



10 REASONS TO USE A BINARY REPOSITORY MANAGER WHEN DEVELOPING WITH



Introduction

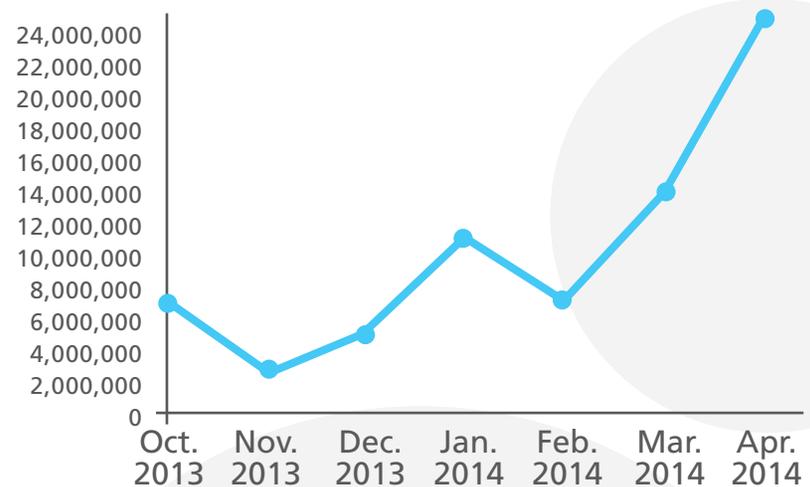
Over the last several years software development has evolved from focusing on in-house coding to making extensive usage of binary components such as free open-source libraries and commercial libraries as well as proprietary libraries built in-house. Already in a survey conducted back in 2010, 98% of enterprise level companies reported that they use open source software¹, and a typical software project today will be comprised of more assembled components than proprietary code.

Companies developing on the .NET framework are no exception. Since its introduction in 2010, NuGet Gallery usage has grown showing over 11 Million packages downloaded in January 2014 skyrocketing to over 25 Million packages in April². While there are many benefits to using NuGet³, there are still several challenges that it does not address. Some of these are:

- Long and network intensive build processes
- Offline access to packages and metadata
- Security and access control for NuGet packages
- Sharing internal and external NuGet packages
- Binary version tracking to reproduce builds
- Searching for packages based on different criteria
- Stability and reliability of systems hosting NuGet packages
- Customized handling of packages
- Maintenance and monitoring of artifacts

Artifactory is a Binary Repository Manager that manages all NuGet packages within your organization, whether developed in-house, downloaded from the NuGet Gallery or downloaded from other 3rd party resources. This white paper describes how Artifactory addresses the above issues substantially reducing development and build time while requiring very little effort from your organization.

Packages Downloaded Per Month



01

Reduce network traffic and optimize builds

Since much of your code is likely to be assembled rather than built, you want to make sure that your usage of packages downloaded from the NuGet Gallery is optimized. It makes no sense for two (or two hundred) developers using the same package to download it separately.

Artifactory is an intermediary between developers and the NuGet Gallery, and handles it as a **remote repository**. Once a package has been downloaded, Artifactory stores it in a local cache. Upon receiving subsequent requests, Artifactory performs a smart checksum search for the requested package, and if it has already been downloaded, then the locally cached copy is provided. Therefore, any package is only downloaded once and is then locally available to all other developers in the organization (thus reducing network traffic). Naturally, this is all transparent to the individual developer. Once the NuGet client is configured to access packages through Artifactory, the developer can get on with what she does best and leave the package management to Artifactory.

If we look at network traffic from the point of view of a build server, the benefits are clear. A typical project may need tens if not hundreds of packages from the NuGet Gallery. For the server to build these projects, all the packages must be downloaded and made available to the server environment which may generate Megabytes (if not Gigabytes) of data traffic on the network. Downloading all these required packages takes a significant amount of time which delays the build process. By caching all those packages locally, the build process is much quicker and incurs much less networking.

Remote Repositories

A remote repository serves as a caching proxy for a repository managed at a remote site, such as the NuGet Gallery or other NuGet repositories. Artifacts are stored and updated in remote repositories according to various configuration parameters that control the caching and proxying behavior.

