To **automatically delete snapshot repositories** in **Sonatype Nexus Repository Manager** once a **release version is created**, you can achieve this using a combination of scripting and built-in capabilities. Here are a few reliable approaches:

**✅ Recommended Approach: Use a Nexus Script + Scheduled Task**

**1. Use the Nexus Scripting API**

Nexus 3 supports Groovy scripts via the REST API for automation. You can write a script to delete snapshot components or repositories based on version or age.

**🔧 Example Groovy Script to Delete SNAPSHOT Artifacts:**

groovy

CopyEdit

import org.sonatype.nexus.repository.storage.Query

def SNAPSHOT\_REGEX = /.\*-SNAPSHOT$/

def components = repository.browse().each { component ->

if (component.version ==~ SNAPSHOT\_REGEX) {

log.info("Deleting component: ${component.group}:${component.name}:${component.version}")

repository.deleteComponent(component)

}

}

You can enhance this to:

* Check if a **corresponding release version exists**.
* Then, **delete the SNAPSHOT version only if the release is found**.

**2. Upload Script via REST API**

Use curl or Postman to upload and run this script:

bash

CopyEdit

curl -u admin:admin123 -X POST -H "Content-Type: application/json" \

http://localhost:8081/service/rest/v1/script \

-d @delete\_snapshots.json

Example delete\_snapshots.json:

json

CopyEdit

{

"name": "deleteOldSnapshots",

"type": "groovy",

"content": "/\* your groovy code here \*/"

}

**3. Create a Scheduled Task in Nexus UI**

* Go to **Administration → Tasks → Create task**
* Type: Run script
* Script name: deleteOldSnapshots
* Schedule it (e.g., every day at midnight)

**🔄 Alternative Approach: CI/CD Integration**

If you're using Maven or Gradle in a CI/CD pipeline:

1. **Trigger a Nexus cleanup API** after the release step.
2. Use a plugin like the **Nexus REST API** or **Nexus Staging Maven Plugin** to clean up old SNAPSHOTs.

**🧩 Optional: Use Community Plugins or Third-Party Tools**

* **nexus-cleanup-plugin** (for Nexus 2.x): Works only with older versions.
* **nexus3-cleanup** (unofficial, GitHub): Custom tools that allow more advanced retention policies.

**✅ Bonus: Use Nexus "Cleanup Policies" (if you're on Nexus Pro)**

If you have **Nexus Repository Pro**, this is the easiest and safest way:

* Go to **"Cleanup Policies"**.
* Create a policy to:
  + Match snapshot artifacts
  + Retain only latest X versions
  + Automatically delete older ones
* Apply it to snapshot repositories.

**Summary**

| **Method** | **Tools Needed** | **Notes** |
| --- | --- | --- |
| Groovy Script + Scheduled Task | Nexus Script API | Most flexible |
| CI/CD Integration | Maven/Gradle + API calls | Integrates well with your pipeline |
| Cleanup Policies (Nexus Pro) | Nexus Pro License | Easiest to maintain |

Would you like a full working Groovy script that checks for a corresponding release version before deleting snapshots?

1/2

4o

**You said:**

write complete gradle plugin to cleanup snapshot version once release version is created

**ChatGPT said:**

Here's a **complete custom Gradle plugin** written in Kotlin (Kotlin DSL) that will:

* Connect to a **Nexus 3+ repository via its REST API**.
* Check for both **SNAPSHOT and RELEASE versions** of a given artifact.
* If a RELEASE version exists, it will delete the corresponding SNAPSHOT via the Nexus REST API.

**⚠️ Requirements**

* Nexus 3.x with REST API enabled.
* Your artifacts are stored in a **hosted Maven** repository.
* You have authentication credentials.
* Nexus base URL and repository name are known.

