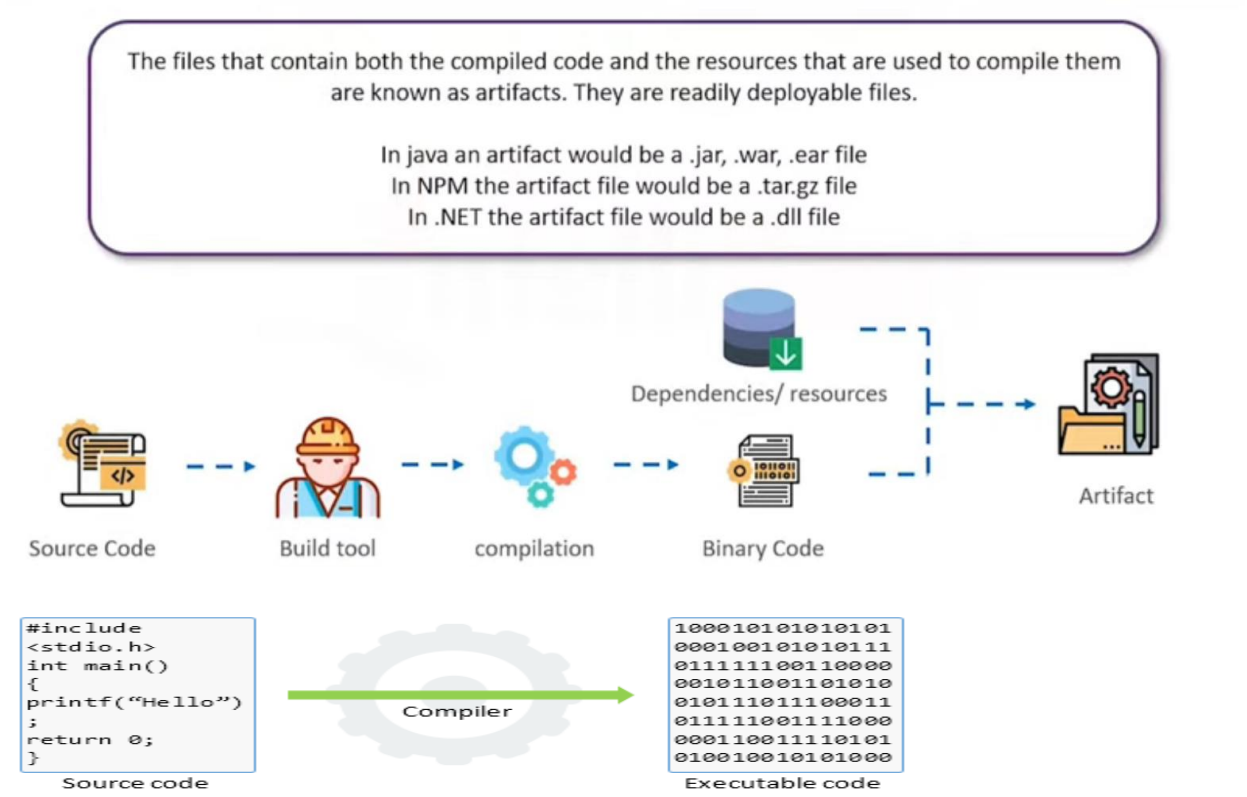
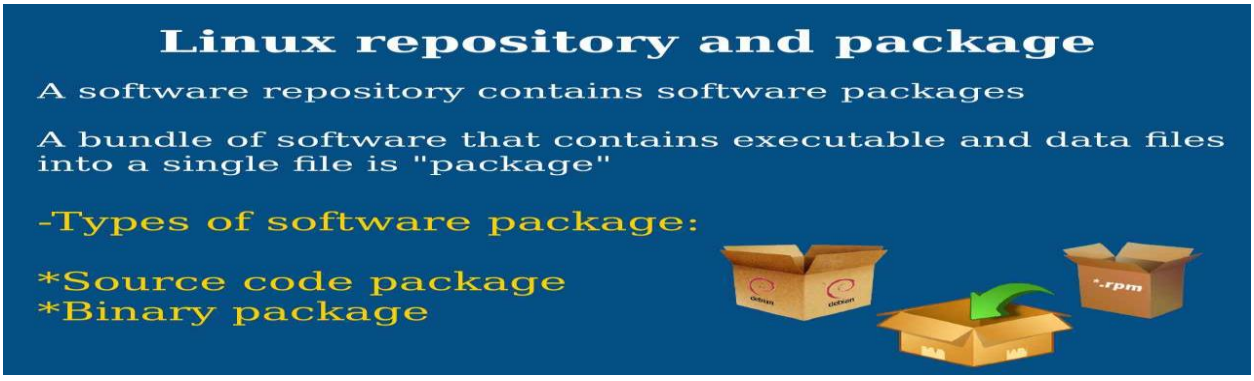
Binary/Repository Management with JFrog Artifactory

What are artifacts?

