ResourceSpace Object-Based Simple Storage Service (S3) Support
November 19, 2020
SVN Branch: https://svn.resourcespace.com/svn/rs/branches/sbowman/20201026_sbowman_s3_3/

Introduction

Object-based storage, such as Amazon Web Services (AWS) Simple Storage Service (S3) can provide very cost-effective storage of files, particularly large files. The *..S3_3* branch incorporates support for S3 storage using the AWS SDK for PHP Version 3 (https://aws.amazon.com/sdk-for-php/, Apache License v2.0) to enable S3 API support. Since AWS developed the S3 object-based storage API, the AWS SDK was used for maximum compatibility. Other provider APIs could be integrated; however, many of these reuse the AWS API code. ResourceSpace (RS) can connect to and use S3 storage from AWS, DigitalOcean, and other S3-compliant storage providers or appliances. As this is an open-source project, the additional RS code has been made provider agnostic, not favoring one provider over another.

Users of your ResourceSpace system have no access to S3 storage or the files (objects) stored there. ResourceSpace natively handles all of the connections and file management using AWS APIs for enhanced security and transfers between the local server and the S3 provider are encrypted by default.

Storage of original resource files is 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 (other providers may be different); 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.

Original Resource File S3 Storage Setup

1. Verify your PHP web server meets ResourceSpace and AWS (for the PHP SDK) requirements that include PHP v7.0 or later, the SimpleXML PHP extension, and cURL v7.16.2 or later. If your web server does not meet these requirements, you will not be able to use S3 storage in ResourceSpace, so stop here. It is highly recommended to use the PHP OPCache extension for performance.

   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

2. You will need to setup an account with your S3 provider to create an use a S3 bucket For AWS, create a new account at https://aws.amazon.com/ and click on the '*Create an AWS Account*' orange button. 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. Setup a new S3 bucket with your provider; for AWS, see: https://console.aws.amazon.com/s3/home. Bucket names are globally unique within a S3 system, so several tries may be needed to determine a unique name. Additional information on AWS S3 bucket naming is available at https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html. It is usually best to pick a 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.

   AWS S3 Bucket Creation and Configuration Settings
   General Configuration
   - Bucket Name: Your unique bucket name.
   - Region: Select the region the bucket will be created in and where the objects will be stored.
   - Copy Settings From Existing Bucket: Ignore

   Check Block All Public Access.
   Bucket Versioning: Disable
   Tags: none defined
   Default Encryption
   - Server-Side Encryption: Disable

   Advanced Settings
   - Object Lock: Disable

   Then click the *Create Bucket* button. On the Buckets page that will now be displayed, click on your new bucket and the Object section should be empty, showing no objects have been uploaded. You can use this display to verify correct object upload. You may need to refresh the list by using the circle arrow button.

4. Create an security Access Key pair. For AWS, go to: *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 access key pair file in a secure location. 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 S3 bucket and all of its contents (objects).

5. Decide on a S3 Storage Class. For AWS and 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.

6. If you are converting an existing ResourceSpace installation to S3 storage, follow Steps 6a to 6d; otherwise, skip to Step 7:

   6a – 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).

   6b – 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) window.

6c – 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) window.

6d – 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) window.

7. Set the following ResourceSpace configuration parameters in the *../include/config.php* file:

```
$storagedir = '';              // Full path to the ../filestore folder.
$purge_temp_folder_age = x;    // Time in days to clear the tmp folder.
$exiftool_write = true;        // Enable ExifTool metadata writing?
$exiftool_write_metadata = true; // Force Exiftool metadata writing?
$custompermshowfile = false;
```

8. Set the following ResourceSpace S3 configuration parameters in the *../include/config.php* file:

```
$s3_enable = true;             // Enable S3 storage?
$s3_provider = '';             // S3 provider code.
$s3_endpoint = '';             // S3 provider endpoint URL.
$s3_region = '';               // S3 region the bucket is located in.
$s3_key = '';                  // Security access key.
$s3_secret = '';               // Security secret key.
$s3_bucket = '';               // S3 bucket name.
$s3_storage_class = '';        // S3 storage class code.
$s3_temp_purge = x;            // Time in days to clear S3 temp folder.
$s3_stats = true or false;     // Enable S3 transfer statistics?
```

9. Verify S3 storage connectivity and settings by using the S3 Installation Check at *Admin, System, S3 Installation Check* (http://localhost/resourcespace/pages/admin/admin_system_s3.php). Resolve any lines reporting FAIL before uploading new resources or converting an existing filestore; otherwise, you may lose files.

