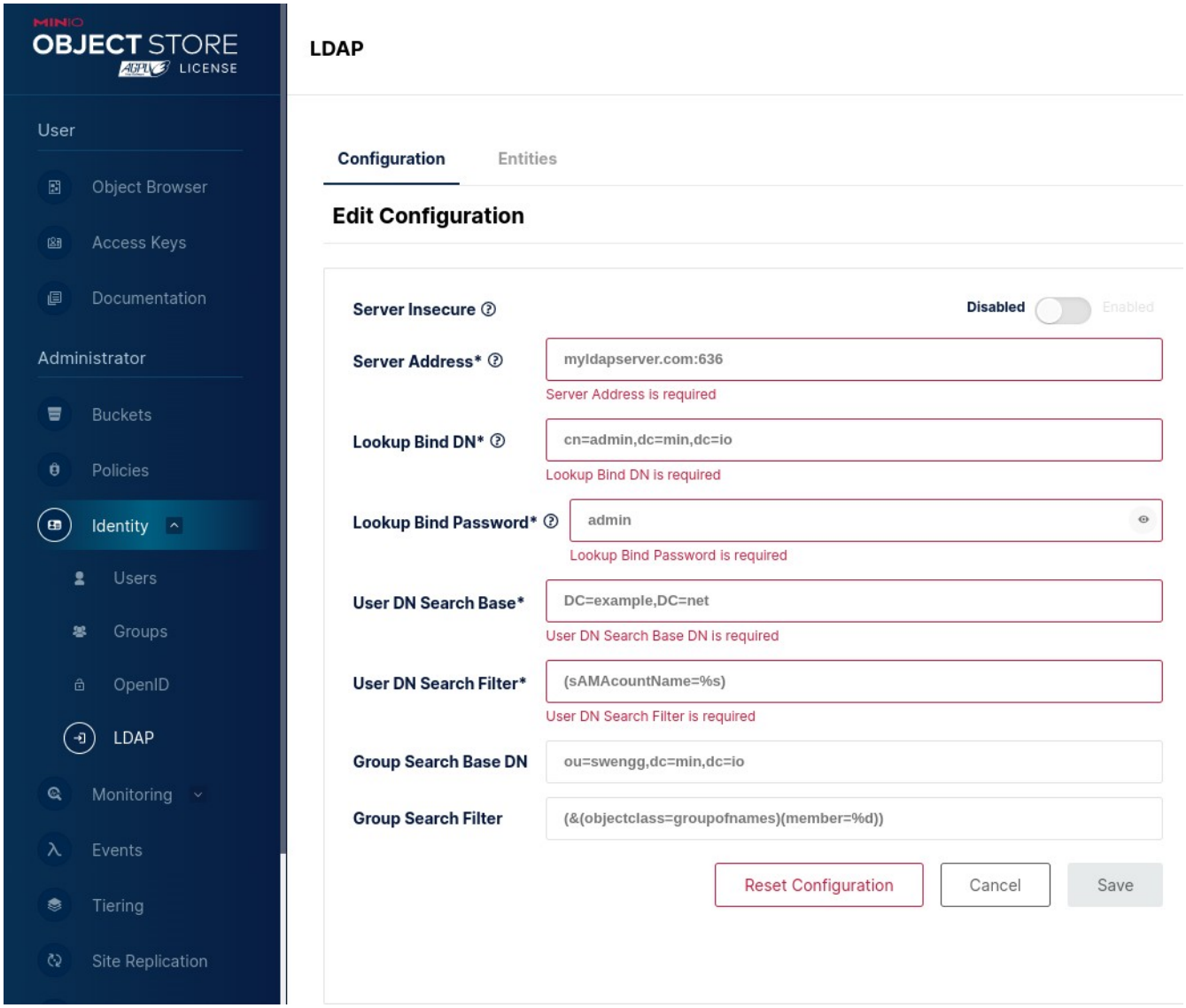


Los siguientes dos métodos de autentificación que nos ofrece MinIO son OpenID un protocolo en línea que permite a los usuarios autenticarse en diferentes sitios web utilizando una única identidad digital, y LDAP/AD el cual puede que sea el más utilizado por su versatilidad. Ambos veremos donde se pueden configurar pero no los configuraremos ya que no utilizaremos estos servicios.

OpenID: Para crear la configuración tendremos que irnos a “Identity” > “OpenID” y pulsar en “Create Configuration”. Se nos abrirá un menú con todos los parámetros para rellenar:



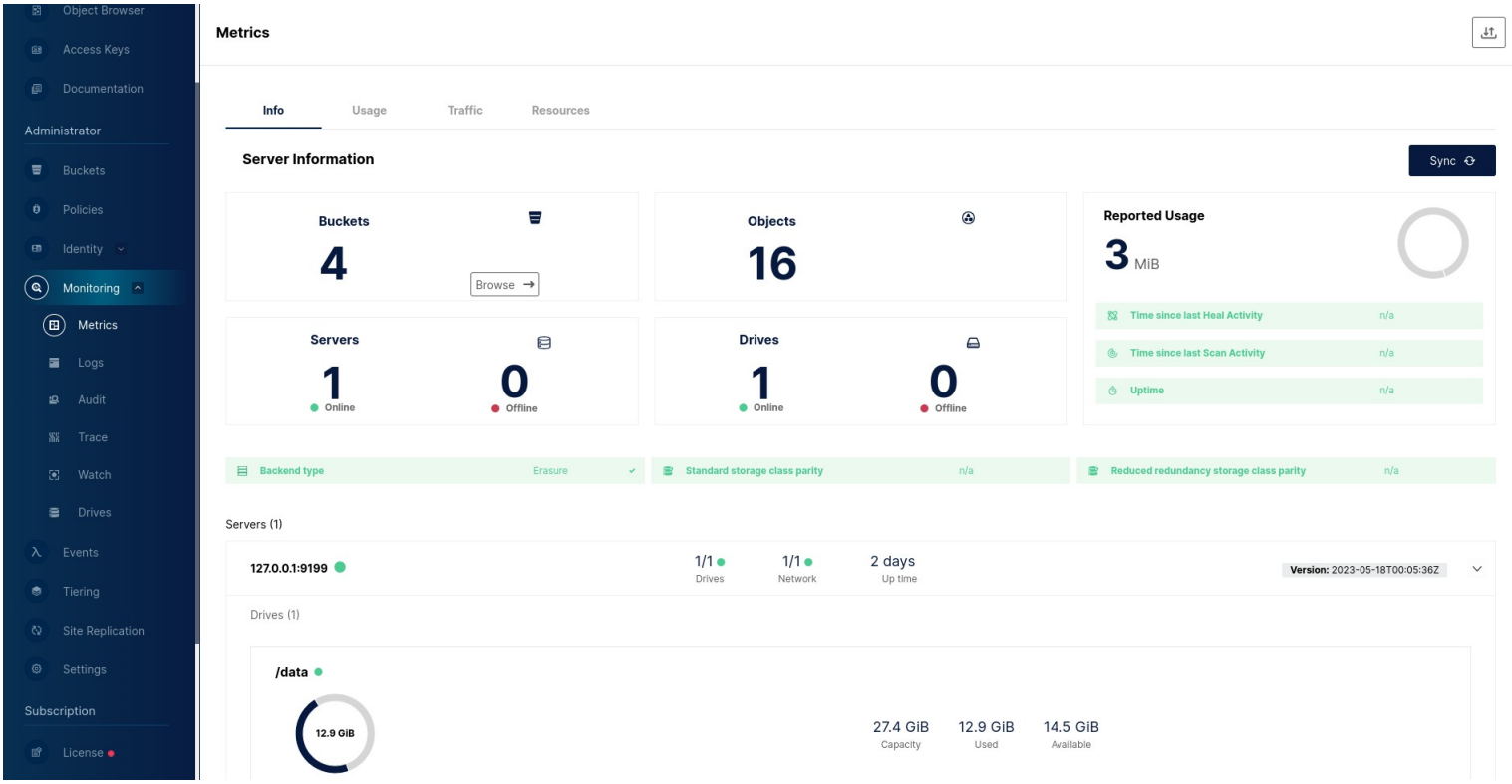
AD/LDAP: Para configurar este servicio iremos al apartado “Identity” > “LDAP”. Una vez ahí damos en “Edit Configuration” y nos aparecerá el menú para poder configurarlo:



6.5.6.Monitoring

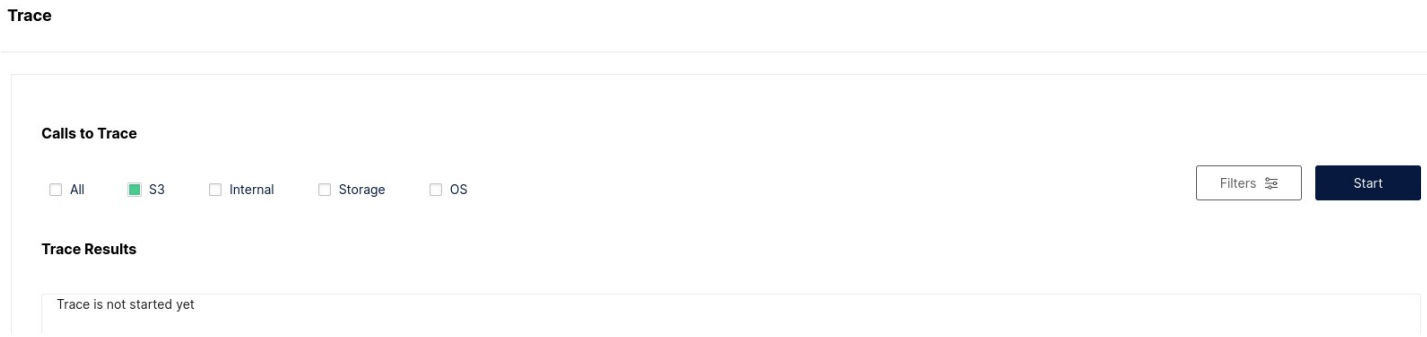
En este apartado veremos las opciones que nos ofrece MinIO para la monitorización de nuestro servicio.

La ventana de Metrics nos ofrece un dashboard en tiempo real de las métricas más importantes de nuestro servicio.

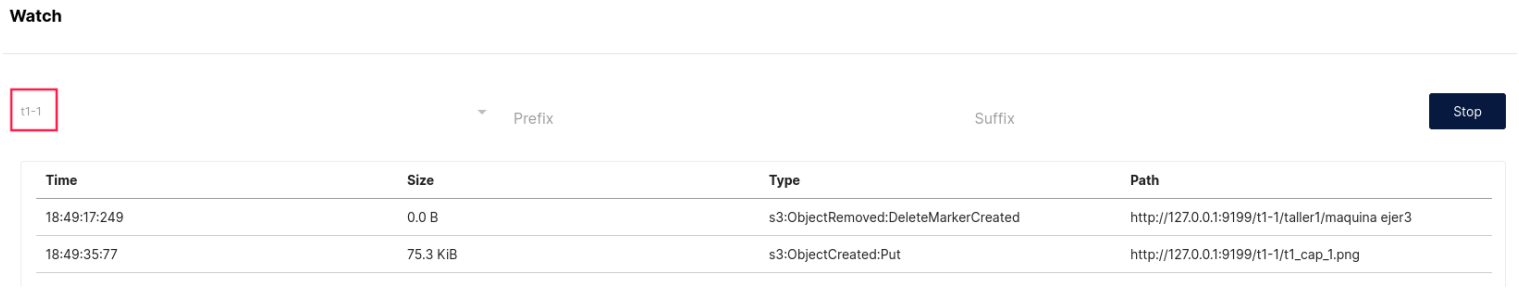


También podemos recopilar logs del servicio en la ventana de “Logs” y auditarlos posteriormente en la ventana “Audit”.

Cuenta además con “Trace” el cual permite habilitar el registro de eventos y trazas de actividad en el servidor MinIO. Proporciona una visibilidad detallada de las operaciones realizadas en el sistema de almacenamiento. Se utiliza principalmente con fines de depuración, monitoreo y análisis de rendimiento.

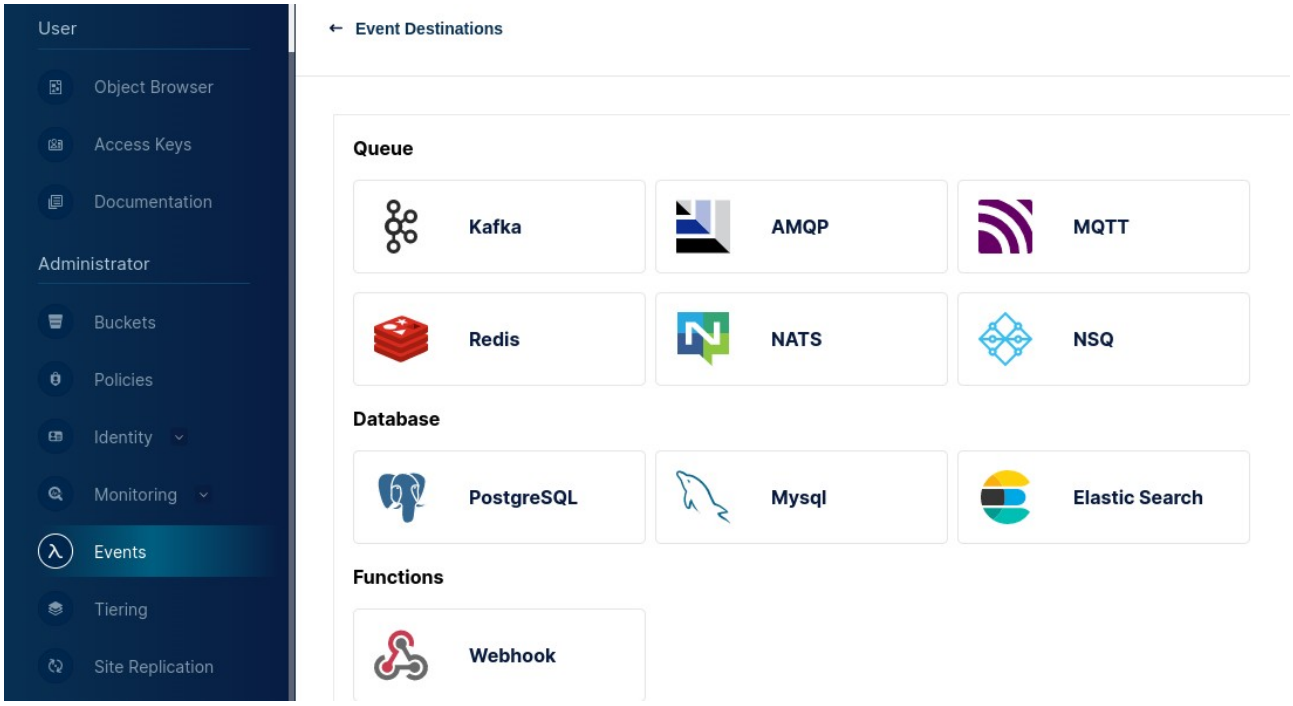


En la ventana “Watch” podremos observar en tiempo real todas las acciones que se realicen sobre nuestros buckets. Voy a subir una nueva imagen al bucket ts-1 y eliminarla al instante y veremos como queda registrado. Para comenzar el monitoreo tendremos que seleccionar en la primera columna el bucket que deseamos monitorizar y dar en “Start”:



