

# A Repository Management Systems

## An Introduction



**Rajesh Kumar**

**[www.RajeshKumar.xyz](http://www.RajeshKumar.xyz)**

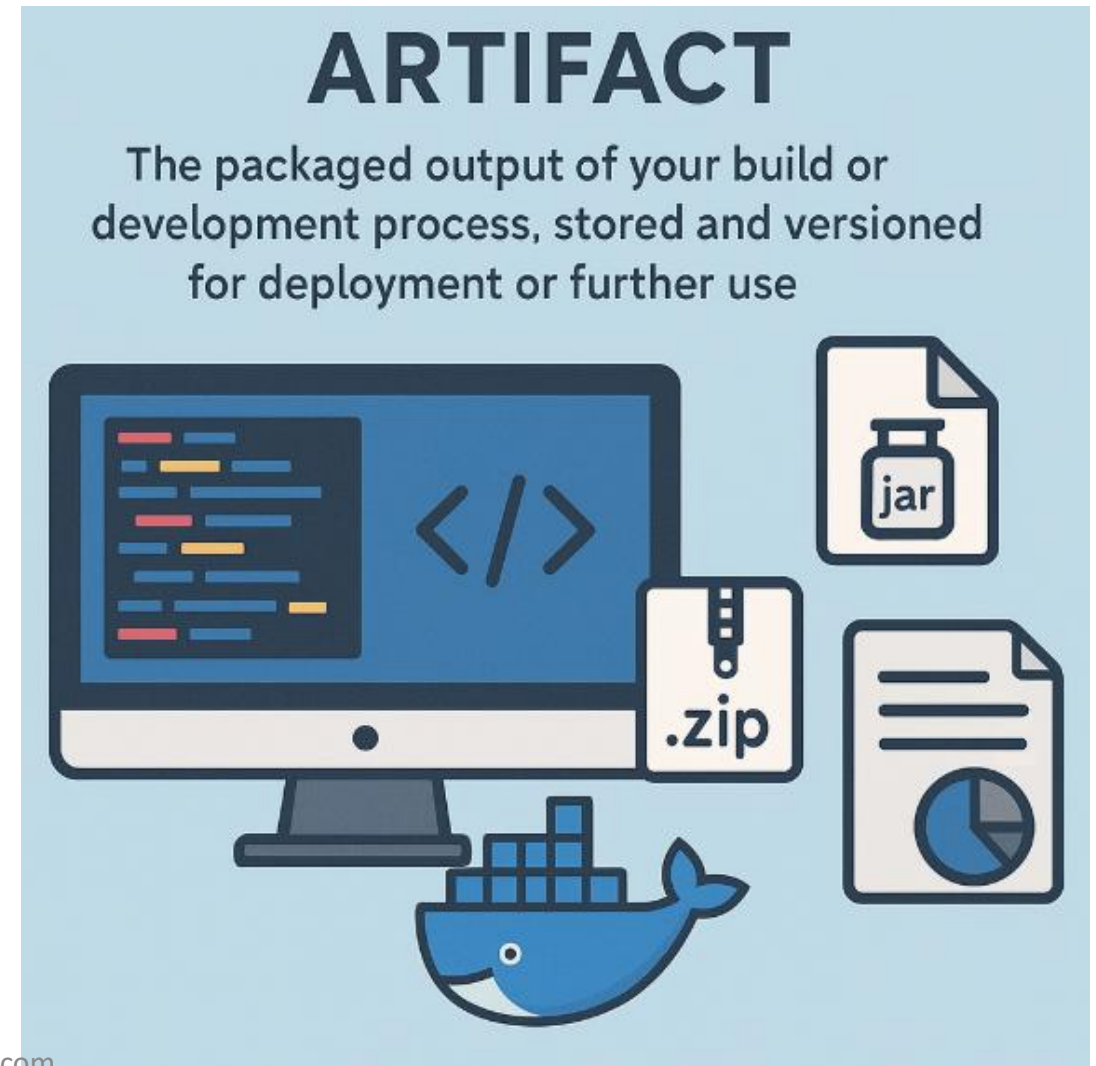
**[DevOps@RajeshKumar.xyz](mailto:DevOps@RajeshKumar.xyz)**

# What is an Artifact?

An **artifact** is a **file or collection of files generated during the build process** of software.