10. If you have an existing ResourceSpace installation and filestore to convert to S3 storage, run the *../pages/tools/filestore_to_s3.php* script in a console (command line interface); otherwise, skip to Step 11 below. This script will 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 in the *../filestore/tmp/s3_cache* folder named by the datetime timestamp the script was started. 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 successfully processed and errors.

11. 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 S3 storage. Most of the time, this delay is minimal and unless very large files are used, most users will not notice the difference.

12. If issues arise, check the ResourceSpace Installation Check (*Admin, System, Installation Check*; *../pages/check.php*) and S3 Installation Check (*Admin, System, S3 Installation Check*; *../pages/admin/admin_system_s3.php*) 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* parameter. Check for irregularities, including with S3 operations that have S3 in the name.

13. The S3 Storage Dashboard (*../pages/admin/admin_system_s3dashboard.php*) at *Admin, System, S3 Storage Dashboard* can be used to help manage S3 storage. The S3 Provider and Bucket Parameters section shows the S3 connection parameters and their status. If AWS S3 storage is used, the Amazon Web Services (AWS) CloudWatch S3 Metrics section shows the total bucket file size, number of stored objects in the current bucket, and other information (metrics) on the bucket and its use. CloudWatch metrics are posted once each day; as a result, data may not be accurate at the date and time of the page load. A *?* value indicates that specific metric is not currently available.

ResourceSpace Operations Tested with S3 Storage

Upload Resource
Upload Alternative File
Download Resource
Download Alternative File
Download Collection
Delete Resource
Delete Alternative File
Delete All Resources in a Collection
Replace Resource
Save Original as Alternative
Recreate Previews
Installation Check
S3 Installation Check
S3 Storage Dashboard
Filestore to S3 Storage Tool Script – more testing needs be performed before using in production environments.

New S3 PHP Variables to Set in ../include/config.php

$s3_enable = true/false [default]
Boolean switch to turn S3 storage usage on or off.

$s3_provider = ‘AWS’ [default]
Enter the provider name here, use ‘AWS’ for Amazon Web Services. For other providers, you may need to modify the *s3_storage_class* function to account for various names the providers use for their storage levels.

$s3_endpoint = ‘’
If using a provider other than AWS, enter the provider endpoint URL here. See your storage provider documentation for more information.

$s3_region = ‘’

Region the S3 bucket is located in. See your storage provider documentation for more information.

$s3_key = ‘’
Security key for accessing S3 storage. See your storage provider documentation for more information.

$s3_secret = ‘’
Secret key for accessing S3 storage. See your storage provider documentation for more information.

$s3_bucket = ‘’
Unique name of the S3 bucket to use. See your storage provider documentation for more information.

$s3_storage_class = ‘STANDARD’ (default), ‘INTELLIGENT_TIERING’, ‘STANDARD_IA’, ‘ONEZONE_IA’, ‘REDUCED_REDUNDANCY’, ‘OUTPOSTS’, or ‘NONE’
Storage class name to use, for AWS, INTELLIGENT_TIERING is recommended for most use cases. See your storage provider documentation for more information.

$s3_temp_purge = 1 [default]
Time in days to clear the S3 temp folder and based on last access.

$s3_stats = true [default]/false
Enable adding network transfer statistics to errors and results when *$debug_log = true;*? May not be applicable all S3 storage providers. See your storage provider documentation for more information.

Technical Details of the S3 Storage Workflow in the ResourceSpace Code

Workflows for uploading, downloading, deleting, and replacing resources are the same as the current RS workflow. No changes are apparent to the user, other than a slight delay when files are uploaded or downloaded from S3 storage.