**🛠 Full Gradle Plugin (Kotlin DSL)**

**✅ build.gradle.kts for the plugin project:**

kotlin

CopyEdit

plugins {

`kotlin-dsl`

`java-gradle-plugin`

}

repositories {

mavenCentral()

}

dependencies {

implementation("com.fasterxml.jackson.module:jackson-module-kotlin:2.16.1")

implementation("com.squareup.okhttp3:okhttp:4.12.0")

}

**✅ src/main/kotlin/NexusSnapshotCleanerPlugin.kt**

kotlin

CopyEdit

import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper

import com.fasterxml.jackson.module.kotlin.readValue

import okhttp3.Credentials

import okhttp3.OkHttpClient

import okhttp3.Request

import okhttp3.RequestBody.Companion.toRequestBody

import okhttp3.MediaType.Companion.toMediaType

import org.gradle.api.Plugin

import org.gradle.api.Project

class NexusSnapshotCleanerPlugin : Plugin<Project> {

override fun apply(project: Project) {

project.tasks.register("cleanupSnapshots") {

group = "nexus"

description = "Deletes snapshot versions from Nexus if release version exists"

doLast {

val nexusUrl = project.findProperty("nexusUrl")?.toString() ?: error("Missing nexusUrl")

val repository = project.findProperty("repository")?.toString() ?: error("Missing repository")

val groupId = project.findProperty("groupId")?.toString() ?: error("Missing groupId")

val artifactId = project.findProperty("artifactId")?.toString() ?: error("Missing artifactId")

val username = project.findProperty("nexusUser")?.toString() ?: error("Missing nexusUser")

val password = project.findProperty("nexusPassword")?.toString() ?: error("Missing nexusPassword")

val client = OkHttpClient()

val mapper = jacksonObjectMapper()

val auth = Credentials.basic(username, password)

val queryUrl = "$nexusUrl/service/rest/v1/components?repository=$repository&group=$groupId&name=$artifactId"

val request = Request.Builder()

.url(queryUrl)

.addHeader("Authorization", auth)

.build()

val response = client.newCall(request).execute()

if (!response.isSuccessful) error("Failed to fetch components: ${response.message}")

val json = response.body?.string() ?: ""

val components: Map<String, Any> = mapper.readValue(json)

val items = components["items"] as List<Map<String, Any>>

val snapshotVersions = items.filter { (it["version"] as String).endsWith("-SNAPSHOT") }

val releaseVersions = items.filterNot { (it["version"] as String).endsWith("-SNAPSHOT") }

val releaseSet = releaseVersions.map { it["version"] as String }.toSet()

snapshotVersions.forEach { snapshot ->

val snapVersion = snapshot["version"] as String

val releaseVersion = snapVersion.removeSuffix("-SNAPSHOT")

if (releaseSet.contains(releaseVersion)) {

val componentUrl = snapshot["url"] as String

val deleteReq = Request.Builder()

.url(componentUrl)

.delete()

.addHeader("Authorization", auth)

.build()

val delResp = client.newCall(deleteReq).execute()

if (delResp.isSuccessful) {

println("Deleted SNAPSHOT: $snapVersion")

} else {

println("Failed to delete $snapVersion: ${delResp.message}")

}

}

}

}

}

}

}

**🔧 Example gradle.properties in Your Project**

properties

CopyEdit

nexusUrl=https://nexus.example.com

repository=my-maven-releases

groupId=com.example

artifactId=my-library

nexusUser=admin

nexusPassword=admin123

**🚀 How to Use It**

1. Add this plugin to your buildSrc or publish it to your internal plugin repository.
2. Apply the plugin to your build:

kotlin

CopyEdit

plugins {

id("your.custom.nexus-cleaner")

}

1. Run:

bash

CopyEdit

./gradlew cleanupSnapshots

**📌 Notes**

* This only deletes components from the given repository.
* It assumes SNAPSHOT and release versions follow the same versioning except for the -SNAPSHOT suffix.
* Add logging, retries, or smarter logic as needed.

Would you like a **Groovy DSL version** of this plugin or a **Java version** for broader compatibility?

4o

**You said:**

Java version

**ChatGPT said:**

Here is the **Java version** of the custom Gradle plugin that deletes **SNAPSHOT versions** from a **Nexus 3 repository** if a **corresponding release version exists**.

**✅ What It Does**

* Connects to Nexus via its REST API.
* Lists all components in a given repository.
* Finds \*-SNAPSHOT versions and checks if the corresponding release version exists.
* Deletes the snapshot if a release exists.