## Examples:

- Compiled binaries (.jar, .exe, .dll)
- Docker images
- .zip or .tar.gz packages
- Test reports, log files

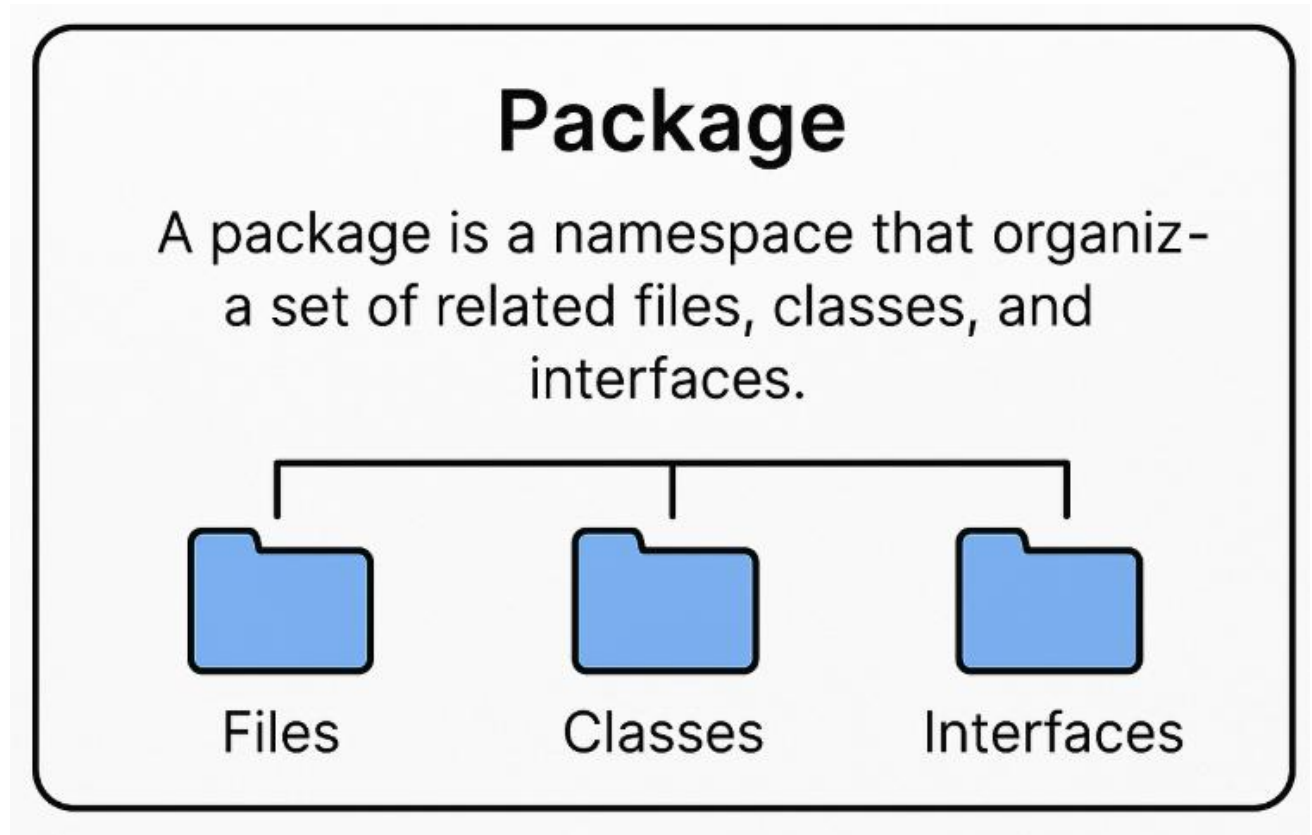


# What is Package?

A **package** is a way to **organize and group related files, classes, or modules under a common namespace** so they can be easily managed, reused, and distributed.