S3-Specific PHP Functions (*../include/s3_storage_functions.php*)

New PHP functions to enable S3 storage support in RS are contained in *../include/s3_storage_functions.php* that is loaded by *../include/db.php* if *$s3_enable = true* in *../include/config.php*. *s3_storage_functions.php* uses the AWS PHP Version 3.158.6 SDK and its autoloader to load the necessary classes from *../lib/aws_php_sdk_3.158.6*. Future versions of the SDK (https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/getting-started_installation.html#installing-by-using-the-zip-file) can be used by saving the specific SDK to a new *../lib* folder , deleting the older SDK folder, and changing Line 5 in *s3_storage_functions.php* to the new folder name. Should a new SDK use a new S3 API version, that can be changed on Line 18; or a new CloudWatch API version, that can be changed on Line 19.

New S3 parameters have been added to *../include/config.defualt.php* in the SIMPLE STORAGE SERVICE (S3) OBJECT-BASED STORAGE PARAMETERS section. Existing filestore-specific parameters have been consolidated under a new section titled RESOURCE FILESTORE PARAMETER CONFIGURATION, just below the S3 parameters section.

Once the SDK is loaded, *$shared_config* contains the configuration parameters used for setting up S3 and AWS CloudWatch clients, and then a new SDK class (*$sdk*) is setup. The remainder of the *s3_storage_functions.php* file contains the following S3-specific functions:

s3_check_parameters()

Function to check that the required S3 API parameters are set in *../include/config*. Returns TRUE on success or FALSE on fail.

s3_get_api($sdk)
Function to get the S3 API description. Returns an array of parameters or FALSE on fail.

s3_get_region($sdk)
Function to get the S3 region we are connected to. Returns an array of parameters or FALSE on fail.

s3_get_endpoint($sdk)
Function to get the S3 endpoint URL we are connected to. Returns an array of parameters or FALSE on fail.

s3_storage_class($s3_storage_class, $format = false)
Function to convert the S3 storage class code to a standard name and subcode. $format enables text formatting of the result. Returns an array [code, name, and status] or FALSE on fail.

s3_bucket_owner($sdk, $s3_bucket)
Function to get the owner of a specific S3 bucket. Returns an array [name, id, and status] or FALSE on fail.

s3_bucket_head($sdk, $s3_bucket)
Function to check if a specific S3 bucket exists. Returns an array or FALSE on fail.

s3_bucket_location($sdk, $s3_bucket)
Function to get the region location name of a specific S3 bucket. Returns an array of parameters or FALSE on fail.

s3_bucket_list($sdk)
Function to list the available S3 buckets. Returns an array of parameters or FALSE on fail.

s3_object_path($path, $trailing_slash = false)
Function to create a S3 object path from the normal filestore path. Original files in a S3 bucket are stored using the same structure as in the local filestore. $trailing_slash (boolean) determines if a trailing slash is added to the returned string. Returns the object path (string) or FALSE on fail.
Example: Converts */var/www/resourcespace/filestore/1/1_c4714a4e8517a49/11_4308ef3dccb1ae2.jpg* to *1/1_c4714a4e8517a49/11_4308ef3dccb1ae2.jpg*

s3_object_exists($sdk, $s3_object)
Function to check if a specific object (file) exists in a specific S3 bucket. $s3_object is the object name obtained from the s3_object_path function. Returns an array or FALSE on fail.

s3_object_head($sdk, $s3_object)
Function to get metadata on a specific object in a S3 bucket. $s3_object is the object name obtained from the s3_object_path function. Returns an array of parameters or FALSE on fail.

s3_object_list($sdk, $s3_object_prefix)
Function to list objects with a certain name prefix ($s3_object_prefix) in a S3 bucket. Returns an array of parameters or FALSE on fail.

s3_object_upload($sdk, $fs_path, $s3_object, $s3_storage_class = 'STANDARD')

Function to upload a local file to a S3 bucket. $fs_path is the full path to the local file to upload, $s3_object is the object name obtained from the s3_object_path function, and $s3_storage_class is the storage class to use for the new object. This function will choose either the standard uploader or the multipart uploader, based on the size of the local file. Returns an array of parameters or FALSE on fail.

s3_object_download($sdk, $s3_object, $fs_file)

Function to download an object from a S3 bucket as a local file. $s3_object is the object name obtained from the s3_object_path function and $fs_file is the full, local path and filename for the saved file. Returns an array of parameters or FALSE on fail.

s3_object_delete($sdk, $s3_object)

Function to delete an object from a S3 bucket. $s3_object is the object name obtained from the s3_object_path function. Returns TRUE on success or FALSE on fail.