**🧱 Project Structure**

You’ll place the plugin in a **separate Gradle module** (e.g., buildSrc) or a standalone project.

css

CopyEdit

nexus-snapshot-cleaner/

├── build.gradle

└── src/

└── main/

└── java/

└── com/

└── example/

└── NexusSnapshotCleanerPlugin.java

**🛠 Step 1: build.gradle (Groovy DSL)**

groovy

CopyEdit

plugins {

id 'java-gradle-plugin'

}

repositories {

mavenCentral()

}

dependencies {

implementation 'com.squareup.okhttp3:okhttp:4.12.0'

implementation 'com.fasterxml.jackson.core:jackson-databind:2.16.1'

implementation 'com.fasterxml.jackson.core:jackson-core:2.16.1'

implementation 'com.fasterxml.jackson.core:jackson-annotations:2.16.1'

}

**🧠 Step 2: NexusSnapshotCleanerPlugin.java**

java

CopyEdit

package com.example;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import okhttp3.\*;

import org.gradle.api.Plugin;

import org.gradle.api.Project;

import org.gradle.api.tasks.TaskAction;

import java.io.IOException;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

public class NexusSnapshotCleanerPlugin implements Plugin<Project> {

@Override

public void apply(Project project) {

project.getTasks().register("cleanupSnapshots", task -> {

task.setGroup("nexus");

task.setDescription("Deletes snapshot versions from Nexus if a release version exists");

task.doLast(t -> {

try {

runCleanup(project);

} catch (IOException e) {

throw new RuntimeException("Failed to clean up Nexus snapshots", e);

}

});

});

}

private void runCleanup(Project project) throws IOException {

String nexusUrl = (String) project.findProperty("nexusUrl");

String repository = (String) project.findProperty("repository");

String groupId = (String) project.findProperty("groupId");

String artifactId = (String) project.findProperty("artifactId");

String username = (String) project.findProperty("nexusUser");

String password = (String) project.findProperty("nexusPassword");

if (nexusUrl == null || repository == null || groupId == null || artifactId == null || username == null || password == null) {

throw new IllegalArgumentException("Missing required properties");

}

OkHttpClient client = new OkHttpClient();

ObjectMapper mapper = new ObjectMapper();

String apiUrl = nexusUrl + "/service/rest/v1/components?repository=" + repository +

"&group=" + groupId + "&name=" + artifactId;

Request request = new Request.Builder()

.url(apiUrl)

.header("Authorization", Credentials.basic(username, password))

.build();

Response response = client.newCall(request).execute();

if (!response.isSuccessful()) {

throw new IOException("Failed to fetch components: " + response.message());

}

JsonNode root = mapper.readTree(response.body().string());

JsonNode items = root.get("items");

Set<String> releaseVersions = new HashSet<>();

Set<JsonNode> snapshotComponents = new HashSet<>();

for (JsonNode item : items) {

String version = item.get("version").asText();

if (version.endsWith("-SNAPSHOT")) {

snapshotComponents.add(item);

} else {

releaseVersions.add(version);

}

}

for (JsonNode snapshot : snapshotComponents) {

String snapshotVersion = snapshot.get("version").asText();

String releaseVersion = snapshotVersion.replace("-SNAPSHOT", "");

if (releaseVersions.contains(releaseVersion)) {

String componentUrl = snapshot.get("url").asText();

Request deleteRequest = new Request.Builder()

.url(componentUrl)

.delete()

.header("Authorization", Credentials.basic(username, password))

.build();

Response deleteResponse = client.newCall(deleteRequest).execute();

if (deleteResponse.isSuccessful()) {

System.out.println("Deleted SNAPSHOT version: " + snapshotVersion);

} else {

System.out.println("Failed to delete " + snapshotVersion + ": " + deleteResponse.message());

}

}

}

}

}

**🧪 Step 3: Usage Example**

Add the following properties to your gradle.properties:

properties

CopyEdit

nexusUrl=https://nexus.example.com

repository=my-maven-releases

groupId=com.example

artifactId=my-library

nexusUser=admin

nexusPassword=admin123

Then run:

bash

CopyEdit

./gradlew cleanupSnapshots

**✅ Optional Enhancements**

* Add pagination to handle >100 components (Nexus API default page size).
* Handle nested components or subgroups.
* Add retries and better error handling.