[Learn more >](#)

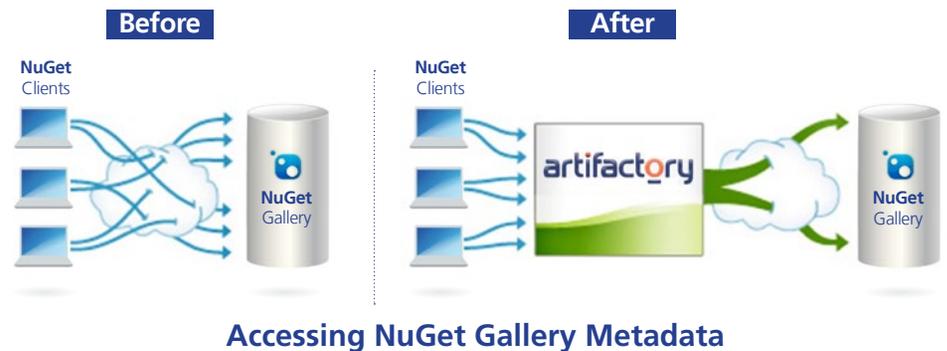
02

Reliable access to NuGet Gallery

As .NET developers, NuGet Gallery is an invaluable resource that you need on a regular basis. But since NuGet Gallery is a remote resource, what do you do if it goes down or if there is an issue with the network. And what happens if “you” are actually a **build server**?

Artifactory provides rich and extensive support for the NuGet API, and is therefore transparent to developers working with NuGet Gallery. To Artifactory, the Nuget Gallery is just another remote repository. When you request a NuGet package or metadata, Artifactory can provide it from the local cache effectively screening you from any issues with the Gallery or the network. You will

always have access to the packages in your system, and your builds won't be held up by issues with the network or the NuGet Gallery.



Build Server Integration

Artifactory has tight integration with MSBuild projects and fully supports the most popular CI systems available today such as Jenkins, Bamboo and TeamCity. The build server uses Artifactory to resolve dependencies when creating the build, but also uses Artifactory as a target to deploy build output to the corresponding local repository. Artifactory takes this one step further, and provides full visibility of deployed artifacts, dependencies, environment and system properties to support fully reproducible builds.

[Learn more >](#)

03

Full support for Docker

As Docker technology continues to evolve, its usage continues to grow. If you are not yet using Docker in your organization, it is likely you will do so soon. So now, in addition to managing NuGet packages, you also need to manage Docker images. But there's no need to onboard and maintain another tool. Artifactory is a fully-fledged Docker repository supporting all Docker Registry APIs. This allows the Docker client to work with Artifactory directly, presenting several benefits for enterprise Docker users.

Using local repositories, you can **distribute and share images** within your organization to make managing images between different teams easy. You can even replicate your Artifactory Docker repositories to remote instances of Artifactory to share images with colleagues in geographically distant sites.

Artifactory offers **fine-grained access control** to your organization's images with secure "docker push" and "docker pull" effectively providing **secure, private Docker repositories** that exceed the security offered by Docker Trusted Registry.

Using Artifactory, instead of private repositories on Docker Hub, removes any issues related to internet connectivity resulting in **reliable and consistent access to images**. And with Artifactory running in a **High Availability configuration** you get system stability and availability of your Docker images that is unmatched in the industry.

Artifactory's **smart search** makes it easy to find any Docker image stored in your system. Full support for the Docker Registry API supports basic search with the Docker client, but Artifactory offers much more. Built in searches answer common needs with single-click operations, custom properties provide the flexibility to meet a variety of specific needs, and Artifactory Query Language offers a simple way to formulate complex queries letting you find images based on any set of criteria.

Whether you're already on board with Docker or just evaluating how to introduce it to your organization, once you're using Artifactory to manage your NuGet packages, you're already covered for Docker images.

04

Security and Access Control

Every organization needs to implement security policies so that people can only access internal resources that they are authorized to use. But how do you control what people in your organization download from the Nuget Gallery or other external sources? How do you control which external resources are accessed in the first place? And then, how do you control where people in the organization put different packages they downloaded or are working on?

Artifactory can provide security and access control at different levels. From complete repositories down to a single artifact, and from a group of any size down to a single developer.

As a first line of defence, Artifactory supports virtual repositories. By going through virtual repositories you can ensure that your developers only access reliable 3rd party resources that have been approved. You can also optimize package resolution by defining the underlying repository order so that Artifactory will first look through local repositories, then remote repository caches, and only then Artifactory will go through the network and request the package directly from the remote resource. For the developer it's simple. Just request the package by name, and Artifactory will safely and optimally access it according to your organization's policies.

For more fine-grained access control, Artifactory also provides a flexible

mechanism to define "Excludes" or "Includes" for download using regular expressions. This lets you define anything from a whole repository to be excluded from your organization's access, to including a single artifact within a repository which may be critical for your development efforts. Once you have decided what can be downloaded to your servers, you can then define which users or groups, as defined in your Active Directory server, can access it. Through this comprehensive set of security and access control capabilities, Artifactory helps you manage your development process by ensuring that downloaded packages are safe and developers can only deploy packages to repositories for which they are authorized to. For example you can ensure that developers can deploy release targets to a QA repository, but only authorized QA staff, who have ensured that a release candidate has met the required standard, can promote it to the "releases" repository.

