MiaRec

Miarec Data-Backup-Guide

MiaRec, Inc.

Table of contents

1. MiaRec Data Backup Maintenance Guide

This guide describes the process of how to use utilities, Amazon Web Services and MiaRec native application features to backup the MiaRec database and audio files. It also covers the process of restoring a database in various scenarios in case of a catastrophic failure.

MiaRec stores data in two forms:

- **database** entries containing call metadata and configuration data; and
- **audio files** with raw audio recordings of calls.

It is possible to use existing utilities and native MiaRec features to dump all information from the MiaRec database and ensure redundancy of audio files. These actions can be completed on-demand or can be scheduled as reoccurring actions.

MiaRec recommends using an offsite storage facility, such as an **Amazon Web Services (AWS) S3 bucket** because it provides exceptional 99.999999999% durability and 99.99% availability and supports WORM (write-once-read-many) model to prevent corruption or tampering of backup files. Other offsite storage mechanisms exist, like SFTP, NFS, but they are not covered in these instructions.

1.1 Database

MiaRec uses a PostgreSQL Database to store call metadata and configuration data. It is possible to use the utilities `pg_dump` and `pg_restore`, which are included in the `postgresql` package, to backup and restore the MiaRec database.

1.1.1 What is in the database?

The database holds the following data:

- Tenant, Group and User configuration
- Call Recording Details/State
- System Configuration
- Branding information
- Job Configuration (export, replication, relocation)
- Other data

1.1.2 What is NOT in the database?

The database does not include the following:

- **Audio Files** Audio files are stored on the file system rather than in a database. The database only stored a path to the files. Therefore audio files should be backed up and restored using other means. As long as the file path entries in the database are still valid, after a database restoration, the audio files will be accessible.

1.2 Audio Files

Audio files can be stored:

- locally on the same server that recorded the calls
- externally on remote storage like FTP, SFTP, S3. In such a case, a background process automatically relocates audio files from local file storage to remote storage. A file relocation job is normally run by a schedule (every 5 minutes or so).

1.2.1 External storage

When external storage is used for audio files, such storage normally provides built-in redundancy, whether it is a NAS server with multiple disks in a RAID array or an Amazon S3 bucket with 99.999999999% durability. In most situations, the provided redundancy of the external storage is sufficient and we must focus on MiaRec database backup/restore only. During a database restoration, the configuration for this external storage target would be retained and audio files would be accessible at the same path.

> **Info**
>
> Make sure to check for IP filtering rules on your external storage device, in the event of a disaster recovery the IP address of the MiaRec server may change.

MiaRec provides built-in support of external storage support. This is achieved by using a file relocation job that runs periodically and moves audio files from the local recording server to external storage like FTP, SFTP, FTPS or Amazon S3.

Note, that there is a short moment when files are kept locally on the recording server(s) before they are relocated to the external storage. If a disk failure occurs, such non-relocated audio files may be lost. We recommend using a RAID 1 disk array for physical servers, which would provide local redundancy to those local files in case of disk failures.

External storage is almost always a preferred solution except for the cases where the reliability of the network connection to the external storage is in question.

1.2.2 Local storage

Using local storage on recording server instances is not considered fully redundant (although some form of redundancy can be achieved by using RAID 1 disk arrays). If there is a failure in one or all of the recorder instance(s), audio files would be lost and unrecoverable.

Redundancy can be achieved with local storage using the following methods:

- MiaRec built-in replication mechanism that synchronizes audio files between two MiaRec clusters. In such a case, audio files exist in two copies on two servers, which can be geographically distributed.
- Third-party file synchronization utilities like `rsync`, periodically create copies of the local audio files on external storage.

In both of the mentioned methods, each audio file is duplicated, i.e. one copy is stored in the local storage and another copy is stored in remote storage or 2nd cluster.

Local storage is accessible reliably and with low latency, so it is ideal in scenarios with unreliable networks.

2. Requirements

- Console/ssh root access to the server that is running the PostgreSQL database service.
- Admin access to the MiaRec web portal.
- Amazon AWS IAM Account with privileges to create IAM users, IAM policies and S3 buckets.

3. Backup Database

3.1 Backup database

This section provide instructions how to create a dump of the MiaRec database and copy it to S3 backet for long-term storage using both manual and automated ways.

3.2 Prerequisites

To support the offload of MiaRec backups, an S3 bucket must be provisioned.

Database backups should be stored using the WORM (*write-once-read-many*) model to prevent corruption or tampering. To support WORM storage, **S3 Object Lock** has to be enabled, this is defined at the bucket level, requiring separate buckets.

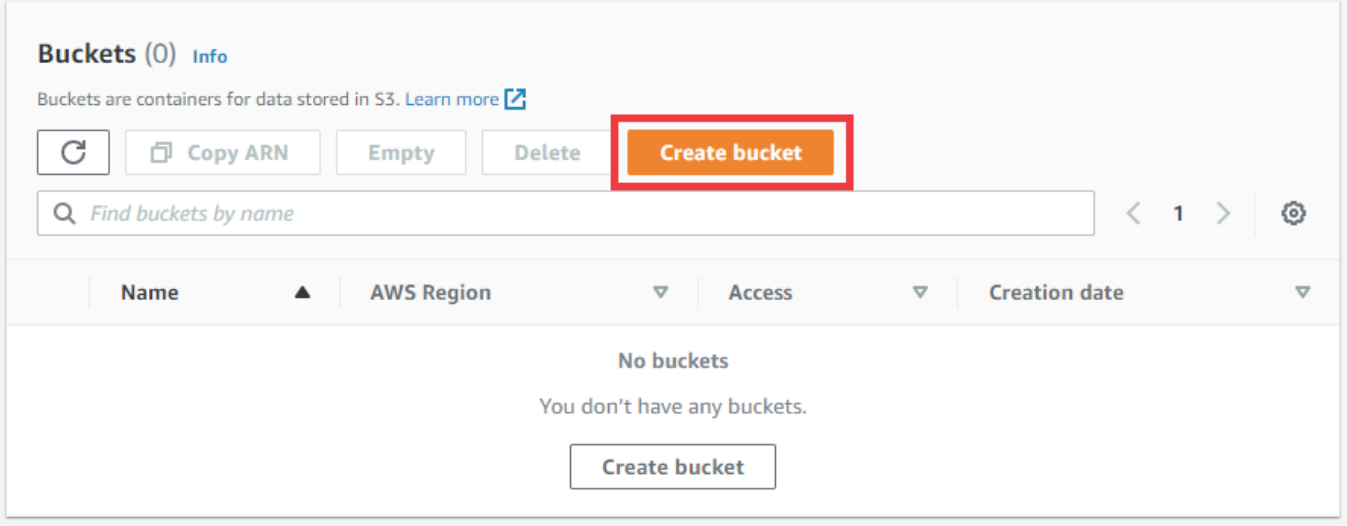
More information about S3 Object Lock can be found at the following link. https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

3.2.1 Create a bucket

An S3 bucket with the Object lock must be created as a destination target for database backups.

From Amazon S3 console at https://console.aws.amazon.com/s3/.

1. Choose **Create bucket** from the console top menu to create a new Amazon S3 bucket.

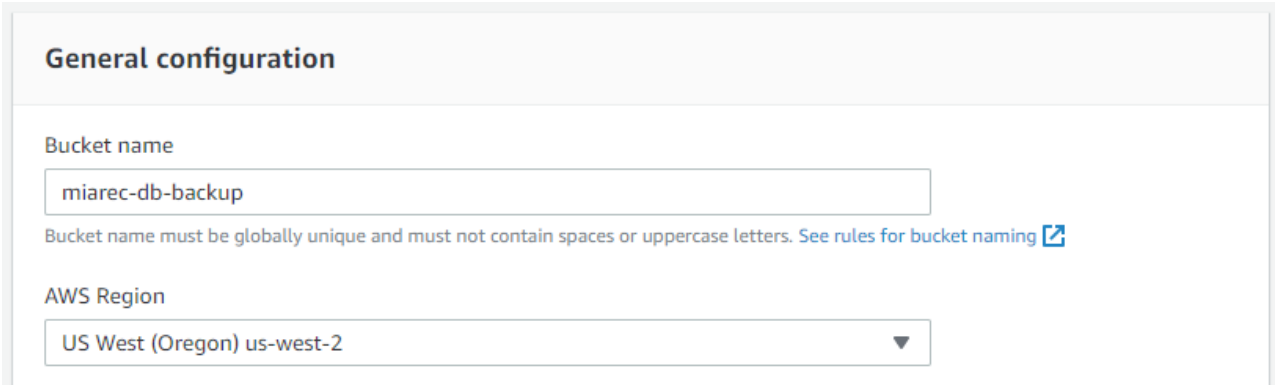


2. On the *Create bucket setup* page, perform the following actions:

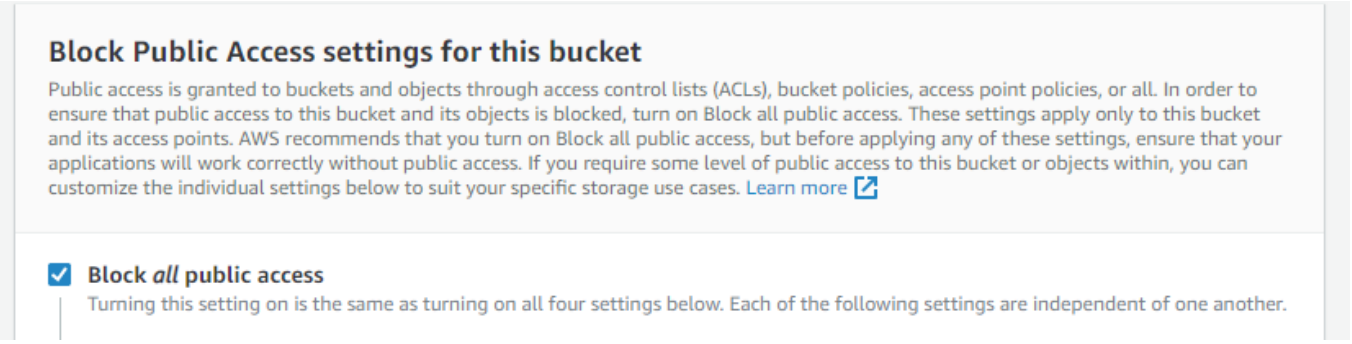
   a. For *General configuration:*

      i. Provide a unique name for the new bucket in the **Bucket name** box.

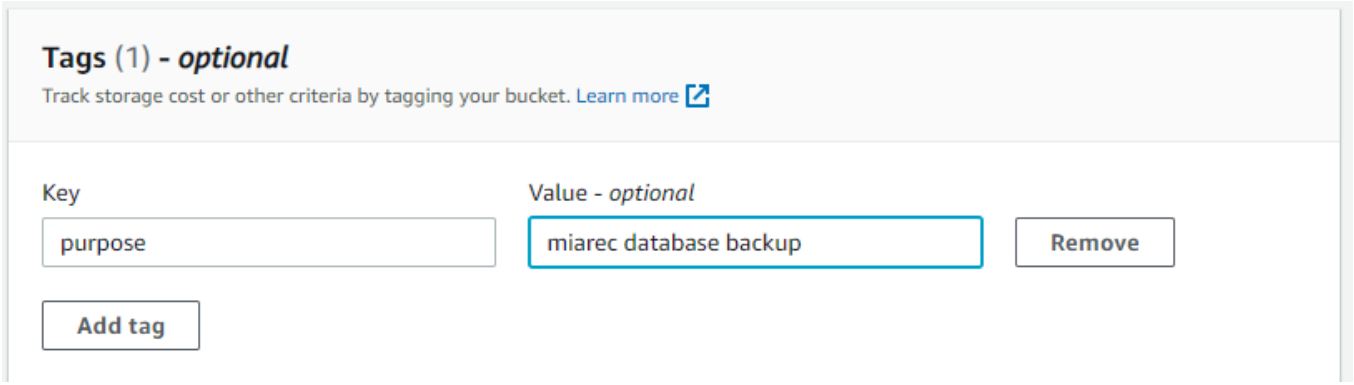
      ii. From the **AWS Region** dropdown list, select the AWS cloud region where the new S3 bucket will be created.



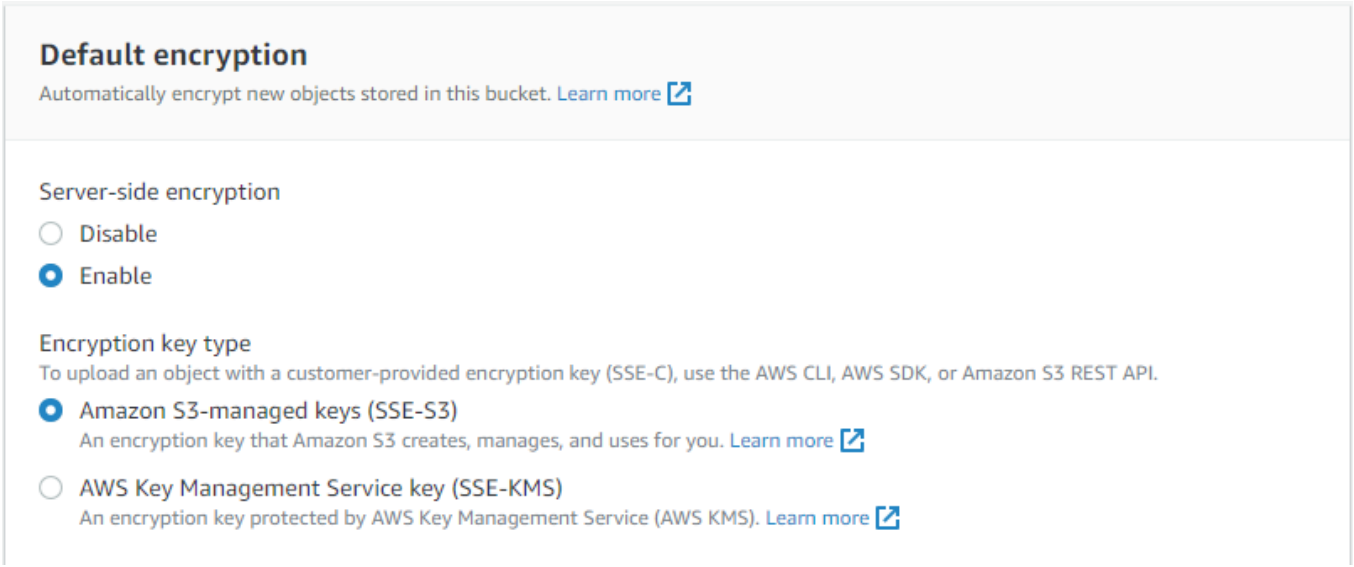
3. For *Block Public Access settings for this bucket*, select **Block all public access** to ensure that all public access to this bucket and its objects is blocked.