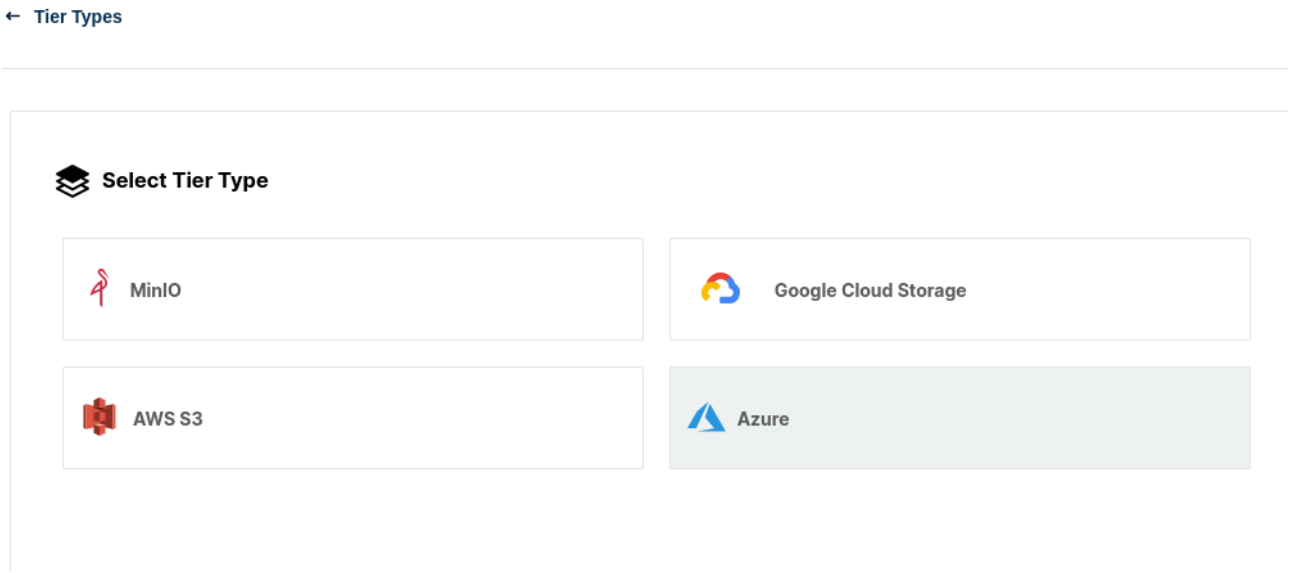
6.5.7.Events

Podemos configurar también notificaciones cuando ocurran eventos determinados en el apartado “Events”. MinIO ofrece una gran variedad de destinos donde enviar dichas notificaciones.



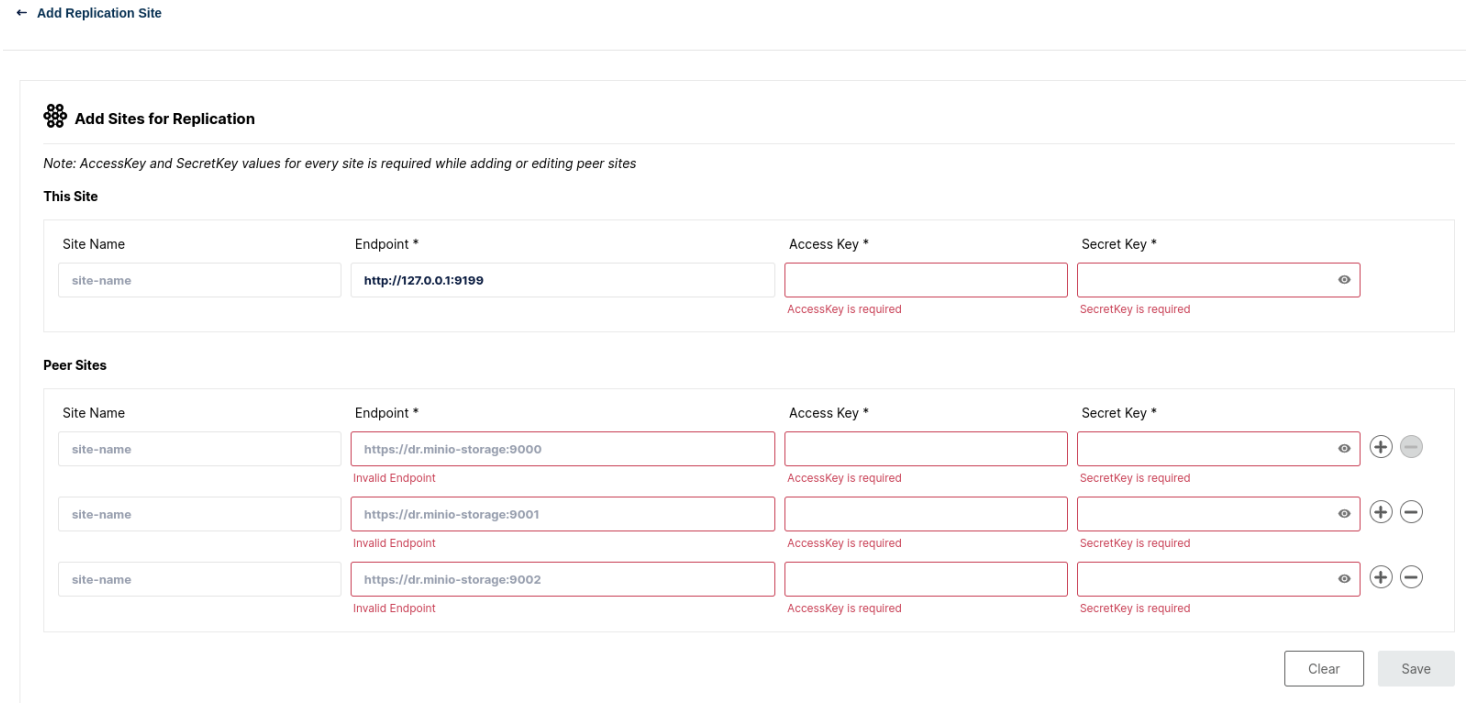
6.5.8.Tiers

En el apartado “Tiers” ofrece la posibilidad de configurar un destino remoto para aquellos buckets que tengan configurados un ciclo de vida especifico, y se transaladan automáticamente. Estos son los servicios remotos que nos ofrece:



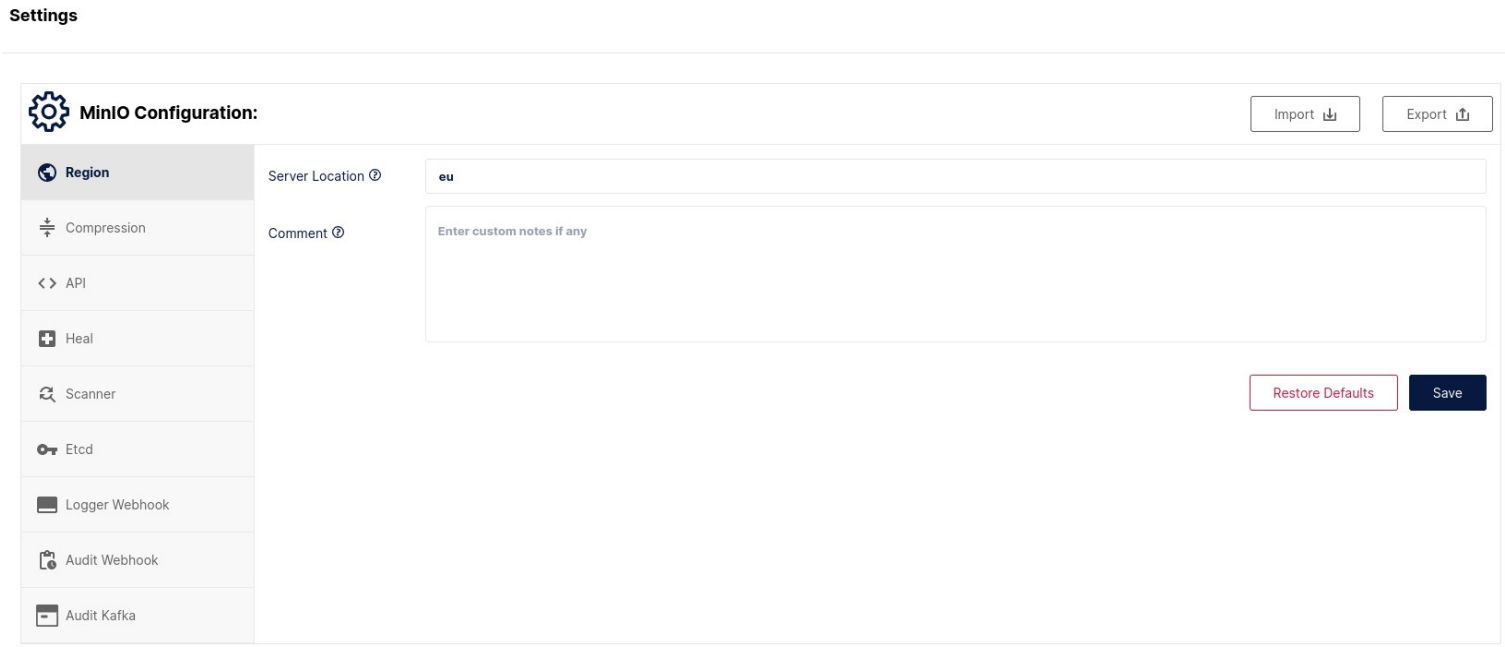
6.5.9.Site Replication

En el apartado “Site Replication” podemos configurar uno o múltiples nodos de replicación con lo que podremos tener nuestro servicio en alta disponibilidad.



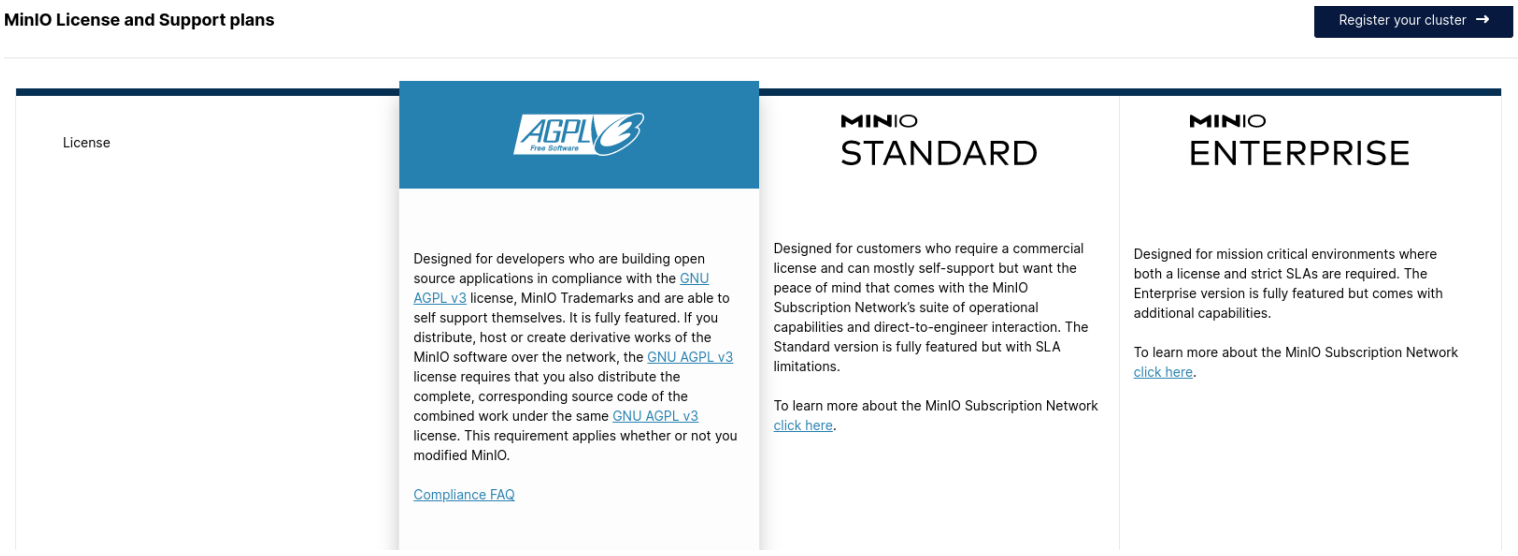
6.5.10.Settings

En este último apartado podremos configurar algunos parámetros para nuestro servidor como la localización de nuestro servidor, los métodos de compresión para los diferentes ficheros, configuración de su API, además de la configuración de varias aplicaciones para recolección de logs como Webhook o para auditoría como Apache Kafka.



Como en casi todo tipo de servicio tenemos una opción para poder exportar o importar la configuración de nuestro servidor.

En este caso estamos utilizando la licencia gratuita de MinIO que nos ofrece todo lo que hemos visto hasta ahora, pero además nos ofrecen soporte oficial con dos tipos de licencias de pago y con algunas opciones más de monitoreo.