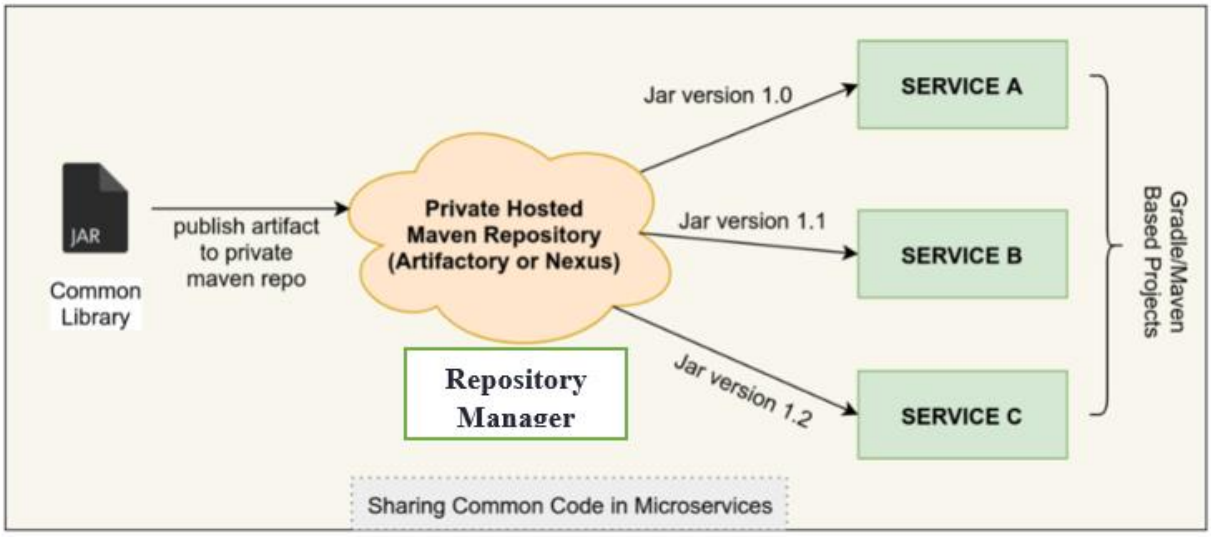




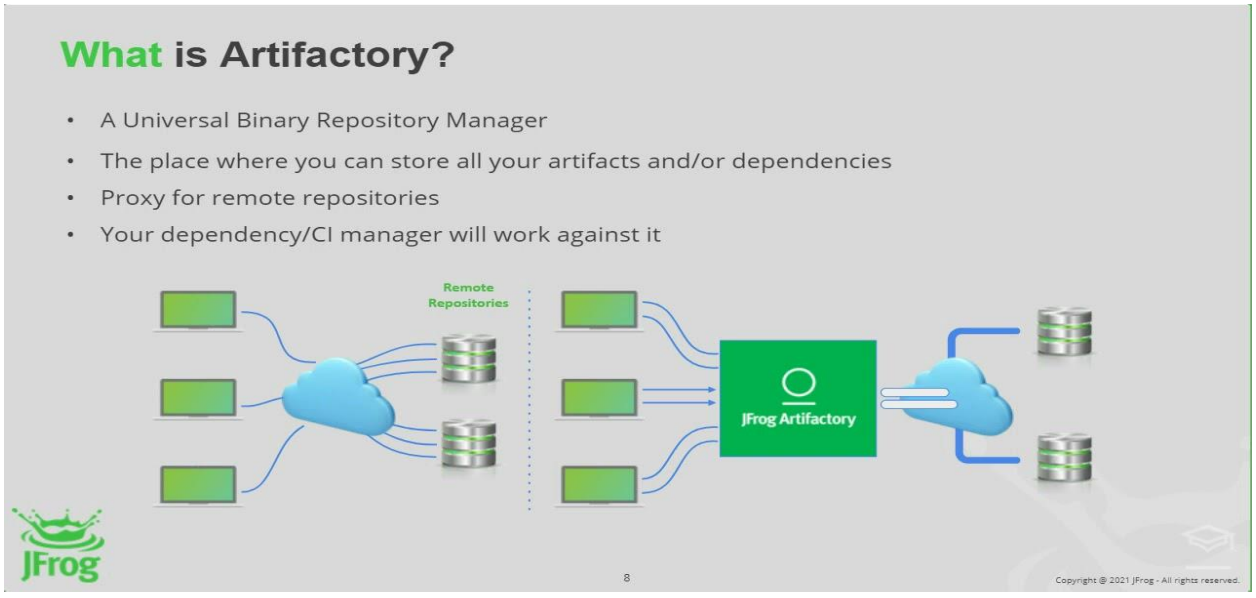
Why Source Repositories?

Source repositories are designed simply to manage source code. A well-built source repository therefore boasts a feature set tailored to source code management (e.g.: differing versions, tracking deleted or overwritten files, branching, and tagging).

Why (Binary) Repository Managers?



- Increasing build performance due to a wider distribution of software and locally available parts.
- Reducing network bandwidth and dependency on remote repositories.
- Insulating your company from outages on the internet, outages of public repositories (Maven Central, npm, etc.), or even removal of an open-source component.

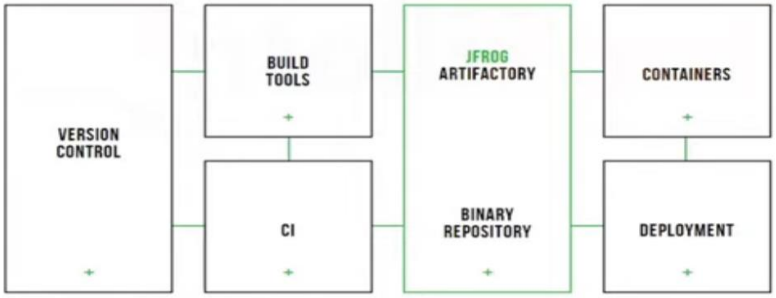


Why Artifactory?