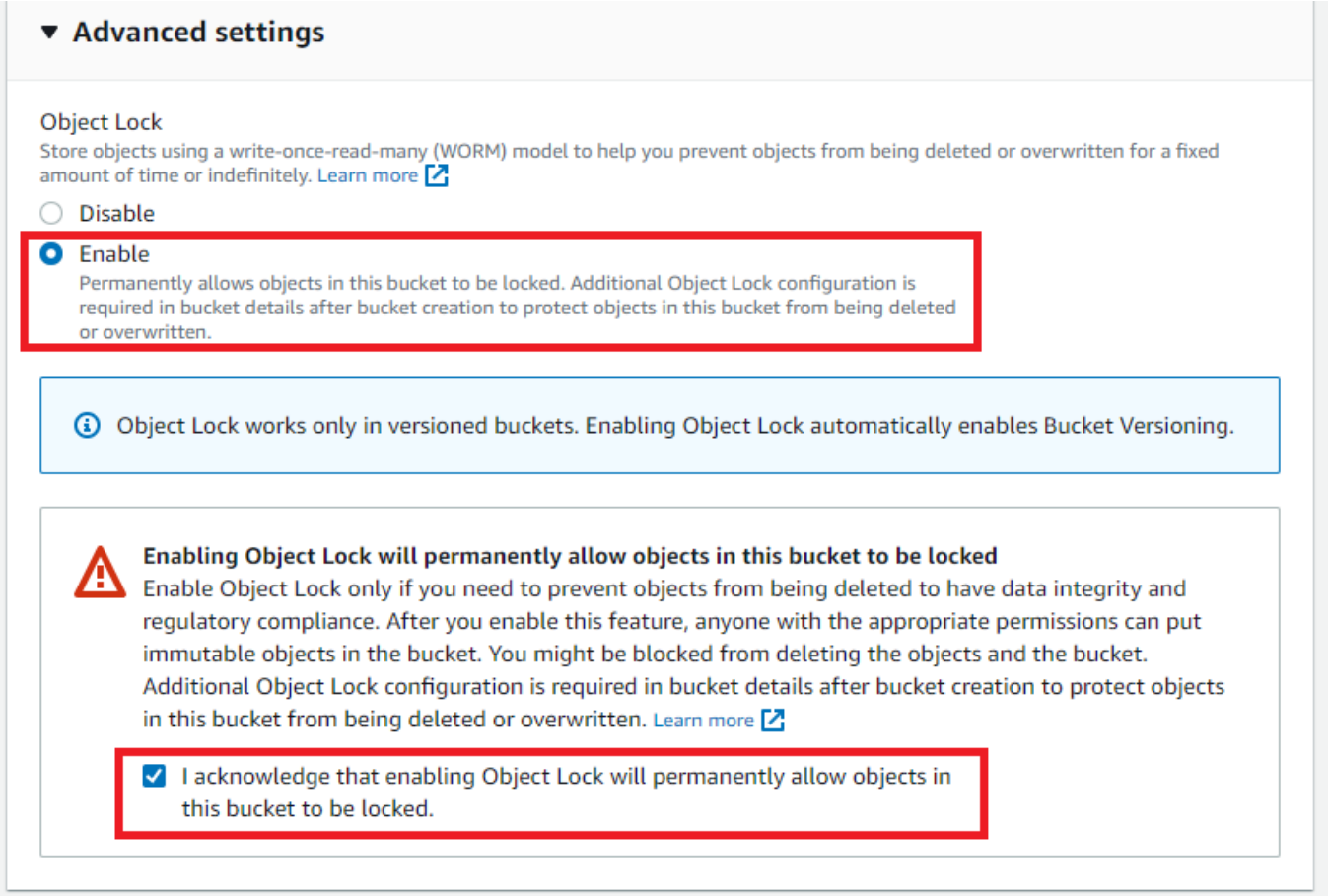
4. (Optional) *Tags*, use the **Add tag** button to create and apply user-defined tags to the S3 bucket. You can track storage costs and other criteria by tagging your bucket.



5. (Recommended) For *Default encryption*, select **Enable** under *Server-side encryption,* and choose *Amazon S3-managed keys (SSE-S3)*.



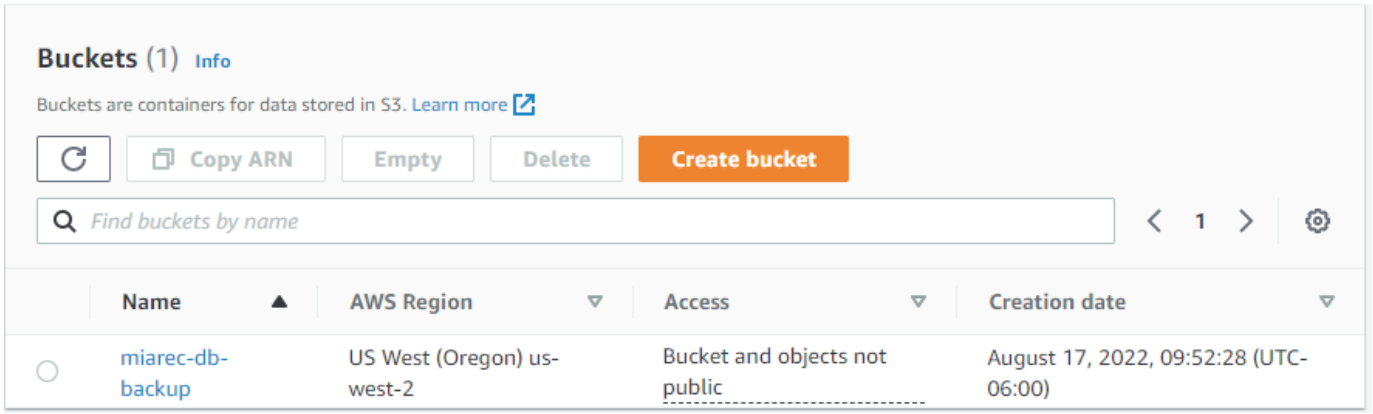
6. For *Advanced settings*, choose **Enable** under *Object Lock* to enable the Object Lock feature. Choose *I acknowledge that enabling Object Lock will permanently allow objects in this bucket to be locked* for confirmation.



7. Choose **Create bucket** to create your new Amazon S3 bucket.

Result

S3 bucket will be created and displayed in the console.

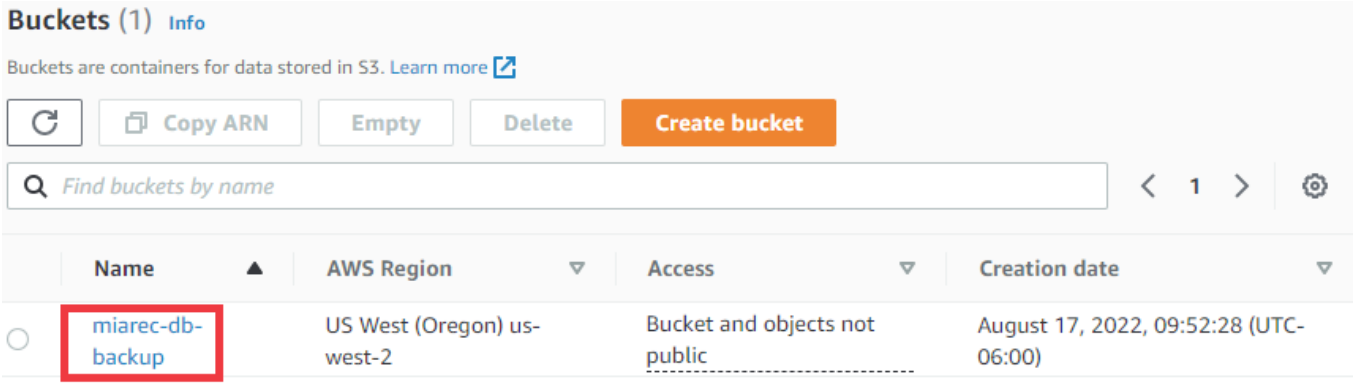


3.2.2 Set Object Lock Retention policy

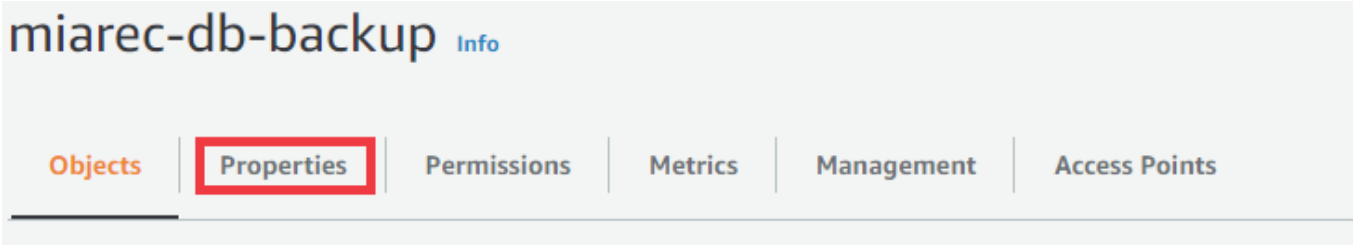
Object Lock Retention policy needs to be configured to ensure items can be eventually removed and guarantees data integrity for a defined period.

From Amazon S3 console at https://console.aws.amazon.com/s3/.

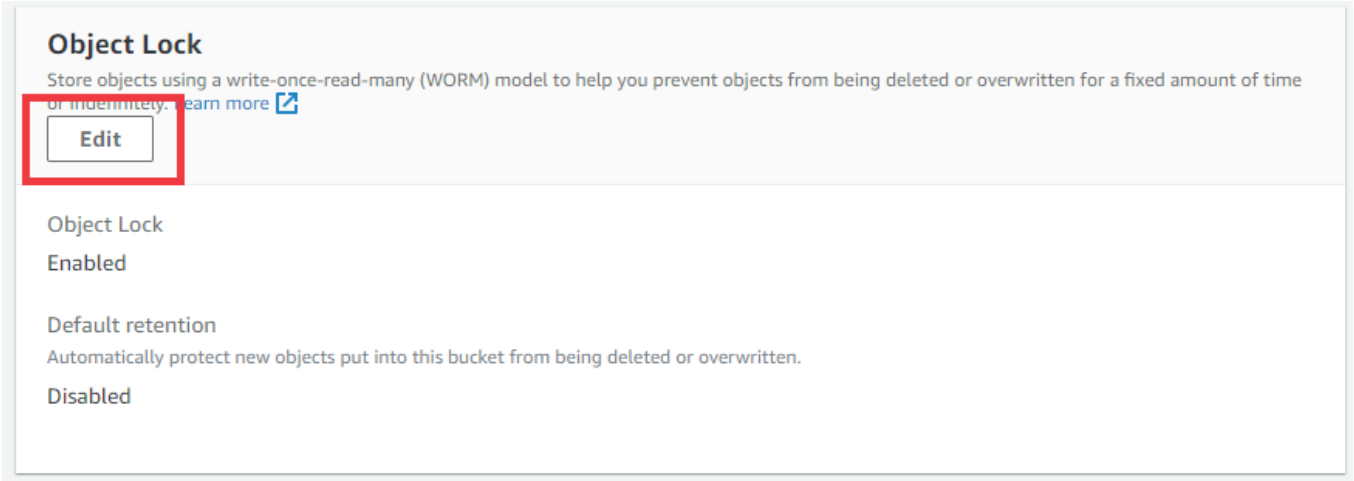
1. Click on the name of the newly created Amazon S3 bucket.



2. Select the **Properties** tab from the console menu to access the bucket properties.



3. In the *Object Lock* section, choose **Edit** to configure the feature settings available for the S3 objects that are uploaded without Object Lock configuration.



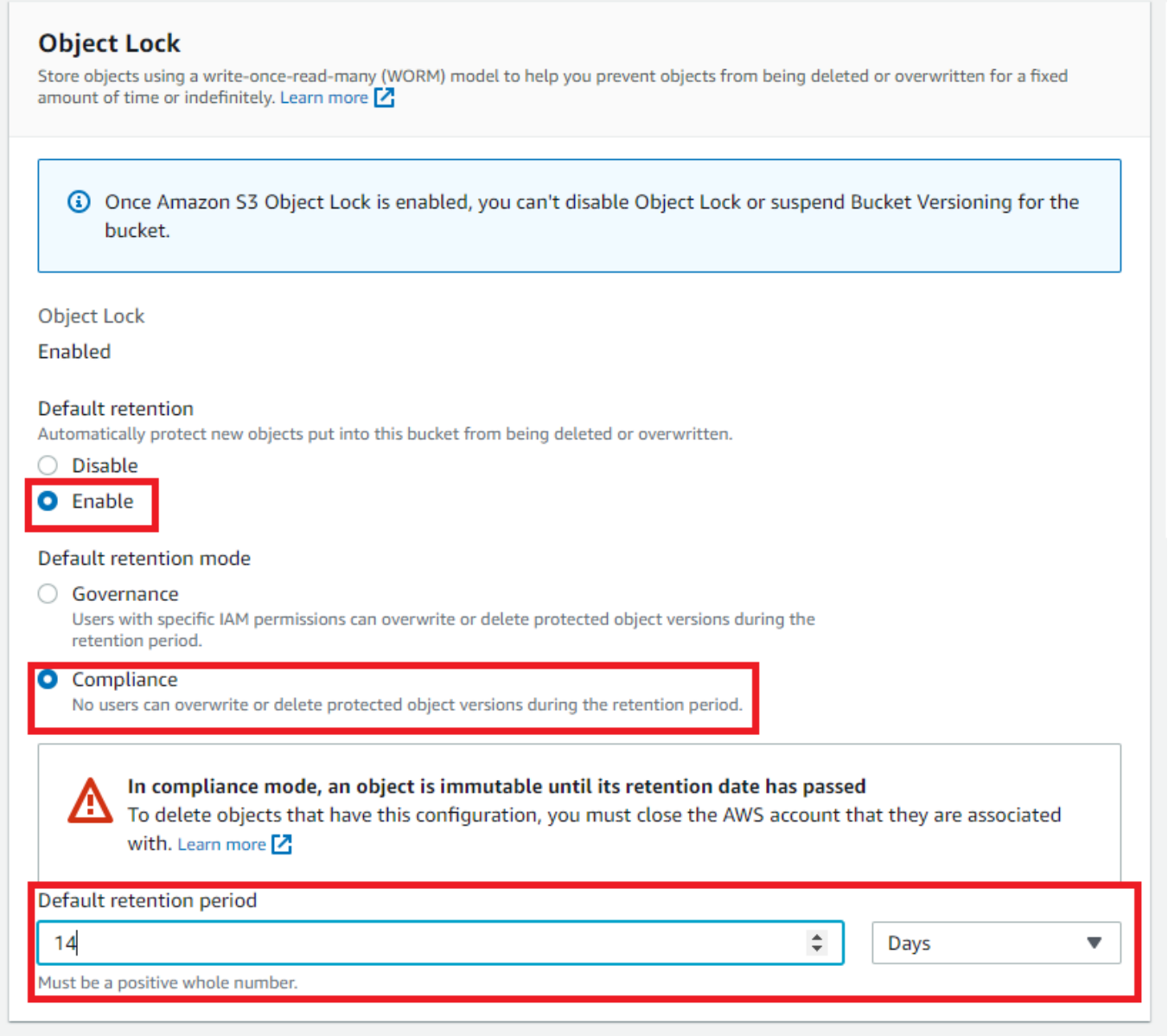
4. Within the *Object Lock* configuration section.

