12 REASONS

to use a Binary Repository Manager When Developing with python™

White Paper

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 and commercial libraries as well as proprietary libraries built in-house. The emergence of tools that automate processes such as build tools and CI servers, have further helped to fire up the usage of components, and a typical software project today will be comprised of more assembled components than proprietary code. While this has obvious benefits in terms of costs and code quality, it also presents a number of challenges:

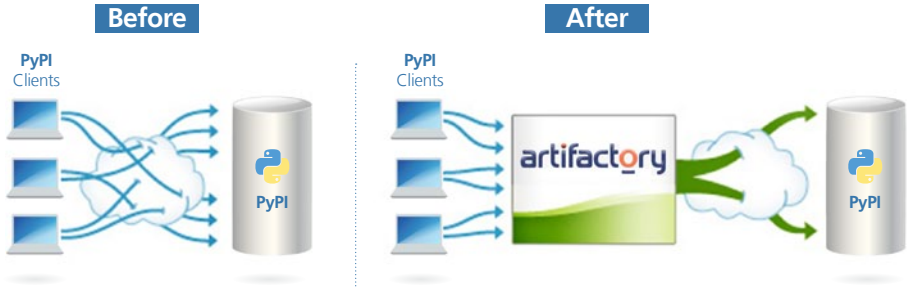
- How can you access components if your remote repositories become unavailable?
- How can you optimize long and network intensive build processes?
- How do you manage security and control access to components?
- How do you share components efficiently across your organization?
- How do you find a component once it has been downloaded by someone?
- How do you implement customized behavior around usage of components?
- How do you ensure compliance with a variety of license requirements?
- How do you ensure that both your proprietary components and downloaded components are always available to your teams?

The answer to all of these questions is Artifactory, a Binary Repository Manager that functions as a single access point organizing all of your binary resources including proprietary libraries, remote artifacts and other 3rd party resources, including PyPI in particular. Fully supporting standard Python tools such as pip and DistUtils, Artifactory transparently replaces your Python repositories to meet these challenges and boost your organization's productivity when developing with Python and open source libraries.

01 Reliable and consistent access to remote artifacts

As Python developers, remote repositories such as PyPI are invaluable resources that you need on a regular basis. But what do you do if these resources go down or if there is an issue with the network.

Artifactory is an intermediary between developers and external resources. As a developer, all of your requests are directed to Artifactory which gives you quick and consistent access to remote artifacts by caching them locally in a **remote repository**. While this sounds like an oxymoron, it actually does make sense. A "remote repository" in the context of Artifactory, refers to the local cache which is a proxy for those remote artifacts. Since remote artifacts are readily available from the cache on your local network, you are independent of external networking issues, and are not affected if the remote resource goes down. Even in the extreme case that a remote resource ceases to exist altogether, any artifacts already downloaded to the local cache are still available to you. As a developer, you can continue your development efforts, and your builds won't be hampered by network issues or a repository going down.



Accessing Python Package Repositories

Remote Repositories

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

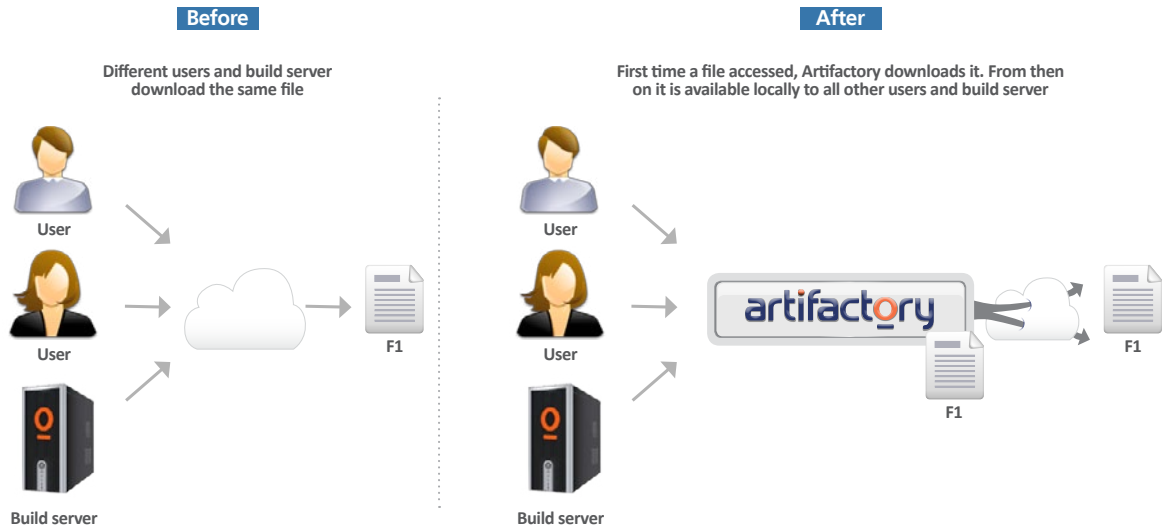
Learn more >

02 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 PyPI or other Python resources is optimized. It makes no sense for two (or two hundred) developers using the same package to download it separately.