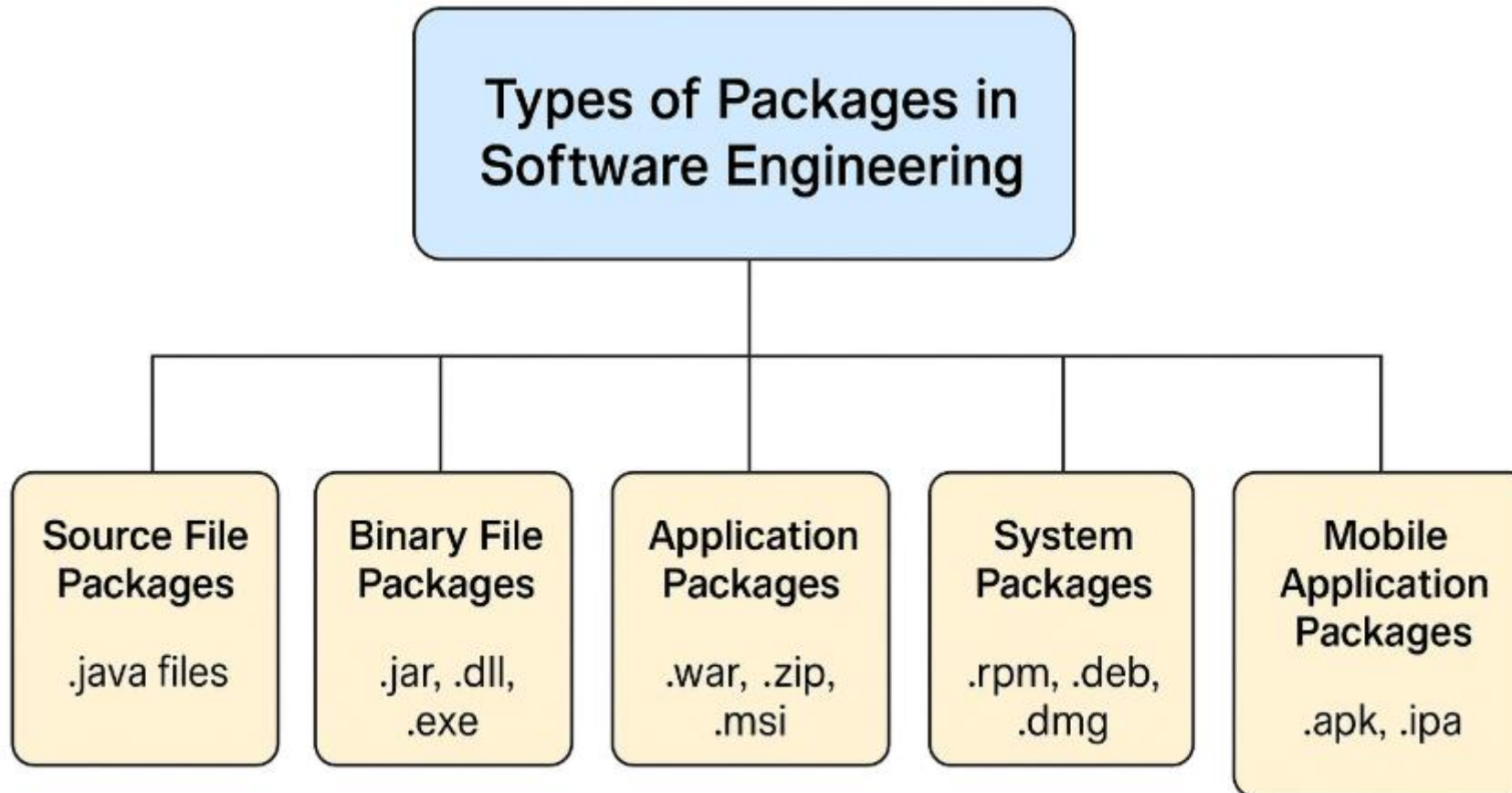
## Key Points:

- A package acts like a **folder or directory** that contains related components.
- Provides a **namespace** to avoid name conflicts between classes or files.
- Helps in **maintaining code structure** and managing dependencies.



# Types of Packages

In **Software Engineering**, packages are used to organize, distribute, and deploy software components. The major types of packages include:



# Types of Packages:Source File Packages

## 1 Source File Packages

Contain human-readable source code files.

Used during development and for generating documentation.

### **Example:**

Java: .java files grouped in a package.

Python: A directory with .py files.

