Sonatype

Repository Management: Stages of Adoption

Adopting Repository Management Best Practices

Introduction

Organizations that use third-party and/or open source software in their application development are increasingly adopting component lifecycle management strategies to gain control over the software components they use they acquire. Repository managers are a fundamental building block to this strategy, used to store and manage binary software artifacts, similar to the way source control management tools like Git and Subversion store and manage source code. Driven by the growth of using open source components as part of applications, repository managers are quickly gaining advanced capabilities to support the modern software development lifecycle. This paper focuses on Sonatype Nexus Pro the industry's most widely adopted repository manager.

Adopting a repository manager is not an all or nothing proposition, and there are various levels (or stages) of adoption that can be distinguished when approaching repository management. On one end of the adoption spectrum is the organization that installs a repository manager just to improve build speeds, control and consolidate access to a set of remote repositories. On the other end of the spectrum is the organization that has integrated the repository manager into a broader component lifecycle management strategy to improve visibility and control across their software lifecycle.

This document outlines the stages of adoption, and provides organizations with a roadmap for adopting best practices in repository management for each phase. The phases are proxying, hosting, and lifecycle integration. Proxying and hosting can be started without any disruption or significant process change, and as the organization progresses to lifecycle Integration the benefits increase.

In addition to outlining the stages of adoption, this document also provides guidance on repository manager adoption.

Stage Zero: Before Using a Repository Manager

While this isn't a stage of adoption, Stage Zero is a description of the way software builds work in the absence of a repository manager. When a developer decides that a he needs a particular open source software component, he will download it from the component's website, read the documentation, and find the additional software that his components rely on (referred to as "dependencies"). Once he has manually assembled a collection of dependencies from various open source project web sites and proprietary vendors, he will place all these components somewhere on the network so that he, his team members, the build script, the QA team, and the production support team can find it. At any time, other developers may bring in other components, sometimes with overlapping dependencies, placing them in different network locations. The instructions to bring all of these ad-hoc, developer managed components libraries together in a software build process can become very complicated and hard to maintain.

Maven and other build tools were introduced to improve the build process by introducing the concept of structured repositories from which the build scripts can retrieve the software components. In Maven language, these software components or dependencies are referred to as "artifacts", a term which can refer to any generic software type, including components, libraries, frameworks, containers, etc. Maven can identify artifacts in repositories, understand their dependencies, retrieve all that are needed for a successful build, and deploy its output back to repositories when done.

Driven by the adoption of Maven, public repositories of open source software artifacts have sprung up and become very widely used. The canonical Central Repository contains the vast majority of all open source Java software and receives more than 10 billion requests per year from more than 100,000 organizations.

Developers using Maven and other build tools without a repository manager find most of their software artifacts and dependencies in the Central Repository. If they happen to use another remote repository or if they need to add a custom artifact, the solution, in Stage Zero, is to manually manipulate the files in a local repository and share this local repository with multiple developers. While this approach may yield a working build for a small team, managing a shared local repository doesn't allow an organization to scale a development effort. There is no inherent control over who can set up a local repository, who can add to them or change or delete from them, nor are there tools to protect the integrity of these repositories.

That is, until repository managers were introduced.

Stage One: Proxying Remote Repositories

This is the easiest stage to understand both in terms of benefits to an organization and action required to complete this stage. All you need to do to start proxying a remote repository is to deploy the Nexus repository manager and start the server with the default configuration. Configure your Maven clients to read from the Nexus public repository group, and Nexus Pro will automatically retrieve artifacts from remote repositories, such as the Central Repository, caching them locally.

While proxying saves both bandwidth and time, it also consolidates access to external resources to a local repository manager. Without a repository manager, your organization might have hundreds of developers independently downloading the same artifacts from public, remote repositories. With a repository manager, these artifacts can be downloaded once and stored locally. In addition to the savings in bandwidth, you will also notice that your builds run considerably faster.

To understand the time savings, one must understand how builds interact with a repository. When a Maven build is running, Maven is continuously interacting with a repository whenever it needs to download an artifact, read artifact metadata, or check for newer versions of dependencies. Often, running a local Nexus repository manager can improve the response time for artifacts and metadata by a factor of ten or more. More responsive, faster builds often translate to better tested software and more productive developers. If you are looking for a faster, more responsive build to facilitate a more efficient development cycle the easiest way to achieve this goal is to install a local repository manager.

Once you've installed Nexus Pro and you've configured all of your organization's clients to use it as a single point of access to remote repositories, you begin to realize that it now provides you with a central configuration point for the artifacts used throughout your organization. Once you've started to proxy, you can start to think about using Nexus Pro as a tool to control policy and what dependencies are allowed to be used in your organization. Nexus Pro provides a procurement plugin that allows for fine-grained control over which artifacts can be accessed from a remote repository. This procurement feature is described in more detail in the section that deals with Lifecycle Integration.

Stage Two: Hosting a Repository Manager

Once you have started to proxy remote repositories and you are using Nexus Pro as a single, consolidated access point for remote repositories you can start to deploy your own artifacts to Nexus Pro hosted repositories. Most people approach repository management to find a solution for proxying remote repositories, and while proxying is the most obvious and immediate benefit of installing a repository manager, hosting internally generated artifacts tends to be the stage that has the most impact on collaboration within an organization.

To understand the benefits of hosting an internal repository, one must understand the concept of managing binary software artifacts. Software development teams are very familiar with the idea of a source code repository or a source code management tool. Versioning control systems such as Subversion, Clearcase, Git, and CVS provide solid tools for managing the various source artifacts that comprise a complex enterprise application, and developers are comfortable checking source out from source control to build enterprise applications. However, past a certain point in the software development lifecycle, source artifacts are no longer relevant. A QA department trying to test an application or an operations team attempting to deploy an application to a production network no longer needs access to the source code. QA and Operations are more interested in the compiled end-product of the software development lifecycle: the binary software artifacts. A repository manager allows you to version, store, search, archive, and release binary software artifacts often derived from the source artifacts stored in a source control system. A repository manager allows you to apply the same systematic operations on binary software artifacts that you currently apply to your source code.