Would you like help adding **pagination** or converting this into a **standalone CLI tool** instead of a Gradle plugin?

4o

**You said:**

Groovy Script to Delete SNAPSHOT Artifacts only when release version is exists so include code to Check if a corresponding release version exists Then, delete the SNAPSHOT version only if the release is found.

**ChatGPT said:**

Here is a **Groovy script** for **Sonatype Nexus Repository Manager 3.x** that:

* **Finds SNAPSHOT artifacts** in a given repository.
* **Checks if a corresponding release version exists** (i.e., same version string but without -SNAPSHOT).
* **Deletes the SNAPSHOT** component **only if the release version is found**.

**✅ Nexus Groovy Script: Delete SNAPSHOT if Release Exists**

groovy

CopyEdit

import org.sonatype.nexus.repository.storage.Query

import org.sonatype.nexus.repository.storage.StorageFacet

import org.sonatype.nexus.repository.Repository

import org.sonatype.nexus.repository.storage.Component

import org.sonatype.nexus.repository.storage.Asset

// CONFIGURATION

def repositoryName = 'maven-releases' // Change to your repository name

// FETCH REPOSITORY

def repo = repository.repositoryManager.get(repositoryName)

def tx = repo.facet(StorageFacet).txSupplier().get()

tx.begin()

try {

def components = tx.findComponents(Query.builder().where('version').like('%-SNAPSHOT').build(), [repo])

def allVersionsByGAV = [:].withDefault { [] }

// Organize components by G:A (artifact coordinates)

components.each { component ->

def key = "${component.group()}:${component.name()}"

allVersionsByGAV[key] << component

}

// Iterate through each artifact group

allVersionsByGAV.each { gav, snapshots ->

def (group, name) = gav.split(':')

// Fetch all versions for this artifact

def allVersions = tx.findComponents(

Query.builder().where('group').eq(group).and('name').eq(name).build(), [repo]

)\*.version

snapshots.each { snapshot ->

def snapshotVersion = snapshot.version

def releaseVersion = snapshotVersion.replace("-SNAPSHOT", "")

if (allVersions.contains(releaseVersion)) {

log.info("Deleting SNAPSHOT version ${snapshotVersion} for ${group}:${name} because release ${releaseVersion} exists.")

tx.deleteComponent(snapshot)

} else {

log.info("Keeping SNAPSHOT version ${snapshotVersion} for ${group}:${name} — no corresponding release version found.")

}

}

}

tx.commit()

} catch (e) {

log.error("Error while deleting SNAPSHOTs: ${e.message}", e)

tx.rollback()

} finally {

tx.close()

}

**📌 How to Use This Script**

1. Go to **Nexus UI** → **Admin** → **System** → **API** → **Scripts**
2. Create a new script:
   * Name: deleteSnapshotsIfReleaseExists
   * Type: Groovy
   * Content: *(paste the above script)*
3. Create a **task** under **Admin → Tasks**:
   * Type: Run script
   * Script: deleteSnapshotsIfReleaseExists
   * Schedule: e.g. **daily at midnight**

**🔧 Parameters You Might Adjust**

* repositoryName = 'maven-releases': set this to the repository you want to clean up.
* Add filters for group, name, etc., if you want more control.

Would you like a version that works **across all Maven repositories**, or filters by **project group ID**?

4o

**You said:**

yes

**ChatGPT said:**

Great! Below is an **enhanced version of the Groovy script** for **Nexus Repository Manager 3.x** that:

* Works across **all hosted Maven repositories** (you don’t need to hard-code a single repo).
* Optionally filters by a **specific groupId prefix** (e.g., com.example) if you want.
* Deletes **SNAPSHOT components only when the corresponding RELEASE version exists**.