- **Universal**: Integrates with 27+ package managers and CI servers
- **System of Record**: Full metadata for all supported package formats
- **Storage Optimization**: Checksum based storage
- **Automation**: REST API, CLI, AQL, User Plugins
- **Integrations**: Built in integrations with automation tools for CI/CD

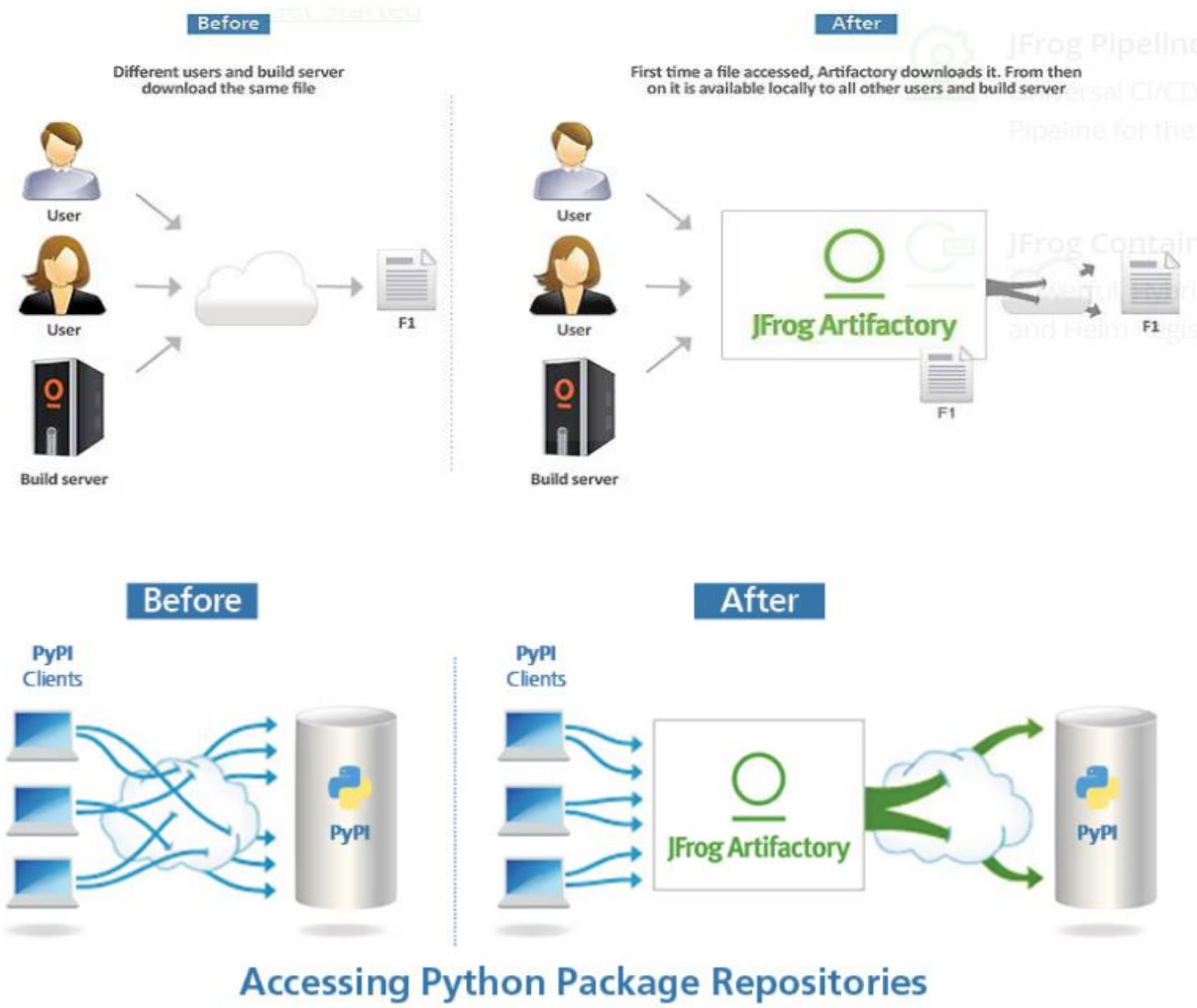
JFrog

JFrog Artifactory is a tool used in DevOps methodology for multiple purposes. One of it's main purpose is to store artifacts(readily deployable code) that have been created in the code pipeline. Another one of it's purpose is to act as a sort of a buffer for downloading dependencies for the build tools and languages.



