



Safeguarding the Software Supply Chain:

Best Practices for Nexus Repository

Executive Summary

[Sonatype Nexus Repository](#) isn't just a development convenience. It gives development teams fast, reliable access to the components they need while helping keep unsafe components out of builds. Misconfigured or poorly governed repositories can introduce rework, disrupt pipelines, delay releases, and expose organizations to [malicious attacks](#) and compliance failures.

Nexus Repository combines centralized artifact management with advanced security capabilities, including proactive blocking of malicious and suspicious packages, role-based access control, TLS encryption, SAML/SSO support, audit logs, artifact immutability, and credential encryption.

This guide outlines key practices for keeping developers productive, builds reliable, and releases moving while reducing risk and scaling repository operations. Each practice is backed by Sonatype's real-world data and experience securing tens of thousands of software pipelines.

Why Nexus Repository Matters More Than Ever

Your [artifact repository](#) is the gateway through which every third-party component enters your development ecosystem. Every component choice can affect build stability, developer productivity, and release speed. Nexus Repository provides a centralized point of control that gives teams consistent access to trusted components while protecting artifact integrity and traceability.

Data Point: In 2025, Sonatype found repository abuse appeared in 55.9% of all malicious packages logged, showing how attackers increasingly use public registries as distribution platforms.

[Modern DevOps](#) depends on fast, reliable access to trusted components. Treat Nexus Repository as production infrastructure. It directly affects whether teams can build, test, and ship software without unnecessary delay or uncertainty.

Best Practices

1. Block Unsafe Components Before They Slow Delivery

Developers and AI agents now discover, select, and introduce open source components at increasing speed. Without automated controls, one bad component can create broken builds, remediation work, and release delays. Blocking malicious, vulnerable, and noncompliant components at the repository boundary keeps unsafe software out while allowing teams to continue working at development speed.

Risk:

Malicious, suspicious, or noncompliant components can enter builds before developers can identify them. Once introduced, they can affect multiple applications and environments, creating rework and delaying delivery.

Best Practice:

- ✓ Nexus Repository should act as a production infrastructure. It keeps developers, pipelines, and releases moving with reliable access to the components they need.
- ✓ Configure Nexus Repository's advanced security capabilities to evaluate component requests and automatically block malicious components before they reach developers or CI/CD pipelines.

Benefit:

Keep unsafe components out of development, reduce downstream remediation, and prevent component risk from disrupting builds and releases.

2. Eliminate Shadow Downloads

Pulling [dependencies](#) directly from public registries may seem like the fastest way to keep developers moving. But direct downloads bypass repository controls, create inconsistent sourcing practices, and make it harder to prevent poisoned packages from entering builds. Leading organizations route every download through a governed repository so developers have one fast, reliable source for trusted components.

Risk:

Developers and CI pipelines fetching components directly from public sources (e.g., npm, PyPI) bypass security controls, increasing the chance of malware, license violations, and audit gaps. Sonatype's Repository Health Check data shows that 50% of unprotected repositories already contain one or more pieces of malware. This demonstrates how quickly uncontrolled downloads can compromise an environment.

Best Practice:

- ✓ Route all downloads through Nexus Repository to give developers a consistent source for components and eliminate shadow risk.
- ✓ Protect build environments by restricting direct external registry access.
- ✓ Keep malicious packages out of shadow downloads with Nexus Repository's integration with Zscaler to block anything malicious at the edge

Benefit:

Give developers fast, consistent access to trusted components while blocking malicious packages before they disrupt builds.

3. Secure Access Without Slowing Teams

As development organizations scale, shared accounts and inconsistent permissions create both risk and administrative overhead. Centralized SSO and role-based access help the right people reach the artifacts they need without relying on manual access management.

Risk:

Weak access controls expose critical artifacts, slow onboarding and offboarding, and make repository activity harder to trace.

Best Practice:

- ✓ Use Nexus Repository's built-in role-based access control to limit users to authenticated, least-privilege roles.
- ✓ Integrate Nexus Repository with enterprise SAML/SSO or LDAP for centralized identity management.
- ✓ Enforce least privilege: no broad admin rights; assign deploy permissions per team.
- ✓ Minimize credential risk through token-based authentication with rotation.
- ✓ Use TLS encryption, encrypted stored credentials, and detailed audit logs to protect access and track repository activity.