Estás son las diferencias entre cada una de ellas:

FEATURES			
Unit Price		**$10 per TiB per month** (Minimum of 100TiB)	**$20 per TiB per month** (Minimum of 100TiB)
Software Release	Upstream	1 Year Long Term Support	5 Years Long Term Support
SLA	No SLA	<48 Hours (Local Business Hours)	<1 hour
Support	Community: Slack + GitHub	L4 Direct Engineering support via SUBNET	L4 Direct Engineering support via SUBNET, Phone, Web Conference
Critical Security and Bug Detection	Self	Continuous Scan and Alert	Continuous Scan and Alert
Panic Button		1 Per year	Unlimited
Health Diagnostics		24/7/365	24/7/365
Annual Architecture Review			✔
Annual Performance Review			✔
Indemnification			✔
Security and Policy Review			✔
	Join Slack	Subscribe	Subscribe

6.6.Administración mediante línea de comandos(cliente).

Con el cliente de MinIO podremos realizar todas las acciones que realizamos en el apartado anterior, incluso algunas más. Esta incorporación es muy útil ya que es un cliente compatible con S3 y con otros servicios como el de google a los cuales podremos conectarnos y gestionar nuestros servidores independientemente del proveedor.

La sintaxis básica del comando es:

```
USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]
```

Los comandos que contiene son los siguientes:

```
COMMANDS:
  alias      manage server credentials in configuration file
  ls         list buckets and objects
  mb         make a bucket
  rb         remove a bucket
  cp         copy objects
  mv         move objects
  rm         remove object(s)
  mirror     synchronize object(s) to a remote site
  cat        display object contents
  head       display first 'n' lines of an object
  pipe       stream STDIN to an object
  find       search for objects
  sql        run sql queries on objects
  stat       show object metadata
  tree       list buckets and objects in a tree format
  du         summarize disk usage recursively
  retention  set retention for object(s)
  legalhold  manage legal hold for object(s)
  support    support related commands
  license    license related commands
  share      generate URL for temporary access to an object
  version    manage bucket versioning
  ilm        manage bucket lifecycle
  quota      manage bucket quota
  encrypt    manage bucket encryption config
  event      manage object notifications
  watch      listen for object notification events
  undo       undo PUT/DELETE operations
  anonymous  manage anonymous access to buckets and objects
  tag        manage tags for bucket and object(s)
  diff       list differences in object name, size, and date between two buckets
  replicate  configure server side bucket replication
  admin      manage MinIO servers
  idp        manage MinIO IDentity Provider server configuration
  update     update mc to latest release
  ready      checks if the cluster is ready or not
  ping       perform liveness check
  od         measure single stream upload and download
  batch      manage batch jobs
```

En primer lugar deberemos saber como obtener la ayuda para formular las ordenes. Para obtener la ayuda principal ejecutamos:

```
mc -h
```

Y para consultar la ayuda de una determinada acción únicamente tendremos que poner:

```
mc COMANDO -h
```

Ejemplo: Consultar ayuda sobre el comando mb.

```
mc mb -h
```

```
NAME:
  mc mb - make a bucket

USAGE:
  mc mb [FLAGS] TARGET [TARGET...]

FLAGS:
  --region value                specify bucket region; defaults to 'us-east-1' (default: "us-east-1")
  --ignore-existing, -p         ignore if bucket/directory already exists
  --with-lock, -l               enable object lock
  --with-versioning             enable versioned bucket
  --config-dir value, -C value  path to configuration folder (default: "/home/paco/.mc")
  --quiet, -q                   disable progress bar display
  --no-color                    disable color theme
  --json                        enable JSON lines formatted output
  --debug                       enable debug output
  --insecure                    disable SSL certificate verification
  --limit-upload value          limits uploads to a maximum rate in KiB/s, MiB/s, GiB/s. (default: unlimited)
  --limit-download value        limits downloads to a maximum rate in KiB/s, MiB/s, GiB/s. (default: unlimited)
  --help, -h                    show help

EXAMPLES:
  1. Create a bucket on Amazon S3 cloud storage.
     $ mc mb s3/mynewbucket

  2. Create a new bucket on Google Cloud Storage.
     $ mc mb gcs/miniocloud

  3. Create a new bucket on Amazon S3 cloud storage in region 'us-west-2'.
     $ mc mb --region=us-west-2 s3/myregionbucket

  4. Create a new directory including its missing parents (equivalent to 'mkdir -p').
     $ mc mb /tmp/this/new/dir1

  5. Create multiple directories including its missing parents (behavior similar to 'mkdir -p').
     $ mc mb /mnt/sdb/mydisk /mnt/sdc/mydisk /mnt/sdd/mydisk

  6. Ignore if bucket/directory already exists.
     $ mc mb --ignore-existing myminio/mynewbucket
```

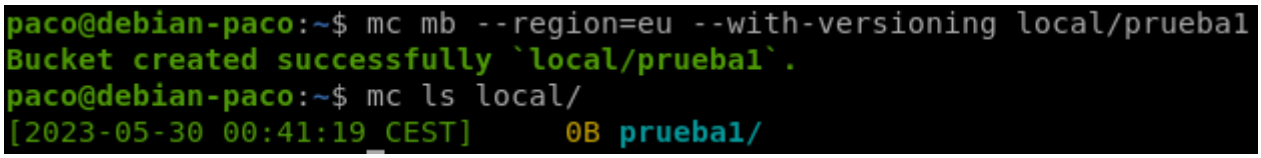
Además de la sintaxis y las distintas opciones que podemos añadir a la orden podemos ver como hay un apartado donde nos indica ejemplos para entender mejor el uso de esa acción. Sin duda esto es algo muy útil para entender y aprender esta tecnología.

Vamos a aprender un uso básico para poder gestionar nuestros servidores de almacenamiento con nuestro cliente.

6.6.1.Crear un bucket.

Vamos a crear un bucket con región en Europa y habilitando el control de versiones.

```
mc mb --region=eu --with-versioning local/prueba1
```

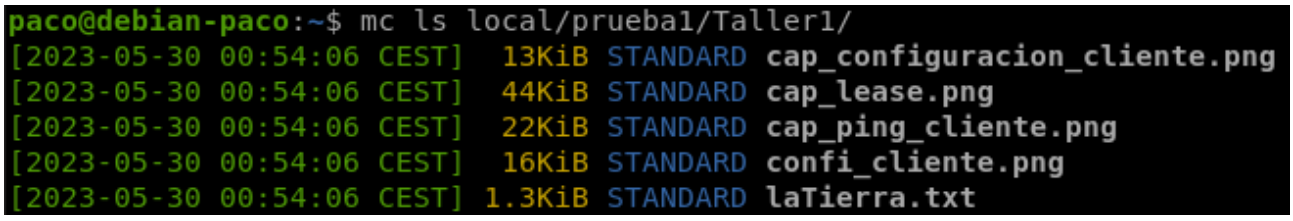


6.6.2.Subir y bajar objetos.

Vamos a subir y bajar(copiar) una carpeta con varios objetos.

```
mc cp -r ./Taller1 local/prueba1
```

```
mc cp -r local/prueba1/Taller1/* downloads/
```



6.6.3.Listar objetos.

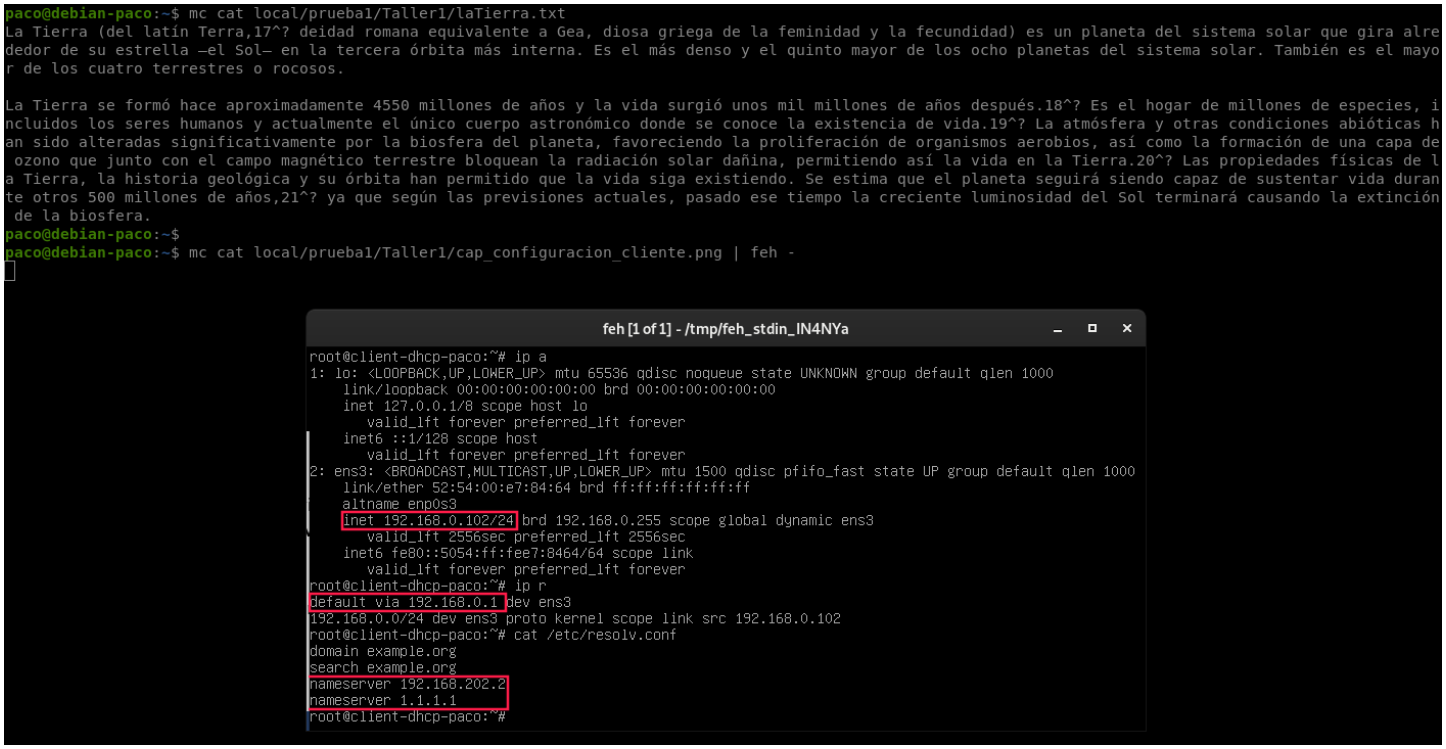
Vamos a listar los objetos que contiene el bucket prueba1.

```
mc ls --summarize --versions -r local/prueba1
```

6.6.4.Mostrar contenido de los objetos.

Vamos a mostrar el contenido de un fichero de texto que hemos subido al bucket y una imagen con la ayuda de un programa externo.

```
mc cat local/prueba1/Taller1/laTierra.txt
mc cat local/prueba1/Taller1/cap_configuracion_cliente.png | feh -
```



6.6.5.Mostrar metadatos de un objeto.

Vamos a mostrar los metadatos de los objetos que subimos.

```
mc stat local/prueba1
```

```
paco@debian-paco:~$ mc stat local/prueba1
Name      : prueba1
Date      : 2023-05-30 00:59:20 CEST
Size      : N/A
Type      : folder

Properties:
  Versioning: Enabled
  Location: eu
  Anonymous: Disabled
  ILM: Disabled

Usage:
      Total size: 96 KiB
   Objects count: 5
  Versions count: 5

Object sizes histogram:
   0 object(s) less than 1024 bytes
   5 object(s) between 1024 bytes and 1 MB
   0 object(s) between 1 MB and 10 MB
   0 object(s) between 10 MB and 64 MB
   0 object(s) between 64 MB and 128 MB
   0 object(s) between 128 MB and 512 MB
   0 object(s) greater than 512 MB
paco@debian-paco:~$ mc stat local/prueba1/Taller1/laTierra.txt
Name      : laTierra.txt
Date      : 2023-05-30 00:54:06 CEST
Size      : 1.3 KiB
ETag      : 83988d58735182f78608c101257bd0f9
VersionID : d8ea51f1-9c9f-4cf0-a6b7-ae274999aa83
Type      : file
Metadata  :
  Content-Type: text/plain
```

6.6.6.Buscar objetos.

Vamos a buscar objetos con la extensión .png en el bucket que subimos.

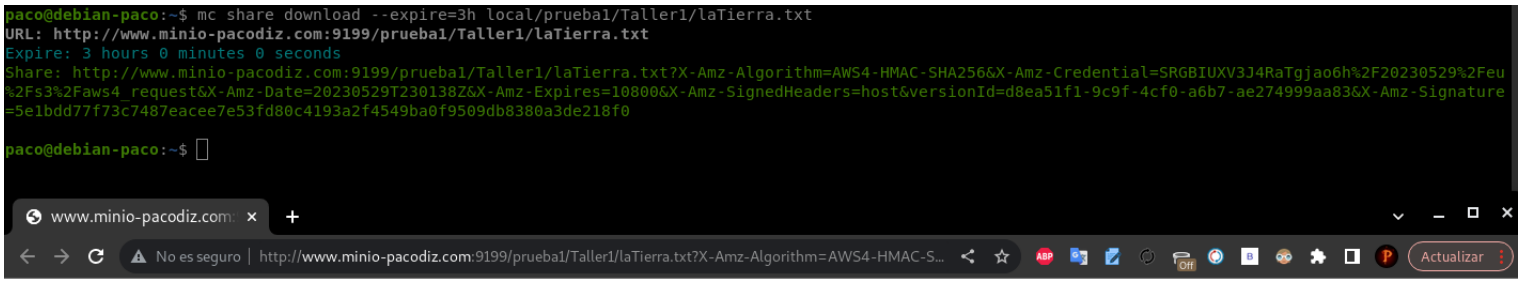
```
mc find --name *.png local/prueba1
```

```
paco@debian-paco:~$ mc find --name *.png local/prueba1
local/prueba1/Taller1/cap_configuracion_cliente.png
local/prueba1/Taller1/cap_lease.png
local/prueba1/Taller1/cap_ping_cliente.png
local/prueba1/Taller1/confi_cliente.png
```

6.6.7.Compartir un objeto temporalmente.