Virtual Repositories

A virtual repository encapsulates any number of local and remote repositories and represents them as a unified repository accessed from a single URL. Packages requested are resolved from the underlying repositories in the order in which they are defined in the virtual repository.

[Learn more >](#)

05

Share proprietary packages across your organization with local repositories

As already mentioned, most of your product is likely to be assembled from components, however you still want to make the most of your proprietary code. If you create a package, you want to be able to easily share it with other developers in your team and across your organization.

Artifactory lets you create **local repositories** where you can deploy all of your proprietary NuGet packages for easy access. Developers only need to configure their environments once to access NuGet packages through Artifactory, and from then on it is transparent. All requests for a specific package will go through Artifactory. The developer just specifies which package is needed, and Artifactory knows where to find it, and will always provide the same single copy stored in the local repository. But what if you want to share your packages with colleagues who are in geographically remote sites of your organization?

Artifactory supports replication of your repositories to another instance of Artifactory which is outside of your local network. Replicated repositories are automatically synchronized with their source periodically so that your artifacts can be made available to different teams wherever they may be located around the world.

Local Repositories

Local repositories are physical, locally-managed repositories into which you can publish artifacts. Typically these are used to publish internal and external releases as well as development build artifacts. Using local repositories, all of your internal NuGet packages can be made available across your organization from a single access point.

[Learn more >](#)

06

Fully reproducible builds

“Bugs”, “defects” or “issues”, however you refer to them - they haunt us. And sometimes, they only turn up after we have released them to the world, and then fixing them becomes urgent. But then, to debug an issue, we want to be able to reproduce the specific release on which it was found. Given the modules we develop and download, along with all the ensuing dependencies, and build environment, that may be a daunting task.

Artifactory provides built-in version tracking by storing exhaustive build information which makes it easy to faithfully reproduce any build. Information stored includes specific package versions, dependencies, system properties, environment variables, user information, timestamps and more. But Artifactory gives you even more than that. With built-in “Build Diff” tools you can compare builds and therefore know exactly what changes were introduced to the version in which the bug was reported.



07

Smart search for packages using build number and custom properties

Given the multitude of packages in your system, finding something specific can sometimes get quite complex.

Artifactory provides you with flexible search capabilities that let you find packages based on any combination of inherent properties such as name, version, timestamp, checksum and more. Artifactory also provides some common built-in searches. For example, you can ask Artifactory for the “latest” version of any package without having to specify a particular version number. Artifactory knows how to compare all the different versions of a package in any of its repositories and provide the latest one available. Artifactory’s build search takes this a step further and lets you search for packages by build number, very much like using the version tag assigned to source files in source code control systems.

This powerful feature enables you to find all the specific artifacts that went into any build according to the build number. But the full power of smart search comes with the flexibility that Artifactory provides you with custom properties that you can

assign to your packages, and then use in your searches. For example, you could define a property to classify the status of build artifacts indicating if they have completed QA or not. Then, when deciding which artifacts to upload to production, you could make sure that your search only provides packages that have been approved by your QA team. Artifactory’s smart search, gives you the flexibility to search for packages using virtually any set of rules relevant to your workflow.

Checksum-based search

Searching for a package by its checksum is a powerful feature supported by Artifactory thanks to a unique method of storing files by their checksum. Even if a package has been renamed, moved or even deployed outside of your organization, you can trace it back to the original version and obtain its complete build information. Simply run the package through a checksum tool (both MD5 and SHA1 are supported) and run a “Checksum” search in Artifactory to retrieve the original version.

[Learn more >](#)

System stability and reliability with Artifactory High Availability (HA)

Playing such a central role in development of your products, the servers hosting your NuGet packages (whether you downloaded them or developed them) can become mission-critical components of your organization meaning that any downtime can have severe consequences.

Artifactory supports a High Availability configuration with a cluster of 2 or more Artifactory servers on the same Local Area Network. This redundant network architecture means that there is no

single-point-of-failure, and your system can continue to operate as long as at least one of the Artifactory nodes is operational. This maximizes your uptime and can take it to levels of up to “five nines” availability. Your system can also accommodate larger load bursts with no compromise to performance. With horizontal server scalability, you can easily increase your capacity to meet any load requirements as your organization grows, and by using an architecture with multiple servers, Artifactory HA lets you perform most maintenance tasks with no system downtime.

High Availability Systems

Systems that are considered mission-critical to an organization can be deployed in a **High Availability** configuration to increase stability and reliability. This is done by replicating nodes in the system and deploying them as a redundant cluster to remove the complete reliability on any single node. In a High Availability configuration there is no single-point-of-failure. If any specific node goes down the system continues to operate seamlessly and transparently to its users through the remaining, redundant nodes with no down time or degradation of performance of the system as a whole.

[Learn more >](#)

Advanced package management with “watches” and user plugins

While organizations may share common best practices in how they manage their packages, each organization is different and may have some unique requirements.