**✅ Nexus Groovy Script (All Repos, Optional Group Filter)**

groovy

CopyEdit

import org.sonatype.nexus.repository.storage.Query

import org.sonatype.nexus.repository.storage.StorageFacet

import org.sonatype.nexus.repository.Repository

import org.sonatype.nexus.repository.storage.Component

// === CONFIGURATION ===

def groupIdPrefixFilter = 'com.example' // Set to null or '' to disable group filter

def snapshotSuffix = '-SNAPSHOT'

// === EXECUTION ===

repository.repositoryManager.browse().findAll {

it.format == 'maven2' && it.type.name() == 'hosted'

}.each { repo ->

log.info("Processing repository: ${repo.name}")

def tx = repo.facet(StorageFacet).txSupplier().get()

tx.begin()

try {

def snapshotComponents = tx.findComponents(

Query.builder().where('version').like("%${snapshotSuffix}").build(), [repo]

)

def groupedByGA = [:].withDefault { [] }

snapshotComponents.each { component ->

if (!groupIdPrefixFilter || component.group()?.startsWith(groupIdPrefixFilter)) {

def key = "${component.group()}:${component.name()}"

groupedByGA[key] << component

}

}

groupedByGA.each { gaKey, snapshots ->

def (group, name) = gaKey.split(':')

def allComponents = tx.findComponents(

Query.builder().where('group').eq(group).and('name').eq(name).build(), [repo]

)

def allVersions = allComponents\*.version.toSet()

snapshots.each { snapshot ->

def snapshotVersion = snapshot.version

def releaseVersion = snapshotVersion.replace(snapshotSuffix, '')

if (allVersions.contains(releaseVersion)) {

log.info("Deleting ${group}:${name}:${snapshotVersion} from ${repo.name} — release ${releaseVersion} exists.")

tx.deleteComponent(snapshot)

} else {

log.debug("Skipping ${group}:${name}:${snapshotVersion} — no release version found.")

}

}

}

tx.commit()

} catch (e) {

log.error("Error cleaning snapshots in repository ${repo.name}: ${e.message}", e)

tx.rollback()

} finally {

tx.close()

}

}

**🔧 Configuration Options**

| **Option** | **Description** |
| --- | --- |
| groupIdPrefixFilter = 'com.example' | Optional: filters only artifacts from this group ID prefix. Use null or empty string to disable filtering. |
| snapshotSuffix = '-SNAPSHOT' | Allows tweaking the suffix pattern. Normally -SNAPSHOT. |

**✅ How to Use in Nexus**

1. Go to **Administration → System → API → Scripts**
2. Create a new script:
   * **Name**: deleteSnapshotsWithReleases
   * **Type**: Groovy
   * **Content**: *(paste the script above)*
3. Save and then go to **Tasks → Create Task**
   * **Type**: Run Script
   * **Script**: deleteSnapshotsWithReleases
   * **Schedule**: e.g., Daily or Hourly

Would you like a version that logs deletions to a remote endpoint or emails a report afterward?

4o

**You said:**

yes

**ChatGPT said:**

Here's an enhanced version of the **Nexus 3 Groovy cleanup script** that:

**✅ Deletes SNAPSHOTs when matching RELEASE versions exist**

**✅ Filters by groupId prefix (optional)**

**✅ Logs deletions to a remote HTTP endpoint**

**🚀 Features**

* **Supports all hosted Maven repositories**
* **Optional group filter** (e.g., "com.example")
* **Remote logging** of deleted artifact info (e.g., to Splunk, ELK, or custom API)

**🔧 Configuration**

Edit these values in the top of the script:

groovy

CopyEdit

def groupIdPrefixFilter = 'com.example' // Set to null or '' to disable filter

def remoteLogUrl = 'https://your-logging-endpoint.example.com/snapshot-deletions' // Logging API

def remoteLogAuthToken = 'Bearer your\_token\_here' // Optional auth

def snapshotSuffix = '-SNAPSHOT'

**📜 Final Script**

groovy

CopyEdit

import org.sonatype.nexus.repository.storage.Query

import org.sonatype.nexus.repository.storage.StorageFacet

import org.sonatype.nexus.repository.Repository

import org.sonatype.nexus.repository.storage.Component

import groovy.json.JsonOutput

import java.net.HttpURLConnection

import java.net.URL

// === CONFIGURATION ===

def groupIdPrefixFilter = 'com.example' // Set to null or '' to disable filtering

def snapshotSuffix = '-SNAPSHOT'