Artifactory Features

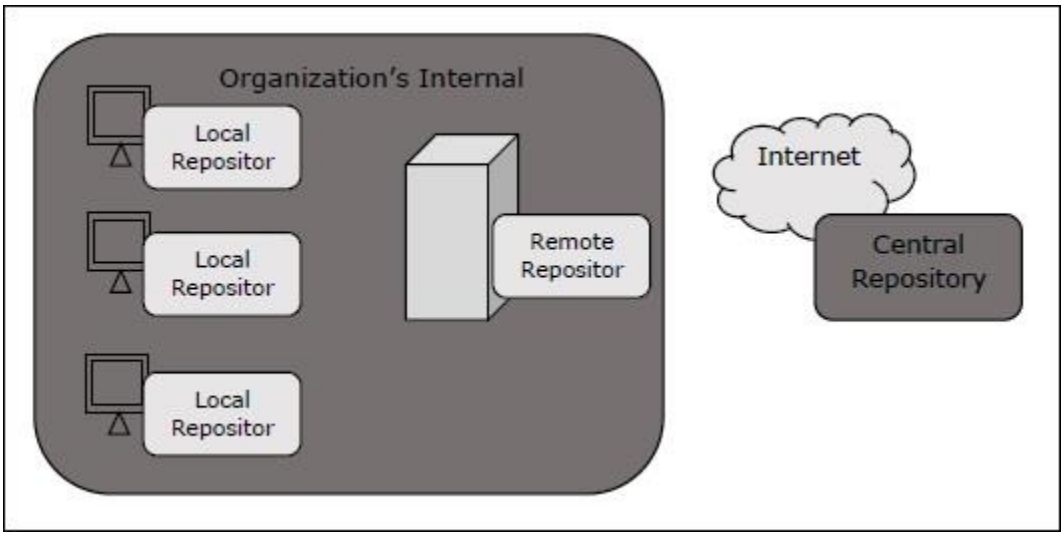
- Reliability
- Efficiency
- Security
- Stability
- Productivity
- XPlat Installation (Windows or other)



Repository vs. Repository Manager

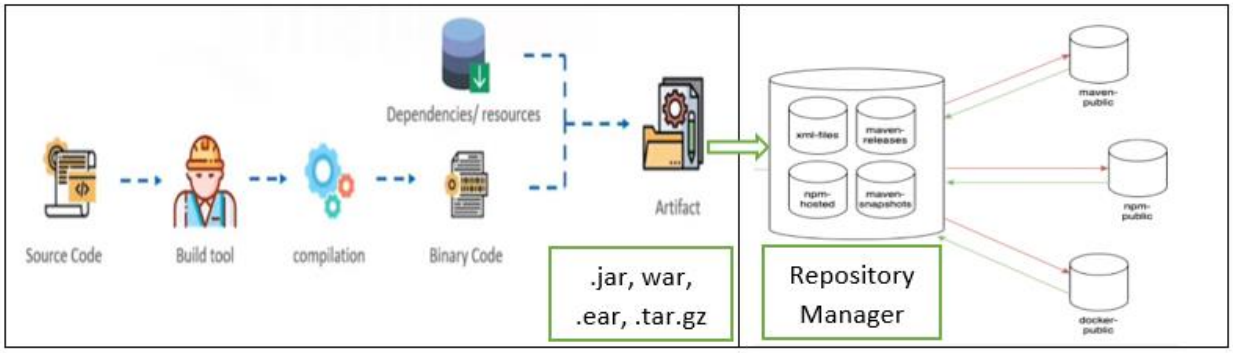
A ***repository*** is a storage location where components such as packages, libraries, binaries, or containers are retrieved so they can be installed or used. In order to ease consumption and usage of open-source components, they are aggregated into collections on public repositories, and are typically available on the internet as a service. Examples of such repositories are:

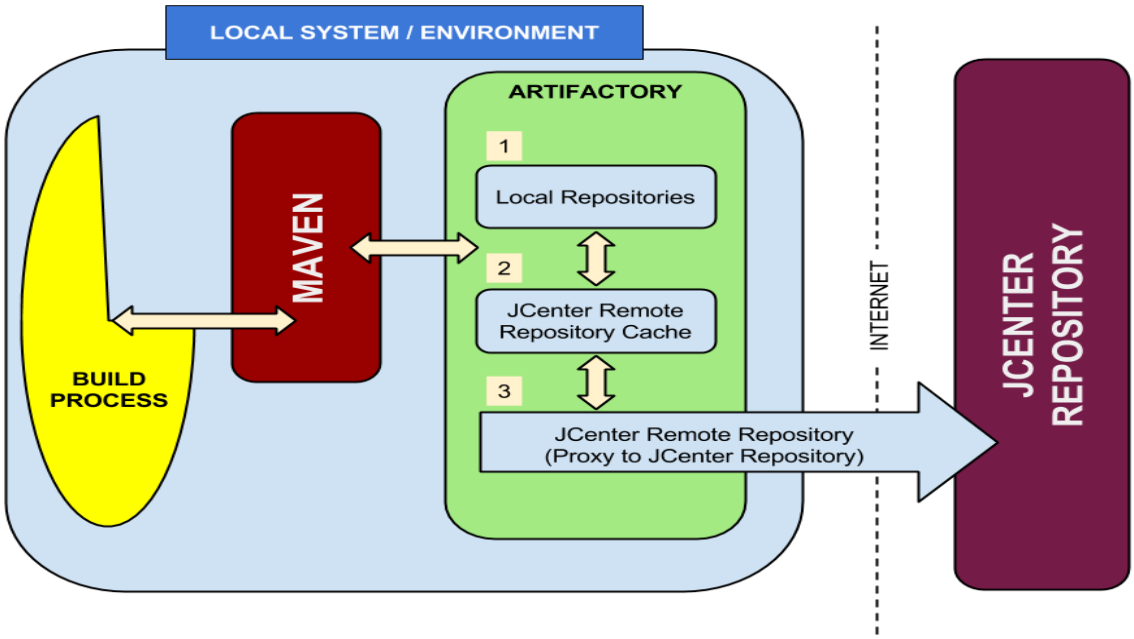
- Central Repository, also known as Maven Central
- NuGet Gallery
- RubyGems.org
- npmjs.org



