Automatically removing old files from a remote archive

Without a cleanup policy, a remote archive grows indefinitely. Storage costs accumulate quietly, and eventually someone has to do a manual purge under time pressure. This article shows how to write a scheduled AFT! script that connects to an S3 bucket, inspects the modification time of every file in an archive directory, and deletes anything older than a configurable retention threshold. The same pattern works for any remote service that AFT! supports; just swap `S3Client` for `SftpClient`, `AzureClient`, or `GCSClient`.

How It Works

The script connects to S3 with credentials from the Secrets store, lists the archive directory using `LIST_FILES` (which excludes directory prefixes), and iterates over the results. For each file it computes the age in milliseconds by comparing `f.TimeStamp` against the current time. Files within the retention window are skipped; files that have exceeded it are deleted.

```
// Remote archive cleanup
// Connects to an S3 bucket and deletes files in the archive directory that are
// older than the configured retention period. Intended to run as a scheduled
// cron job, for example once a day after the nightly backup completes.

var retentionDays = 90;        // delete files older than this many days
var archiveDir    = "/archive"; // remote directory to clean up

var cli = new S3Client();
cli.Region    = GetSecret("s3-region");
cli.AccessKey = GetSecret("s3-access-key");
cli.SecretKey = GetSecret("s3-secret-key");
cli.Bucket    = GetSecret("s3-bucket-name");

if (!cli.Connect()) {
   Log.Error("failed to connect to S3");
   Exit(1);
}
```

```
try {
    // LIST_FILES requests file entries only; directory prefixes are excluded.
    var files = cli.ListDir(archiveDir, LIST_FILES);
    Log.Info("scanning " + files.length + " files in " + archiveDir);

    var nowMs       = new Date().getTime();
    var retentionMs = retentionDays * 24 * 60 * 60 * 1000;
    var deleted     = 0;
    var retained    = 0;

    for (var i = 0; i < files.length; i++) {
        var f     = files[i];
        var ageMs = nowMs - new Date(f.TimeStamp).getTime();

        if (ageMs < retentionMs) {
            retained++;
            continue;
        }

        // Build the full remote path; ListDir returns bare filenames when
        // operating on a specific directory.
        var fullPath = archiveDir + "/" + f.Name;
        var ageDays  = Math.floor(ageMs / (24 * 60 * 60 * 1000));

        if (cli.Delete(fullPath)) {
            Log.Info("deleted: " + f.Name + " (age: " + ageDays + " days)");
            deleted++;
        } else {
            Log.Error("failed to delete: " + fullPath);
        }
    }

    Log.Info("cleanup complete: " + deleted + " deleted, " + retained + " retained");
} finally {
    cli.Close();
}
```

`new Date(f.TimeStamp).getTime()`: `f.TimeStamp` is the modification time of the S3 object as reported by the API. Passing it directly to the `Date` constructor produces a JavaScript `Date` object you can compare arithmetically. The difference in milliseconds divided by the appropriate factor gives you age in seconds, minutes, hours, or days.

`LIST_FILES`: passing this constant to `ListDir` tells AFT! to return only file objects. S3 "directories" are just key prefixes, not real objects, and they have no meaningful timestamp. Excluding them avoids a spurious deletion attempt.

The `try/finally` block ensures `cli.Close()` is always called even if the body throws an exception.

Running across subdirectories

Replace `ListDir` with `ListDirR` to scan the archive recursively:

```
var files = cli.ListDirR(archiveDir, LIST_FILES);
```

`ListDirR` returns the relative path from the listing root in `f.Name`, so the `fullPath` construction and `cli.Delete(fullPath)` calls work without any other changes.

Testing before you delete

Add a dry-run flag when you first deploy this to a production environment:

```
var dryRun = true;   // set to false to enable actual deletion


if (dryRun) {
   Log.Info("dry run: would delete " + fullPath + " (age: " + ageDays + " days)");
} else {
   if (cli.Delete(fullPath)) { deleted++; }
}
```

Run it once with `dryRun = true`, review the job log, and flip the flag only when you are satisfied the retention threshold is correct.

Using MinIO or another S3-compatible service

Add `Endpoint` and `UsePathStyle` before calling `Connect`:

```
cli.Endpoint     = "https://minio.internal.example.com";
cli.UsePathStyle = true;
```

Everything else stays the same.

Revision #1
Created 12 April 2026 14:53:15 by DevTeam
Updated 12 April 2026 15:02:54 by DevTeam