// Remote logging endpoint

def remoteLogUrl = 'https://your-logging-endpoint.example.com/snapshot-deletions'

def remoteLogAuthToken = 'Bearer your\_token\_here'

// === FUNCTIONS ===

def sendLog(String repoName, String group, String name, String version) {

def payload = [

repository: repoName,

group : group,

artifact : name,

version : version,

timestamp : new Date().toString()

]

def json = JsonOutput.toJson(payload)

try {

def url = new URL(remoteLogUrl)

def conn = url.openConnection() as HttpURLConnection

conn.setRequestMethod("POST")

conn.setDoOutput(true)

conn.setRequestProperty("Content-Type", "application/json")

if (remoteLogAuthToken) {

conn.setRequestProperty("Authorization", remoteLogAuthToken)

}

conn.outputStream.withWriter("UTF-8") { writer ->

writer.write(json)

}

def responseCode = conn.responseCode

if (responseCode >= 200 && responseCode < 300) {

log.info("Logged deletion to remote endpoint: ${group}:${name}:${version}")

} else {

log.warn("Failed to log deletion (HTTP ${responseCode})")

}

} catch (Exception e) {

log.warn("Error sending remote log: ${e.message}")

}

}

// === EXECUTION ===

repository.repositoryManager.browse().findAll {

it.format == 'maven2' && it.type.name() == 'hosted'

}.each { repo ->

log.info("Scanning repository: ${repo.name}")

def tx = repo.facet(StorageFacet).txSupplier().get()

tx.begin()

try {

def snapshots = tx.findComponents(

Query.builder().where('version').like("%${snapshotSuffix}").build(), [repo]

)

def grouped = [:].withDefault { [] }

snapshots.each { c ->

if (!groupIdPrefixFilter || c.group()?.startsWith(groupIdPrefixFilter)) {

def key = "${c.group()}:${c.name()}"

grouped[key] << c

}

}

grouped.each { key, comps ->

def (group, name) = key.split(':')

def allVersions = tx.findComponents(

Query.builder().where('group').eq(group).and('name').eq(name).build(), [repo]

)\*.version.toSet()

comps.each { comp ->

def snapshotVer = comp.version

def releaseVer = snapshotVer.replace(snapshotSuffix, '')

if (allVersions.contains(releaseVer)) {

log.info("Deleting ${group}:${name}:${snapshotVer} from ${repo.name} (release ${releaseVer} exists)")

tx.deleteComponent(comp)

sendLog(repo.name, group, name, snapshotVer)

} else {

log.debug("Skipping ${group}:${name}:${snapshotVer} — no matching release version found")

}

}

}

tx.commit()

} catch (Exception e) {

log.error("Error processing repository ${repo.name}: ${e.message}", e)

tx.rollback()

} finally {

tx.close()

}

}

**🔐 Logging Endpoint Requirements**

Your endpoint should:

* Accept POST JSON requests like:

json

CopyEdit

{

"repository": "maven-releases",

"group": "com.example",

"artifact": "my-lib",

"version": "1.2.3-SNAPSHOT",

"timestamp": "Fri May 16 15:34:00 UTC 2025"

}

* Respond with 2xx on success

**🧪 Testing**

Before scheduling this as a task:

1. Set groupIdPrefixFilter to a narrow scope like com.example.test
2. Watch Nexus logs
3. Monitor your logging endpoint

Would you like to extend this to **email a daily summary**, or push logs to **Slack/MS Teams** as well?

4o

**You said:**

extend this to email a daily summary

**ChatGPT said:**

Here’s how you can **extend the Nexus 3 Groovy script** to **email a daily summary** of deleted SNAPSHOT artifacts.