A ***repository manager*** is a dedicated server application used to manage all the repositories your development teams use throughout the course of development. At its core, a repository manager does the following:

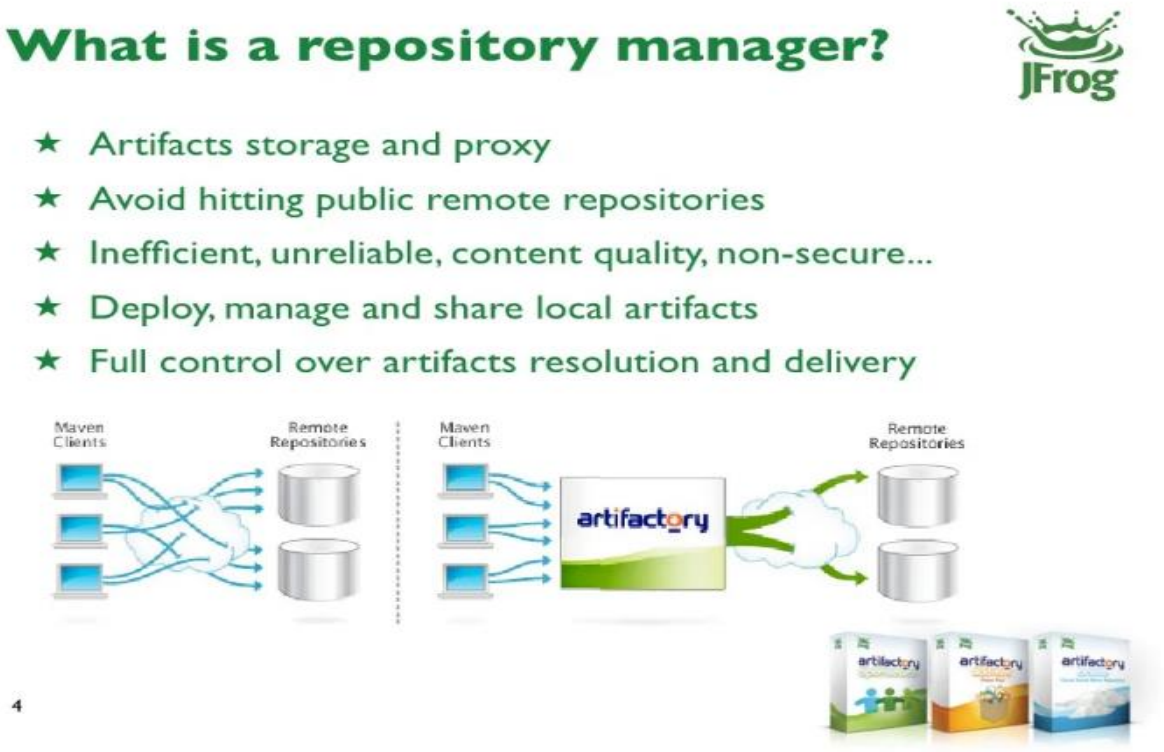
- Proxies remote repositories and caches contents.
- Hosts internal repositories.
- Groups repositories into a single repository.



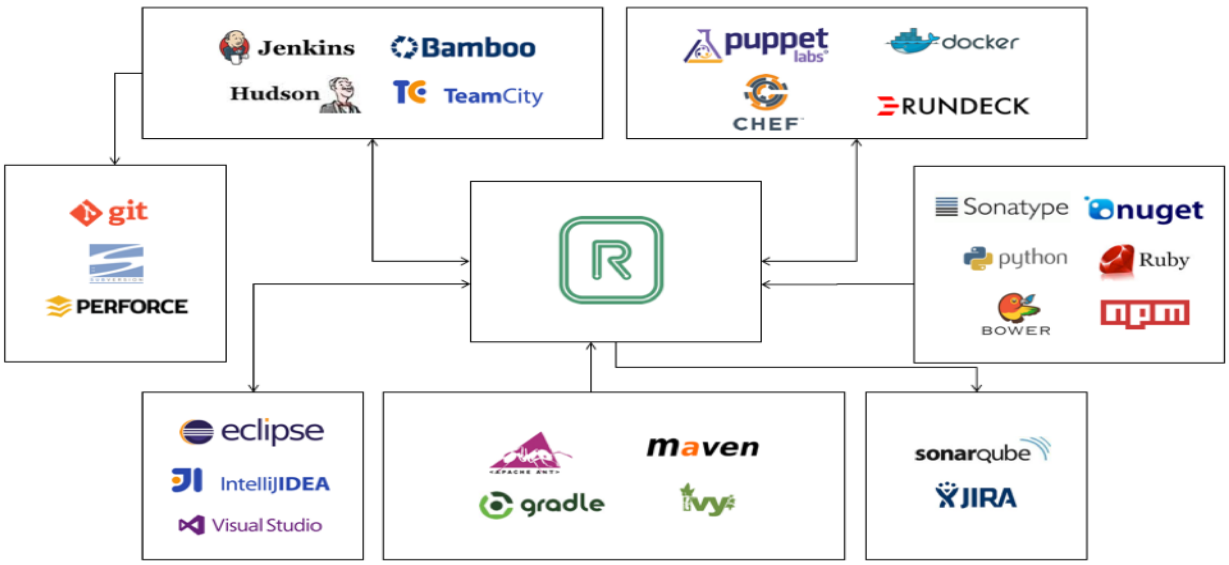


Why do I need a Repository Manager?