Vamos a compartir un objeto para descargar por 3 horas.

```
mc share download --expire=3h local/prueba1/Taller1/laTierra.txt
```



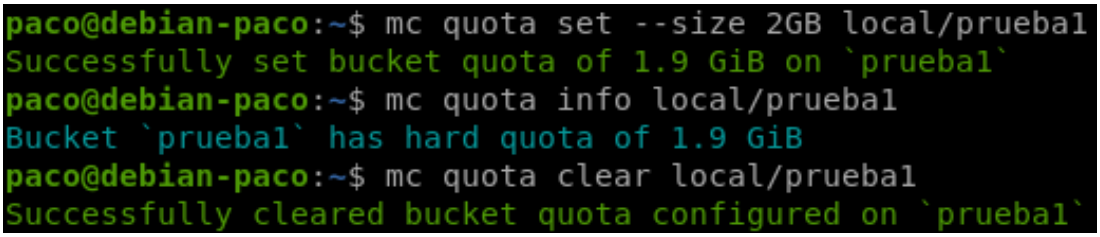
La Tierra (del latÃn Terra,17â€‹ deidad romana equivalente a Gea, diosa griega de la feminidad y la fecundidad) es un planeta del sistema solar que gira alrededor de su estrella â€"el Solâ€" en la tercera Ã³rbita mÃ¡s interna. Es el mÃ¡s denso y el quinto mayor de los ocho planetas del sistema solar. TambiÃ©n es el mayor de los cuatro terrestres o rocosos.

La Tierra se formÃ³ hace aproximadamente 4550 millones de aÃ±os y la vida surgiÃ³ unos mil millones de aÃ±os despuÃ©s.18â€‹ Es el hogar de millones de especies, incluidos los seres humanos y actualmente el Ãºnico cuerpo astronÃ³mico donde se conoce la existencia de vida.19â€‹ La atmÃ³sfera y otras condiciones abiÃ³ticas han sido alteradas significativamente por la biosfera del planeta, favoreciendo la proliferaciÃ³n de organismos aerobios, asÃ como la formaciÃ³n de una capa de ozono que junto con el campo magnÃ©tico terrestre bloquean la radiaciÃ³n solar daÃ±ina, permitiendo asÃ la vida en la Tierra.20â€‹ Las propiedades fÃsicas de la Tierra, la historia geolÃ³gica y su Ã³rbita han permitido que la vida siga existiendo. Se estima que el planeta seguirÃ¡ siendo capaz de sustentar vida durante otros 500 millones de aÃ±os,21â€‹ ya que segÃºn las previsiones actuales, pasado ese tiempo la creciente luminosidad del Sol terminarÃ¡ causando la extinciÃ³n de la biosfera.

6.6.8.Establecer una quota en un bucket.

Vamos a asignarle al bucket prueba1 una cuota de 2GB, veremos la información de la cuota y por último la eliminaremos.

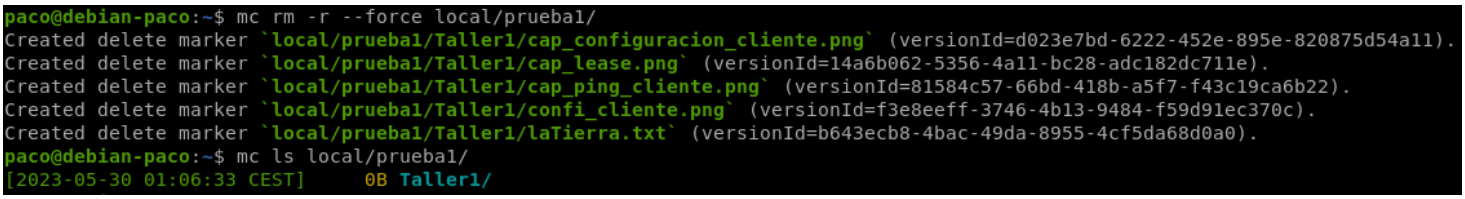
```
mc quota set --size 2GB local/prueba1

mc quota info local/prueba1

mc quota clear local/prueba1
```



6.6.9.Eliminar objetos.

Vamos a eliminar todos los objetos que subimos.

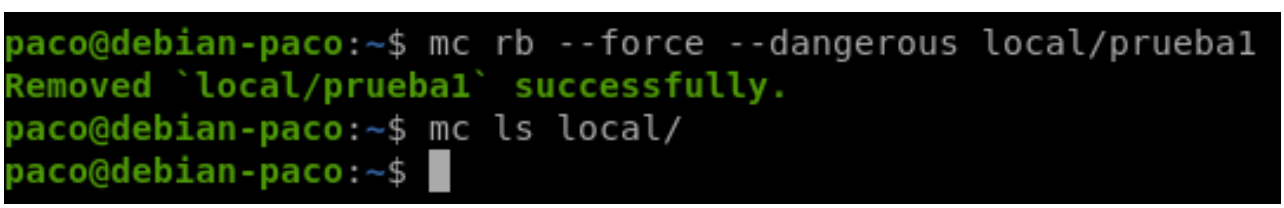
```
mc rm -r --force local/prueba1/
```



6.6.10.Eliminar bucket.

Vamos a eliminar el bucket prueba1.

```
mc rb local/prueba1
```

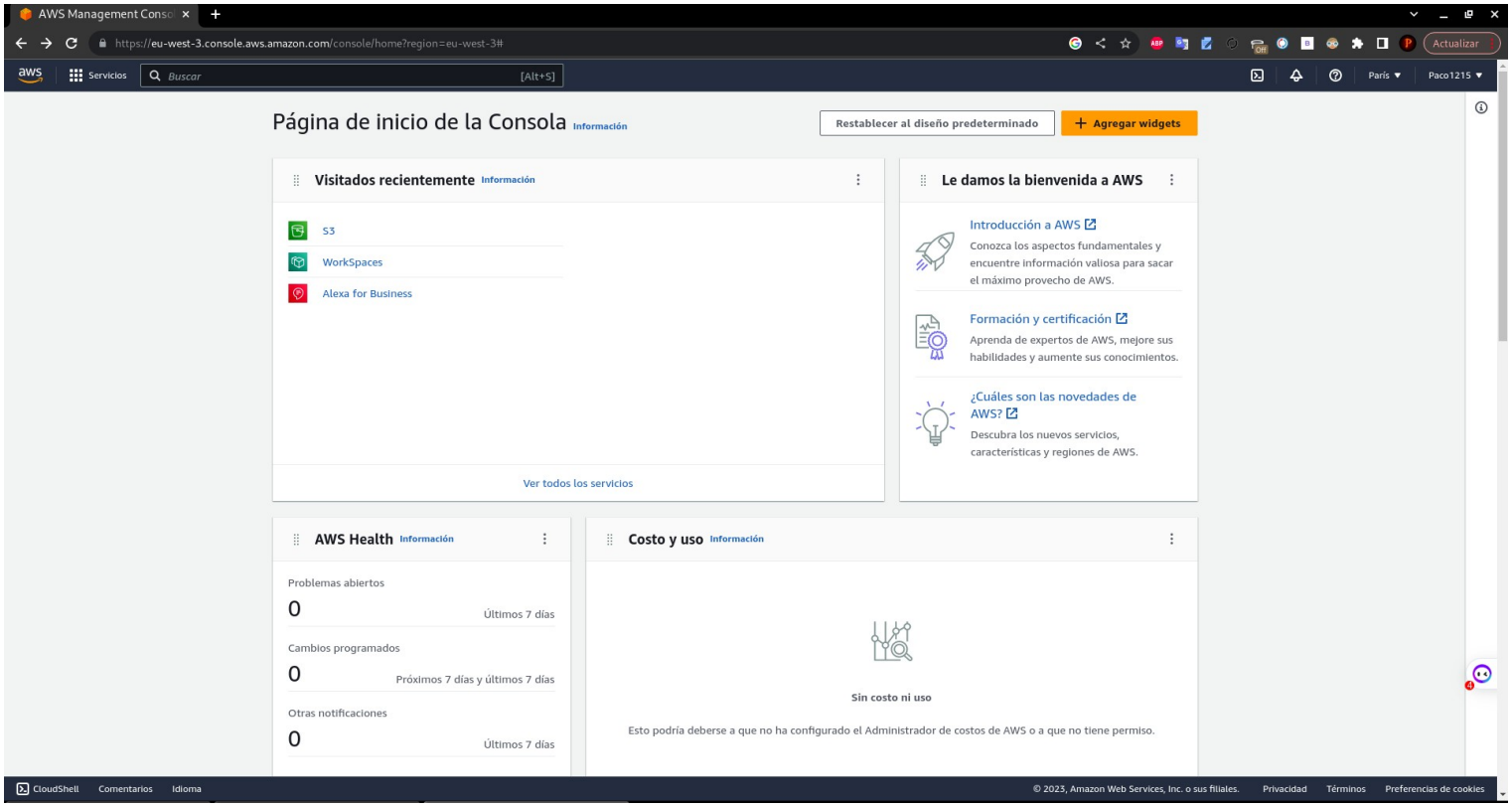


6.7.Almacenamiento de objetos en S3.

En este punto veremos algunos conceptos sobre la interfaz web de S3 AWS que se asemejan a lo que ya vimos en el servidor de MinIO. Con esto veremos que no existe mucha diferencia entre servidores de distintos proveedores y saber que con los conocimientos adecuados sobre uno podríamos utilizar el resto.

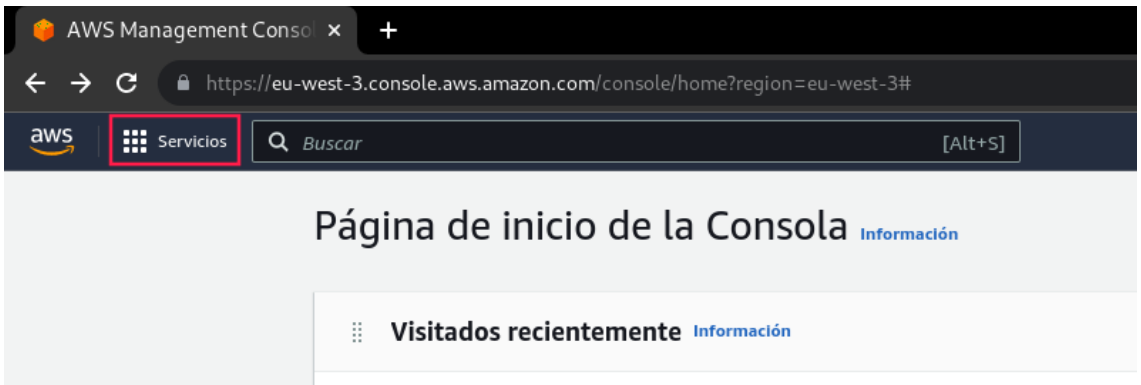
6.7.1.Conceptos básicos para administrar S3 desde consola web.

Tras iniciar sesión con nuestra cuenta de Amazon nos aparecerá la siguiente ventana:

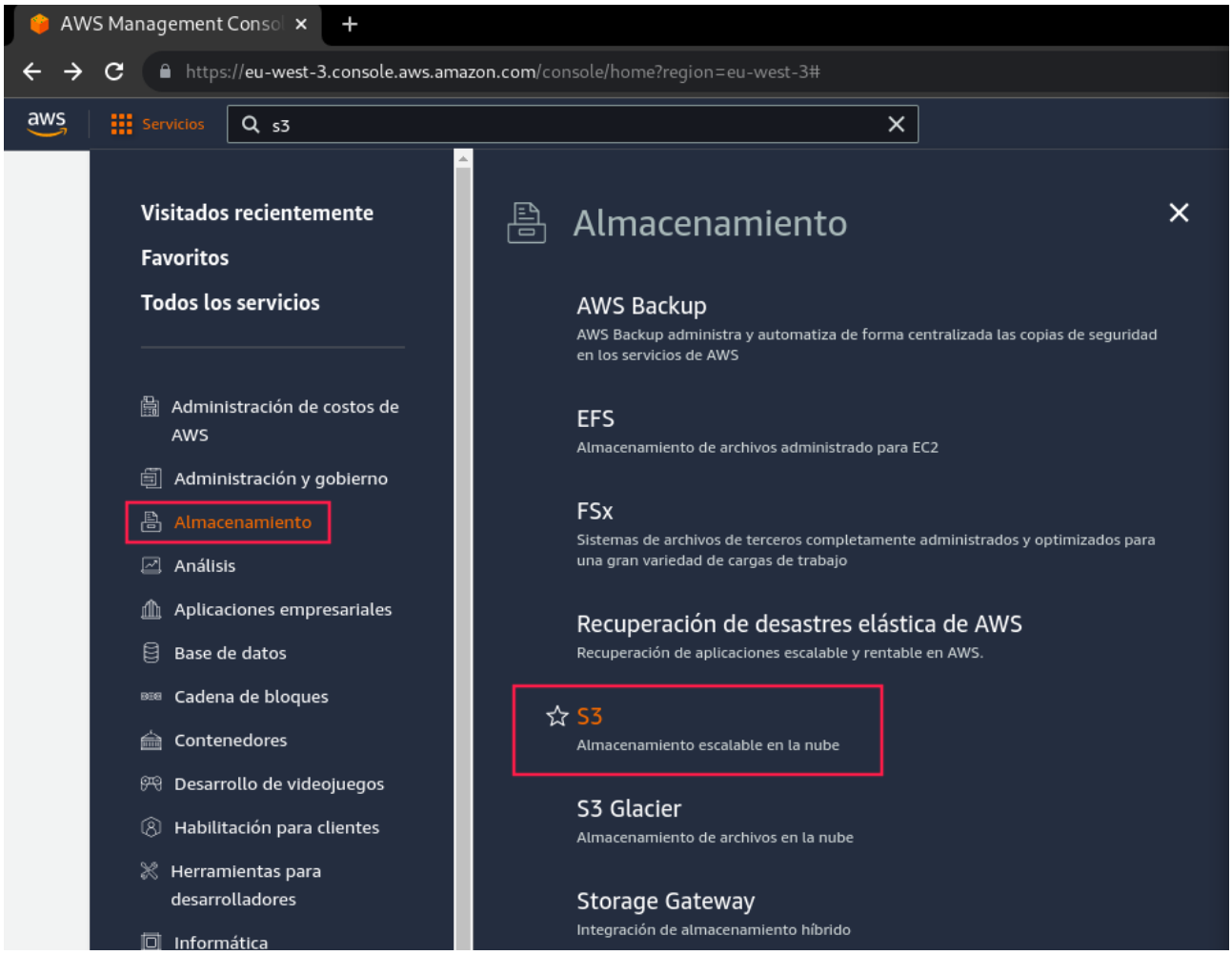


La región en este caso no nos importa ya que por defecto en S3 nos asigna “Global”.