s3_object_copy($sdk, $object_from, $object_to, $s3_storage_class)

Function to copy an existing object in a S3 bucket to a new object in the same S3 bucket. $object_from is the existing object filename, $object_to is the new object filename, and .$s3_storage_class is the storage class to use for the new object. Returns an array of parameters or FALSE on fail.

s3_file_placeholder($fs_path)

Function to delete a filestore original file and create a zero-byte placeholder file in its place. $fs_file is the full, local path and filename for the original file to be processed. Returns an array of parameters or FALSE on fail.

s3_file_tempname($path, $uniqid = ‘’)

Function to determine a filestore temporary random filename for a given file. $uniqid has the same properties as used in the existing get_temp_dir function. Returns the temp path (string).

s3_create_cache()

Function to create a new folder in the temp folder location. Used for *../pages/tools/filestore_to_s3.php* logs. Returns the temp folder path or FALSE if fail.

s3_folder_cleanup($folder = ‘’, $min_age = 5, $time_type = ‘change’, $max_age = ‘’)

Function to cleanup specified folder by purging files, but not those in subfolders, based on file age ($time_type = ‘change’) or the last access ($time_type = ‘access’). $folder is the specific folder to cleanup; $min_age and $max_age in minutes. Returns TRUE on success or FALSE if fail.

s3_folder_scleanup($path, $filename_text)

Function to clean a folder of files meeting a specific search parameter. $path is the folder to cleanup and $filename_text is the search parameter. Returns TRUE on success or FALSE if fail.

s3_original_download($ref, $alternative = ‘’)

Function to provide original or alternative resource downloads via the API for the bulk_download plugin. $ref is the resource ID and $alternative is the resource alternative ID. Returns the URL or FALSE if fail.

tmp_file_search($subfolder, $search_file)

Function to enable future local caching of files downloaded from a S3 bucket. Returns TRUE on success or FALSE on fail.

s3_metric_statistics($sdk, $namespace, $metric_name, $dimensions, $start_time, $end_time, $period, $statistics, $unit)

Function to get AWS S3 CloudWatch metric statistics if using AWS S3 storage. Returns an array of parameters or FALSE on fail.

cw_dimension($name1, $value1, $name2, $value2)

Function to create the input dimension parameters for the AWS CloudWatch metric statistics. Returns an array of parameters.

folder_file_size($folder, $string = false)

Function to determine the total number and file size of all files in a folder, not including subfolders. For future use enabling local caching of files downloaded from a S3 bucket. Returns an array of parameters or FALSE on fail.

Uploading a New Resource

Once each file has been uploaded to the local RS server via *../pages/upload_plupload.php* and the *upload_file, create_previews,* and the *create_previews_using_im* functions (*../include/image_processing.php*), the original file is uploaded to a S3 bucket and the filestore original file is deleted and replaced with a zero-byte file with the same name as the deleted original file.

Downloading a Resource

On resource download (*../pages/download.php*), if using S3 storage, the object is downloaded to the same temp folder as normal filestore downloads, so metadata can be written using Exiftool.

Deleting a Resource

On resource delete (*../pages/delete.php*), files in the filestore are deleted as normal, then objects in S3 storage associated with that resource are deleted. A listing of files in the filestore and objects in S3 storage are shown to the user before deleting. This listing can also be enabled when not using S3 storage and only filestore files are shown by setting *$show_files_delete = true* in *../include/config.php*.

Replacing an Original Resource File

On replacing an original resource file, the original S3 object is deleted and the new file is uploaded after all of the new preview generation in complete.

Recreating Previews

On preview recreate for S3 stored original files, the original file is downloaded to the temp folder, in a similar manner as copying the file from the filestore location. The previews are recreated and the S3 downloaded file in the temp folder is deleted.

Bulk Download Plugin

The bulk_download plugin enables the download of many, large original files from either the normal filestore or those in S3 storage and replaces the existing shopping cart (E-commerce) bar at the bottom of the page. This plugin is intended for public, non-logged users, where *$anonymous_login="guest"* is set. Resources are added to the download bar as previous functionality and resources added can be from any collection. The user will need to install a Python-based executable on their local computer, using the link in the Download Bar (these files are also in the *../plugins/bulk_download/ python_executable* folder). Once the user has added all of

the files they want to download, they will then click the Download button and receive a ResourceSpace Download configuration text file (.rsd).