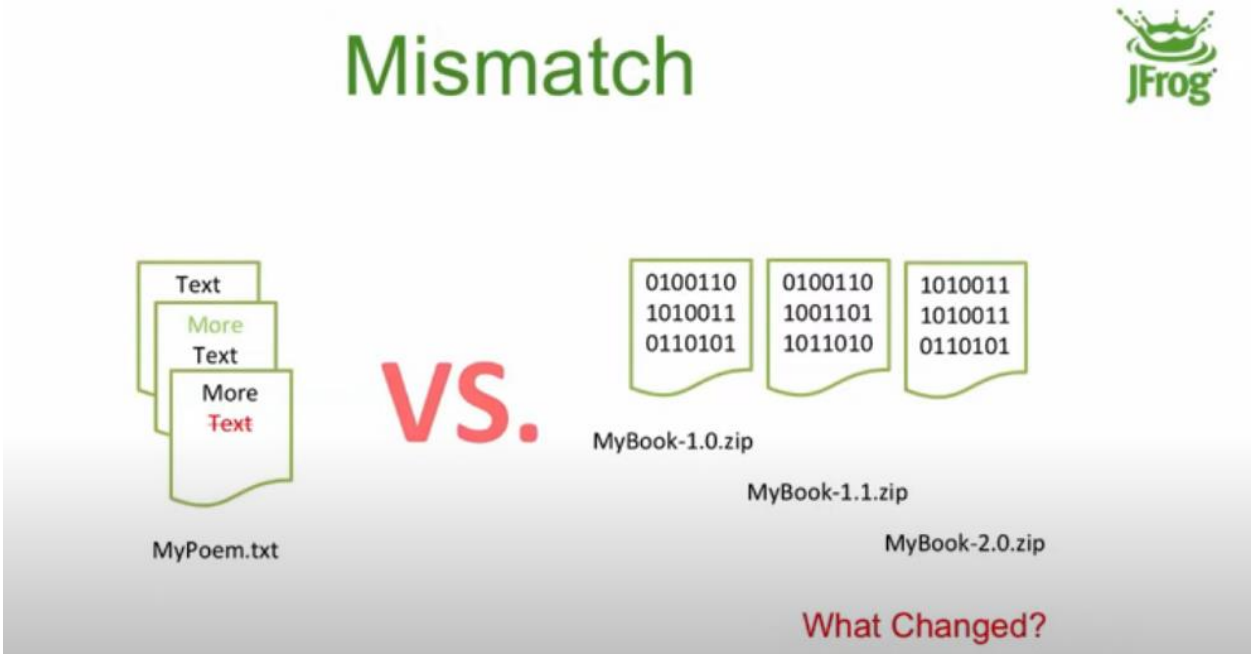
- Most of today's software is assembled using open source, proprietary, and third-party code. Because of this, many organizations rely on repository management to efficiently source, store, share, and deploy software parts.
- Rather than direct download from public repositories, repeated downloads, or manual distribution, using a repository manager makes it the central access and management point for any component usage in your software development life cycle. This central role makes it easy for everyone in your organization to understand where things are stored.
- Using a repository manager allows you to manage parts required for software development, application deployment, and automated hardware provisioning.
- It also fulfills a central role for application life cycle management.
- A repository manager greatly simplifies the maintenance of your own internal repositories and access to remote repositories.
- In addition, integrating a repository manager into your organization lets you accomplish the following:
- Manage binary software code through the software development life cycle.
    - Search and browse software components.
    - Control code releases with rules and add automated notifications.
    - Integrate with external security systems, such as LDAP or Atlassian Crowd.
    - View component metadata.
    - Host internal items not available in remote repositories.
    - Control access by authenticating and authorizing specific sets or groups of users.
    - Relying on a repository manager lets you gain new visibility into the quality of items flowing through your software supply chains, helping to avoid downstream technical debt and unplanned work.



Repository Managers in the DevOps Toolchain



Sources vs Binaries



Advanced repository features

- ★ Checksum-based storage
- ★ Handling download bursts
- ★ Verifying uploaded artifacts
- ★ Locking
- ★ Cleanup bad remote repo references
- ★ Built-in metadata

CHECKSUM BASED STORAGE ADVANTAGES

- Deduplication
- "Free" copy and move
- "Free" delete (complimentary GC)
- Efficient uploads, downloads and replication
- Filesystem performance
- Search stability and performance
- Freedom of layouts

Checksum-based storage

- ★ Many identical artifacts are produced and stored numerous times in the repositories
  - Unique snapshots that are exactly the same between subsequent builds
  - Other artifacts that are copied
    - Mainly needed for more natural security control
- ★ Artifactory uses a checksum-based storage
- ★ Identical artifacts are stored on the server exactly once!
  - No matter how many references are there
- ★ Copy and move are very cheap
  - Pointer operations

Concurrent downloads/ request bursts

- ★ New snapshot dependency available/ POM updated with a new dependency version
- ★ Dependency can be as big as hundreds of megs
  - Assemblies
- ★ All clients download at once
  - Network blockage (DOSing)
- ★ Artifactory will identify this
  - Queue all incoming request until the first one finishes
  - Others will get the cached version

Checksum for uploaded artifacts

- ★ No way to verify uploaded artifacts
- ★ Maven approach:
  - The repository is passive
  - All calculations done on client
  - Repository to accept and store client checksum