**✅ What This Does:**

* Scans all Maven hosted repositories.
* Deletes SNAPSHOT artifacts **only if matching release version exists**.
* **Collects a summary** of deletions.
* **Emails a daily summary** with:
  + Repository
  + Group ID
  + Artifact ID
  + Version
  + Timestamp

**📧 Requirements**

You must configure **email settings** in Nexus first:

1. Go to **Administration → Email Server**
2. Configure SMTP settings (Gmail, corporate SMTP, etc.)
3. Test the connection

**✉️ Final Script (with Email Summary)**

groovy

CopyEdit

import org.sonatype.nexus.repository.storage.Query

import org.sonatype.nexus.repository.storage.StorageFacet

import org.sonatype.nexus.repository.Repository

import org.sonatype.nexus.repository.storage.Component

import org.sonatype.nexus.email.MailService

import org.sonatype.nexus.email.MailConfiguration

import groovy.json.JsonOutput

import java.text.SimpleDateFormat

// === CONFIGURATION ===

def groupIdPrefixFilter = 'com.example' // set to null or '' to disable

def snapshotSuffix = '-SNAPSHOT'

// Email config

def toAddress = 'devops@example.com'

def fromAddress = 'nexus@example.com'

def subject = "Daily Nexus SNAPSHOT Cleanup Summary"

// Summary buffer

def deletedItems = []

// === EXECUTION ===

repository.repositoryManager.browse().findAll {

it.format == 'maven2' && it.type.name() == 'hosted'

}.each { repo ->

log.info("Scanning repository: ${repo.name}")

def tx = repo.facet(StorageFacet).txSupplier().get()

tx.begin()

try {

def snapshots = tx.findComponents(

Query.builder().where('version').like("%${snapshotSuffix}").build(), [repo]

)

def grouped = [:].withDefault { [] }

snapshots.each { c ->

if (!groupIdPrefixFilter || c.group()?.startsWith(groupIdPrefixFilter)) {

def key = "${c.group()}:${c.name()}"

grouped[key] << c

}

}

grouped.each { key, comps ->

def (group, name) = key.split(':')

def allVersions = tx.findComponents(

Query.builder().where('group').eq(group).and('name').eq(name).build(), [repo]

)\*.version.toSet()

comps.each { comp ->

def snapshotVer = comp.version

def releaseVer = snapshotVer.replace(snapshotSuffix, '')

if (allVersions.contains(releaseVer)) {

log.info("Deleting ${group}:${name}:${snapshotVer} from ${repo.name} (release ${releaseVer} exists)")

tx.deleteComponent(comp)

deletedItems << [

repo : repo.name,

group : group,

artifact : name,

version : snapshotVer,

timestamp: new Date().toString()

]

} else {

log.debug("Skipping ${group}:${name}:${snapshotVer} — no matching release version found")

}

}

}

tx.commit()

} catch (Exception e) {

log.error("Error processing repository ${repo.name}: ${e.message}", e)

tx.rollback()

} finally {

tx.close()

}

}

// === EMAIL SUMMARY ===

if (!deletedItems.isEmpty()) {

def mailService = container.lookup(MailService.class.name)

def summary = new StringBuilder()

summary << "The following SNAPSHOT components were deleted:\n\n"

deletedItems.each {

summary << "- ${it.repo}: ${it.group}:${it.artifact}:${it.version} @ ${it.timestamp}\n"

}

summary << "\nTotal deletions: ${deletedItems.size()}"

mailService.send(fromAddress, [toAddress], subject, summary.toString())

log.info("Summary email sent to ${toAddress}")

} else {

log.info("No SNAPSHOT artifacts deleted today. No email sent.")

}

**📝 Email Output Example**

Subject:

pgsql

CopyEdit

Daily Nexus SNAPSHOT Cleanup Summary

Body:

yaml

CopyEdit

The following SNAPSHOT components were deleted:

- maven-releases: com.example:my-lib:1.2.3-SNAPSHOT @ Fri May 16 12:45:02 UTC 2025

- maven-releases: com.example:my-lib-util:1.2.3-SNAPSHOT @ Fri May 16 12:45:03 UTC 2025

Total deletions: 2

**✅ How to Use**

1. Save this script under **Administration → System → API → Scripts**
2. Name: deleteSnapshotsAndSendEmail
3. Create a new task:
   * Type: Run Script
   * Script: deleteSnapshotsAndSendEmail
   * Schedule: **daily**, e.g., 2 AM