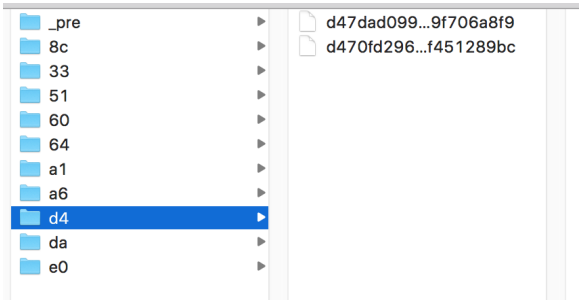
Checksum-Based Storage

Overview

Artifactory uniquely stores artifacts using checksum-based storage.

A file that is uploaded to Artifactory, first has its SHA1 checksum calculated, and is then renamed to its checksum. It is then hosted in the configured filestore in a directory structure made up of the first two characters of the checksum. For example, a file whose checksum is "ac3f5e56..." would be stored in directory "ac"; a file whose checksum is "dfe12a4b..." would be stored in directory "df" and so forth. The example below shows the "d4" directory that contains two files whose checksum begins with "d4".



In parallel, Artifactory creates a database entry mapping the file's checksum to the path it was uploaded to in a repository. This way of storing binaries optimizes many operations in Artifactory since they are implemented through simple database transactions rather than actually manipulating files.

Page Contents

Deduplication

Artifactory stores any binary file only once. This is what we call "once and once only storage". First time a file is uploaded, Artifactory runs the required checksum calculations when storing the file, however, if the file is uploaded again (to a different location, for example), the upload is implemented as a simple database transaction that creates another record mapping the file's checksum to its new location. There is no need to actually store the file again in storage. No matter how many times a file is uploaded, the filestore only hosts a single copy of the file.

Copying and Moving Files

Copying and moving a file is implemented by simply adding and removing database references and, correspondingly, performance of these actions is that of a database transaction.

Deleting Files

Deleting a file is also a simple database transaction in which the corresponding database record is deleted. The file itself is not directly deleted, even if the last database entry pointing to it is removed. So-called "orphaned" files are removed in the background by Artifactory's garbage collection processes.

Upload, download and replication

Before moving files from one location to another, Artifactory sends checksum headers. If the files already exist in the destination, they are not transferred even if they exist under a different path.

Filesystem Performance

Filesystem performance is greatly improved because actions on the filestore are implemented as database transactions, so there is never any need to do a write-lock on the filesystem.

Checksum Search

Searching for a file by its checksum is extremely fast since Artifactory is actually searching through the database for the specified checksum.

Flexible Layout

Since the database is a layer of indirection between the filestore and the displayed layout, any layout can be supported, whether for one of the standard packaging formats such as Maven1, Maven2, npm, NuGet etc. or for any custom layout.

SHA-256 Support

From version 5.5, Artifactory natively supports SHA-256. An artifact's SHA-256 checksum is calculated when it is deployed to Artifactory, and is maintained in persistent storage as part of the database. The Set Item SHA256 Checksum REST API endpoint (which sets an artifact's SHA-256 checksum as one of its properties) is still supported for backward compatibility, however, this endpoint will eventually be deprecated.

Artifactory's support for SHA-256 checksums is fully-featured and is evident in several ways:

- They can be used in AQL queries, and are returned in corresponding responses
- They are included as download header information
- They can be used in the Deploy Artifact and Deploy Artifact by Checksum REST API endpoints.
- They are included when downloading a folder
- They are displayed in the General Information tab of the Artifact Repository Browser
- The can be used in a variety of REST API endpoints used for search

After upgrading to version 5.5 (or above), Artifactory will be fully capable of utilizing an artifact's SHA-256 checksum for any of the features mentioned above.

Making full use of Artifactory's native support for SHA256

New artifacts that are uploaded will automatically have their SHA-256 checksum calculated, however, artifacts that were already hosted in Artifactory prior to the upgrade will not have their SHA-256 checksum in the database yet.

To make full use of Artifactory's SHA-256 capabilities, you need to run a process that migrates Artifactory's database making sure that the record for each artifact includes its SHA-256 checksum.

Migrating the Database to Include SHA-256

Migrating the database may be a resource intensive operation

Depending on the size of your database, this process may be resource intensive. To mitigate the possible load on your system, you may configure the process using several system properties listed below or the REST APIs. We strongly recommend reading through the entire process migration process to ensure the optimal configuration for your system.