- ★ Artifactory
  - Compare client checksum with the one calculated on the repository
  - If in conflict return 409 until a good checksum is found
  - This behavior is configurable

Locking

- ★ Artifactory applies RWLocks on all items
- ★ Avoid concurrent writes & dirty reads
- ★ Spans to metadata as well - cannot create metadata conflicts
- ★ Built-in utility for debugging lock contention in runtime (zero overhead)

POM cleanup

- ★ Many common third party POMs contain remote repository references making controlled resolution a nightmare
- ★ Global mirroring is not a solution
  - Forces a unified repository for releases/snapshots/plugins
- ★ Artifactory can facade POMs through a Virtual Repository
  - Can configure remote repo references to be removed
  - Original POM is intact

Metadata backed into the core

- ★ Every repository element can hold metadata definitions
  - Both files and folders
- ★ Metadata is any XML document
  - Can be queried using XPath
  - All exposed via UI and REST API
- ★ Properties tagging
  - Similar to SVN props
  - Internally stored as XML
  - "Strongly typed" props can be defined via UI
    - Open/closed lists, multi-select, single-select etc.
  - Applied via UI or REST
  - Also on deploy time with zero build tool tweaking!

Smart staging and promotion

- ★ Repository has two main roles
  - Proxy remote artifacts
  - Host deployed artifacts
    - Artifacts (should) come from CI server
- ★ Promotion of artifacts starts with a build
- ★ *The Binaries Repository & the CI Server are always interconnected*

Search-based management

- ★ Makes bulk artifact management a lot easier
- ★ Shopping-cart of artifacts
- ★ Fill-up the cart by searching and saving the search results - any search type!
- ★ Do other searches and add/subtract the results from the original
- ★ Can tweak the result manually
  - E.g. remove sources
- ★ Once done move/promote/copy/export the result
- ★ Does not enforce narrow concepts to support only specific limited use cases
  - E.g. promotion

Repository Types

- **Local** - Physical, locally-managed repository
- **Remote** - Proxy of another network repo/another artifactory (lazy cache)
- **Virtual** - Aggregation of repos under one URL (Local and/or Remote)

Repository Types: Local Repository

- Physical, locally-managed repository
- You deploy artifacts on it
- You consume artifacts from it
  (usually those other teams produced)

Repository Types: Remote Repository

- Proxy of another network repository (Lazy, not mirror)
- Caches foreign repository
- Can be another Artifactory

Repository Types: Virtual Repository

- Aggregates a number of repositories under a single URL
- An entry point for client reads and deployment to the default local Repository.
- Simplifies client configuration
- Supports nesting
- Considered as a best practice

BUILDS & CI INTEGRATION

- Consume binaries from Artifactory as well as deploying them.
- Publish BuildInfo and see all published builds.
- Explore build's artifacts and corresponding dependencies.
- Obtain information about the build environment.
- Bidirectional links between build and artifact information, inside the build server and Artifactory pages.

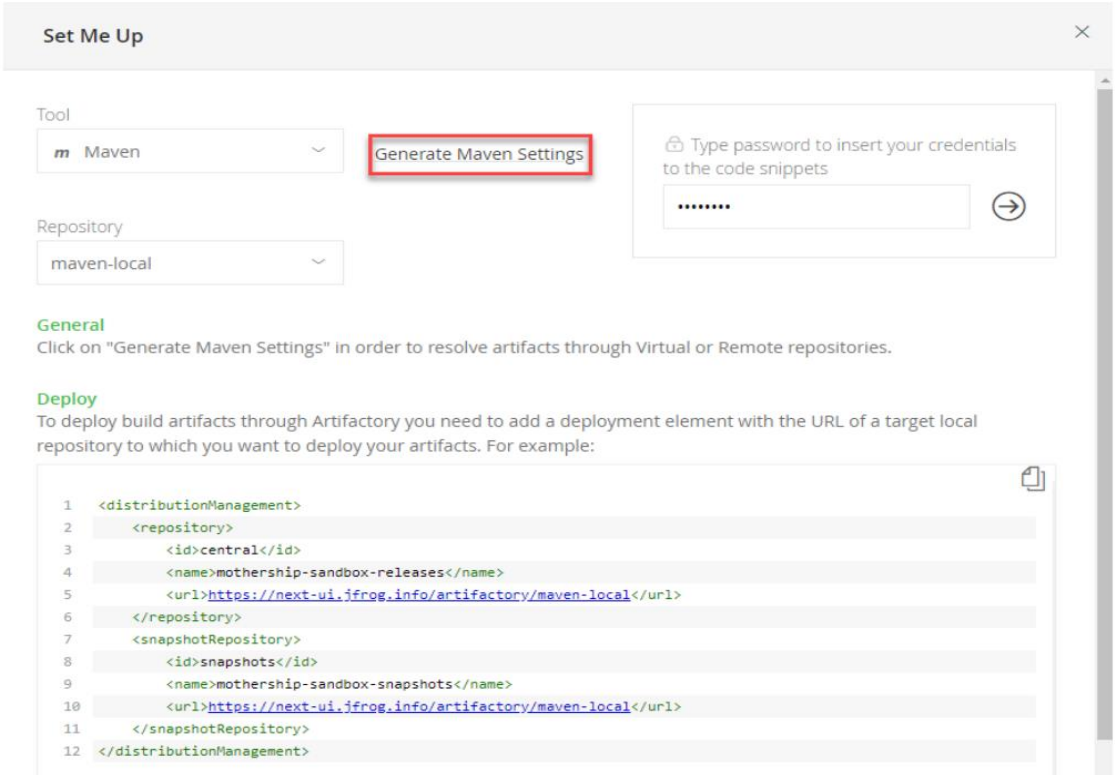
Deploying Artifacts Through Artifactory

maven element – distribution Management

Distribution management' section, which is exactly where you need to add your **POM** to get a proper connection to Artifactory.