When the Python executable is launched, the user selects the downloaded RSD file and a folder on their local computer they want files to be downloaded to. Clicking the Download button in the Python downloader will start the download of each file listed in the RSD file, one file at a time. If there are problems with the download, the process can be restarted, and files already downloaded will be skipped. The user also has the option of having the files downloaded to subfolders named by the collections they are in.

This process was developed to enable downloading many, fairly large files (>100MB), without creating large ZIP or TAR files that increase the filestore size significantly and reduce processing overhead on the server. As the plugin was developed for a project where the original resource files are already internally lossless compressed, ZIP and GZ/TAR packing will not reduce download sizes. The executable was written in Python, so it could easily be compiled for use in various operating systems. Python tkinter was used for the GUI and while not as modern an interface as possible, made it easy to create. The executable provided do not require a local installation of Python to be as portable as possible. The Python source code is also provided, so future versions and/or compilation to other OSs can be easily created.

On configuring the plugin, two options for the RSD configuration file are possible: 1) a standard mode where the anonymous guest user's API key is present in the configuration file and 2) a high-security mode where the API key is not added to the file, but the pre-generated and -signed resource API queries are added to the file.

Some additional testing and development of this plugin is on-going and will be competed shortly.

Existing ResourceSpace Code Cleanup

As part of the integration of object-based, S3 storage into the existing RS code, I cleaned up the existing code to make it easier to understand and conform to the ResourceSpace coding standards described at https://www.resourcespace.com/knowledge-base/developers/coding_standards, so the S3 integration could be relatively seamless and future maintenance easier. Specifically, the following changes were generally made:

- Four spaces used for indent, in place of a tab.
- Line length generally limited to 120 characters.
- Curly braces used for all control structures.
- Function calls formatted as described on the referenced web page, Function Calls section.
- MySQL statement capitalization (SELECT, DELETE, INSERT INTO, AND, ORDER BY, etc.).
- Readability of code blocks.
- Proper line indentation.

In the spirit of Montala's environmental policy (https://www.montala.com/environmental_policy) on energy and water, I converted double quotes (") to single quotes (') whenever possible, as PHP performs a check on double-quoted text for variables each time they are encountered. Where possible, variable post-incrementing ($i++ or $i--) was changed to pre-incrementing (++$i or --$i), as PHP creates a temporary variable and uses a extra processing step with pre-incrementing (extra RAM memory and processor usage). In addition, where PHP for loops used a count function inside the declaration, I created a new $xxxx_count variable just above the for loop, so the count function only gets called once, instead of the number of times the loop continues.

This leads to unnecessary processor overhead and environmental impact (increased energy use and subsequent emissions). While likely minor on a single machine, over thousands of ResourceSpace installations worldwide, those small, incremental impacts quickly add up, particularly for heavily-used servers. It also made the code execute a bit faster to help offset the small delay in accessing S3 storage objects. It should help those using multiple, virtualized machines running on limited resource server blades and similar virtualized environments.

Reviewing Code Changes (for Dan and other Montala staff)

Since there are quite a few changes related to implementing object-based storage in the ResourceSpace code and reviewing using each SVN revision will likely be cumbersome, I suggest having one window (say on the left monitor) show the existing trunk file in a code editor and the corresponding file from this branch in a code editor on the right monitor. Then, you can just scroll each window down a bit a visually review the changes and merge in to the trunk as needed.

Branch SVN Revisions (may not be a complete description)

R16204 – Initial on 26/10/2020.

R16205 – Updated the Installation Check and related files (*../pages/check.php*, *../include/config.default.php*, and *../languages/en.php*).

R16207 – Added the AWS PHP SDK (/lib/aws_php_sdk_3.158.6/) and *../include/s3_storage_functions.php* files. Updated *include/config.default.php* and *include/general_functions.php* files.

R16208 – Added *../pages/admin/admin_system_s3.php* and *../pages/admin/admin_system_s3dashboard.php* files. Updated *../pages/admin/admin_home.php*, *../languages/en.php*, and *../include/db.php* files.

R16210 – Updated the Installation Check files.

R16211 – Additional Installation Check file updates.