   a. Choose **Enable** under *Default retention*.

   b. Select **Compliance** so that a protected object version cannot be overwritten or deleted by any user, including the root account user. Once an S3 object is locked in Compliance mode, its retention mode cannot be reconfigured and its retention period cannot be shortened. This retention mode ensures that an object version can't be overwritten or deleted for the duration of the retention period

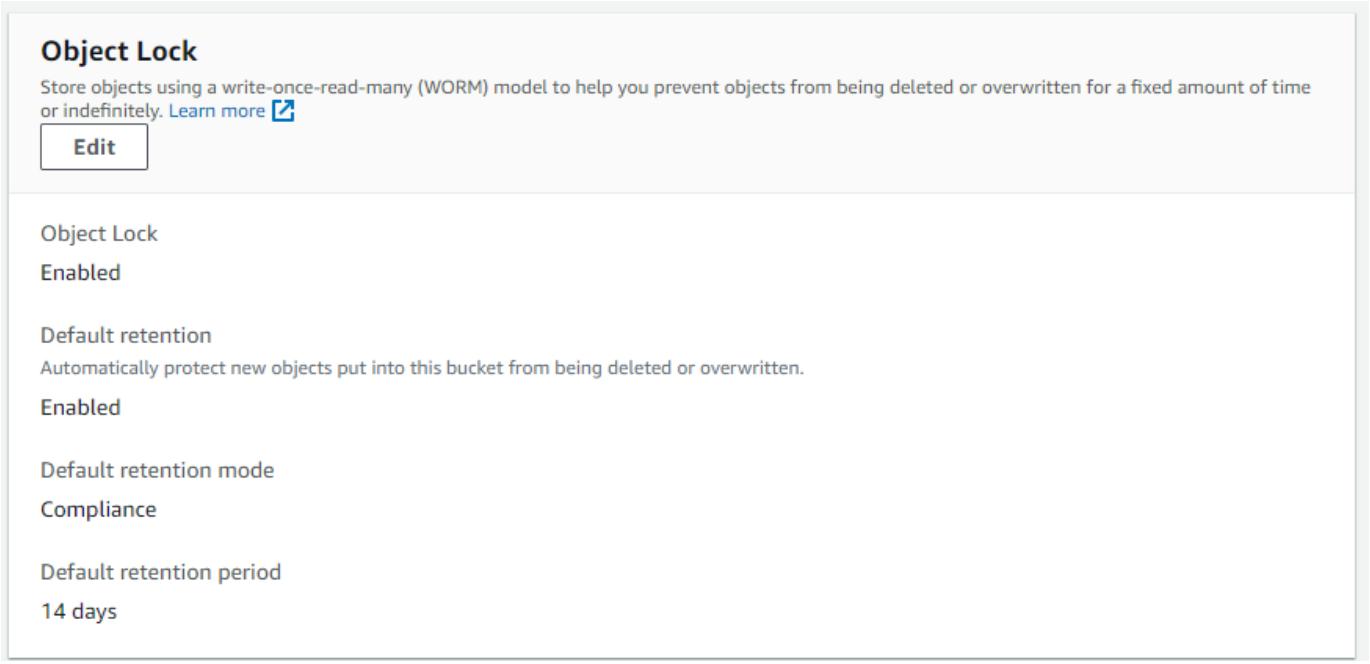
   c. Define the **Default retention period**.

   d. Click **Save changes** to apply the configuration changes.



Result

Object Lock Configuration will be modified.

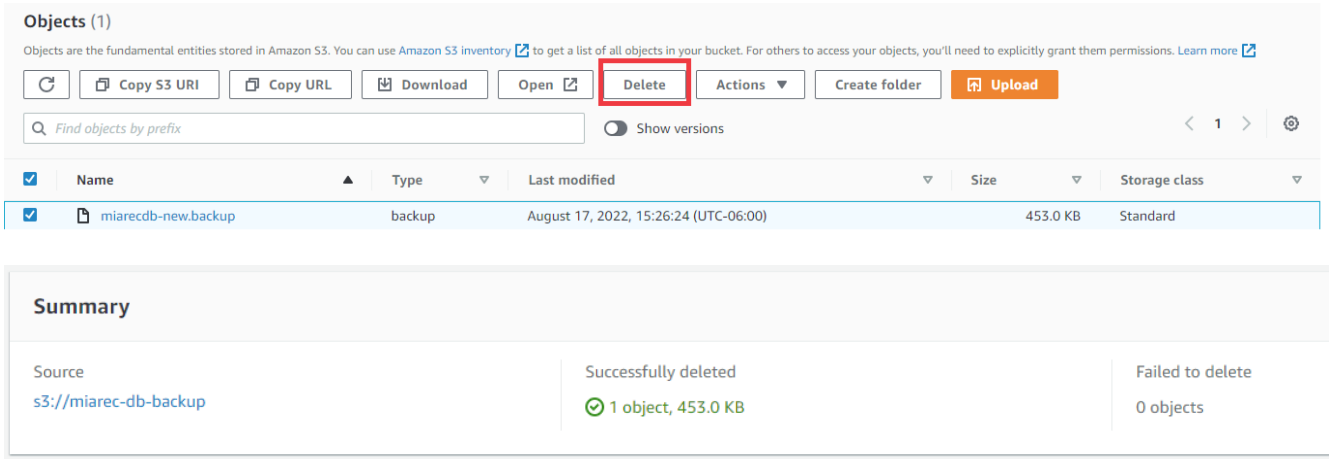


Note, that the Object Log Retention configuration doesn't delete files after the specified retention period. It just prevents the deletion of the files during the specified retention period. To delete the old backup files, you must use Amazon S3 lifecycle policies.

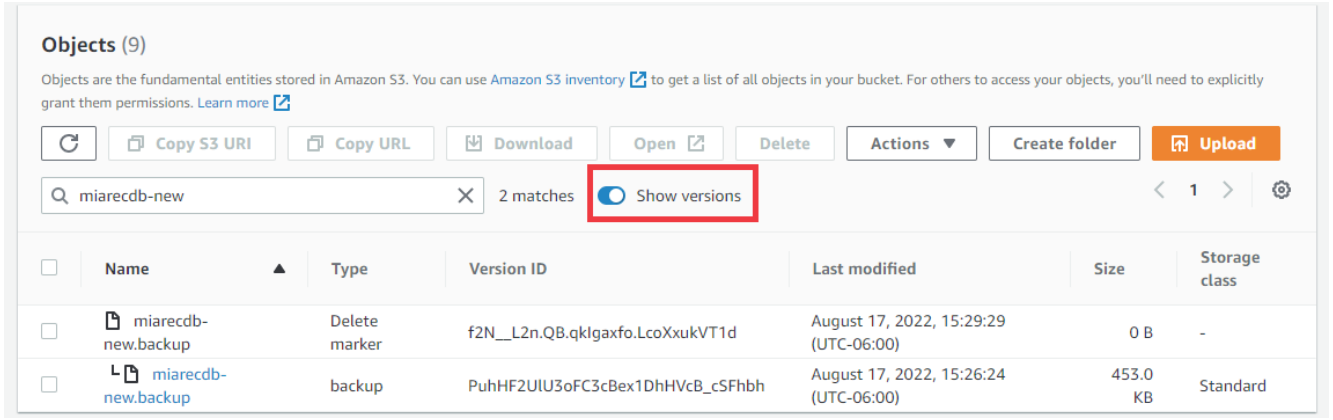
How to delete files in S3 bucket when Object lock is present?

It is possible to "delete" an object that is currently in Object lock in the same manner you normally delete a file.

Select the object, and click **Delete**, then confirm the action.



This will display a *successful delete*, however, this is misleading, by toggling the **Show Version**, you can see what happened is that a *Delete marker* was applied, and the previous version is still available for download.



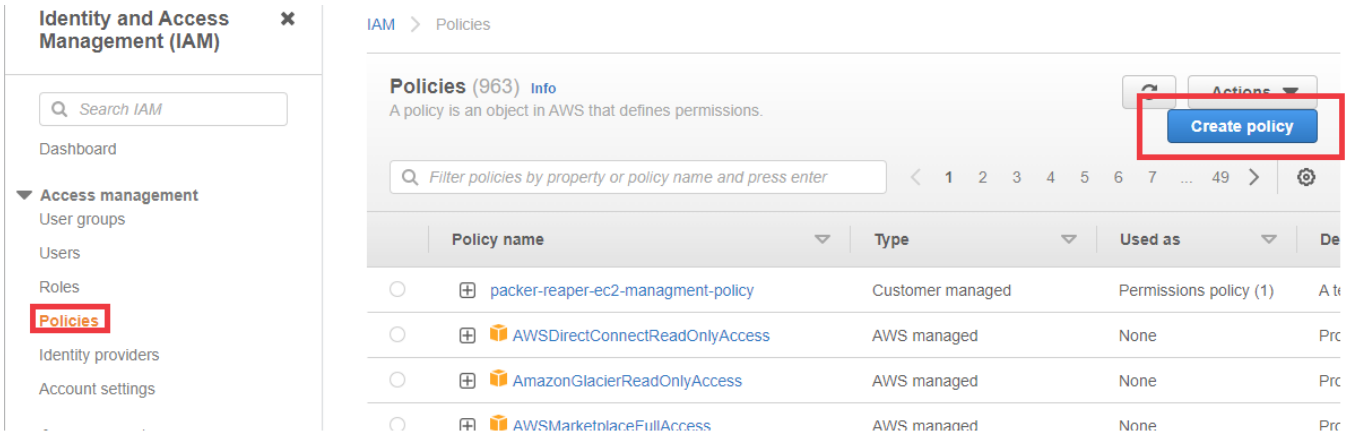
However, this is temporary, after the object lock retention period expires the object will be deleted.

3.2.3 Create IAM policy for access to the database backup bucket

An IAM Policy has to be created and assigned to an IAM user so that objects can be added to the S3 bucket by that IAM user.