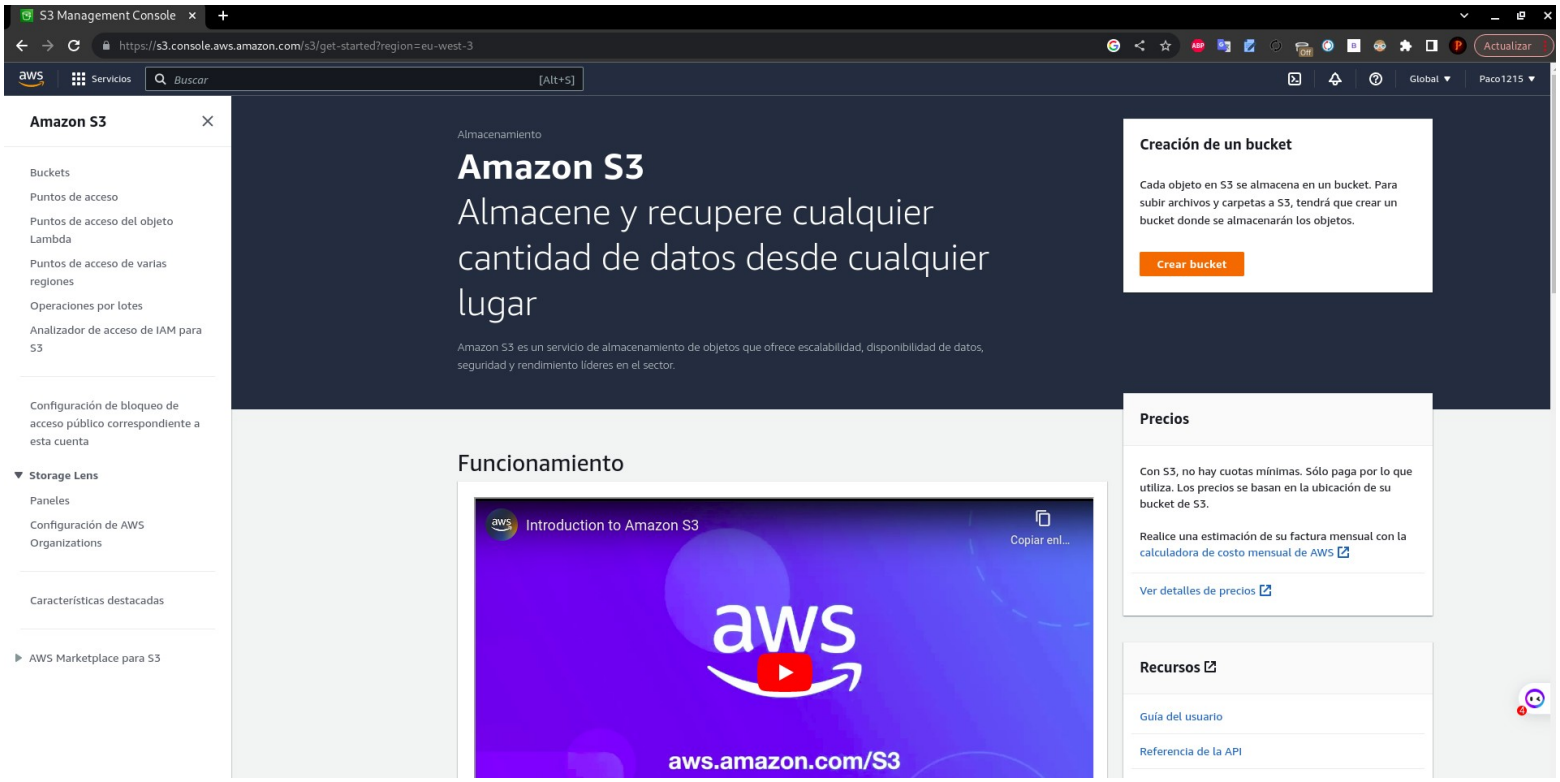
Seguidamente para entrar en S3 nos iremos a la parte superior izquierda de la ventana y haremos clic en “Servicios”:



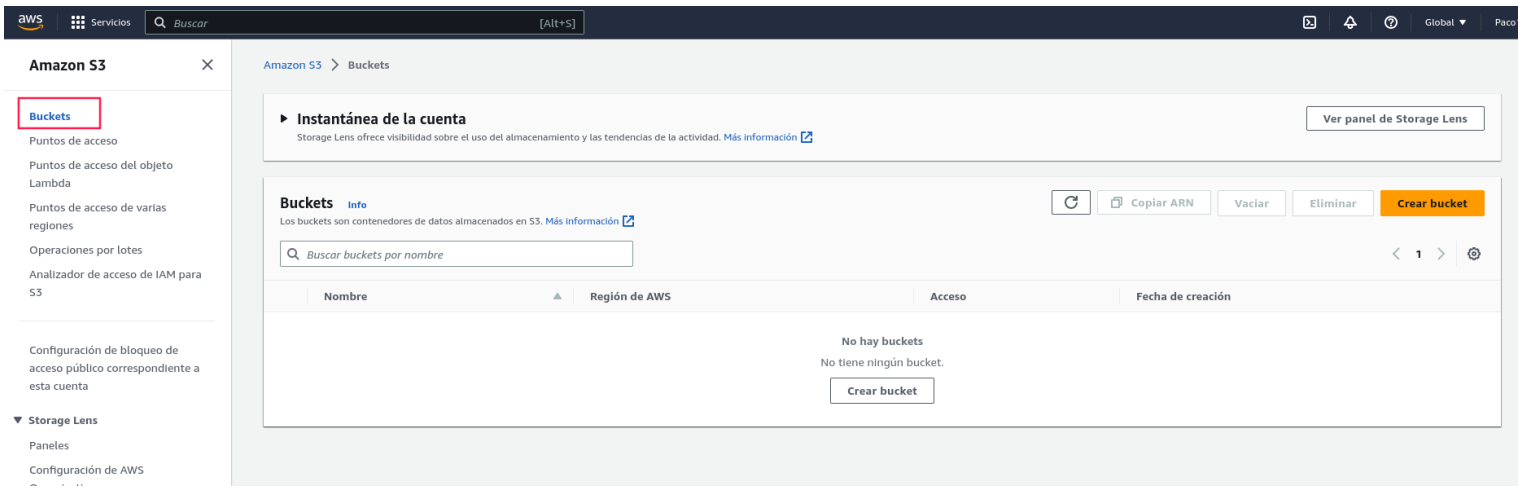
Y en el apartado de “Almacenamiento” encontraremos “S3”, daremos encima para entrar:



Una vez dentro veremos la interfaz principal de S3:

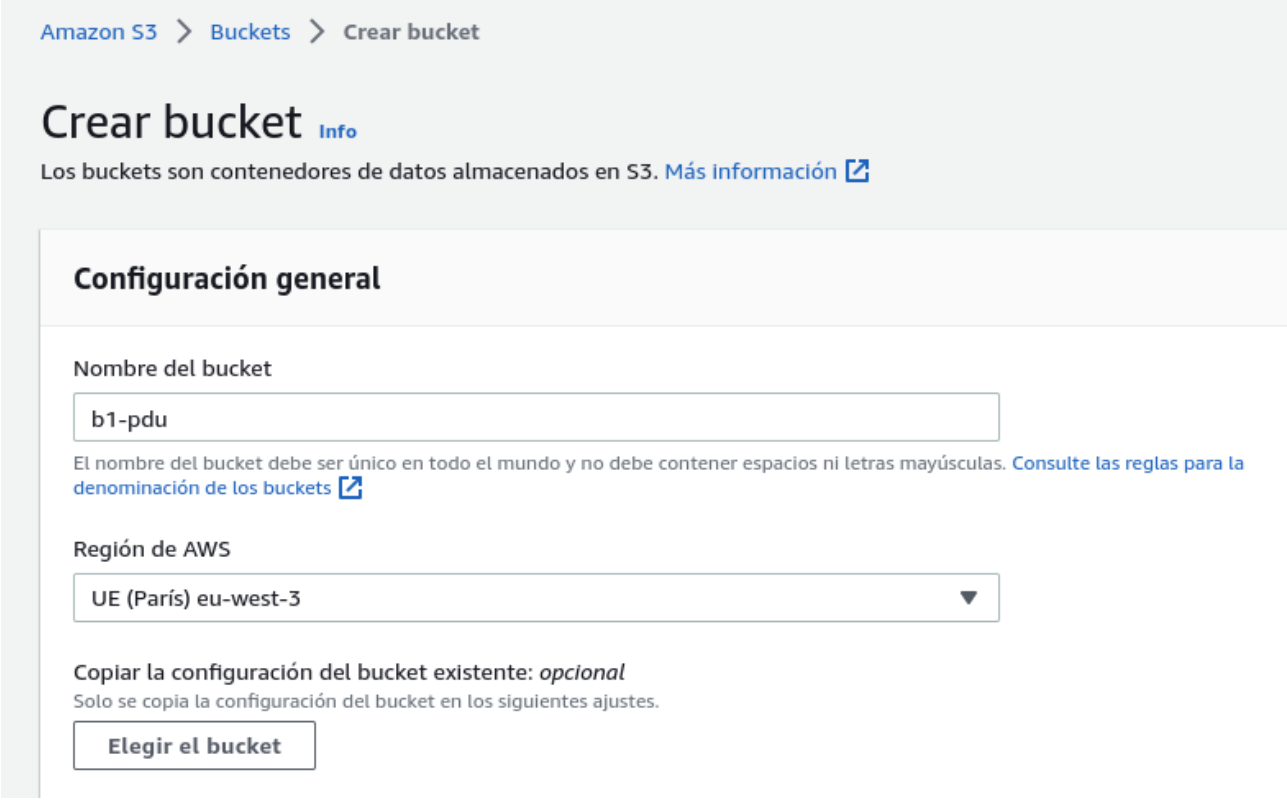


Pasaremos a crear nuestro primer bucket. Para ello entraremos al apartado “Buckets”:



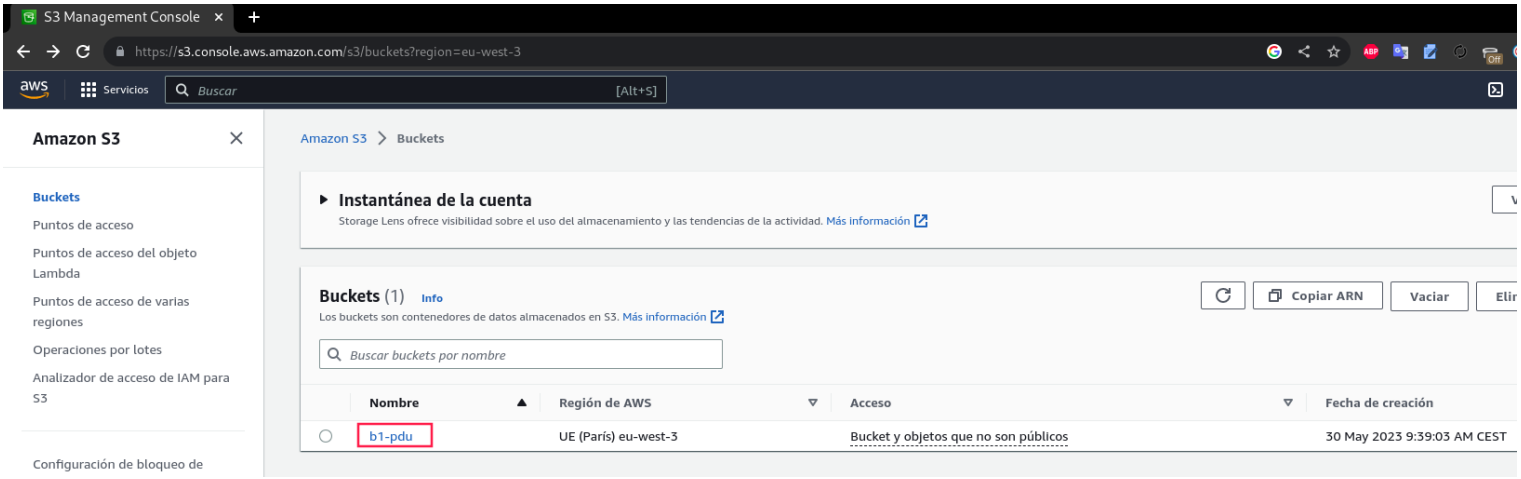
Seguidamente daremos en “Crear bucket” y se nos abrirá un menú para configurarlo. En este caso vamos a realizar una configuración simple, la cual veremos en las siguientes imágenes:

Es muy importante que sepamos que el nombre del bucket tiene que ser único en el mundo por lo que tendremos que elegir bien. También es importante es seleccionar una región cercana a nuestra localización para agilizar la latencia.