R16410 – Updated *../include/image_processing.php*, *../include/resource_functions.php*, *../include/resource_functions.php*. *../pages/collection_download.php*, *../include/config.default.php*, *../include/db.php*, *../include/s3_storage_functions.php*, *../languages/en.php*, *../pages/dowload.php*, and *../pages/upload_plupload.php* files.

R16411 – Updated *../include/api_bindings.php* file.

R16423 – Updated *../include/resource_functions.php* (delete_resource and delete_alternative_file functions) file.

R16424 – Added the Bulk Download plugin (*../plugins/bulk_download/*).

R16425 – Updated *../include/resource_functions.php* (replace_resource_file function) file.

R16426 – Updated *../include/resource_functions.php* (repalce_resource_file and save_original_as_alternative functions) file.

R16427 – Updated *../include/resource_functions.php* (repalce_resource_file and save_original_as_alternative functions) file.

R16428 – Updated *../pages/edit.php* and *../include/image_processing.php* (create_previews and create_previews_using_im functions, and GD processing) files.

R16439 – Fix recreate previews and previous WebP image format issues.

R16440 – Added ../pages/tools/filestore_to_s3.php filestore to S3 storage conversion script.

R16441 – Fix in *../include/s3_storage_functions.php* file.

R16442 – Cleaned up *../pages/search.php* file.

R16443 – Additional clean up in *../pages/search.php* file.

R16444 – Cleaned up *../include/config.default.php* and *../include/s3_storage_functions.php* files.

R16479 – Minor clean up in *../include/config.default.php*, *../include/general_functions.php*, *../include/image_processing.php*, *../pages/admin/admin_home.php*, *../pages/collection_download.php*, *../pages/download.php*, and *../pages/edit.php* files.

New ResourceSpace Files

RS_S3_3_Branch_Information.odt
This explanatory document.

lib/aws_php_sdk_3.158.6
Provides library files for the AWS PHP SDK v3.158.6.

include/s3_storage_functions.php
Adds S3-specific storage functions.

pages/admin/admin_system_s3.php
Adds the S3 Installation Check page.

pages/admin/admin_system_s3dashboard.php
Adds the S3 Storage Dashboard page.

pages/tools/filestore_to_s3.php
Adds the *Filestore to S3 Storage* script tool.

plugins/bulk_download/
Adds functionality to enable remote bulk downloads via a user locally-installed Python-based executable.

Modified, Existing ResourceSpace Files

include/api_bindings.php
Lines 819-835: Added *api_s3_original_download($ref, $alternative = '')* function, no other changes.

include/collection_functions.php

Lines 283-389: Cleaned up *add_resource_to_collection* function, no other changes.

include/config.default.php

Lines 178-209: Added new S3 parameters to new SIMPLE STORAGE SERVICE (S3) OBJECT-BASED STORAGE PARAMETERS section.

Lines 211-249: Added a new section and consolidated existing filestore-specific parameters under RESOURCE FILESTORE PARAMETER CONFIGURATION.

Lines 1907-1908: Added *$show_files_delete = false;* parameter.

Line 3157: Added *$show_upload_log_expand = true;* parameter.

include/db.php

Lines 72-79: Added an include for *s3_storage_functions.php* and set *$show_upload_log = true; $show_upload_log_expand = true;* and *$show_files_delete = true;* parameters.

include/general_functions.php

Lines 4175-4216: Added *boolean_convert($input, $type)*, *html_status_code($code, $show_text = false)*, and *get_server_memory_usage()* functions.

Lines 700-745: Cleaned up *formatfilesize* function and added petabyte symbol.

Lines 141-204: Cleaned up *nicedate* function, no other changes.

include/image_processing.php

Lines 14-607: Cleaned up *upload_file* function.

Lines 1127-1447: Cleaned up *create_previews* function, added new input *$s3_recreate = false* parameter and global *$s3_enable* parameter

Lines 1139-1161: New code to download original file from S3 storage to the filestore to allow preview re-creation.

Lines 1438-1443: New code to delete the previously downloaded S3 file and create a new zero-byte placeholder file.

Lines 1450-2307: Cleaned up *create_previews_using_im* function, no other changes.

include/resource_functions.php

Lines 5-426: Cleaned up *get_resource_path* function, no other changes.