### **Purpose:**

Organize code and provide namespaces.

Allow reuse and easy maintenance.

# Types of Packages: Binary File Packages

## 2 Binary File Packages

Contain compiled, machine-readable files ready for execution or linking.

### Examples:

Java: .jar (Java Archive).

Windows: .dll (Dynamic Link Library).

Executables: .exe, .bin.

### Purpose:

Distribute built software without exposing source code.

Optimize for deployment.

# Types of Packages: Application Packages

## 3 Application Packages

Bundled software for deployment or installation.

Include application code, resources, and metadata.

### Examples:

Java: .war (Web Archive), .ear (Enterprise Archive).

.NET: .msi, .exe.

Compressed: .zip, .tar.gz.

### Purpose:

Deliver a ready-to-use software application.

Simplify installation and updates.

# Types of Packages: System Packages

## 4 System Packages

Packages tailored for operating system-level installation.

### Examples:

Linux: .rpm (RedHat/CentOS), .deb (Debian/Ubuntu).

macOS: .dmg.

Windows: .msi, .exe.

### Purpose:

Integrate with OS package managers.

Support version control, dependencies, and uninstallation.

# Types of Packages: Mobile Application Packages

## 5 Mobile Application Packages

Used for mobile platforms.

### Examples:

Android: .apk (Android Package), .aab (App Bundle).

iOS: .ipa (iOS App Archive).

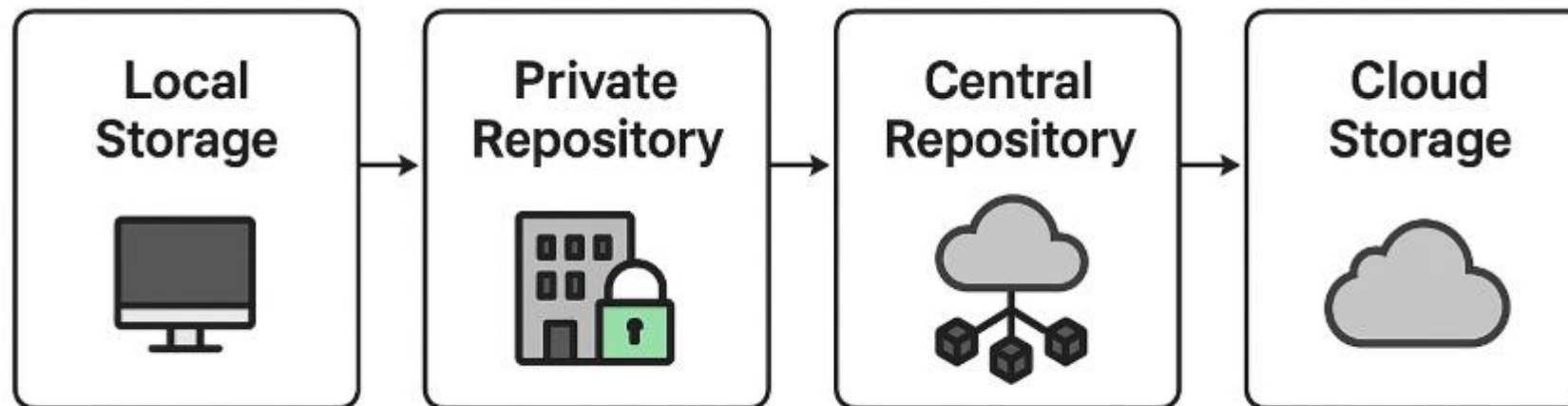
### Purpose:

Deliver mobile apps via app stores.

Manage signing and distribution.