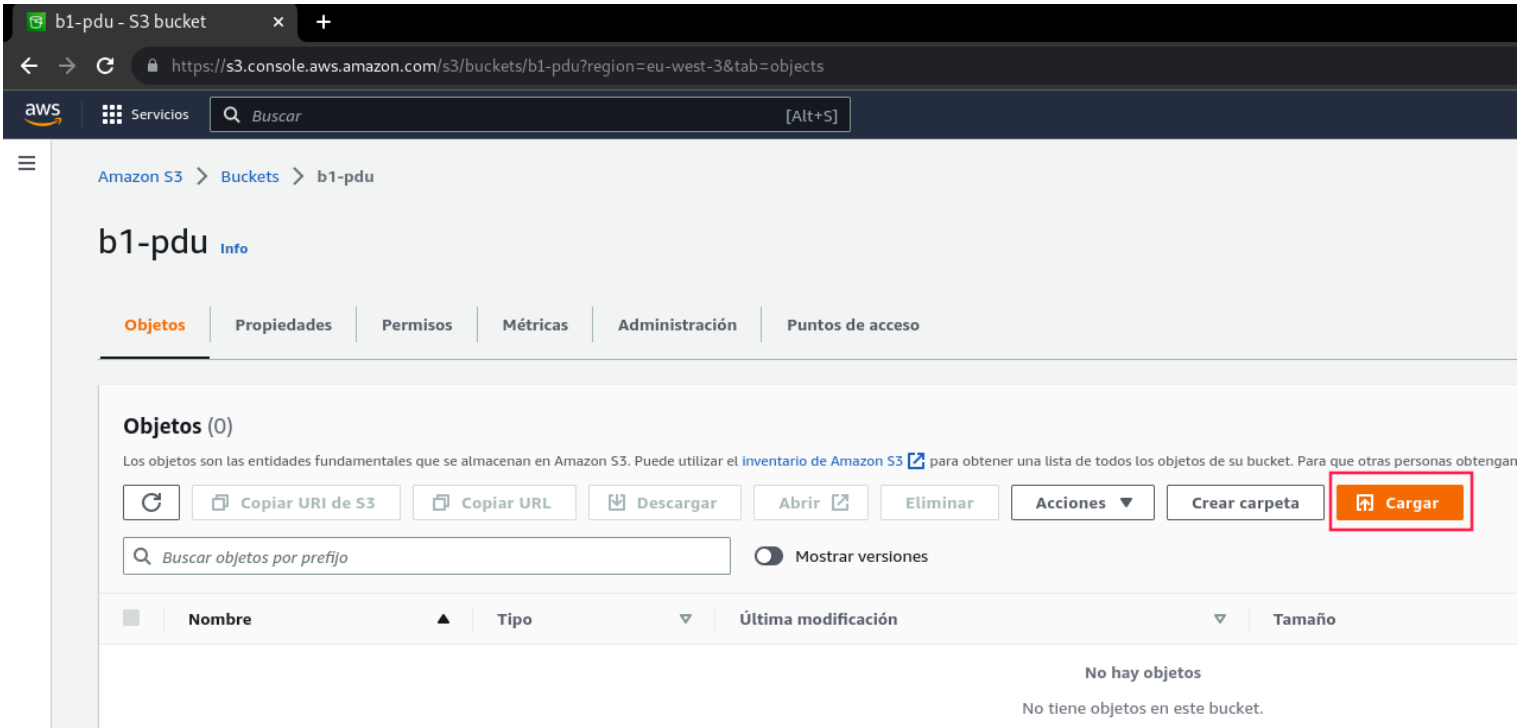


El resto lo dejaremos por defecto.

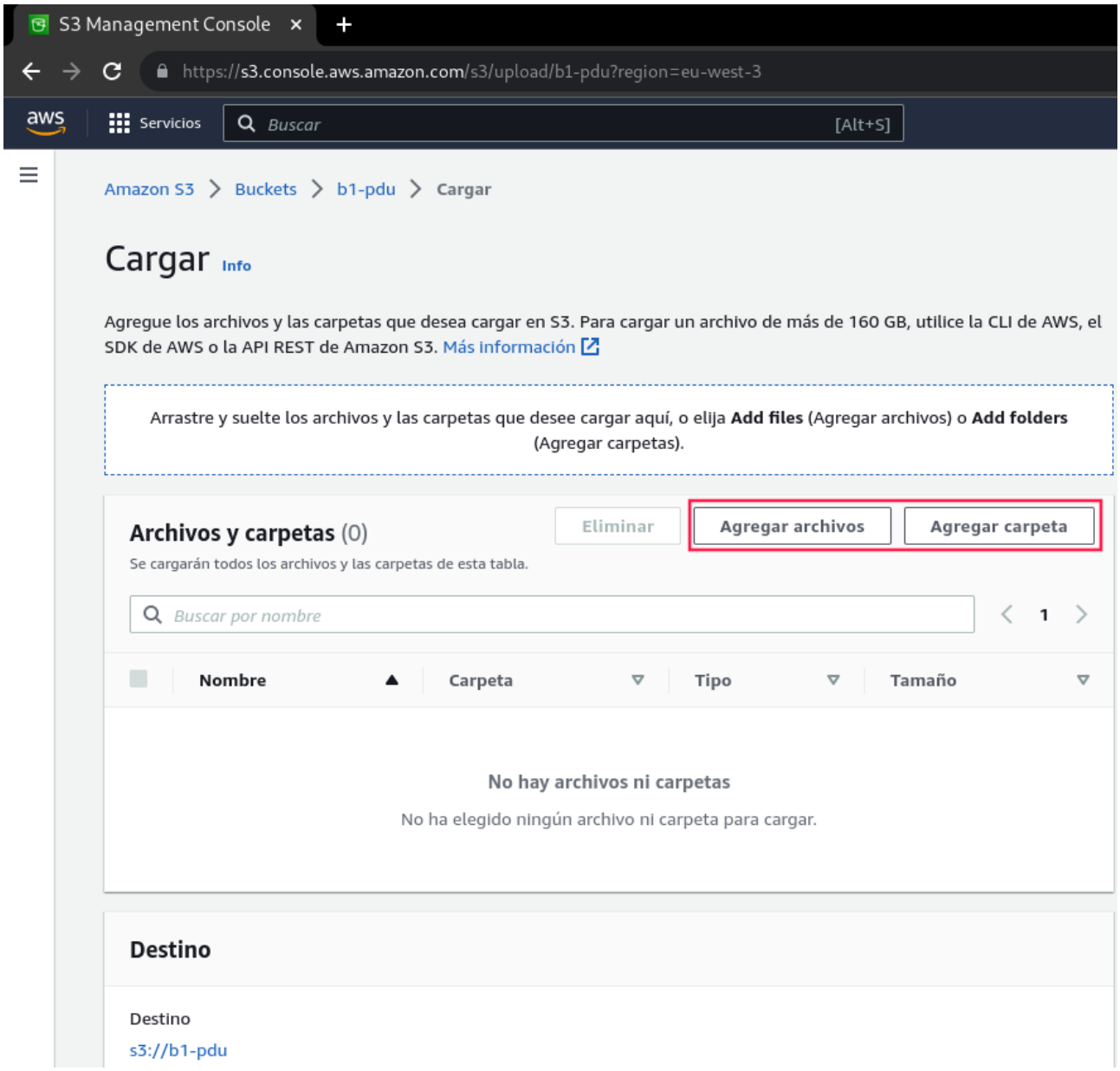
Con esto ya tendríamos disponible nuestro bucket en S3. Para subir nuevos objetos solo tendremos que seleccionar el bucket:



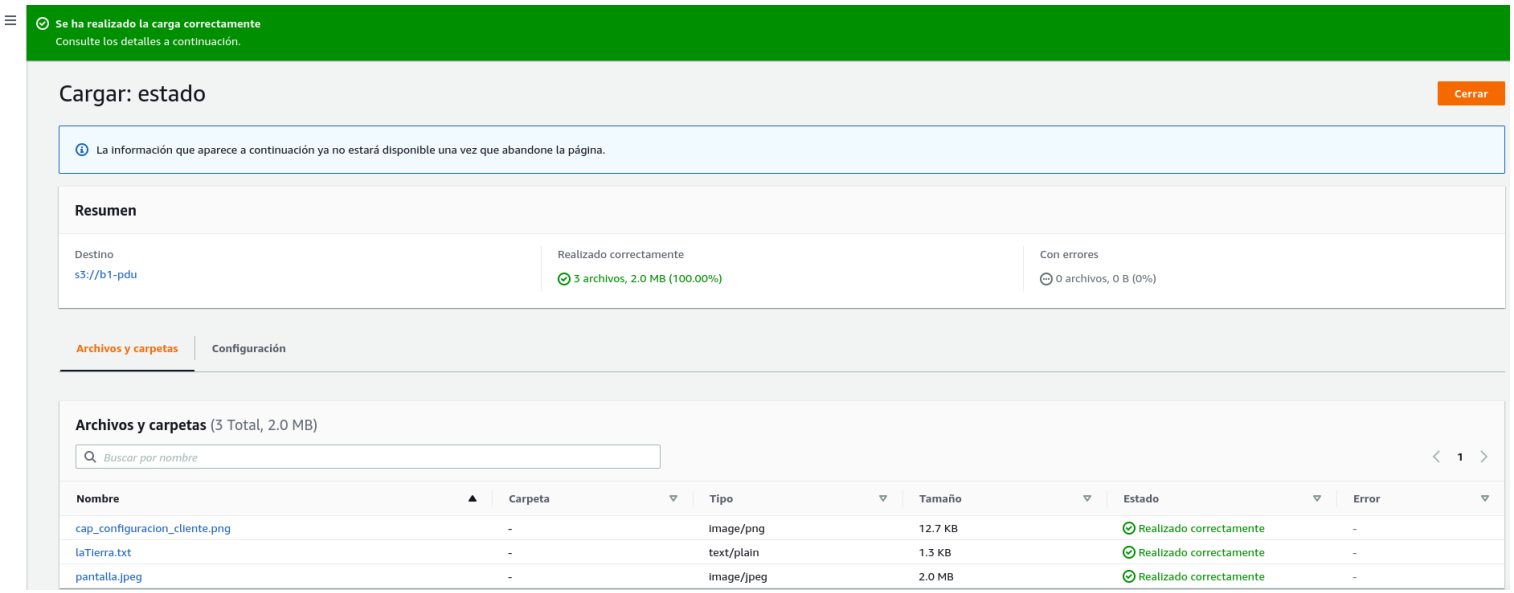
Damos en “Cargar”



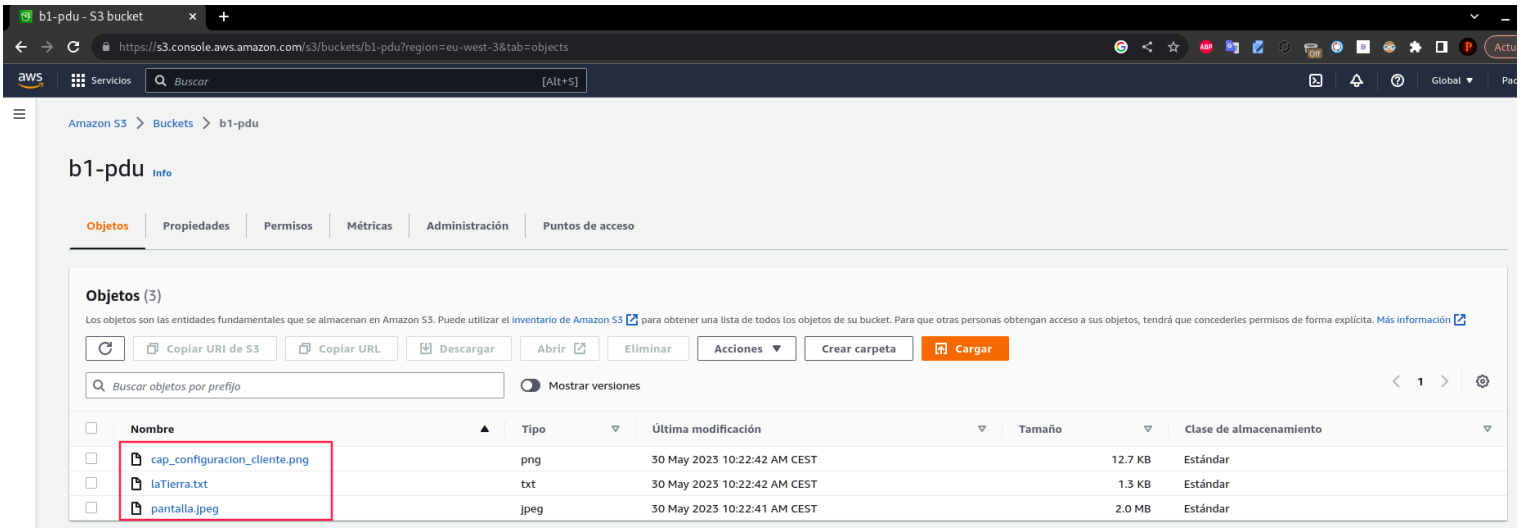
Se nos abrirá una nueva ventana en la que podremos agregar carpetas completas y/o ficheros.



Una vez termine el proceso de subida no abrirá una ventana con el estado de la subida:



Y ya podemos ver como aparecen dentro del bucket:



Al igual que en MinIO disponemos de todo tipo de ajustes de permisos, accesos anónimos, propiedades de cada objeto y bucket, métricas,etc.

Para eliminar el bucket es muy sencillo e intuitivo, seleccionaremos primero el contenido y lo eliminaremos. Seguidamente ya podremos eliminar el bucket.

La consola de S3 es muy extensa y nos daría para realizar un proyecto de esta sola fácilmente, así que solo veremos estas nociones básicas en S3 que nos servirán el funcionamiento que tendrá S3 en las demos.

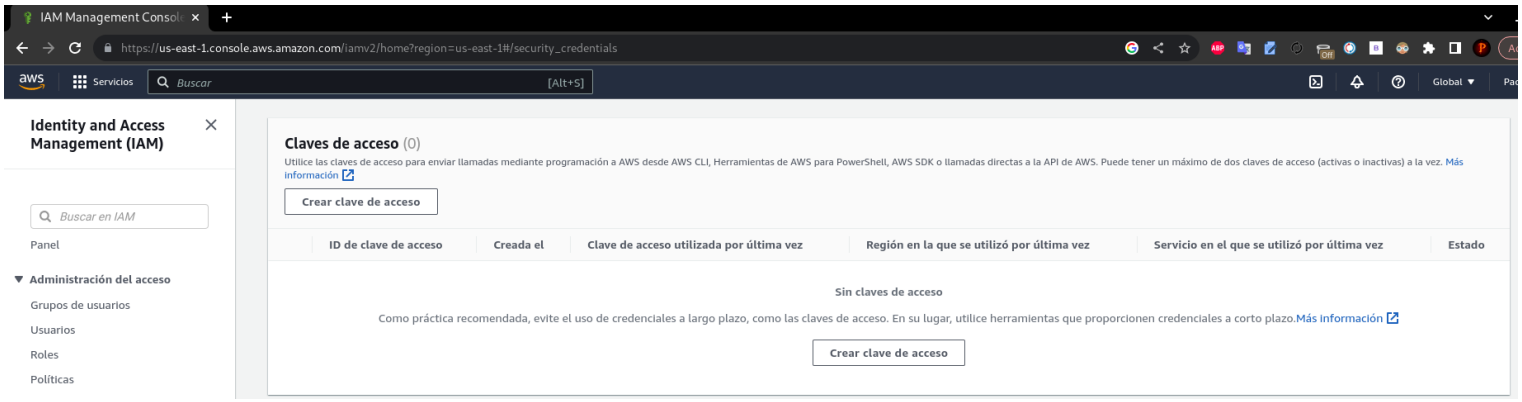
6.7.2.Integración del cliente MinIO con S3.

