ResourceSpace KnowledgeBase
AWS S3 Storage
Amazon Web Services (AWS) Simple Storage Service (S3) Object-Based Original Files Storage
August 16, 2019

Object-based storage, such as Amazon Web Services (AWS) Simple Storage Service (S3) can provide very cost-effective storage of files, particularly large files. Native support of AWS S3 is provided in ResourceSpace for storage of original resource files in a specified S3 bucket (a container for file objects). Preview and other resized images are stored in the existing ../filestore/.. location as before. Object-based storage systems do not use the concept of a folder or directory structure, each file stored in them is referred to as an object. However, the object name can contain the desired path structure, preserving the original folder structure used elsewhere. Original resource files stored in S3 are named by their unique ResourceSpace pathname after the filestore folder name to preserve the filestore structure for easy conversion from one storage system to another. Files uploaded to or downloaded from S3 are sent encrypted for security.

There is no limit to the number of files or the total file size stored in an AWS S3 bucket; however, individual files are limited to a maximum size of 5 TB each. Individual files are not able to be uploaded in ResourceSpace greater than 5 TB when AWS S3 storage is used.

AWS S3 storage is currently billed monthly. Pricing information on the use of S3 storage is available at https://aws.amazon.com/s3/pricing/, depending upon the region is the bucket is located in (physical location of a group of AWS data centers), the storage class, total storage size, and downloads.

Users of your ResourceSpace system have no access to AWS S3 or the files (objects) stored there. ResourceSpace natively handles all of the connections and file management using AWS APIs for enhanced security.

AWS S3 Storage Setup Instructions

1. Verify your PHP web server meets AWS requirements that include PHP v5.5.0 or later and cURL v7.16.2 or later. If your web server does not meet these requirements, you will not be able to use AWS S3 storage in ResourceSpace, so stop here. It is recommended to use the PHP OPCache extension.

2. Setup a new AWS S3 bucket (https://console.aws.amazon.com/s3/home). Bucket names are globally unique within the S3 system, so several tries may be needed to determine a unique name. Additional information on bucket naming is available at https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html. It is usually best to pick an AWS region the bucket will be located in physically closest to your ResourceSpace server to reduce latency. Use the default bucket settings and permissions allowing access to the bucket only by your account and no public access.

You will need an AWS account to setup a new bucket. If needed, create a new AWS account at https://aws.amazon.com/ and click on the 'Create an AWS Account' orange button at the page top left. For new AWS users, you may want to consider the AWS Free Usage Tier that currently includes 5GB of S3 Standard storage; 15GB of data transfer out of S3 storage per month; and 20,000 GET requests, 2000 PUT, Copy, POST, or LIST requests over one year.

3. Create an AWS Access Key pair at User, My Security Credentials/Access Keys (https://console.aws.amazon.com/iam/home). Once an Access Key pair (access key ID and secret access key) is created, the secret key is not retreivable afterwards, so make sure to save the pair file. Otherwise, you will need

to create a new Access Key pair. Do NOT share this key pair with anyone or else they will have full access to the AWS S3 bucket.

4. Decide on an AWS S3 Storage Class. If unsure, use INTELLIGENT_TIERING, which works well for most ResourceSpace use cases. More information is available at https://docs.aws.amazon.com/AmazonS3/latest/dev/storage-class-intro.html.

5. If you are converting an existing ResourceSpace installation to AWS S3 storage, follow Steps 5a to 5d; otherwise, skip to Step 6:

5a – Make sure to perform a full backup of the ../filestore folder before proceeding and ensure that no one is using the system (all users should be logged out).

5b – Ensure that all resources in the deleted state are deleted permanently, see Admin, Manage Resources, View Deleted Resources, which should be empty. If not, delete these files permanently. You can also run the ../pages/tools/purge_deleted_resources.php script in a Console (command line).

5c – Cleanup the existing ResourceSpace database by deleting unused and orphaned rows in the database by running the ../pages/tools/database_prune.php script in a Console (command line).

5d – Cleanup the existing ResourceSpace filestore for directories without associated database entries by running the ../pages/tools/prune_filestore.php script script in a Console (command line).

6. Install curl and php-curl for your specific web server OS:

Ubuntu/Debian Linux: `sudo apt-get install curl php-curl`
CentOS Linux: `sudo yum install curl php-curl`
Windows-based: install the php_curl.dll PHP extension

7. Enable ResourceSpace AWS S3 storage functionality in the ../include/config.php file by setting all of the following parameters:

```
$storagedir = “”;   // Filestore location, such as “/var/www/resourcespace/
filestore”.
$originals_separate_storage = false;
$purge_temp_folder_age = ‘’;
$exiftool_write = true;
$exiftool_write_metadata = true;
$replace_resource_preserve_option = false;
$replace_resource_preserve_default = false;
$replace_batch_existing = false;
$custompermshowfile = false;

$aws_s3 = true;          // Enable AWS S3 original file storage?
$aws_bucket = ‘’;        // AWS S3 bucket name.
$aws_region = ‘’;        // AWS region the S3 bucket is located in.
$aws_storage_class = ‘’; // AWS S3 storage class.
$aws_key = ‘’;           // AWS S3 Access Key ID.
$aws_secret = ‘’;        // AWS S3 Secret Access Key.
$aws_tmp_purge = ‘’;     // Time in minutes to purge the AWS tmp folder.
```