From Amazon IAM console at https://console.aws.amazon.com/iam/

1. From the **Policies** menu. Choose *Create Policy* to create a new IAM Policy.

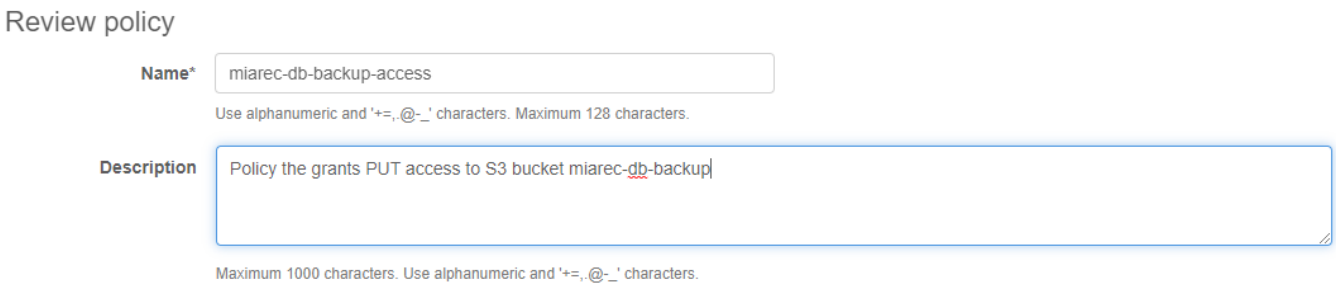


2. Select *JSON* tab, copy the following access policy and paste it into the **JSON** field. **Do not forget to replace** miarec-db-backup **with your bucket name!!!**.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-db-backup\"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-db-backup/*"
        ]
    }
]
}
```

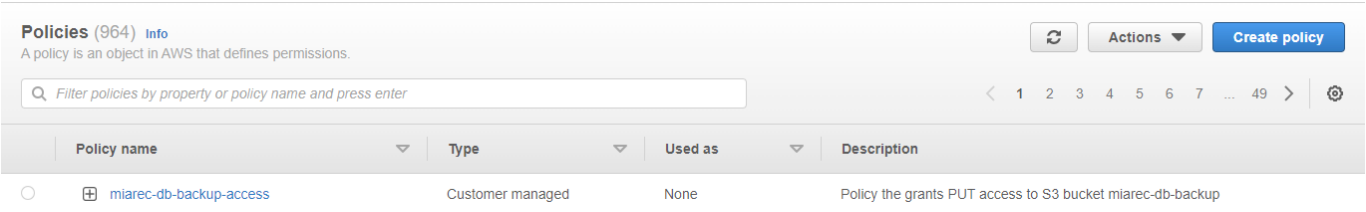
(Optional) *Tags*, use the Add tag button to create and apply user-defined tags to the resource.

Review policy, choose a descriptive name and description for the policy and click **Create policy** button.



Result

The policy will be created and ready to be assigned.

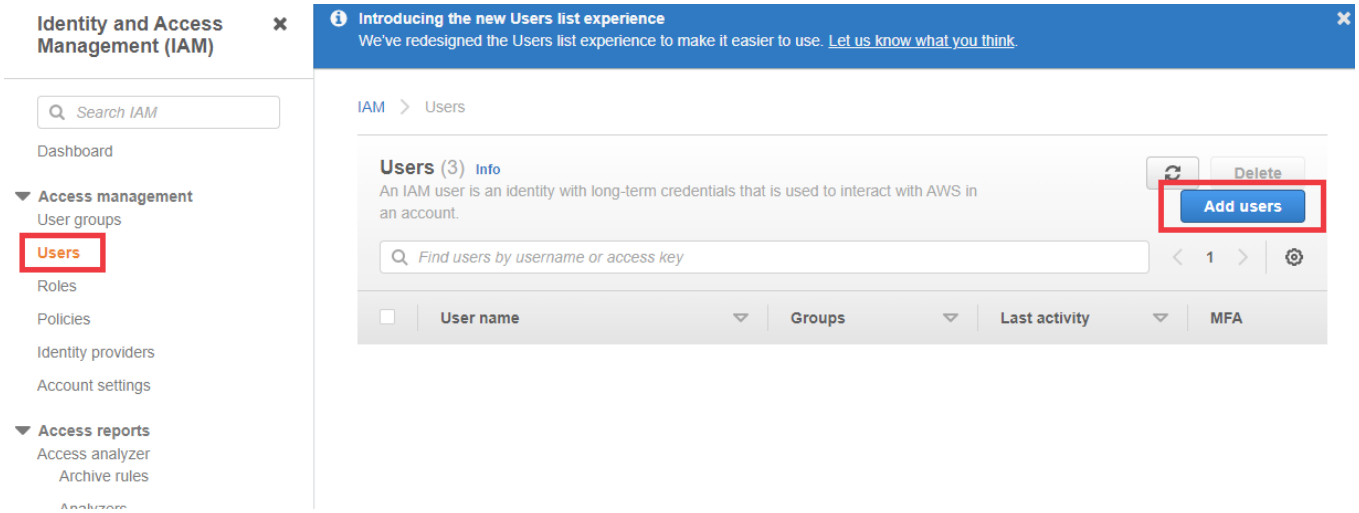


3.2.4 Create IAM User for database backup bucket

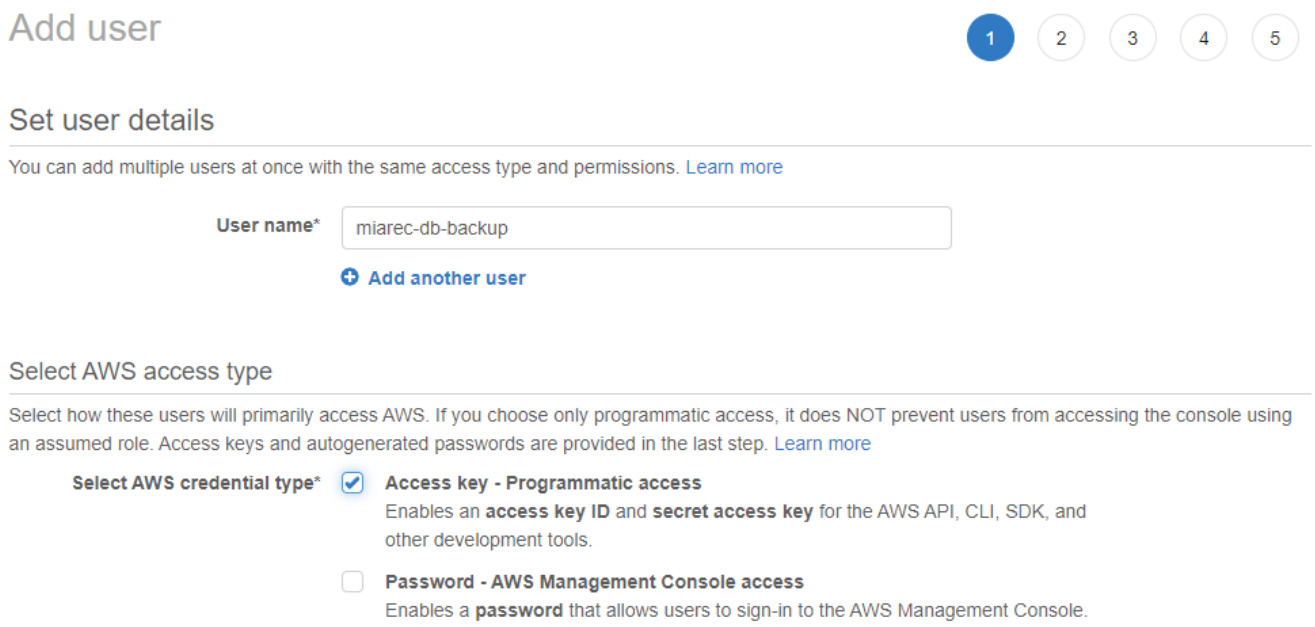
IAM user has to be created that can be used later to push database backups to the S3 bucket.

From Amazon IAM console at https://console.aws.amazon.com/iam/

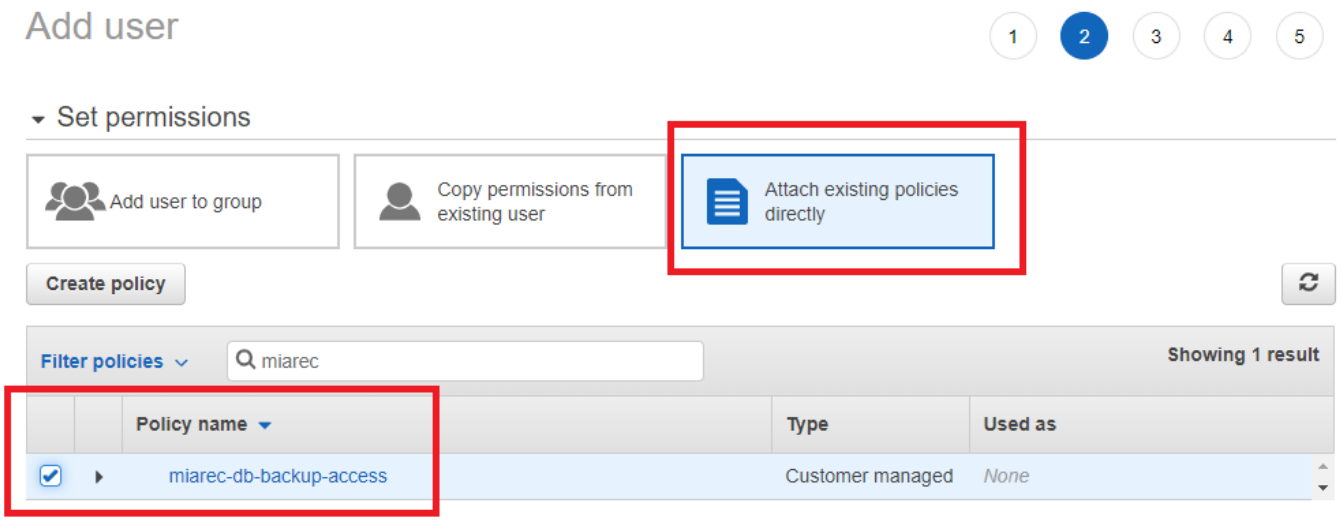
1. From the *Users* menu, choose **Add User** to create a new IAM User.



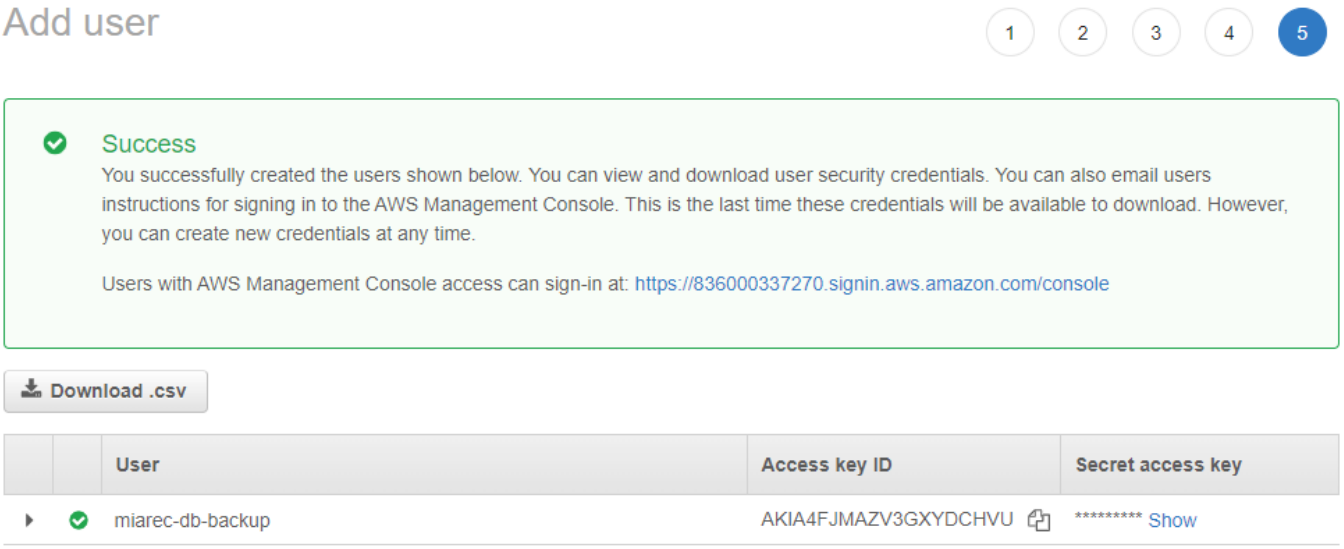
2. *Details*, choose **User name** and enable **Programmatic access**.



3. *Permissions*, select **Attach existing policies directly** and then select the previously created policy from the list. *Use the search box to find the policy by name.*

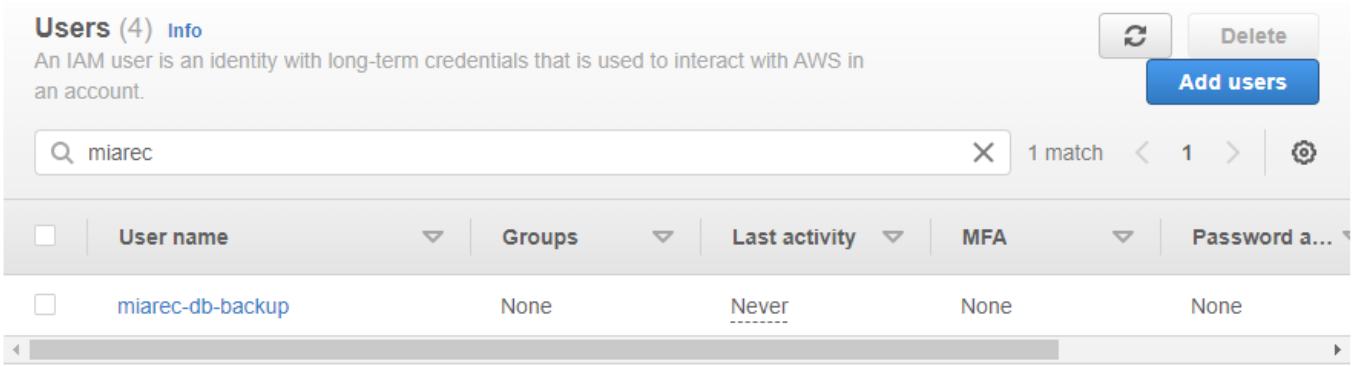


4. (Optional) *Tags*, use the Add tag button to create and apply user-defined tags to the resource.
5. *Review*, confirm the settings and click **Create user**.
6. On the *Complete* screen, copy **Access Key ID** and **Secret access key** and store them in a secure place. This will be used later to push database backup to S3.



Result

The user will be added, access key and the secret access key will be available to use to access the S3 bucket.



3.2.5 Install AWS CLI

`awscli` package is required to transfer database backups to an S3 bucket.

Install `unzip`

```
sudo yum install -y unzip
```

Install `aws-cli`

For the latest version of the AWS CLI, use the following command block:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Verification

```
[centos@miarecdb ~]$ aws --version
aws-cli/2.7.25 Python/3.9.11 Linux/3.10.0-1160.el7.x86_64 exe/x86_64.centos.7 prompt/off
```

3.3 Manual backup of database

3.3.1 Backup database using pg_dump utility

A backup of the MiaRec database can be executed on demand using the `pg_dump` utility. This is a good idea to execute on initial deployment, to verify operation. The `pg_dump` utility is installed as part of the `postgresql` package, so it does not need to be installed.

Execute the pg_dump utility:

```
sudo -iu postgres pg_dump -F c -f /tmp/miarecdb-$(date "+%Y.%m.%d-%H.%M.%S").backup miarecdb
```

Let's break down those options:

- `sudo -iu postgres` instructs the shell to execute the trailing command on behalf of user `postgres`, this is required for authentication to the database.
- `pg_dump` calls the `pg_dump` utility.
- `-F c` Sets an output file format, in this case, a custom archive suitable for input into `pg_restore`. This is the most flexible format in that it allows reordering of loading data as well as object definitions. This format is also compressed by default.
- `-f /path/to/outputfile` Send output to the specified path, this directory will need to be accessed by the `postgres` user, a directory like `/tmp` would be a suitable destination.
- `$(date "+%Y.%m.%d-%H.%M.%S")` will be replaced with the current timestamp, like *2022.08.02-12.13.14*.
- `miarecdb` the target database for dump, this should always be `miarecdb`.

Verification

An archive will be produced at the specified path

```
[centos@miarecdb ~]$ ls -l /tmp
-rw-rw-r--. 1 postgres postgres 1102336 Aug  2 16:35 miarecdb-2022.08.02-12.13.14.backup
[centos@miarecdb ~]$
```

3.3.2 Copy database backup files to S3

Set AWS credentials

Use the previously created user credentials for the the dabase backup bucket. Using environmental variables, these values will be set for this session, but not be retained after the session is ended.