When your build system starts to deploy artifacts to an internal repository, it changes the way that developers and development groups can interact with one another in an enterprise. Developers in one development group can code and release a stable version of an internal library, deploy this library to an internal Nexus release repository, and so share this binary artifact with another group or department. Without a repository manager managing internal artifacts, the relationship between two development groups would consist of a series of ad-hoc, manual communications between independent workgroups. With a repository manager, every developer and every development group within the enterprise understands and interacts with a common collaborative structure: the repository manager. Do you need to interact with the Commerce team's new API? Just add a dependency to your project and Maven will retrieve the library from Nexus Pro automatically.

Developing this collaborative model further, if your application is being continuously built and deployed using a tool like the Hudson continuous integration server, a developer can checkout a specific module from a large multi-module build and not have to constantly deal with the entire source tree at any given time. This allows a software development effort to scale efficiently. If every developer working on a complex enterprise application needs to checkout the entire source tree every time he or she needs to make a simple change to a small component, you are quickly going to find that building the entire application becomes a burdensome bottleneck to progress. The larger your enterprise grows, the more complex your application becomes, the larger the collective burden of wasted time and missed opportunities. A slow enterprise build prevents the quick turnaround or quick feedback loop that helps your developers maintain focus during a development cycle.

One of the other direct benefits of deploying your own artifacts to a repository such as Nexus Pro is the ability to quickly search the metadata and contents of those artifacts both via a web UI and through IDE integration tools such as m2eclipse. When you start to deploy internal artifacts you can synchronize all development groups to a common versioning and naming standard, and you can use the highly configurable authentication and role-based access controls to control which developers and which development groups can deploy artifacts to specific repositories or paths within a repository.

Stage Three: Lifecycle Integration

Once you've configured a repository manager to proxy remote repositories and you are using a repository manager as an integration point between developers and departments, you start to think about the various ways your repository manager can be used to support the decisions that go into software development. You can start using the repository manager to stage releases and supporting the workflow associated with a managed release, and you can use the procurement features of a tool like Nexus Pro to give management more visibility into the origins, characteristics and open source licenses of the artifacts used during the creation of an enterprise application.

Nexus Pro enables organizations to integrate the management of software artifacts tightly with the software development lifecycle: provisioning, compliance, procurement, enterprise security, staging and other capabilities that support the workflow that surrounds a modern software development effort.

Provisioning

- Using Nexus Pro as an integration point between Engineering and Operations means that Engineering can be responsible for delivering solid, tested artifacts to Quality Assurance and Operations via a standard repository format. Often development teams are roped into the production deployment story and become responsible for building entire production environments within a build system. This conflates software engineering with system administration and blurs the line between Engineering and Operations. If you use Nexus Pro as an end-point for releases from Engineering, Operations can then retrieve, assemble, and configure an application from tested components in the Nexus repository.

Compliance

- Procurement, staging, and audit logs are all features that increase the visibility into who and what is involved with your software development effort. Using Nexus Pro, Engineering can create the reports and documents that can be used to facilitate discussions about oversight. Organizations subject to various regulations often need to produce a list of components involved in a software release. Legal departments often require a list of open source licenses being used in a particular software component, and managers often lack critical visibility into the software development process.

Procurement

- The ease with which today's developer can add a dependency on a new open source library and download this library from a central repository has a downside. Organizations large and small are constantly wondering what open source libraries are being used in applications and whether these libraries have acceptable open source licenses for distribution and are safe to use. The procurement features of Nexus Pro give architects and management more oversight over the artifacts that are allowed into an organization. Using the procurement features, a Nexus administrator or procurement manager can allow or deny specific artifacts by group, version, or path. You can use the procurement manager as a firewall between your own organization's development environment and the 400,000+ artifacts available in the Central Repository.

Enterprise Security

- Nexus Pro's LDAP integration allows an enterprise to map existing LDAP groups to Nexus roles and provides Nexus administrators with a highly configurable interface to control which individuals or groups have access to a fine-grained set of Nexus permissions.

Staging

- Nexus Pro adds an important step to the software release workflow, adding the concept of a managed (or staged) release to a hosted repository. When a developer needs to perform a production release, Nexus Pro can isolate the artifacts involved in a release in a staged repository that can then be certified and tested. A manager or a quality assurance tester can then promote or discard a release. The staging feature allows you to specify the individuals that are allowed to promote a release and keeps an audit of who was responsible for testing, promoting, or discarding a software release.

Stage Four: Beyond the Repository Manager

Sonatype Nexus Pro is a core foundation of a component lifecycle management strategy. Nexus Pro provides increased control of components in the repository and enables effective collaboration among teams. Many organizations wish to extend component management across the entire software development lifecycle. Sonatype CLM builds on Nexus Pro to enable visibility and control from design, through development and build, an in to production.

Conclusion

By reducing build times, adding control, and improving collaboration, a repository manager is a must for modern software development organizations. Nexus Pro has long been the industry's most widely used repository manager. With rich enterprise features and 24x7 support, Nexus Pro is the ideal foundation for component lifecycle management.

For more information about Sonatype Pro ™ for Nexus go to: www.sonatype.com/nexus

To try a live demo of Sonatype Pro ™ for Nexus go to: www.sonatype.com/nexus-demo

Sonatype

12501 Prosperity Drive • Suite 350
Silver Spring, MD 20904
1.877.866.2836 • www.sonatype.com