# How to Store Packages?

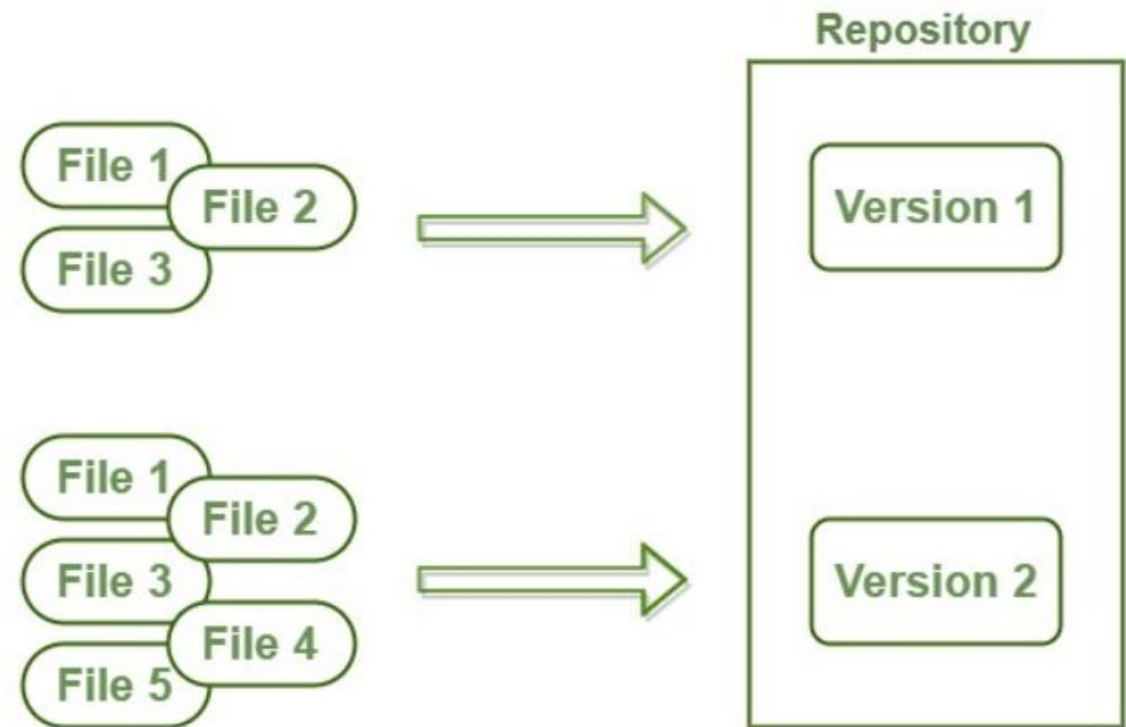
Packages are stored in **repositories** or **artifact storage systems** that allow versioning, access control, and distribution. How you store a package depends on the type (application, system, mobile) and the environment (development, production).



# What is Repository?

## What is a Repository?

A repository is a place where source code and other development artifacts are stored. A repository typically includes a version control system to manage changes to files.