The migration is configured through a **set of properties in Artifactory's *`artifactory.system.properties`* file** as described below, or using the ***Start SHA256 Migration Task*** and ***Stop SHA256 Migration Task*** REST API endpoints, and essentially, does the following:

- Search for all database records that don't have a SHA-256 value.
- For each such record, find all others in the database with the same SHA1 checksum value
  - If any of them have the SHA-256 calculated already, use that to update all the others
  - If none of them have the SHA-256 calculated already, calculate it and then use that to update all others
- If there are no other records with the same SHA1 value, calculate the SHA-256

First run garbage collection to optimize the migration process

The migration process is complete once all database entries have been populated with SHA-256 values. Since your database may contain entries for artifacts that have been deleted, but have not yet been physically removed by garbage collection, we strongly recommend manually invoking garbage collection before invoking the database migration. Removing deleted artifacts can greatly improve performance and total run time of the migration by reducing the number of downloads it generates.

Configuring the Migration Process

The migration process may be configured through the following system properties, or using the ***Start SHA256 Migration Task*** and ***Stop SHA256 Migration Task*** REST API endpoints

By default, the migration will run on the primary node, however, using the `forceRunOnNodeId` property described below, you may configure it to run on a secondary node.

Property Name	Function
`artifactory.sha2.migration.job.enabled`	[Default: false] When true, the process that migrates the database to include SHA-256 checksum for all artifacts will be invoked when the node is restarted.

`artifactory.sha2.migration.job.forceRunOnNodeId`	[Default: null] Only valid for an HA installation. By default, the migration process runs on the primary node. To run the process on any other node, set this value to the corresponding node's ID (as specified in the **node.id** property in the nodes *$ARTIFACTORY_HOME/etc/ha-node.properties* file). **Running the migration on a dedicated node** This gives you the option of dedicating a specific node to run the migration and allocating extra resources allowing it to finish the process faster. **Set the property on both the primary and the corresponding secondary node** To run the migration process on a secondary node, you need to **set this property on BOTH the primary node and the corresponding secondary node**. Artifactory will still only run the process on the corresponding secondary node.
`artifactory.migration.job.dbQueryLimit`	[Default: 100] Specifies the number of rows that should be retrieved each time the migration job queries the database for entries that are missing SHA-256 values.
`artifactory.migration.job.batchSize`	[Default: 10] Artifacts are updated concurrently in batches with new SHA-256 values and then a sleep cycle is initiated. This property specifies the number of artifacts in each batch.
`artifactory.sha2.migration.job.queue.workers`	[Default: 2] Specifies the number of concurrent threads that should execute actual artifact updates. Each concurrent artifact update may incur a download in order to calculate its SHA-256 checksum. However, the artifact will only be downloaded once, first time a database entry is found for it with no SHA-256 value. Subsequent database entries for the same artifact (which therefore have the same SHA1 value) will reuse the SHA-256 value that was already calculated.
`artifactory.migration.job.sleepIntervalMillis`	[Default: 5000 milliseconds] Specifies the duration of the sleep cycle which is initiated after each batch of updates.

A sample snippet you can paste into your artifactory.system.properties is below, adjust the number of workers as appropriate based on I/O and CPU utilization:

Example artifactory.system.properties snippet

```
##SHA2 Migration block
artifactory.sha2.migration.job.enabled=true
artifactory.sha2.migration.job.queue.workers=5
```

Restart required

For changes to the migration configuration to take effect, you need to restart the instance (or node in the case of an HA installation) that will run it. The default values specified above are set to keep your system performing optimally during the migration process. To speed up the migration process, you may tweak these values (keeping hardware limits in mind), however that may come at a cost of system performance.

Monitoring the Migration Process

Depending on the size of your storage, and the migration parameters you have configured, the migration process may take a long time. To enable easy monitoring of the process, status and error messages are printed into a dedicated log file, *`ARTIFACTORY_HOME/logs/sha256_migration.log`*. In addition, some messages (process initiation, startup errors) are also logged in the *`ARTIFACTORY_HOME/logs/artifactory.log`* file.

Post SHA256 Migration

While it's not mandatory, it is recommended to remove all SHA256 related system properties after the migration has completed and perform a restart.