In addition to reliability, another benefit of remote repositories is reduced networking. Once an artifact has been downloaded, it 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 artifacts are accessed through Artifactory, the developer can get on with what she does best and leave the binary management to Artifactory.

If we look at network traffic from the point of view of a build script, the benefits are clear. A typical project may depend on tens if not hundreds of artifacts from external resources. For the tool to build these projects, all remote artifacts must be available to the server environment. Downloading all those required artifacts may generate Gigabytes of data traffic on the network which takes a significant amount of time delaying the build process. By caching remote artifacts locally, the build process is much quicker and incurs much less networking.



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 Python 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 Python packages, you're already covered for Docker images.

04 Full integration with your build ecosystem

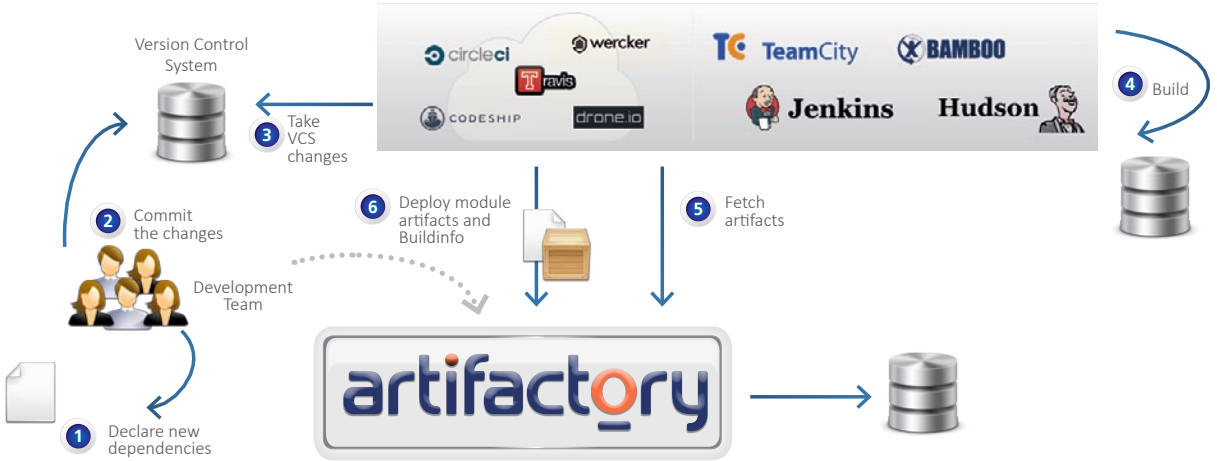
While it's important to make it easy and efficient for your developers to access binary artifacts, it's even more important for your build systems which may be running builds many times a day.

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 the build, and also as a target to deploy build output to the corresponding local repository.

One of the main benefits of running builds through Artifactory is fully reproducible builds. Artifactory stores exhaustive build information including specific artifact versions, modules, dependencies, system properties, environment variables, user information, timestamps and more. With this information, it is easy to faithfully reproduce a build at any time. Moreover, with built-in "Diff" tools you can compare builds and therefore know exactly what changes were introduced from one version to another. These capabilities can be invaluable when trying to track down bugs that were reported in specific versions released.

Artifactory also simplifies release management. A series of simple settings configure things like staging, build promotion, VCS tagging and more, essentially automating the release management process.

But what happens if your are using cloud-based CI systems where you can't apply plugins? In that case, Artifactory provides plugins directly for the build tools themselves, which ultimately provides the same level of build automation. Essentially, since Artifactory is platform agnostic it can be integrated with generic tools across all the build ecosystems within your organization. Finally, once your builds are automated, Artifactory will keep your system free of clutter by cleaning up old builds according to your organization's maintenance policies.



05 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 external resources? 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 artifacts they downloaded or are working on?

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

As a first line of defense, 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 such as PyPI. For more fine-grained access control, Artifactory lets you use naming patterns with wildcard characters to define "Excludes" or "Includes" for download. With this flexible mechanism you can 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 of users can access it with a full set of permissions you can configure. In addition to controlling download of artifacts, you can use permissions for complete access management. You can control where developers can deploy artifacts to, whether they can annotate metadata, whether they can delete artifacts and more. And if it's access to your servers that you're concerned about, Artifactory provides full integration with the most common access protocols such as LDAP, SAML, Crowd and others. The comprehensive security and access control capabilities in Artifactory help you manage your development process by ensuring that developers can only access 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. It gives you a way to manage which repositories are accessed by developers since you have the freedom to mix, match and modify the actual repositories included within the virtual repository. You can also optimize artifact 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 artifact directly from the remote resource. For the developer it's simple. Just request the package, and Artifactory will safely and optimally access it according to your organization's policies.