8. Verify AWS S3 bucket connectivity using the Admin, System, Installation Check and Admin, System, AWS S3 Dashboard. Resolve any lines reporting FAIL before uploading new resources or converting an existing

filestore. The AWS S3 Dashboard contains valuable information on parameter settings and the size of the S3 bucket. As files are uploaded to AWS S3, you can verify this by checking the S3 Management Console page at https://console.aws.amazon.com/s3.

9. If you have an existing ResourceSpace installation and filestore to convert to AWS S3, run the ../pages/tools/filestore_to_s3.php script in a console (command line interface); otherwise, skip to Step 10 below. This script will upload progress through the filestore uploading each original resource and original alternative file(s) to the specified S3 bucket, then verify each file has successfully been uploaded to S3, and if successful, will delete the local filestore original file and save a zero-byte placeholder file. If a file is not verified, it will be skipped. After script completion, rerun the script to catch these files again. It is likely that the file(s) were successfully uploaded to S3; however, due to the eventual consistency nature of object-based storage systems, files may not be immediately available for use. This delay with AWS S3 is usually a very short time period.

As the script runs, all of the console text output will also be saved to a text (.txt) file named by the datetime timestamp the script was started in the ../filestore/tmp/aws_s3 folder. It is recommended to review this file for any errors or other irregularities before resuming use of your ResourceSpace system. The file will also contain a summary of the number of files processed.

10. New files may now be uploaded to ResourceSpace and users allowed to access the system. There is a slight delay when creating new resources or downloading existing resources, as the original files are stored remotely in AWS S3 storage. Most of the time, this delay is minimal and unless very large files are used, most users will not notice the difference.

11. If issues arise, check the ResourceSpace Installation Check and AWS S3 Dashboard pages for any errors, and enable the ResourceSpace debug log in ../include/config.php by adding $debug_log = true; and set the $debug_log_location = “” parameters. Check for irregularities, including with AWS operations that begin with the text AWS or have S3 in the name.

ResourceSpace Operations Tested with AWS S3 Storage

Upload Resource
Upload Resource Alternative File
Download Resource
Download Resource Alternative File
Download Collection
Delete Resource
Delete Resource Alternative File
Delete All Resources in a Collection
Replace (Original Resource) File

ResourceSpace Activated Plugins Tested with AWS S3 Storage

lightbox_preview
image_text

ResourceSpace Plugins Known to Not Work with AWS S3 Storage

transform

New AWS Functions for Developers Provided in ../include/aws_sdk.php

Get an AWS S3 bucket location by AWS region.

function aws_s3_bucket_location($aws_bucket)

Check the accessibility of an AWS S3 bucket.

function aws_s3_bucket_check($aws_bucket)

Get the owner of an AWS S3 bucket.

function aws_s3_bucket_owner($aws_bucket)

Get the AWS S3 bucket metrics using AWS CloudWatch.

function aws_s3_bucket_statistics($cw_metric, $cw_statistic, $cw_unit, $cw_storage = “”)

Convert the AWS S3 storage class code to a name.

function aws_s3_storage_class($aws_storage_class)

Create an AWS object path from the normal filestore path.

function aws_s3_object_path($path)

Check if an AWS S3 object exists in a specified bucket.

function aws_s3_object_check($path)

Upload a local file to a specified AWS S3 bucket.

function aws_s3_object_uploader($fs_path, $s3_path = ‘’, $fs_delete = false)

Delete a filestore original file and create a placeholder file in its place.

function aws_s3_file_placeholder($fs_path)

Determine a local filestore temp filename for a given file with a random temp name.

function aws_s3_file_tempname($path, $uniqid = “”)

Copy an AWS S3 object to another path.

function aws_s3_object_copy($old_path, $new_path, $old_delete = false)

Download an AWS S3 object in a specified bucket to a local file.

function aws_s3_object_download($path, $s3_tmpfile)

Delete an AWS S3 object in a specified bucket.

function aws_s3_object_delete($ref, $fs_path = “”)

Cleanup filestore tmp folder by purging files, not subfolders, based on file age in minutes.

function filestore_temp_cleanup($min_age = 5)

Folder cleanup for a specified filename search parameter.

function filestore_folder_cleanup($path, $filename_text)

Convert a boolean ‘true-false’ value to ‘Ok-FAIL’ or ‘Yes-No’ text.

function boolean_convert($input, $type)