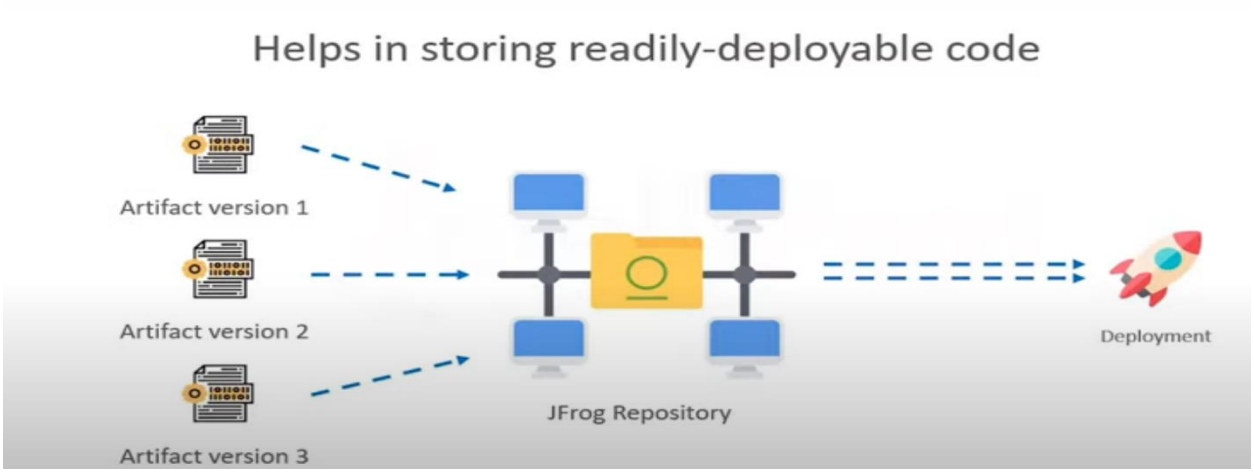
Once you have created your Maven repository, go to **Application | Artifactory | Artifacts**, select your Maven repository and click **Set Me Up.**

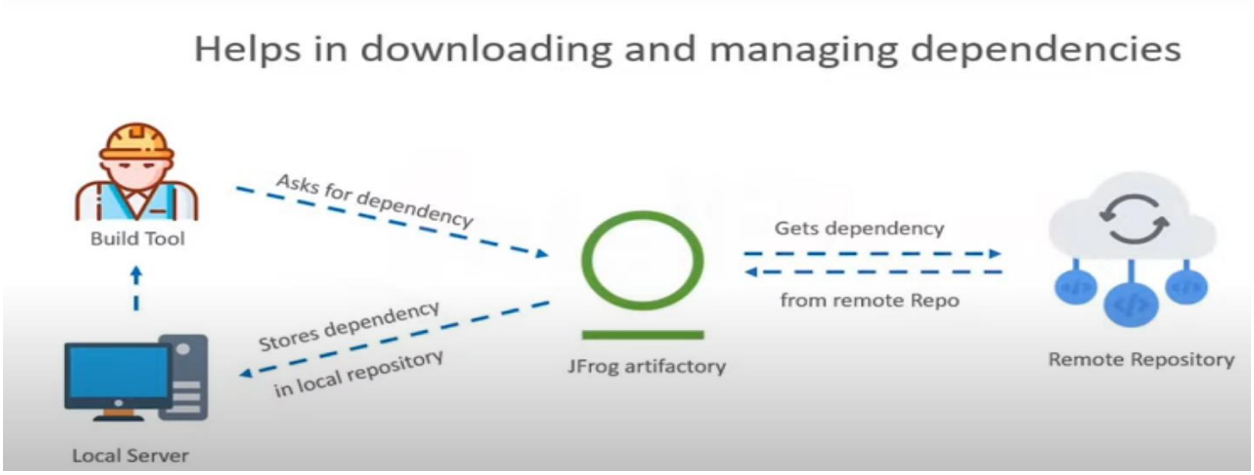
In the Set Me Up dialog, click **Generate Maven Settings**.



Depending on your Artifactory version you need to insert some additional lines, so our Maven Plugins can be deployed to that Artifactory. Make sure you have the following lines included in your `settings.xml` file:

```
<servers>
          <server>
          <id>central</id>
          <username>admin</username>
          <password>password</password>
          </server>
</servers>
```





Advantages-JFROG Artifactory-Artifact Managers

- Integrating a binary repository into a Continuous Delivery pipeline or a Continuous Deployment process
- Standardize the use of a binary artifact library for repeatable, secure, and consistent results
- Ensure only the right artifacts are accessed by the right credentials at the right time
- Promote more reliable rollbacks if a deployment fails
- Perform Continuous Deployments from an immutable, and approved set of binaries

Disadvantages

- On **prem** version needs to be managed
- Could be expensive