```
export AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
export AWS_SECRET_ACCESS_KEY=<AWS_AC_KEY_ID>
```

Execute Copy to S3

```
aws s3 cp /path/to/source s3://{s3-bucket-name}
```

Let's break down those options:

- `aws s3 cp` calls the *aws cli utility* and instructs it to *copy* a file to *s3.*
- `/path/to/source` path and file name of source database dump file.
- `s3://{s3-bucket-name}` Sets a destination of s3 transfer, this should be the bucket created in an earlier step.

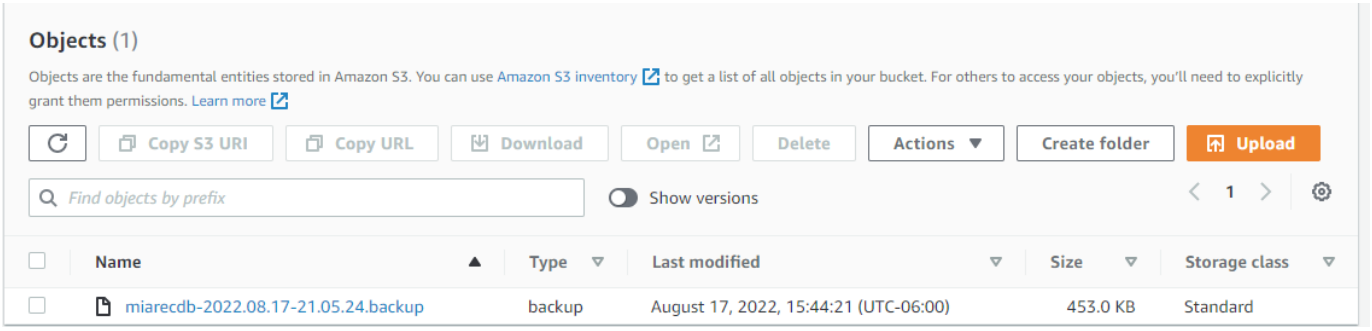
Example. Copy a file from the /tmp directory to S3 bucket `miarec-db-backup`:

```
aws s3 cp /tmp/miarecdb-2022.08.17-21.05.24.backup s3://miarec-db-backup
```

Result

The object will be moved to the defined bucket

```
[centos@miarecdb ~]$ aws s3 cp /tmp/miarecdb-2022.08.25-19.47.14.backup s3://miarec-db-backup
upload: ../../tmp/miarecdb-2022.08.25-19.47.14.backup to s3://miarec-db-backup/miarecdb-2022.08.25-19.47.14.backup
[centos@miarecdb ~]$
```



3.4 Automated backup of database

In this section, we provide instructions how to run `pg_dump` utility and transfer the database dump file to S3 bucket automatically by schedule.

3.4.1 Create a user account backup with specific privileges

Since this is an automated script, the best practice is to create a user with just the permissions needed that can be used to execute the script.

Add user backup

```
sudo useradd backup
```

Give backup privileges to run `pg_dump` as user `postgres`

Modify the `sudoers` file

```
sudo visudo
```

Add the following at the bottom of the file:

```
backup ALL=(postgres) NOPASSWD:/usr/bin/bash,/usr/bin/pg_dump
```

Verification

You should be able to assume user backup, verify the `aws` version and execute `pg_dump` as user `postgres`. Any other commands will prompt for a password and fail to execute with `Permission denied`.

```
[centos@miarecdb ~]$ sudo -iu backup
...
[backup@miarecdb ~]$ /usr/local/bin/aws --version
aws-cli/2.7.25 Python/3.9.11 Linux/3.10.0-1160.el7.x86_64 exe/x86_64.centos.7 prompt/off
...
[backup@miarecdb ~]$ sudo -Hiu postgres pg_dump --version
pg_dump (PostgreSQL) 12.12
```

3.4.2 Create Bash Script

Create secret file

This file will only be accessible by the backup user and super users, it will contain credentials generated in the above step.

```
sudo -u backup vi /home/backup/.backup_secret
```

Insert the following, and be sure to change the information for your deployment.

```
FILEPREFIX=<backup_prefix>
BUCKETNAME=<S3_BUCKET_NAME>
AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
```

Where:

- `FILEPREFIX`, file name prefix that will be used to name all backup files in AWS S3 storage, this should be unique to each instance, suggestion is to include `\$HOSTNAME` var
- `BUCKETNAME`, name of s3 bucket where database backup will be stored
- `AWS_ACCESS_KEY_ID`, AWS Secret Key ID generated earlier
- `AWS_SECRET_ACCESS_KEY`, AWS Secret Access Key generated earlier

Verify

```
[centos@miarecdb ~]$ sudo -u backup cat /home/backup/.backup_secret
FILEPREFIX=miarecdb-$HOSTNAME
BUCKETNAME=miarec-db-backup
AWS_ACCESS_KEY_ID=....
AWS_SECRET_ACCESS_KEY=....
[centos@miarecdb ~]$
```

Create a backup script

```
sudo vi /usr/local/bin/miarec_backup.sh
```

Insert the following:

```
#!/bin/bash

# Read Variables from secret file
set -o allexport
source ~/.backup_secret
set +o allexport

BACKUPDIR=/tmp
TMPFILE=miarecdb.backup
DATE=$(date "+%Y.%m.%d-%H.%M.%S")

echoerr() { echo "$@" 1>&2; }

# Generate DB dump
backup_db (){
  echo "Dumping database to $BACKDIR/$TMPFILE"
  sudo -Hiu postgres pg_dump -F c -f $BACKDIR/$TMPFILE miarecdb
  if [ $? -eq 0 ]
  then
    echo "The database dump was successful!"
  else
    echoerr "There was a problem with the database dump, stopping"
    exit 1
  fi
}

relocate_s3 (){
  echo "Moving files to S3"
  /usr/local/bin/aws s3 cp $BACKUPDIR/$TMPFILE s3://$BUCKETNAME/$FILEPREFIX-$DATE.backup
  if [ $? -eq 0 ]
  then
    echo "Backup was successfully transferred to AWS S3 $BACKDIR/$TMPFILE"
  else
    echoerr "There was a problem with the transfer to S3, stopping"
    exit 1
  fi
}

backup_db
relocate_s3

echo "Completed in ${SECONDS}s"
```

Make the script executable

```
sudo chmod u+x /usr/local/bin/miarec_backup.sh
```

Change ownership to backup user

```
sudo chown backup:backup /usr/local/bin/miarec_backup.sh
```

Result

```
[centos@miarecdb ~]$ ls -l /usr/local/bin/
total 652
...
-rwxr--r--. 1 backup backup    902 Aug 24 17:32 miarec_backup.sh
```

Verify

Manually call script on behalf of user `backup`

```
[centos@miarecdb ~]$ sudo -iu backup /usr/local/bin/miarec_backup.sh
Dumping Database
The database dump was successful
Moving files to S3
upload: ../../tmp/miarecdb.backup to s3://miarec-db-backup/miarecdb-miarecdb.example.com-2022.08.24-17.36.40.backup
```

```
Backup was transferred to AWS S3
Completed in 2s
```

3.4.3 Create a `crontab` job to execute the backup script

```
sudo crontab -u backup -e
```

An editor will be started (vi by default. The file being edited will have one job per line. Empty lines are allowed, and comments start their line with a hash symbol (#).)

Insert the following

```
0 1 * * * /usr/local/bin/miarec_backup.sh
```

Let's break down those options:

- `0 1 * * *` cron expression determines when the job will run, which is 1:00am every day. Help with creating Cron expressions can be found here https://crontab.guru/
- `/usr/local/bin/miarec_backup.sh` location of script

Verification

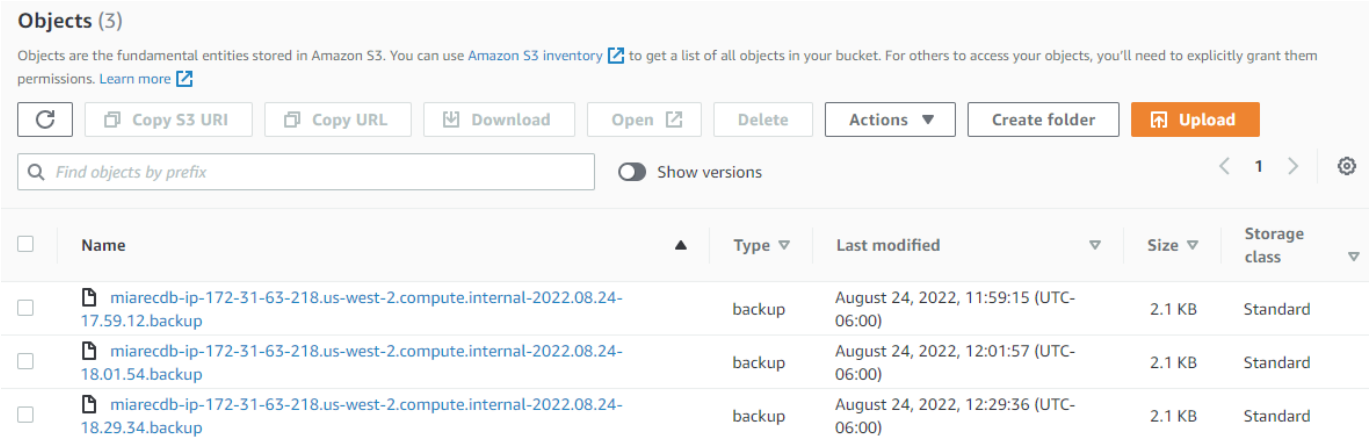
Display `cron` jobs

```
[centos@miarecdb ~]$ sudo crontab -u backup -l
0 1 * * * /usr/local/bin/miarec_backup.sh
```

An archive will be produced at the specified path every night at 1:00am.

```
[centos@marecdb ~]$ ls -l /tmp
-rw-rw-r--. 1 postgres postgres 1102336 Aug  2 16:35 miarecdb.backup
[centos@miarecdb ~]$
```

Navigate to Amazon AWS Console and check the presense of new backup files in the S3 bucket.



4. Restore Database

4.1 Restore database

This section provide instructions how to restore MiaRec database from the previously create backup file.

MiaRec data can be restored in various disaster scenarios. Steps can vary based on the scope of the restoration:

- Complete Reinstall of MiaRec cluster (all-in-one configuration)
- Complete Reinstall MiaRec cluster (decoupled configuration).
- Partial Reinstall of Database component only (decoupled configuration).

4.2 Retrieve database backup from S3

In any restore scenario, you will need to retrieve the database backup file from S3 and load it to the database instance.

This can be accomplished either using AWS CLI directly on the database instance or manually downloading the file via the Amazon AWS Web portal and uploading it to the server via SCP protocol.

4.2.1 Option 1. Copy database backup from S3 to the Database instance using AWS CLI

> **Info**
>
> This will require AWS CLI, if the database instance has been replaced, this will need to be reinstalled. Those steps are listed earlier in this document.

Set AWS credentials

Using environmental variables, these values will be set for this session, but not be retained after the session is ended.

```
export AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>

export AWS_SECRET_ACCESS_KEY=<AWS_AC_KEY_ID>
```

Execute a copy command

```
aws s3 cp s3://{s3-bucket-name}/{filename} /path/to/destination
```

Let's break down those options.

- `aws s3 cp` calls the aws cli utility and instructs it to copy a file from s3 bucket to local file path
- `s3://{s3-bucket-name}` sets a source of the database backup file on S3 bucket.
- `/path/to/source` - a destination path and file name of the restored database dump

Example