Una de las ventaja que tiene MinIO es que tiene una muy buena integración con el servicio de S3. Esto quiere decir que por ejemplo podremos utilizar el cliente de MinIO “mc”, que vimos anteriormente, para administrar S3 en su totalidad.

Para ello tendremos que establecer una conexión de nuestro cliente a S3 y para ello necesitaremos crear una clave de acceso que nos permitirá autenticarnos en el servicio.

La clave la generaremos accediendo a la consola web de AWS y dando clic encima de nuestro nombre de usuario, justo en la esquina superior derecha, y seleccionando “Credenciales de seguridad”.

Bajaremos hasta “Claves de acceso” y daremos en “Crear clave de acceso”.



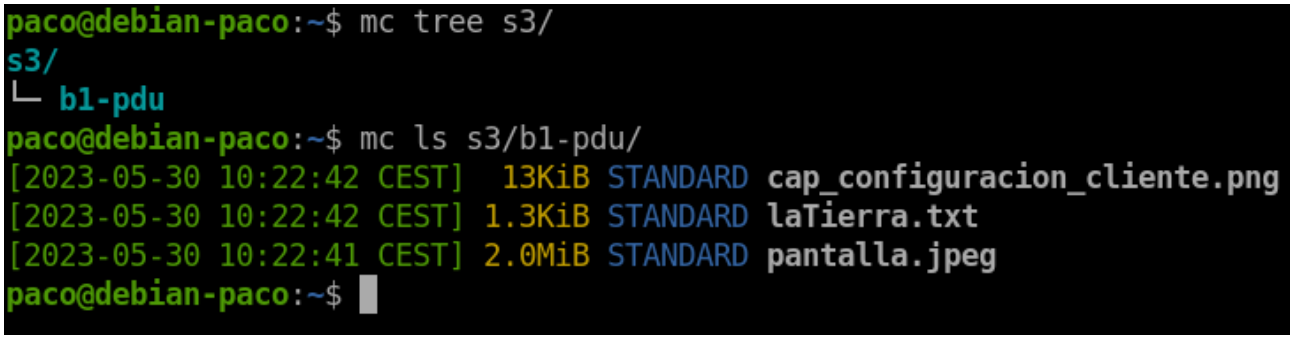
En la siguiente ventana nos avisa que no es del todo seguro, aceptamos y damos en “Siguiente”. Y con esto se nos abrirá otra ventana en la cual nos aparecerá nuestra clave de acceso junto a una clave secreta la cual tendremos que guardar ya que nada más que salgamos de esta ventana no podremos volver a consultarla.

Así nos debería de quedar:

Seguidamente nos iremos a nuestro cliente de MinIO para configurar la conexión. Editaremos el fichero de configuración .mc/config.json añadiendo la clave de acceso al alias S3 que ya viene creado.

```
 GNU nano 5.4                                .mc/config.json
{
        "version": "10",
        "aliases": {
                "gcs": {
                        "url": "https://storage.googleapis.com",
                        "accessKey": "YOUR-ACCESS-KEY-HERE",
                        "secretKey": "YOUR-SECRET-KEY-HERE",
                        "api": "S3v2",
                        "path": "dns"
                },
                "local": {
                        "url": "http://www.minio-pacodiz.com:9199",
                        "accessKey": "SRGBIUXV3J4RaTgjao6h",
                        "secretKey": "oQzOFlZwVOyMX7Vshi8CvJvdgabcKYe2tAJtaMYT",
                        "api": "S3v4",
                        "path": "auto"
                },
                "play": {
                        "url": "https://play.min.io",
                        "accessKey": "Q3AM3UQ867SPQQA43P2F",
                        "secretKey": "zuf+tfteSlswRu7BJ86wekitnifILbZam1KYY3TG",
                        "api": "S3v4",
                        "path": "auto"
                },
                "s3": {
                        "url": "https://s3.amazonaws.com",
                        "accessKey": "AKIAUEQOAEGAZRJET2ZT",
                        "secretKey": "[redacted]",
                        "api": "S3v4",
                        "path": "dns"
                }
        }
}
```

Ahora al usar el cliente con dicho alias tendríamos que poder ver el bucket que creamos en el apartado anterior:



Como podemos ver en la imagen funciona correctamente. Desde aquí ya podríamos utilizar la totalidad de comandos que nos ofrece el cliente para administrar nuestro almacenamiento de objetos en S3.

6.8.Demos.

6.8.1.Demo-1.

Escenario necesario para la demo:

- Servidor MinIO local
- Servidor Web local alojando página web y con cliente MinIO instalado.

Explicación de la demo:

El objetivo de esta demo es el de montar un servidor web y almacenar todo el contenido estático de en un bucket de nuestro servidor MinIO para no tener que ocupar un espacio innecesario en este.

URL de los ficheros: https://github.com/Pacodiz02/PI/tree/main/demos/demo1

Reproducción de la demo:

En primer lugar tendremos que montar y configurar un servidor MinIO, pero en este caso como ya hemos montado uno para la explicación de los anteriores puntos usaremos ese mismo servidor.

6.2.Configuración del servidor MinIO
6.3.Instalación del cliente MinIO

Seguidamente montaremos el servidor web con un apache2 y un virtualhost. Para ello:

Instalamos apache2:

```
sudo apt update && sudo apt install -y apache2
```

Creamos un nuevo virtualhost:

```
cd /etc/apache2/sites-available/
```

```
cp 000-default.conf web.conf
```

Editamos el nuevo fichero y colocamos la siguiente configuración:

```
<VirtualHost *:80>

        DocumentRoot /var/www/web
        ServerAdmin webmaster@localhost
        ServerName www.pre-app1.com

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

  </VirtualHost>
```

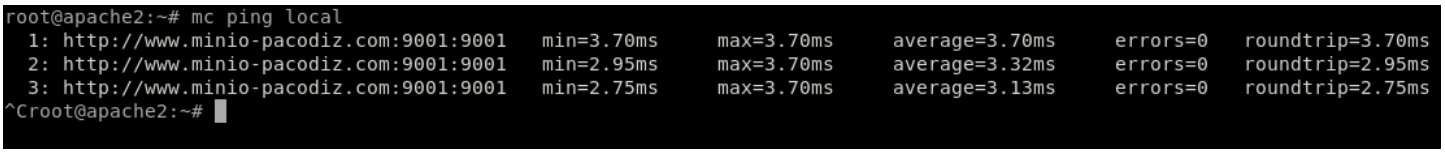
Creamos el DocumentRoot el cual únicamente contendrá los ficheros .html:

```
mkdir /var/www/web/
```

Ahora necesitaremos instalar el cliente MinIO en la máquina donde tenemos el servidor web tal y como explicamos en el punto 6.3.Instalación del cliente MinIO.
Esto lo haremos para poder crear un nuevo bucket para la página web y subir todo el contenido estático a este.

Una vez lo tengamos instalado tendremos que configurar la conexión con el servidor MinIO para la cual utilizaremos la misma clave que en la explicación.

Comprobamos que tenemos conexión con el servidor remoto:



Para crear el bucket ejecutamos:

```
mc mb local/web
```

Y para subir los archivos ejecutamos:

```
mc cp -r ./web-estatico/ local/web
```

Ahora moveremos los ficheros .html al DocumentRoot:

```
cp -r ./web-html/ /var/www/web/
```

Después de esto tendremos que ajustar los ficheros .html para que cojan el contenido estático de las nuevas rutas.

Por último habilitamos el servicio:

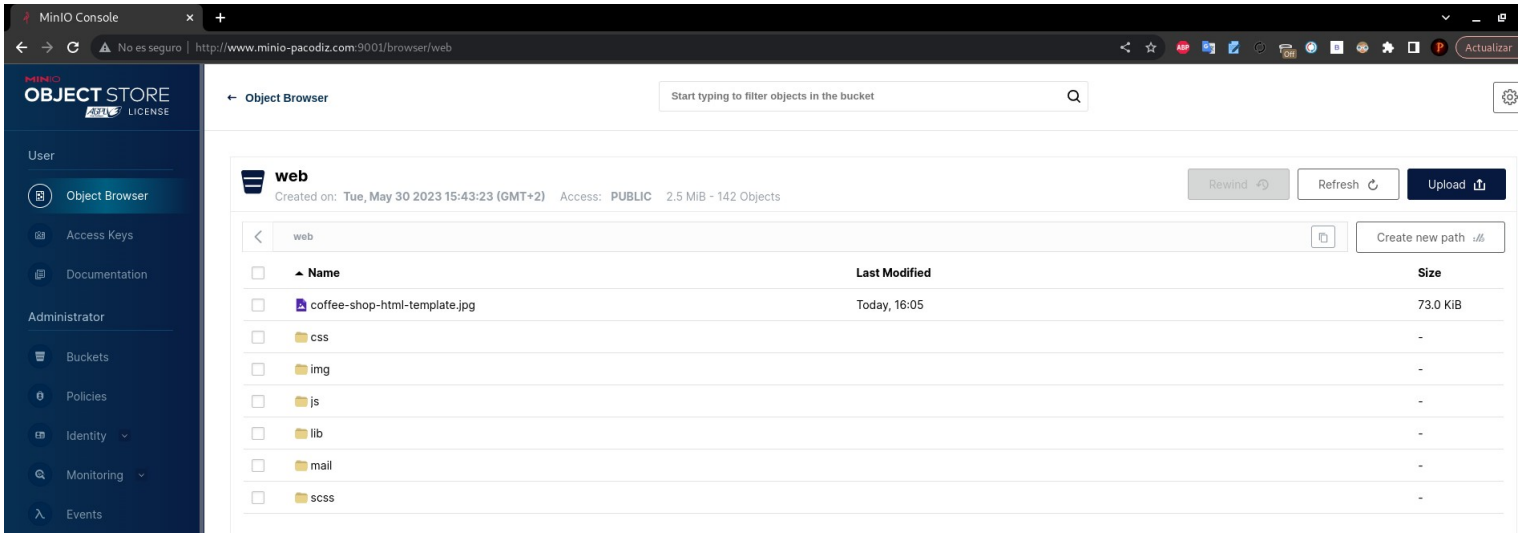
```
ln -s /etc/apache2/sites-available/web.conf /etc/apache2/sites-enabled/
```

Y reiniciamos el servicio para aplicar la configuración:

```
systemctl restart apache2
```

Comprobaciones:

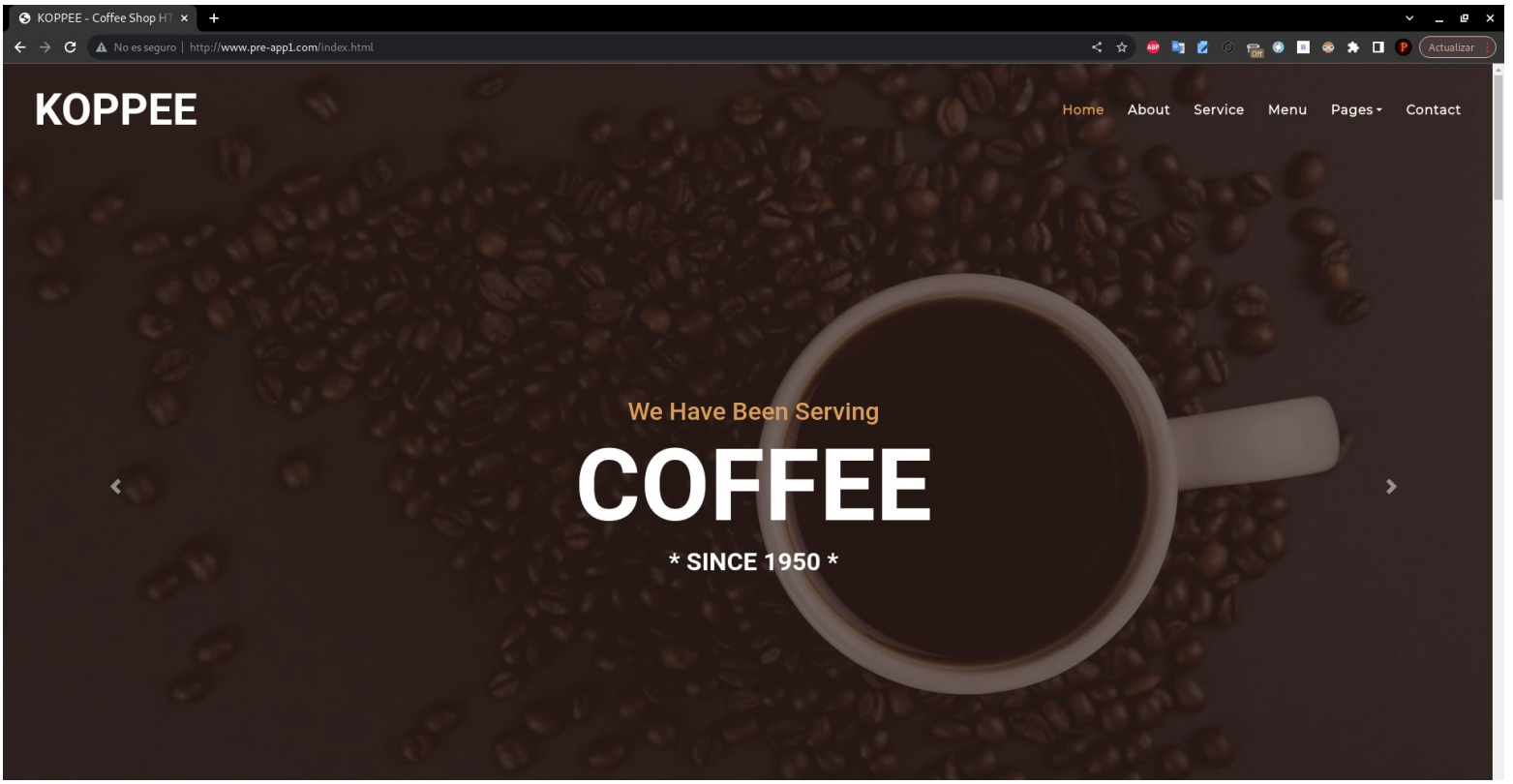
Desde la interfaz web de MinIO podemos ver como se ha creado correctamente el bucket y como contiene todos los archivos que subimos desde el cliente:



Para acceder a la url web local sin un dns previo configurado tendremos que añadir las siguiente línea al fichero /etc/hosts del equipo con el que visitemos la web.

```
192.168.122.215      www.pre-app1.com
```

Comprobamos que el contenido de la página web se ve correctamente:



6.8.2.Demo-2.