In Artifactory you can define “watches” which let you closely monitor any of the critical packages in your system. In case a package is accidentally deleted or moved, you can receive an immediate alert to restore it as required. But Artifactory also provides much more flexible and advanced package management capabilities.

Artifactory generates an event for virtually every action performed on a package including download, deployment, moving, copying and more. For every such event you may write a plugin that will perform any action required by your organization’s policies. For example, each time a package is downloaded from the NuGet Gallery you could run a virus check to make sure your systems do not get infected with something nasty.

User Plugins

User Plugins give you a way to extend the functionality of Artifactory with any custom behavior required by your organization. Plugins are written in Groovy giving you complete flexibility to implement behavior such as manipulating downloaded content, querying security information, executing promotion logic and more. During development, plugin source files can be changed and redeployed on-the-fly, and they can even be debugged using industry standard IDEs.

[Learn more >](#)

10

A universal, end-to-end solution for all binaries

No single packaging format or technology is sufficient to support development in a modern organization. There is a multitude of formats, a variety of build tools, different continuous integration systems and other technologies that go into building a flexible and maintainable software development ecosystem. Managing binaries for all the different packaging formats and integrating with all the moving parts of the ecosystem can become a maintenance nightmare.

Artifactory was designed from the ground up to fit in with any development ecosystem. Uniquely built on checksum-based storage, Artifactory supports any repository layout and can, therefore, provide native-level support for any packaging format. Essentially, regardless of the packaging format you are using, Artifactory can store and manage your binaries, and is transparent to the corresponding packaging client. The client works with Artifactory in exactly the same way it would work with its native repository. For example, if you are working with Docker, Artifactory proxies Docker Hub (or any other public Docker registry), lets you store and manage your own images in local Docker repositories, and works transparently with the Docker client. If you are working with .NET, Artifactory proxies NuGet Gallery (or any other public NuGet repository), lets you store your own packages in local NuGet repositories, and works transparently with the NuGet client. Similarly for Vagrant, npm, Ruby, Debian, YUM, Bower, Python and more.

But development is only one end of the software delivery pipeline. Before a package makes it into a product, it needs to go through processes of build and integration. There are many build and integration

tools on the market, but there is only one product that works with them all. Through a set of plugins, Artifactory provides tight integration with popular CI systems available today such as Jenkins, Bamboo and TeamCity. These systems use Artifactory to supply artifacts and resolve dependencies when creating a build, and also as a target to deploy build output. And to support cloud-based CI systems on which you are not able to apply plugins, Artifactory provides plugins for the build tools you use (such as Maven and Gradle) which ultimately provides the same level of build automation. That takes care of development and deployment, but what about distributing your software once it's ready for consumption. That's where Bintray comes in.

Bintray is JFrog's download center in the cloud offering rapid downloads, fine-grained access control, detailed stats and logs and an extensive REST API. Promoting releases for distribution from Artifactory is a matter of a single-click or API call. Like Artifactory, Bintray is package-agnostic and works seamlessly with all the different package clients, so it can be fully integrated into any continuous integration/continuous delivery ecosystem.

Artifactory is a universal repository. It is the single tool that sits in the center of your development ecosystem and "talks" to all the different technologies, increasing productivity, reducing maintenance efforts and promoting automated integration between the different parts. Together, Artifactory and Bintray are the central components of a fully-automated software distribution pipeline.

Summary

This paper has shown how a Binary Repository Manager such as Artifactory can reduce development and build time while requiring very little effort from your organization. This is done by managing packages, ensuring optimal and reliable access to the NuGet Gallery and more. This is all available either through a comprehensive and intuitive UI

or through a rich and extensive set of APIs supporting build automation. And since Artifactory is agnostic to the artifacts that it manages, it can act as a single access point not only for NuGet packages, but for all of your binary resources whether they come from RubyGems.org, Maven Central or virtually any other third party repository.

For more information on how Artifactory can boost your organization's performance please contact us at: info@jfrog.com

References

1. 2010 Open Source Systems Management Survey. (2010). [e-book] p. 2. Available through: <http://community.zenoss.org/blogs/zenossblog/2010/08/10/2010-trends-in-open-source-systems-management> <http://community.zenoss.org/servlet/JiveServlet/download/38-3009/OpenSourceManagement.pdf> [Accessed: 24 Mar 2014].
2. Nuget.org, (2013). NuGet Gallery | Statistics. [online] Available at: <http://www.nuget.org/stats> [Accessed 11 May. 2014].
3. Grace, A. (2012, 9th May, 2012). An Introduction to NuGet. Coders Barn, [web log] Retrieved from: <http://codersbarn.com/post/2012/05/09/An-Introduction-to-Nuget.aspx> [Accessed: 24 Mar 2014].