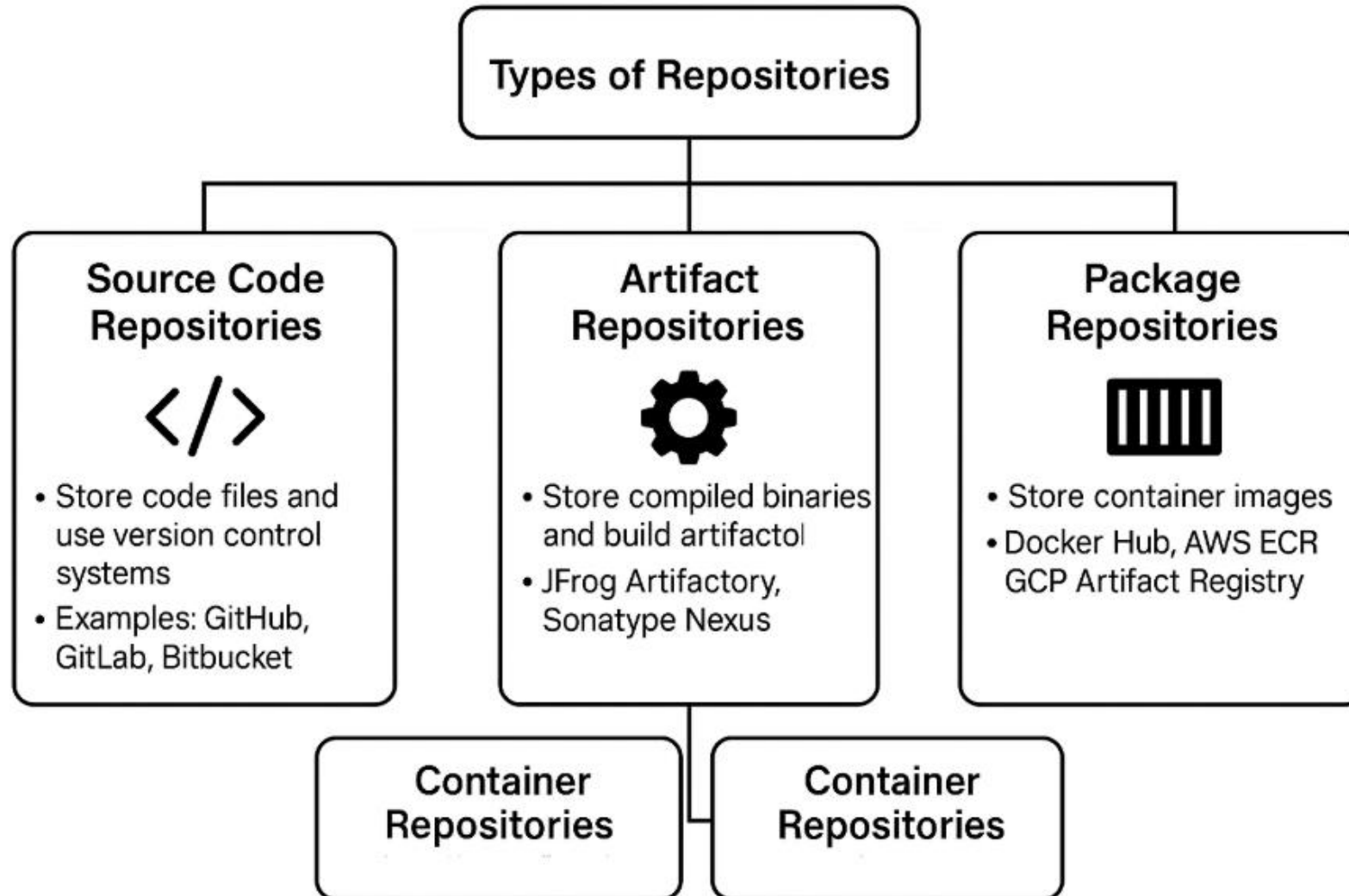
# What is Repository?

A **repository** is a centralized location where **source code, packages, and development artifacts** are stored, managed, and version-controlled. It acts as a structured storage system that allows developers to **organize, share, and track changes** to files over time.

## Key Points:

- Stores **code, binaries, packages, or artifacts**.
- Provides **version control** to track changes.
- Allows **collaboration** among teams.
- Ensures **backup and availability** of project files.

# Type of Repository?



# Type of Repository?

## Types of Repositories:

### 1. Source Code Repositories:

1. Store code files and use version control systems.
2. Examples: GitHub, GitLab, Bitbucket.

### 2. Artifact Repositories:

1. Store compiled binaries and build artifacts.
2. Examples: JFrog Artifactory, Sonatype Nexus.

### 3. Package Repositories:

1. Host language-specific packages.
2. Examples: Maven Central, PyPI, npm Registry.

### 4. Container Repositories:

1. Store container images.
2. Examples: Docker Hub, AWS ECR, GCP Artifact Registry.

# What is Repository Management System?

A **Repository Management System (RMS)** is a software tool or platform used to **store, organize, manage, and distribute software packages, artifacts, and dependencies** in a centralized and secure manner. It acts as a hub between the **development/build process** and the **deployment process**, ensuring that all versions of software components are properly tracked and available.

# What is Repository Management System?

## Key Features of a Repository Management System:

- **Centralized Storage:** Keeps all packages, binaries, and artifacts in one place.
- **Version Control:** Maintains multiple versions of the same package.
- **Dependency Management:** Automatically resolves and stores dependencies for builds.
- **Access Control & Security:** Restricts who can upload/download artifacts.
- **High Availability:** Ensures packages are always accessible.
- **Integration with CI/CD:** Works seamlessly with build pipelines.

# Examples of Repository Management Systems

1. Sonatype Nexus Repository
2. JFrog Artifactory
3. Apache Archiva
4. AWS CodeArtifact
5. Azure Artifacts
6. Azure DevOps
7. Github CI
8. Gitlab CI

# What is Registry?

A specialized type of repository that primarily stores container images or specific package formats.