```
aws s3 cp
s3://miarec-db-backup/miarecdb-db.example.com-2022.08.17-21.05.24.backup/tmp/miarecdb.backup
```

Verification

The object will be moved to the defined path.

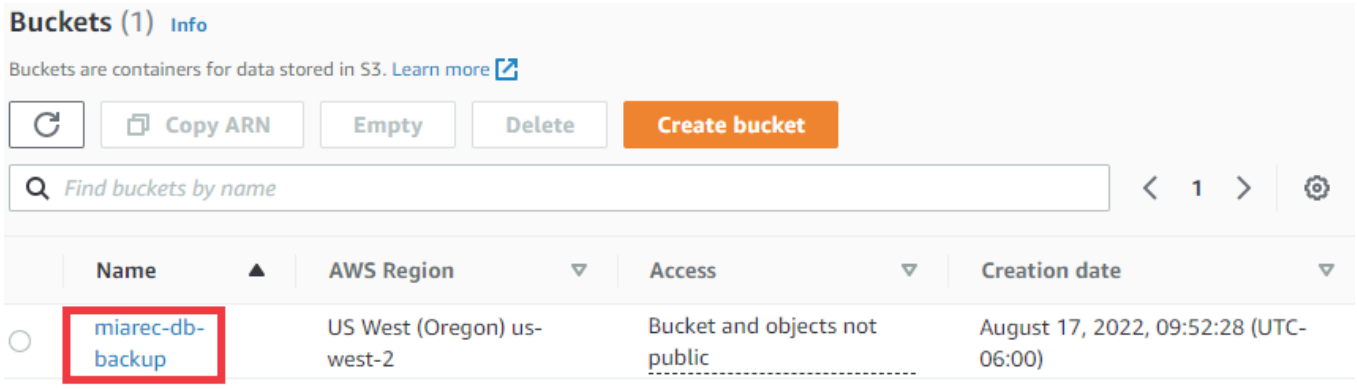
```
[centos@ip-172-31-63-218 ~]$ ls -l /tmp
total 28
...
-rw-rw-r--. 1 centos   centos    2142 Aug 24 17:59 miarecdb.backup
```

4.2.2 Option 2. Download a backup file from the Amazon AWS portal

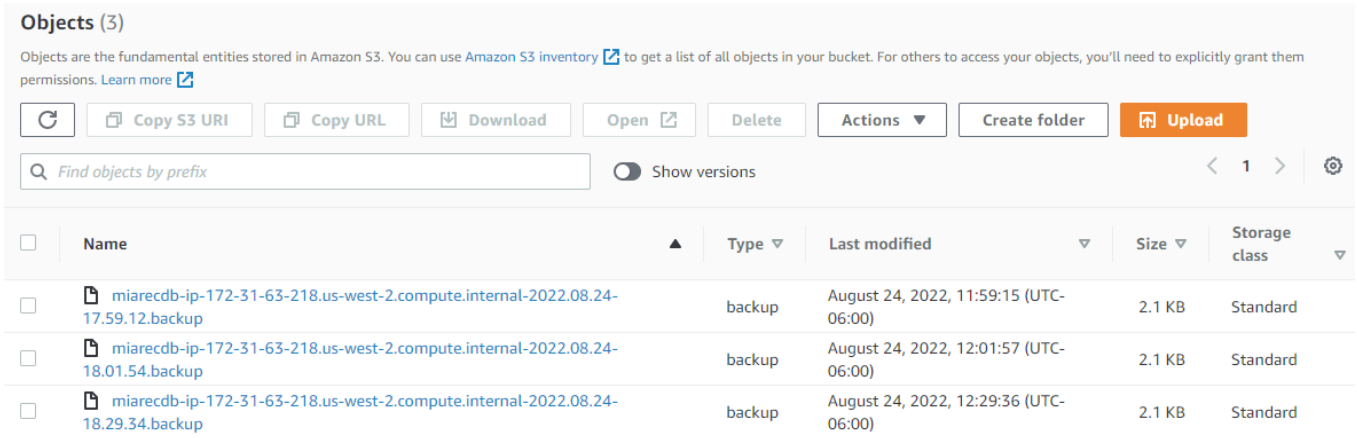
The backup can be downloaded from the AWS S3 console and then transferred to the database instance using a utility like `scp`.

From Amazon S3 console at https://console.aws.amazon.com/s3/.

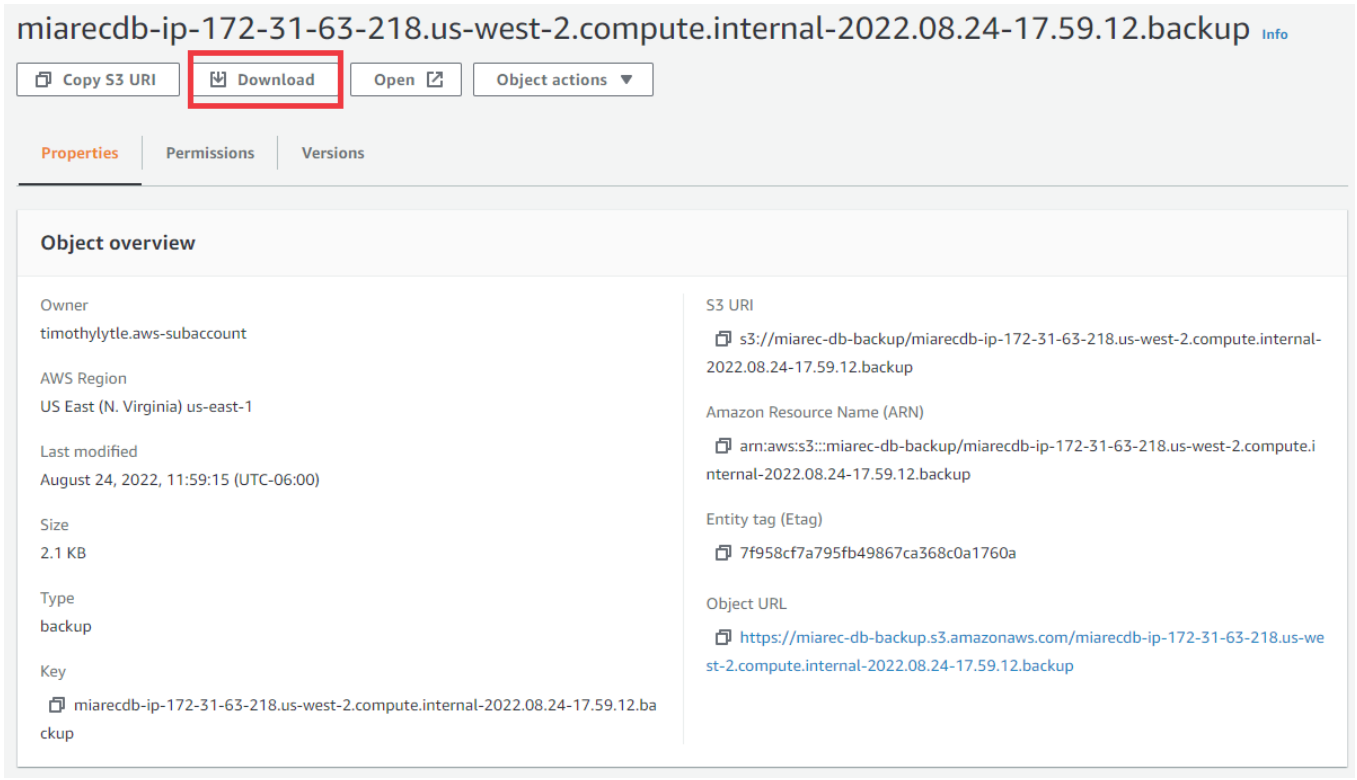
1. Click the name of the Amazon S3 bucket where DB backups are stored.



2. In the **Object** section, locate the corresponding backup file by looking at the timestamp.



3. Select **Download** from the item menu.



4. The file will be downloaded to your current machine. Use an utility like `scp` to transfer the file to the server:

```
scp ~/miarecdb-db.example.com-2022.08.17-21.05.24.backup centos@x.x.x.x:/tmp/miarecdb.backup
```

4.3 Complete Reinstall of MiaRec Cluster (All-In-One and Decoupled)

In the event that an entire cluster, all-in-one or decoupled, is lost and needs to be reinstalled, the following steps can be executed to restore data:

- Install all the software on new servers
- Restore database (configuration, call metadata)
- Restore audio files
- Regenerate license keys

4.3.1 Prepare Ansible Controller Host

This process prepares the `ansible controller` to provision the MiaRec Cluster. The `ansible controller` can be a local machine, the all-in-one MiaRec instance, or one of the instances in a decoupled MiaRec cluster (usually a database instance).

The process is described here

4.3.2 Prepare Target Hosts

This process prepares host(s) to be provisioned. Execute this process on all host(s) in the MiaRec Cluster.

The process is described here

Configure Deployment

This process prepares the ansible inventory which defines what and how the cluster is provisioned. Execute this process on the `ansible controller`.

> **Info**
>
> MiaRec application versions are defined in this step. For successful restoration of the database in later steps, these versions must be the same or newer as the backed-up cluster.

The process is described here

4.3.3 Execute `prepare-hosts.yml` playbook to provision the server(s)

This playbook installs all the required software dependencies for MiaRec applications, including the creation of the `miarecdb` database that can be a target for data restoration in the next step.

Execute Playbook

Execute these commands from the `ansible controller`.

```
cd /opt/ansible-miarec

ansible-playbook -i hosts prepare-hosts.yml
```

Verification

Confirm satisfactory completion with zero items unreachable or failed.

```
PLAY RECAP *********************************************************************
...
miarec                :  ok=79   changed=42   unreachable=0    failed=0
```

Result

All host(s) will be provisioned with supporting software such as Redis, Python, and PostgreSQL. The database `miarecdb` will be created, however, there will be no data present in the database.

Restore Database with `pg_restore` utility

The following commands must be executed from the Database instance. In an all-in-one configuration, this will be a single server instance, in a decoupled architecture, this will be the instance listed in the `[db]` group in the ansible inventory.

Execute `pg_restore` from `db` instance

```
sudo -iu postgres pg_restore -d miarecdb /path/to/backup_file
```

Let's break down those options:

- `suo -iu postgres` instructs the shell to execute the trailing command to run as user `postgres`, this is required for authentication.
- `pg_restore` calls pg_restore utility.
- `-d miarecdb` connects to database `miarecdb` and restores directly into the database, this should always be `miarecdb`.
- `/path/to/backup_file` - the file to restore from, keep in mind this needs to be accessible by the `postgres` user account. Change the file permissions or owner of the backup file if necessary using `chmod` and `chown` utilities.

Verification

The database will be restored, you can verify it by querying the database.

```
sudo -iu postgres psql -d miarecdb -c "SELECT COUNT(*) FROM calls;"
```

The count of all calls in the database should be returned.

```
[centos@miarecdb ~]$ sudo -iu postgres psql -d miarecdb -c "SELECT COUNT(*) FROM calls;"
 count
-------
    34
(1 row)
```

Run `setup-miarec.yml` playbook to install the MiaRec software

This playbook installs the MiaRec applications. This should be executed after the database is successfully restored.

> **Note**
>
> MiaRec application versions specified in the ansible inventory must be the same or newer as the previously backed up cluster.

Execute Playbook

Execute these commands from the `ansible controller`.

```
cd /opt/ansible-miarec

ansible-playbook -i hosts setup-miarec.yml
```

Verification

Confirm satisfactory completion with zero items unreachable or failed.

```
PLAY RECAP *********************************************************************
...
miarec                     :  ok=79   changed=42   unreachable=0    failed=0
```

Result

You should now be able to access the new instance of MiaRec using the previously configured admin account, all configuration and call recordings should be accessible. Audio files may still need to be restored separately depending on an initial storage configuration for audio files.

> **Note**
>
> Depending on the type of recovery, License keys will need to be regenerated, contact Miarec support for that.

Restore License Keys

License keys are stored as database objects. However, with a complete reinstall, previous keys may become invalid on new hardware and new keys must be generated.

Contact MiaRec support for assistance.

4.3.4 Restore Audio Files

When `Relocate Audio Files job` is used to move files to external storage, there is no need to follow the restoration steps. Audio files should be accessible on the same external storage device.

If the files were stored locally on the recording server, then copy the audio files from your existing backup location to the same location as was used previously on the original server.

4.4 Partial Reinstall, Database Component Only (Decoupled configuration)

In the event that a database becomes unavailable or corrupt, the following steps can be executed to reinstall the database instance and accomplish the following:

- Reinstall the Database component
- Restore Database (configuration, call metadata)

4.4.1 Prepare Ansible Controller Host

This might not be needed if the `ansible controller` was not affected.

This process prepares the `ansible controller` to provision the MiaRec Cluster. The `ansible-controller` can be a local machine, the All-In-One MiaRec instance, or one of the instances in a Decoupled MiaRec Cluster.

The process is described here

4.4.2 Prepare Target Hosts

This process prepares the database host to be provisioned.

The process is described here 2. Prepare target hosts

4.4.3 Configure Deployment

This process prepares the ansible inventory which defines how the cluster is provisioned. In this scenario, likely the only update will be to the `db` instance IP address, where a database service should be redeployed.

MiaRec Application versions are defined in this step. For successful restoration of the database in later steps, these versions must be the same or newer as the previously backed-up cluster.

Execute this process on the `ansible controller`.

The process is described here 3. Configure Deployment

4.4.4 Execute `prepare-hosts.yml` playbook to provision the server

This playbook installs all the required software dependencies for MiaRec applications, including the creation of an empty `miarecdb` database.

Execute Playbook

Execute these commands from the ansible controller.

```
cd /opt/ansible-miarec

ansible-playbook -i hosts prepare-hosts.yml
```

Verification

Confirm satisfactory completion with zero items unreachable or failed

```
PLAY RECAP *********************************************************************
...
miarec                :  ok=79   changed=42   unreachable=0    failed=0
```

Result

The database instance will be provisioned with supporting software such as `redis`, `pgbouncer` and `postgresql`. The `postgres` user will be created and the database `miarecdb` will be created, however, there will be no data present.

4.4.5 Restore Database with `pg_restore` utility

The following commands need to be executed from the Database Instance. This will be the instance listed in the `[db]` group in the ansible inventory.

Execute `pg_restore` from `db` instance

```
sudo -iu postgres pg_restore -d miarecdb /path/to/backup_file
```

Let's break down those options:

- `sudo -iu postgres` instructs the shell to execute the trailing command to run as user `postgres`, this is required for authentication.
- `pg_restore` calls pg_restore utility.
- `-d miarecdb` connects to database `miarecdb` and restores directly into the database, this should always be miarecdb.
- `/path/to/backup_file` - the file to restore from, keep in mind this needs to be accessible by the `postgres` user account.

Verification

The database will be restored, you can verify by querying the database.

```
sudo -iu postgres psql -d miarecdb -c "SELECT COUNT(*) FROM calls;"
```

The count of all calls in the database should be returned.

```
[centos@miarecdb ~]$ sudo -u postgres psql -d miarecdb -c "SELECT COUNT(*) FROM calls;"
 count
-------
    34
(1 row)
```

4.4.6 Run `setup-miarec.yml` playbook to modify the existing MiaRec Applications

This playbook installs and/or updates MiaRec applications. In this case, the only action that will be applied updates to the `miarecweb` and `miarec` configuration INI files, directing the applications to the new database IP address.

This should be executed only after the database is successfully restored.

> **Info**
>
> MiaRec Application versions specified in the ansible inventory must be the same or newer as the previously backed up cluster.

Execute Playbook

Execute this command from the `ansible controller`.