Lines 2523- : Cleaned up *delete_resource* function, added global *$s3_enable* and *$sdk* parameters.

Lines 2592-2609: New code to delete the original file(s) in S3 storage.

Lines 4321- : Cleaned up *delete_alternative_file* function, added global *$s3_enable* and *$sdk* parameters.

Lines 4325-4343: New code to delete the alternative file in S3 storage.

Lines 4362-4372: New code to delete the JPG original alternative file in S3 storage.

Lines 4400-4410: New code to delete the MP3 original alternative file in S3 storage.

Lines 6819- : Cleaned up save_original_file_as_alternative function, added global $s3_enable parameter.

Lines 6860-6885: New code incorporating old code to move the old file to the alternative file location. If using S3, check if the original file exists in S3 storage. If so, copy to the new alternative object name and delete the original object.

Lines 6926-6992: Cleaned up *replace_resource_file* function, added global *$s3_enable* parameter.

Lines 6967-6977: New code to upload a file to S3 storage.

Lines 2523-2655: Cleaned up *delete_resource* function, added global *$s3_enable* and *$sdk* parameters.

Lines 2592-2609: New code to delete an existing object in S3 storage.

Lines 4321-4420: Cleaned up *delete_alternative_file* function.

Lines 4333-4343: New code to delete an existing object in S3 storage.

Lines 4362-4372: New code to delete an existing JPG original alternative object in S3 storage.
Lines 4400-4410: New code to delete an existing MP3 original alternative object in S3 storage.
Lines 6819- : Cleaned up *save_original_file_as_alternative* function, added global $s3_enable parameter.
Lines 6860-6885: New code incorporating old code to move the old file to the alternative file location. If using S3, copy the object to the new location and delete the old object.
Lines 6918-6992: Cleaned up *replace_resource_file* function, added *$s3_enable* parameter.
Lines 6967-6977: New code to upload a file to S3 storage.

languages/en.php
Cleaned up file, deleted excess white space, and changed some language definitions with double quotes to single quotes.
Lines 2930-2993: New Simple Storage Service (S3) Storage section with various language definitions to support S3 storage.

pages/admin/admin_home.php
Cleaned up file, deleted excess white space, and fixed indentation.
Lines 93-97: New code to add *S3 Installation Check* and *S3 Storage Dashboard* links.

pages/check.php
Cleaned up file, deleted excess white space, fixed indentation and reordered items into more logical general groupings (ResourceSpace version, server memory, MySQL, Apache, PHP, filestore, S3, ResourceSpace dependencies, and scheduled tasks).

pages/collection_download.php
Cleaned up file, deleted excess white space, and fixed indentation.
Lines 283-293: New code incorporating old code to get the file size.
Line 436: Modified code for S3 storage, added *!$s3_enable &&*
Line 800: Modified code for S3 storage, added *&& !$s3_enable*

pages/collections.php
Line 762: Added hook *<?php hook('addbasketlinks'); ?>*

pages/delete.php
Cleaned up file, deleted excess white space, and fixed indentation.
Lines 131-187: New code to show a list of files in the filestore and in S3 storage to delete before actually deleting. Can enable list if *$s3_enable = false* that just shows filestore files to delete.

pages/download.php
Cleaned up file, deleted excess white space, and fixed indentation.
Lines 8-11: New code for S3, *include s3_storage_functions.php*.
Lines 138-165: New code for downloading an object from S3 storage.
Line 186: Add *&& !$s3_enable*
Lines 352-357: New code to delete the temporary file created from downloading a S3 object.
Cleaned up file, deleted excess white space, and fixed indentation.

pages/edit.php
Minor file cleanup.
Line 7: Added *global $s3_enable;*
Lines 331-337: New code for re-creating previews for originals in S3 storage.

Lines 769-828: Cleaned up code.

Lines 791-796: New code for re-creating previews for originals in S3 storage.

Lines 1149-1165: Cleaned up code.

Lines 1153-1162: New code incorporating old code to show file size top deal with S3 storage.

pages/purchase.php

Minor file cleanup.

Lines 5-6 and 222: Added hook *if(hook(‘replacepurchase’)*)

pages/search.php

Cleaned up file, deleted excess white space, and fixed indentation; no other changes.