Most commonly used for Docker/OCI images or language-specific packages.

## Examples:

1. Docker Hub (container registry)
2. AWS ECR (Elastic Container Registry)
3. GCP Artifact Registry
4. npm Registry (JavaScript packages)
5. PyPI Registry (Python packages)

# RMS Vs Registry

**RMS** = Broader system that can host multiple repository formats (binaries, packages, containers, etc.).

**Registry** = Often refers to a specialized repository, mostly for container images or a single package ecosystem.

## Repository Management System

- Stores and manages various software artifacts, packages and binaries

 nexus

 JFrog  
ARTIFACTORY

 ARCHIVA

## Registry

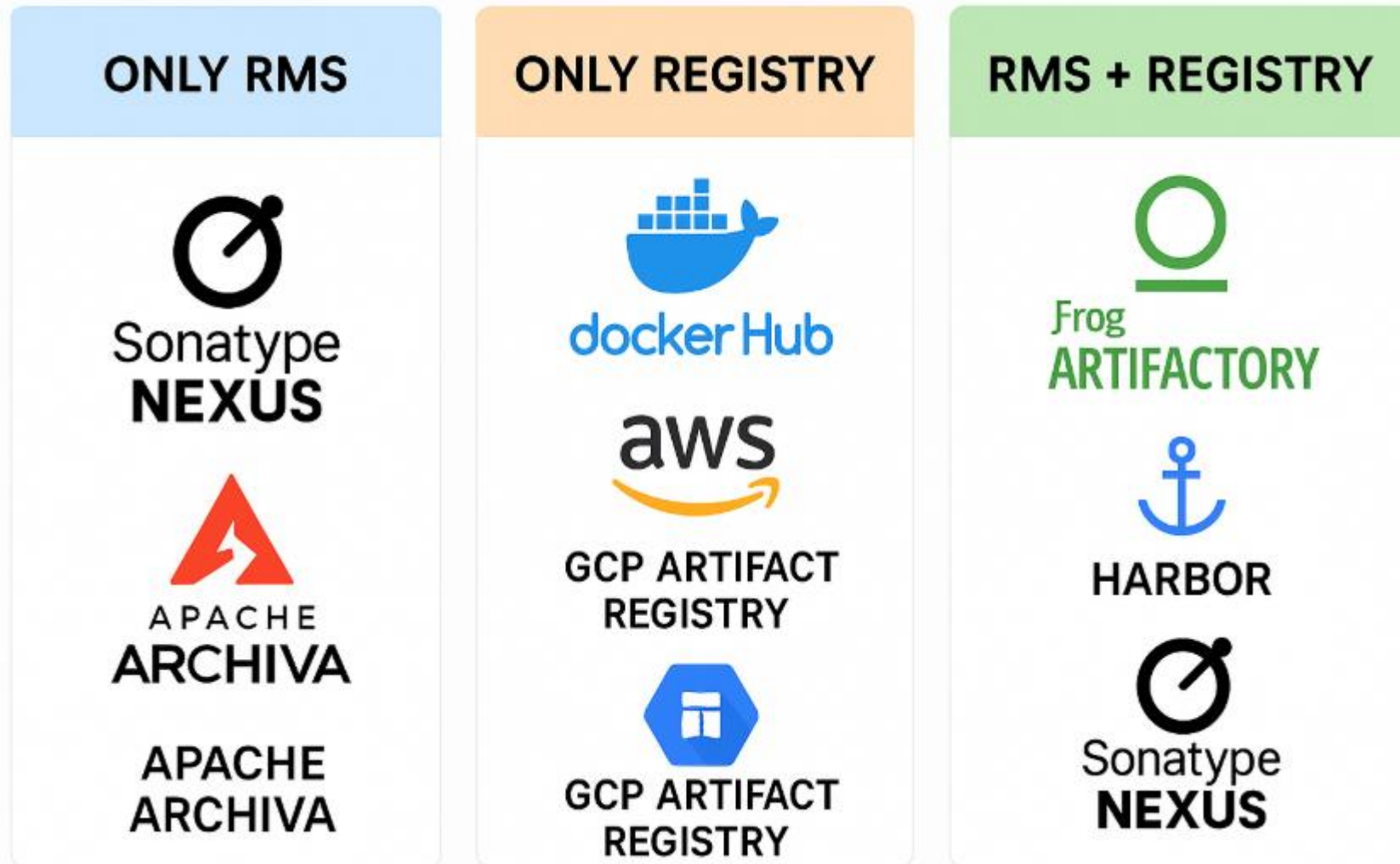
- Stores container images or specific package formats