Escenario necesario para la demo:

- AWS S3
- Servidor Web en la nube con cliente MinIO instalado.

Explicación de la demo:

El objetivo de esta demo es muy similar a Demo-1 con la diferencia de en vez de utilizar como servidor de almacenamiento MinIO, utilizaremos AWS S3 y que lo haremos en internet y no en local.

Simulando así con la Demo1 un entorno de preproducción y Demo2 un entorno de producción.

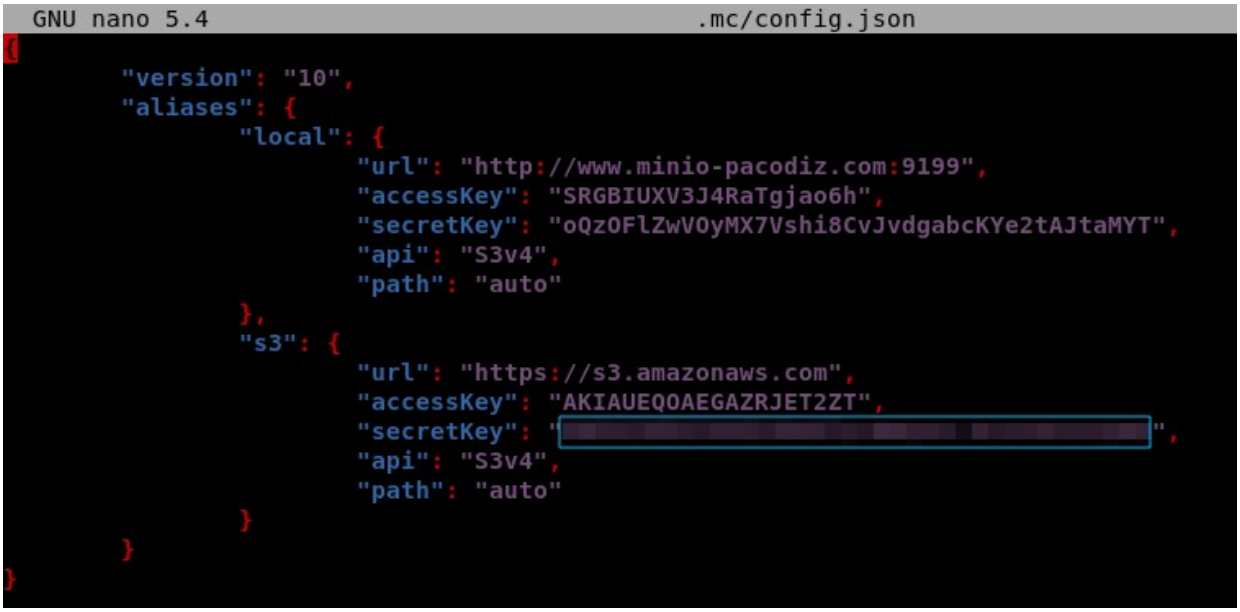
URL de los ficheros: https://github.com/Pacodiz02/PI/tree/main/demos/demo2

Reproducción de la demo:

Antes de nada, tendremos que tener disponible una cuenta de AWS con acceso a S3 y tener creada una clave de acceso. Luego tendremos que disponer de una máquina en internet(VPS) en la que tendremos que tener instalado un servidor web apache2 con la misma configuración que en la Demo1.

En primer lugar, tendremos que pasar los objetos de nuestro servidor minIO(preproducción) hasta S3(producción). Esto lo haremos configurando una nueva conexión a S3 ,tal y como vimos en la explicación anteriormente, en el cliente donde ya tenemos configurada la conexión al servidor de MinIO.

Configuraremos el cliente tal y como vimos en el apartado 6.7.2.Integración del cliente MinIO con S3. Usaremos las mismas credenciales que usamos en este ejemplo. Quedaría así:



Con esto terminado, tendremos que crear un nuevo bucket al que migraremos el contenido estático.

```
mc mb s3/web
```

Seguidamente migraremos los objetos de MinIO a S3:

```
mc cp -r local/web s3/web-pdu
```

```
paco@debian-paco:~$ mc ls s3/web-pdu/
[2023-05-31 12:05:20 CEST]  73KiB STANDARD coffee-shop-html-template.jpg
[2023-05-31 12:09:45 CEST]     0B css/
[2023-05-31 12:09:45 CEST]     0B img/
[2023-05-31 12:09:45 CEST]     0B js/
[2023-05-31 12:09:45 CEST]     0B lib/
[2023-05-31 12:09:45 CEST]     0B mail/
[2023-05-31 12:09:45 CEST]     0B scss/
```

Una vez tengamos esto comenzaremos a configurar nuestro servidor web en el VPS:

Creamos el DocumentRoot y movemos los ficheros .html modificados con la nueva dirección.

```
mkdir /var/www/web-pdu
```

```
cp -r *.html /var/www/web-pdu
```

Creamos un nuevo virtualhost con el siguiente contenido:

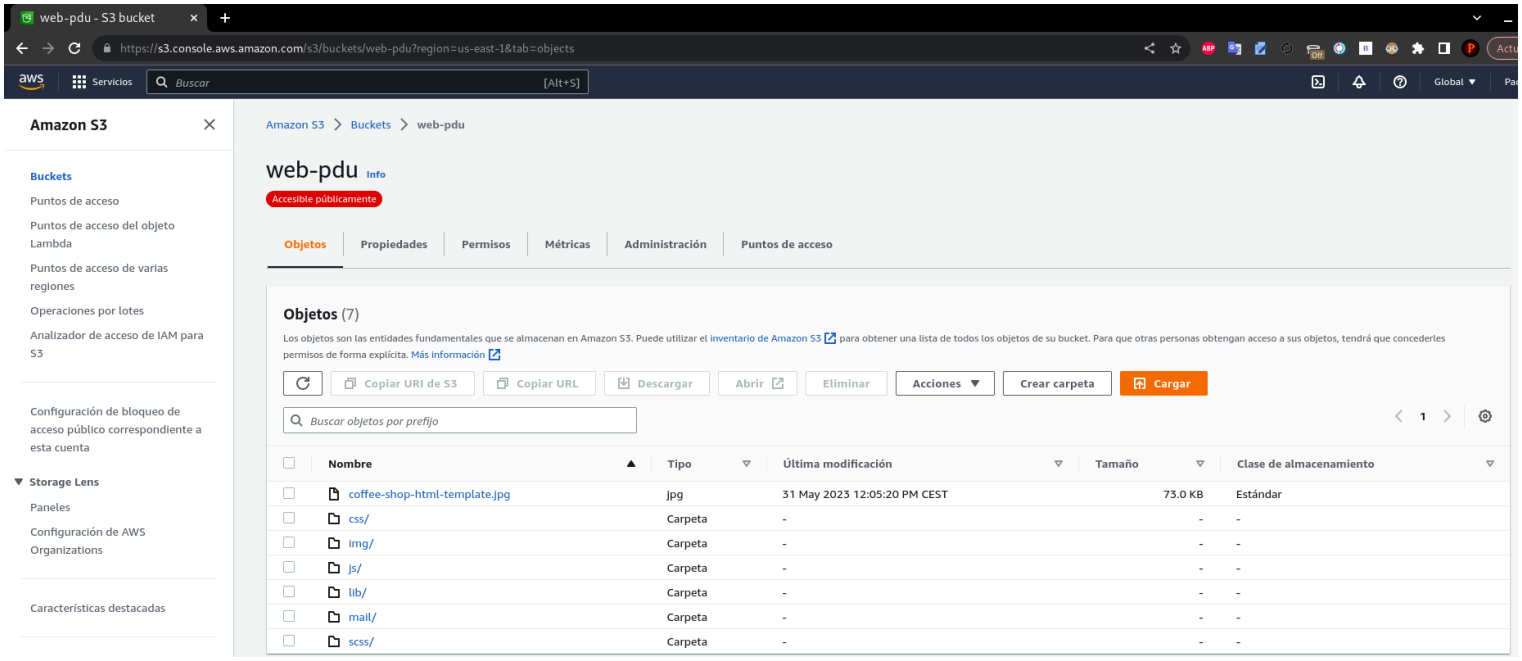
```
cat << EOF > /etc/apache2/sites-available/web-pdu.conf
<VirtualHost *:80>

        DocumentRoot /var/www/web-pdu
        ServerAdmin webmaster@localhost
        ServerName www.tutiendadecafe.com

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

 </VirtualHost>
EOF
```

Activamos el sitio y reiniciamos el servicio de apache2.

```
ln -s /etc/apache2/sites-available/web-pdu.conf
/etc/apache2/sites-enabled/

systemctl restart apache2
```

Comprobaciones:

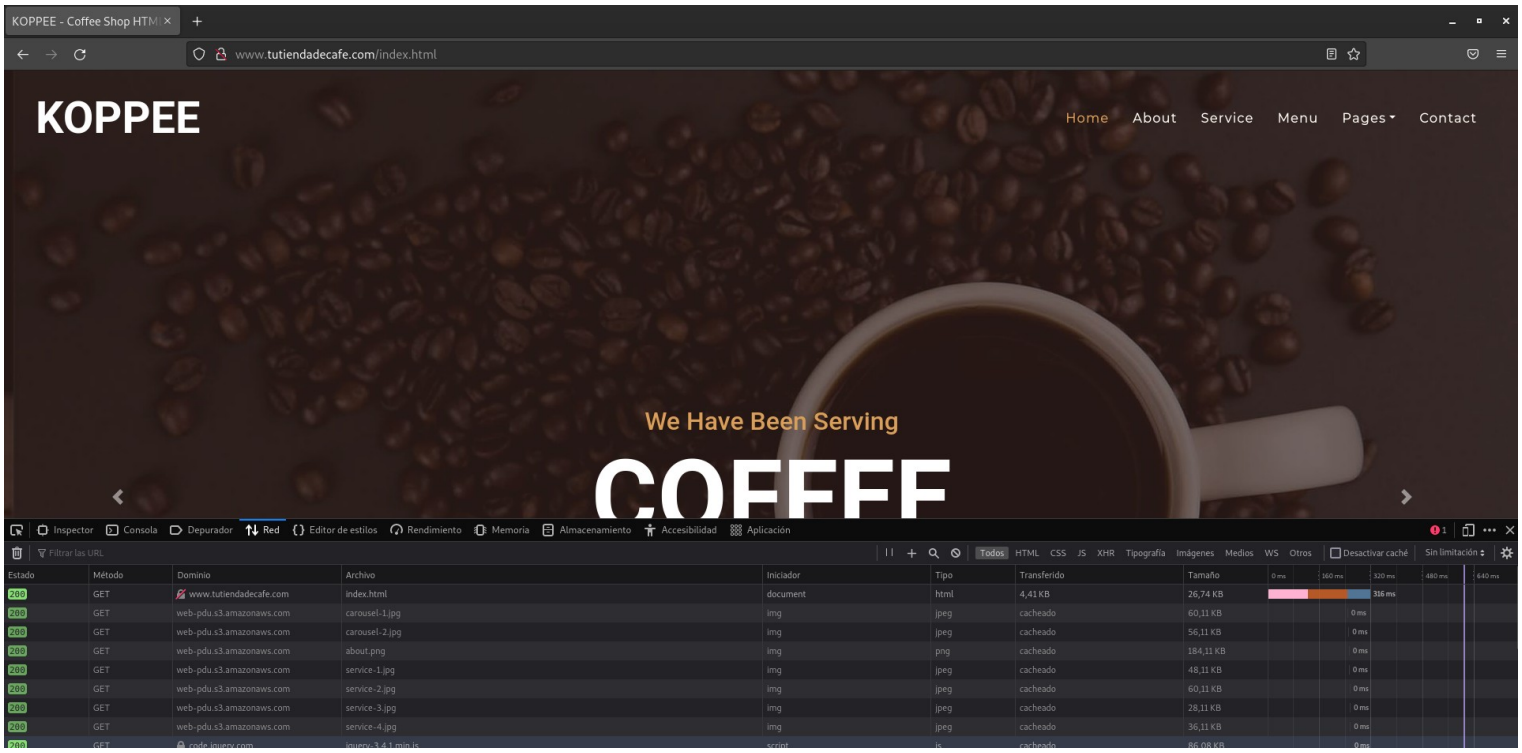
Desde la consola web de AWS S3 podemos ver como se ha creado correctamente el bucket y como contiene todos los archivos que migramos desde MinIO:



Para poder resolver para esta demo a través de internet sin tener que poner un nombre de dominio contratado lo pondremos localmente en /etc/hosts/

```
217.160.155.234      www.tutiendadecafe.com
```

Comprobamos que el contenido de la página web se ve correctamente:



7.Conclusión.

En conclusión, tanto MinIO como S3 son soluciones altamente confiables y eficientes para el almacenamiento en la nube. MinIO es ideal para aquellos que desean controlar su propia infraestructura de almacenamiento y obtener compatibilidad con S3, mientras que S3 es una opción sólida para aquellos que buscan una solución de almacenamiento en la nube administrada y altamente escalable. La elección entre ellos dependerá de los requisitos específicos de almacenamiento y las preferencias de cada organización.

8.URL Repositorio.

La URL del repositorio con los ficheros de configuración y archivos para el desarrollo del proyecto es: https://github.com/Pacodiz02/PI

9.Wiki.

Enlaces utilizados:
https://min.io/docs/minio/linux/index.html?ref=con
https://www.youtube.com/watch?v=FmwfKcjQKeA
https://howtoforge.es/como-instalar-y-configurar-un-servidor-de-almacenamiento-de-objetos-compatible-con-s3-utilizando-minio-en-ubuntu-20-04/
https://docs.aws.amazon.com/s3/index.html