Learn more >

06 Distribute and share artifacts across your organization

While much of your product is likely to be assembled from components, 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.

Using **local repositories**, Artifactory gives you a central location to store your internal binaries. When all teams know that any artifact can be accessed from a single URL, access to local artifacts and managing dependencies between the different teams becomes very easy.
But what if you want to share 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 packages 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 deploy artifacts. Typically these are used to deploy internal and external releases as well as development builds, but they can also be used to store binaries that are not widely available on public repositories such as 3rd party commercial components. Using local repositories, all of your internal resources can be made available from a single access point across your organization from one common URL.

Learn more >

07 Smart search for binaries 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 both through the UI, and using the extensive REST API. You can find packages based on any combination of inherent attributes 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 build 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 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 packages 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 binaries that have been approved by your QA team. With all these capabilities, Artifactory's flexible smart search lets you 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. 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 >

08 User Plugins

While Artifactory provides an extensive set of features to manage binary resources, it's impossible to accommodate all the requirements that different organizations may have. Enter **user plugins**.

User plugins present a long list of entry points which effectively extend the Artifactory REST API providing a simple way to implement complex behavior. This gives you enormous freedom to support virtually any custom requirement in your workflow including scheduling tasks, managing security and authentication, deployment, build integration and promotion logic, maintenance and cleanup and more. To keep things simple, user plugins are written as Groovy scripts and have a simple DSL to wrap them as closures within the extension points. The plugins can be changed and redeployed on-the-fly, and can even be debugged- all from within your favorite IDE.

09 License compliance and open source governance

Ever been held up because someone suddenly realized that there are a bunch of licensing requirements to which you must comply?

Artifactory can help prevent such a scenario. Upon deployment of any package to your repositories Artifactory performs a license check on your artifact and on all ensuing dependencies, and provides immediate feedback on all license requirements. This lets you prepare ahead of time to ensure you comply early on in the development cycle avoiding unnecessary delays at “crunch time” when you want to release. And by integrating with Black Duck Code Center, you can take advantage of a full range of license compliance and open source governance features while managing all of your binary uploads through Artifactory.

10 System stability and reliability with Artifactory High Availability

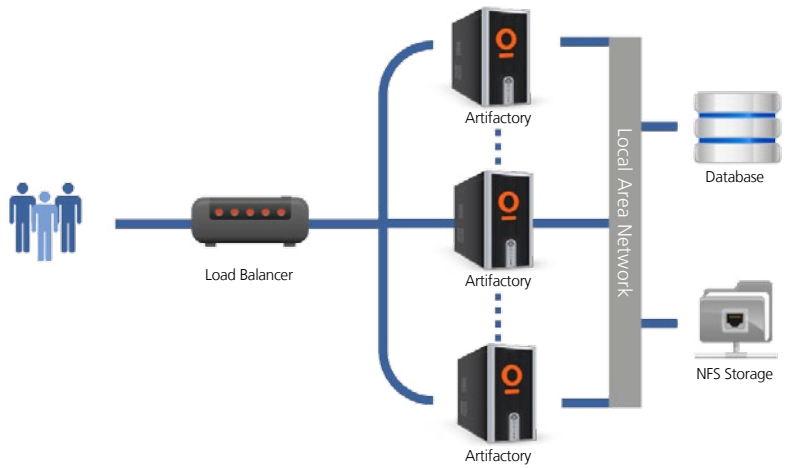
Playing such a central role in the management of packages, your Binary Repository Manager can become a mission-critical component of your organization. Any downtime can have severe consequences to your productivity, and you need to ensure developers can access your Python repositories at all times.

Artifactory supports a **High Availability** network configuration with a cluster of 2 or more Artifactory servers on the same Local Area Network. A 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. Moreover, your system can 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. Finally, 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 system performance as a whole.

Learn more >



11 Maintenance and Monitoring

The number of packages you generate can grow very quickly. Without proper management, your systems can quickly get clogged with old and irrelevant packages.

Artifactory keeps your system organized and free of clutter with automatic, timed cleanup processes. With a few simple settings, you can schedule tasks to clean up old builds and unused packages. You can set restrictions on and monitor disk space usage or define “watches” to receive an alert whenever there is a change to your most critical packages. And with an extensive REST API, Artifactory can support virtually any rule-based cleanup protocol you would want to implement in your organization’s scripts.

12 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 Python, Artifactory proxies PyPI (or any other public Python repository), lets you store your own packages in local Python repositories, and works transparently with the pip client. Similarly for Vagrant, NuGet, npm, Ruby, Debian, YUM, Bower 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 boost productivity of your organization's development and DevOps teams by managing and optimizing access to Python packages. Since Artifactory is agnostic to the binary types that it manages, it can work with any Python package and repositories for virtually any other binary formats such as RPMs, RubyGems, NuGet and more to support the different platforms used by your organization, and provide central control over all binaries, both when uploading builds and when downloading artifacts.

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