 Docker Hub

 aws ECR

 GCP  
Artifact Registry

# RMS can act as a Registry



# Summary

## In Short:

- **Artifacts** are the individual outputs from builds.
- **Packages** are collections of artifacts.
- **Repositories** store versioned packages.
- **RMS** manages repositories, versions, dependencies, and access.

# Summary

## 1 Multiple Artifacts → Form a Package

- Artifacts (binaries, config files, resources) are bundled together into a versioned package.

## 2 Multiple Packages → Stored in a Repository

- These packages are uploaded and stored in a repository with metadata and version history.

## 3 Repository → Managed by RMS

- The repository is organized, secured, and version-controlled through a **Repository Management System (RMS)**.

# Summary



## Top Repository Management Systems



Universal Artifact Repository



Managed Cloud RMS



Enterprise Universal RMS



Managed Cloud RMS



Lightweight / Open Source



 Harbor (Project)  
Kubernetes-Native

# Feature of Repository Management Systems

## Top 10 Features of Repository Management Systems



- 1 Centralized Artifact Storage**  
Stores all packages, binaries, and artifacts in one secure location
- 2 Multi-format Support**  
Handles multiple package types such as Maven, npm, PyPI, Docker
- 3 Version Control for Packages**  
Maintains multiple versions of the same package
- 4 Dependency Management**  
Resolves and caches dependencies automatically
- 5 Integration with CI/CD Pipelines**  
Seamlessly connects with Jenkins, GitLab CI/CD, etc.
- 6 Security and Access Control**  
Enforces role-based access and security policies
- 7 Caching and Proxying Remote Repositories**  
Caches dependencies from remote repositories
- 8 High Availability and Replication**  
Supports HA configurations and geo-replication
- 9 Audit and Traceability**  
Tracks and records all repository activities
- 10 Promotion and Lifecycle Management**  
Manages artifact promotion across different environments

# Feature of Repository Management Systems

## ✓ 1. Centralized Artifact Storage

- Stores all packages, binaries, and artifacts in one secure location.
- Eliminates dependency on scattered local builds.

## ✓ 2. Multi-format Support

- Handles multiple package types:
  - Maven, npm, PyPI, NuGet, Docker, Helm, RubyGems, etc.
- Enables universal artifact management in one tool.

## ✓ 3. Version Control for Packages

- Maintains multiple versions of the same package.
- Allows easy rollback and upgrade of software components.

# Feature of Repository Management Systems

## ✓ 4. Dependency Management

- Resolves and caches dependencies automatically.
- Speeds up builds by providing pre-fetched libraries.

## ✓ 5. Integration with CI/CD Pipelines

- Seamlessly connects with Jenkins, GitLab CI/CD, Azure DevOps, GitHub Actions, etc.
- Automates build, store, and deploy workflows.

## ✓ 6. Security and Access Control

- Implements Role-Based Access Control (RBAC).
- Supports artifact signing, checksum validation, and vulnerability scanning.

# Feature of Repository Management Systems

## ✓ 7. Caching and Proxying Remote Repositories

- Acts as a proxy to central repositories (e.g., Maven Central, npm registry).
- Reduces build time and provides offline access to dependencies.

## ✓ 8. High Availability and Replication

- Supports HA clusters and geo-replication.
- Ensures artifact availability in multi-site and disaster recovery scenarios.

## ✓ 9. Audit and Traceability

- Tracks who uploaded, downloaded, or changed artifacts.
- Provides complete audit logs for compliance.

## ✓ 10. Promotion and Lifecycle Management

- Manages artifact lifecycle (dev → test → prod).
- Supports artifact promotion between environments without rebuilding.

*Thank you!*

*DevOpsSchool Team*



[www.DevOpsSchool.com](http://www.DevOpsSchool.com)



contact@devopsschool.com