```
cd /opt/ansible-miarec

ansible-playbook -i hosts setup-miarec.yml
```

Verification

Confirm satisfactory completion with zero items unreachable or failed

```
PLAY RECAP *********************************************************************
...
miarec               :  ok=79   changed=42   unreachable=0    failed=0
```

Result

You should now be able to access the MiaRec cluster using the previously configured admin account, all configuration and call recordings should be accessible.

4.4.7 Restore License Keys

No Action is required. License keys are stored as database objects, and since the recorder was not changed, license keys are still valid.

4.4.8 Restore Audio Files

No Action is required, since the recorder was not modified, audio files should be intact.

5. Relocate Audio Files

5.1 Prerequisites

5.1.1 Create a bucket for audio files

An S3 bucket must be created as a storage target for the audio files.

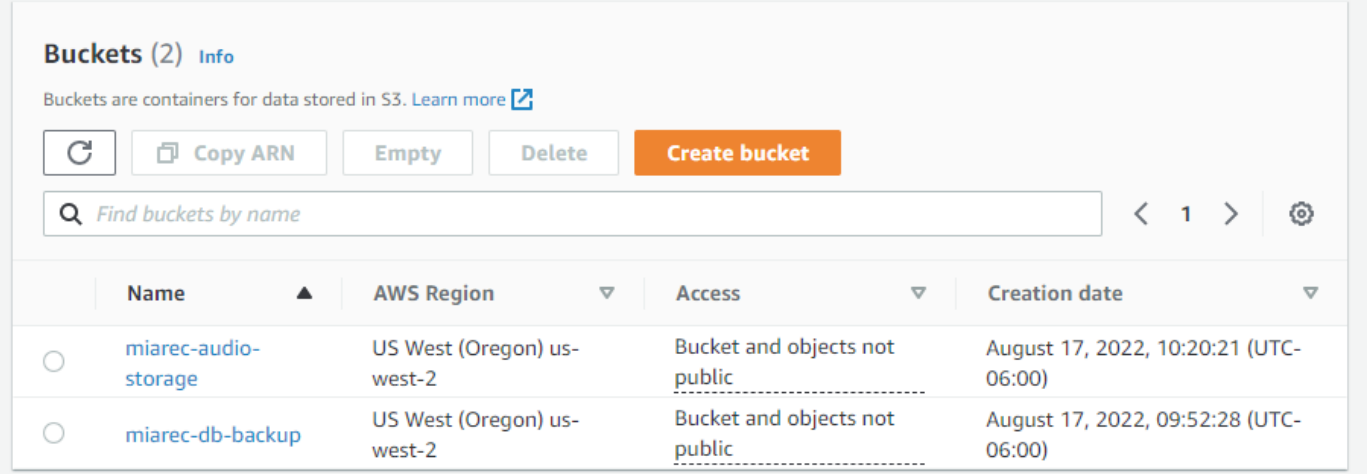
> **Why can't audio files and database backups share the same bucket?**
>
> Database backups should be stored using the WORM (*write-once-read-many*) model to prevent corruption or tampering, whereas audio files will need to be periodically modified or removed depending on the retention policies. To support WORM storage, **S3 Object Lock** has to be enabled, this is defined at the bucket level, requiring separate buckets.
>
> More information about S3 Object Lock can be found at the following link. https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

From Amazon S3 console at https://console.aws.amazon.com/s3/.

1. Choose **Create bucket** from the console top menu to create a new Amazon S3 bucket.
2. On the **Create bucket** setup page, perform the following actions:
   a. For *General configuration:*
      i. Provide a unique name for the new bucket in the **Bucket name** box.
      ii. From the **AWS Region** dropdown list, select the AWS cloud region where the new S3 bucket will be created
   b. For *Block Public Access settings for bucket*, select **Block all public access** to ensure that all public access to this bucket and its objects is blocked.
   c. (Optional) *Tags*, use the Add tag button to create and apply user-defined tags to the S3 bucket. You can track storage costs and other criteria by tagging your bucket.
   d. For *Default encryption*, select **Enable** under **Server-side encryption**, and choose one of the encryption key types available. If you don't know what to choose, then choose *Amazon S3-managed keys (SSE-S3).*
3. Choose **Create bucket** to create your new Amazon S3 bucket.

Result

S3 bucket will be created and displayed in the console.



5.1.2 Create IAM policy for access to the audio bucket

An IAM Policy has to be created and assigned to an IAM user so that objects can be added to the S3 bucket by that IAM user

From Amazon IAM console at https://console.aws.amazon.com/iam/.

1. From the *Policies* menu. Choose **Create Policy** to create a new IAM Policy.
2. Select *JSON* tab, copy the following access policy and paste it into the **JSON** field. **Do not forget to replace** miarec-audio-storage **with your bucket name!!!**.

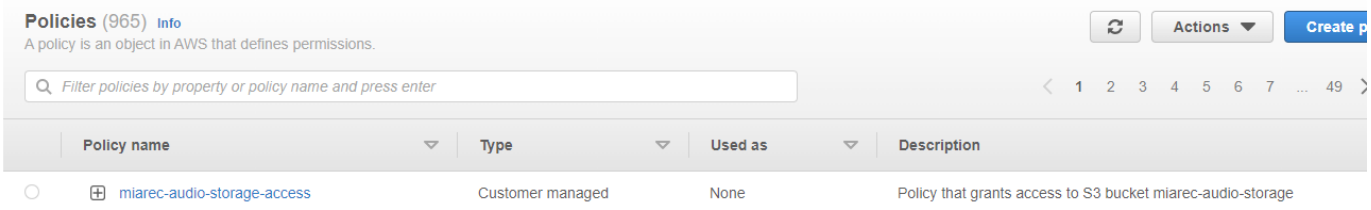
```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-audio-storage"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-audio-storage/*"
        ]
    }
]
}
```

(Optional) *Tags*, use the Add tag button to create and apply user-defined tags to the resource. You can track cost and other criteria by tagging your resource.

Review policy, choose a descriptive name and description for the policy and click the **Create policy** button.

Result

The policy will be created and ready to be assigned

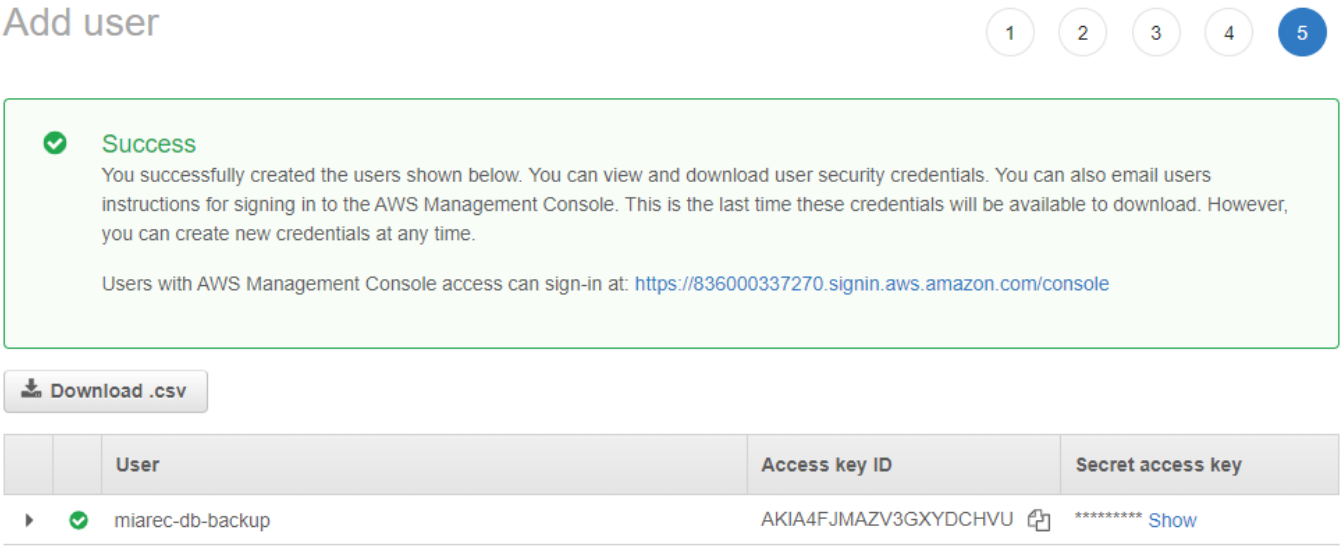


5.1.3 Create IAM User for audio bucket

IAM user has to be created that can be used to relocate audio files from Miarec to S3 storage. We recommend using a separate user account rather than granting the same user access to both database backup and audio file buckets.

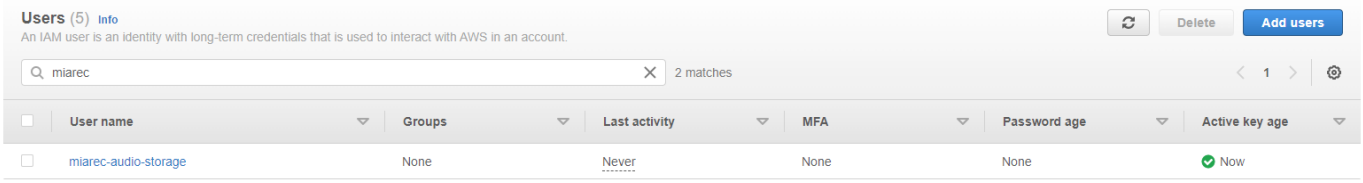
From Amazon IAM console at https://console.aws.amazon.com/iam/.

1. From the *Users* menu, choose **Add User** to create a new IAM User.
2. *Details*, choose **User name** and enable **Programmatic access**.
3. *Permissions*, select **Attach existing policies directly** and then select the previously created policy from the list. *Use the search box to find the policy by name.*
4. (Optional) *Tags*, use the Add tag button to create and apply user-defined tags to the resource.
5. *Review*, confirm the settings and click **Create user**.
6. On the *Complete* screen, copy **Access Key ID** and **Secret access key** and store them in a secure place. This will be used later to configure a Storage target in the MiaRec application.



Result

User will be added, access key and the secret access key will be available to use to access the S3 bucket.

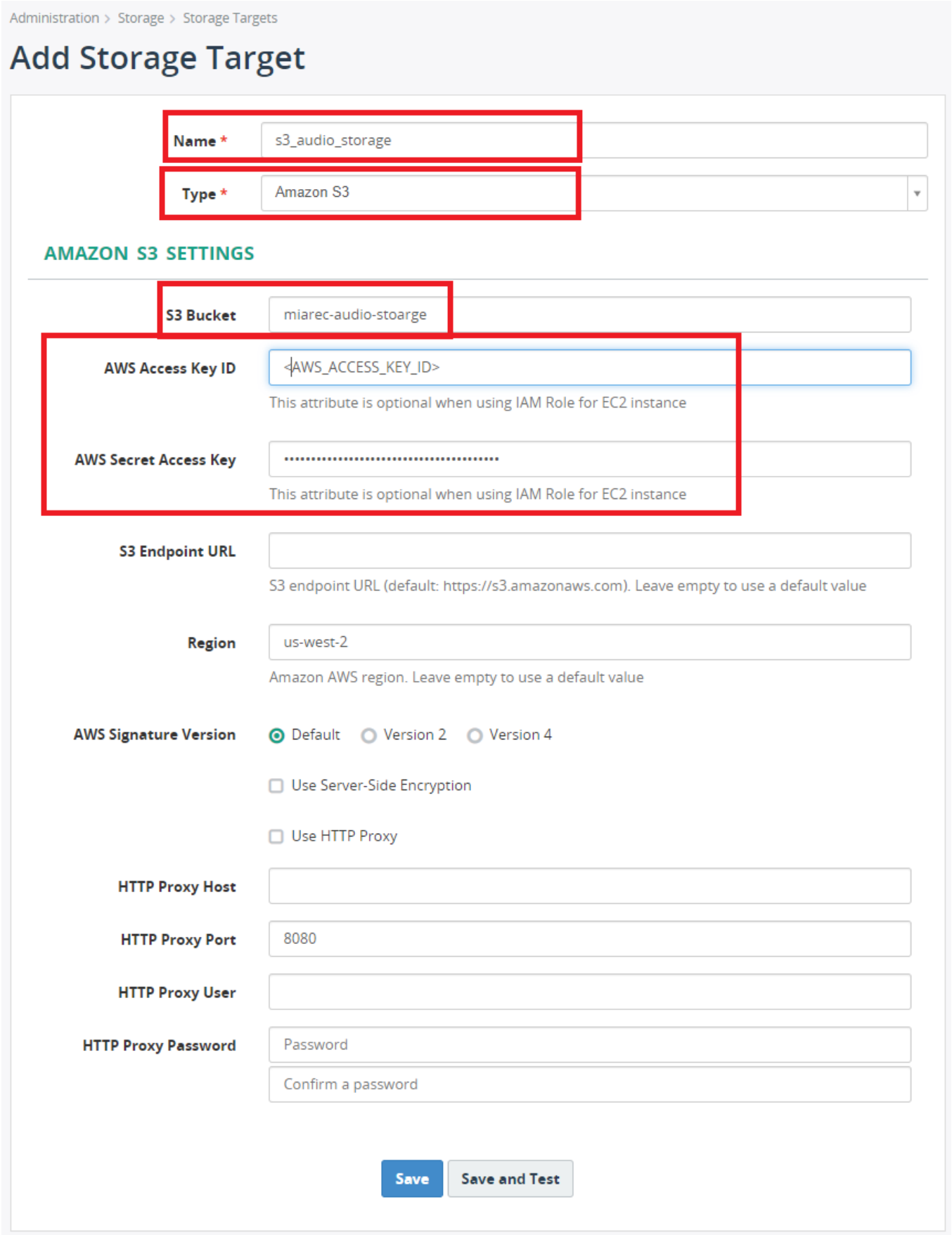


5.2 Automatically relocate audio files to S3 bucket

The MiaRec admin portal offers a feature to automatically relocate files to an external storage target, in this case, an AWS S3 bucket. This function moves an audio file from a local file system to an S3 bucket and updates a file path in the database for each call. Note, that function moves the file rather than creating a copy of it. A redundancy for audio files is achieved by the nature of the Amazon S3 service that provides 99.999999999% durability and 99.99% availability. Optionally, an automatic replication in the S3 bucket can be enabled, that copies asynchronous objects across S3 buckets. To enable such a replication, check the Amazon S3 User Guide.

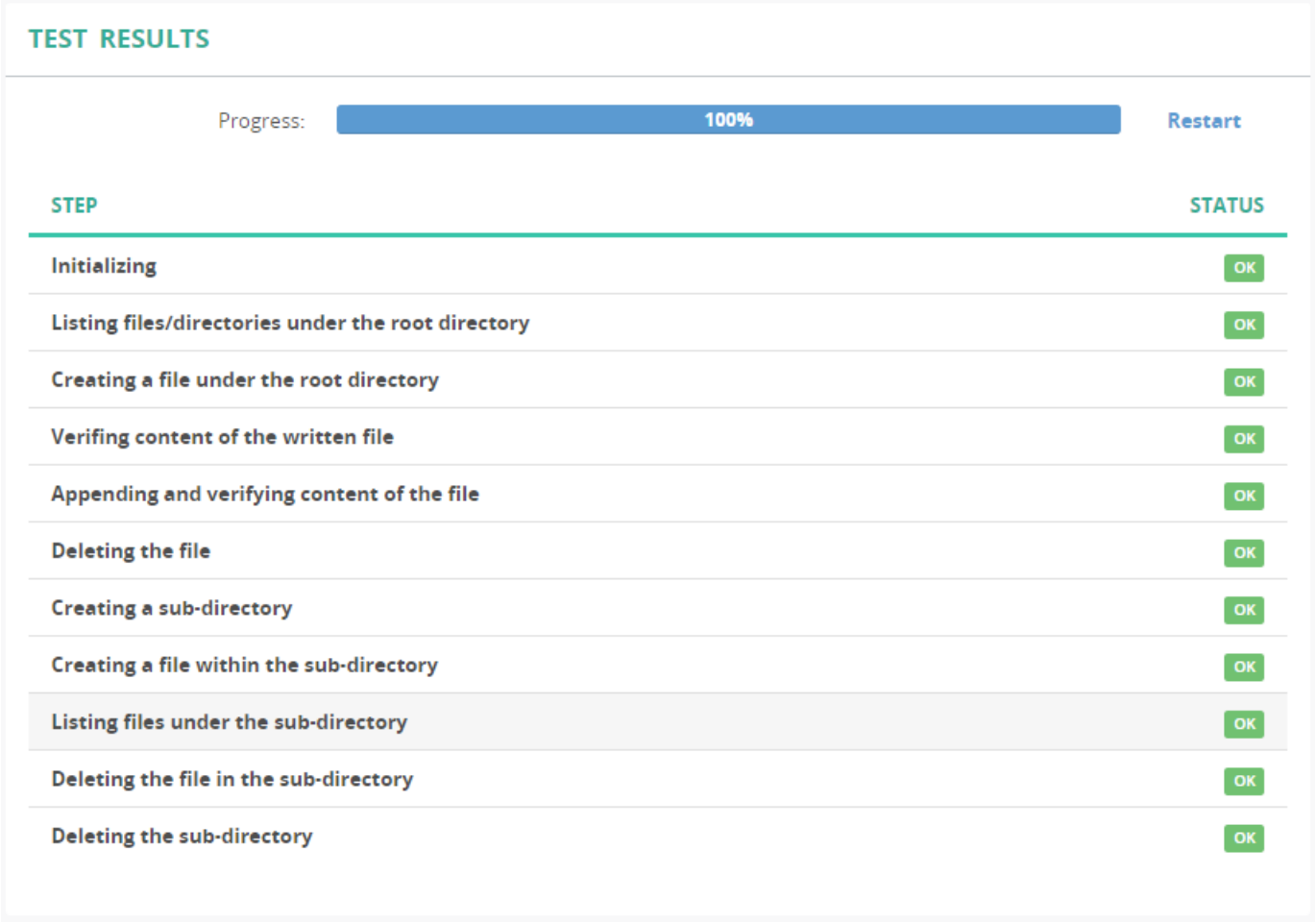
5.2.1 Create a Storage Target

1. Navigate to **Administration > Storage > Storage Target.**
2. Select **Add.**
3. Populate the fields that are appropriate for your deployment.



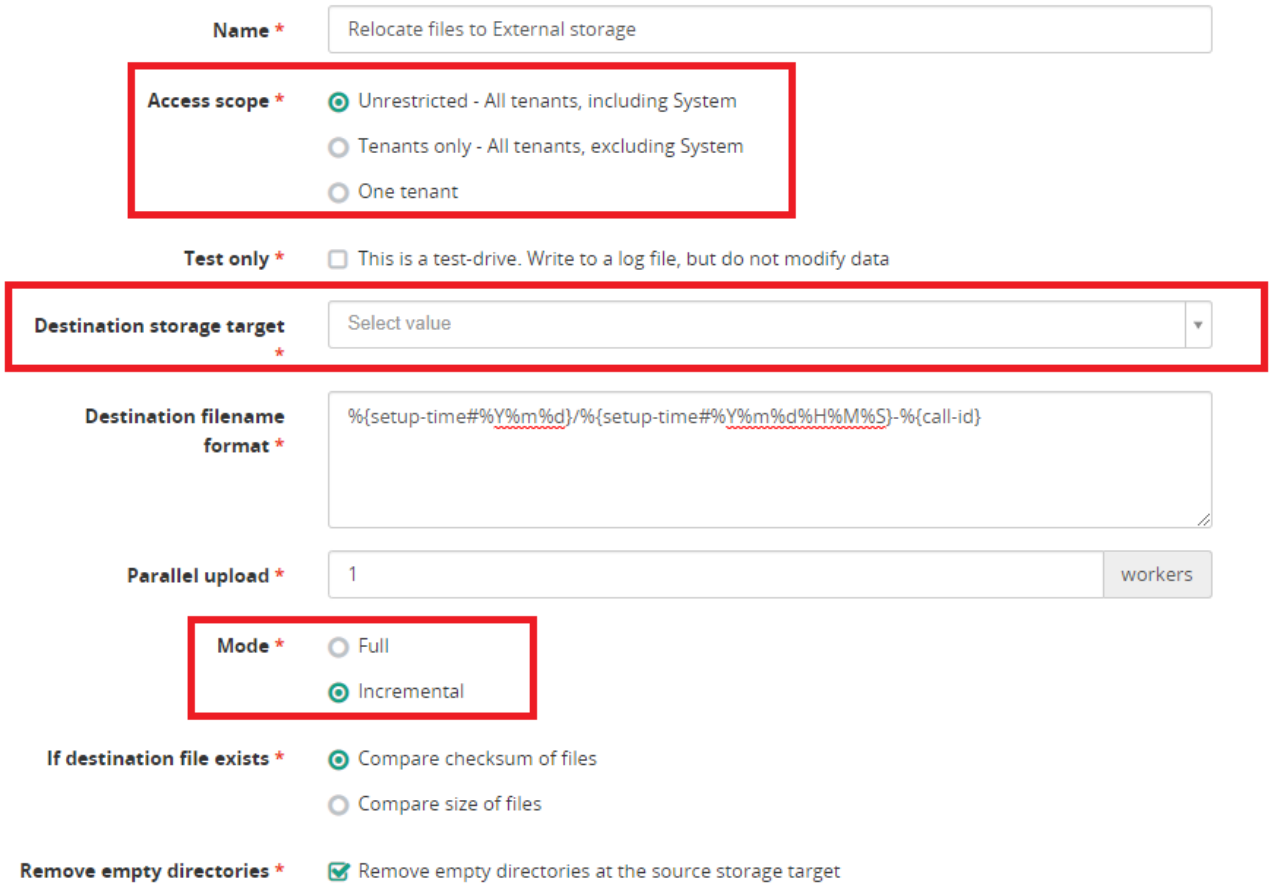
4. `Name` Unique Identifier for this storage target.
5. `Type` Amazon S3.

6. `S3 Bucket` Bucket name defined earlier.
7. `AWS Access Key ID` and `AWS Secret Access` Key Access keys created for IAM user earlier.
8. `Region` bucket region defined earlier.
9. Select **Save and Test.**
10. Verify all tests pass.



5.2.2 Schedule Relocate Audio Files Job

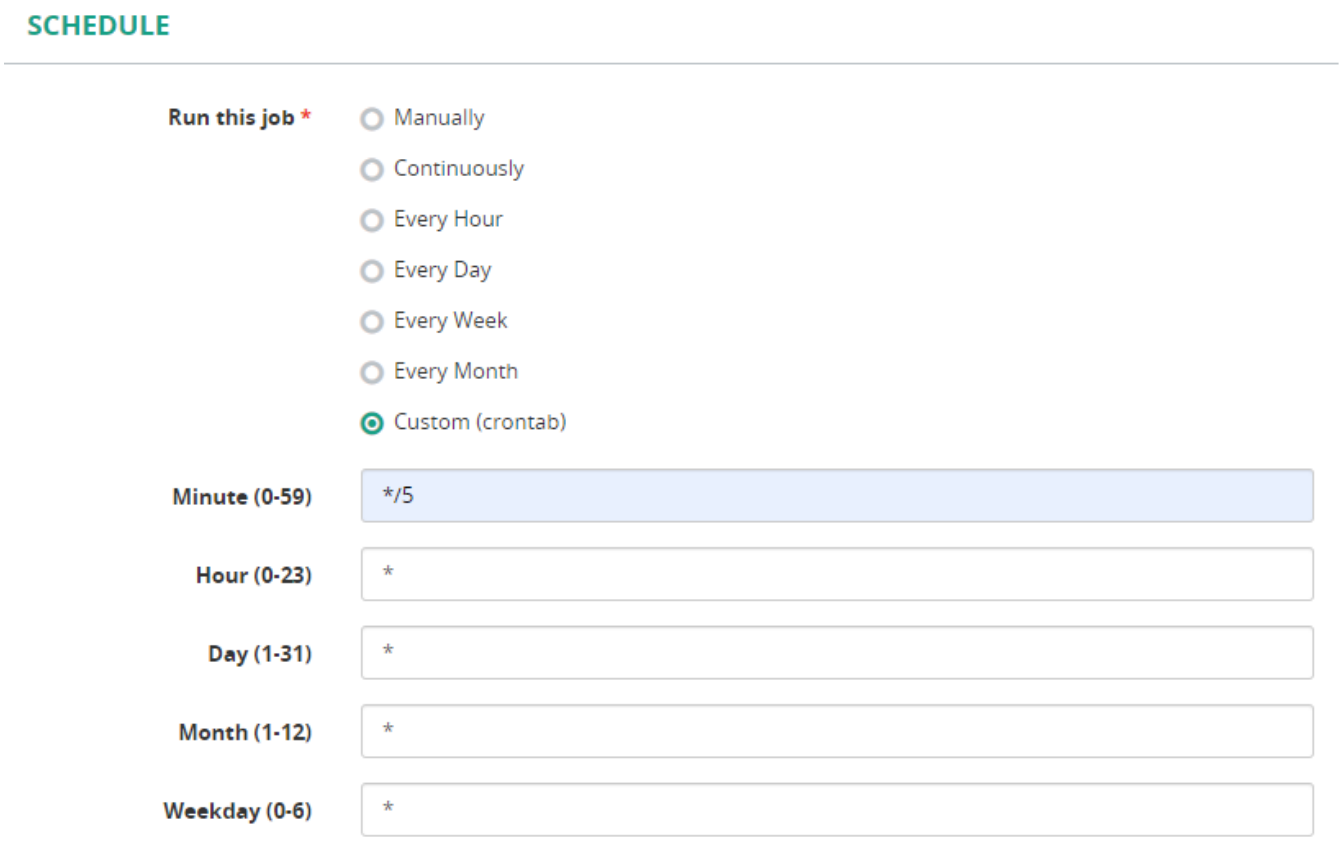
1. Navigate to **Administration > Storage > Relocate Audio Files.**
2. Select **Add.**
3. Define the **Access Scope**, **Mode** and **Destination storage target** (defined in the previous step)



- `Access Scope` In most cases this will be **Unrestricted** seperate relocation jobs can be scheduled for individual tenants if needed.
- `Mode` **Incremental**, system will only target files it has not previously relocated.
- `Destination storage target` Defined in the previous step, is where files will be moved.

4. Apply any filter criteria (optional)

5. Select a Schedule to execute



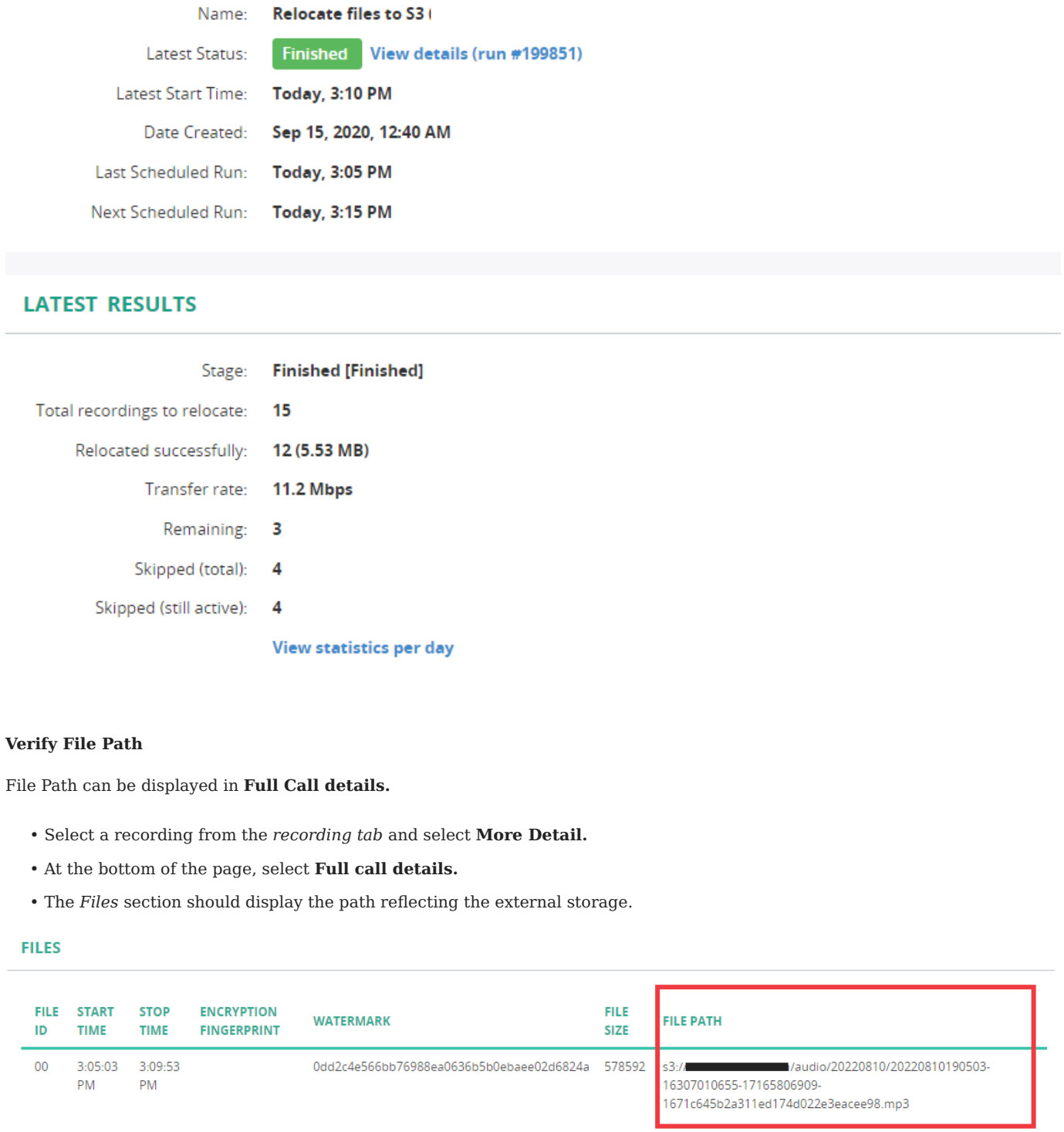
6. Select **Save and Start**

5.2.3 Verification

Calls will be relocated to the external storage target and the file path will be updated in the database.

Verify Relocation Job

See job run results at **Administration > Storage > Relocate Audio Files**.



Verify File Path

File Path can be displayed in **Full Call details.**

- Select a recording from the *recording tab* and select **More Detail.**
- At the bottom of the page, select **Full call details.**
- The *Files* section should display the path reflecting the external storage.