Benefit:

Simplifies access management as teams scale, reduces credential-related risk, and gives developers reliable access to the artifacts they need.

Data Point:

Organizations with centralized governance over software components are significantly more likely to meet regulatory and audit readiness thresholds. Implementing RBAC and SSO helps streamline these controls.

4. Protect Build Continuity With Tested Recovery

A repository outage can stop development across every team that depends on it. Backups are only useful if they can be restored quickly, making documented and tested recovery processes essential to keeping builds and releases moving through infrastructure failures, ransomware events, and cloud outages.

Risk:	Best Practice:	Benefit:
A corrupted or unavailable repository halts development, breaks builds, and could permanently lose proprietary components.	✓ Automate nightly backups of blob stores and the database. ✓ Store backups offsite (cloud storage, secondary region). ✓ Test restore process annually; simulate failure scenarios. Document and periodically validate recovery steps.	Faster recovery, fewer stalled pipelines, and greater confidence that development can continue through an outage.

Data Point:

47% of organizations lack a tested recovery plan for their artifact repository. Those that do recover 2.5x faster after an outage.

5. Design for High Availability

A single-node repository creates a clear point of failure. As pipelines, distributed teams, and automated workflows multiply, dependable access to components becomes essential to keeping delivery moving. [High-availability architectures](#) are now the standard expectation for eliminating single points of failure and ensuring continuous delivery.

Risk:	Best Practice:	Benefit:
A single-node Nexus instance creates a critical single point of failure.	✓ Support uninterrupted availability with high-availability Nexus Repository architecture. ✓ Use an external PostgreSQL database and shared blob store. ✓ Separate nodes across infrastructure to avoid cascading failure.	Zero downtime during maintenance or node failure. Developers stay productive, pipelines stay green.

Data Point:

Resilient infrastructure, including HA configurations, is a key trait of mature software supply chain organizations. While downtime reduction varies, leaders prioritize redundancy and availability to meet SLA expectations.

6. Enforce Component Standards at the Repository Level

Manual reviews and occasional scans cannot keep pace with the volume of open source components flowing through modern development. They also force developers to wait for decisions or remediate issues after components have already entered their builds.

Nexus Repository blocks known malware and quarantines suspicious components based on an organization's risk thresholds. This applies standards at the point of request, keeping bad components out without adding another manual review to the development process.

Risk:	Best Practice:	Benefit:
Malicious, vulnerable, or noncompliant components can create broken builds, remediation work, and release delays when they enter development unnoticed.	✓ Configure organization-specific risk thresholds in Nexus Repository to automatically block known malware and quarantine suspicious components for review. ✓ Use Nexus Repository's built-in access controls, encryption, audit logging, artifact immutability, and traceability as the foundation of repository security. ✓ Periodically review quarantined components and policy violations, then take the appropriate remediation action.	Keeps bad components out, makes trusted choices easier, and reduces the remediation work that slows builds and releases.

Data Point:

Teams using automated policy enforcement and governance tooling remediate vulnerable components 2.7x faster than those relying solely on manual reviews.

7. Automate SBOMs as Release Artifacts

Until recently, [software bills of materials \(SBOMs\)](#) were viewed as optional documentation, produced only when specifically requested. Today, regulations and customer requirements increasingly make them part of the release process. Treating SBOMs as first-class artifacts and automating their generation prevents documentation work from becoming a last-minute delivery bottleneck.

Risk:	Best Practice:	Benefit:
Without current SBOMs, teams can struggle to answer customer questions, respond to incidents, or provide required release documentation on time.	✓ Generate SBOMs using CycloneDX or other supported tooling. ✓ Store SBOMs in a dedicated Nexus Repository. ✓ Automate SBOM generation for each release pipeline.	Keeps releases audit-ready without adding manual work, while accelerating incident response and customer reporting.

Data Point:

SBOM coverage is now a top-five audit item for U.S. federal suppliers (NTIA 2024).

8. Use Cleanup and Retention Policies Strategically

Letting repositories grow without limits was once viewed as harmless, with storage costs seen as negligible. Today, that approach increases infrastructure costs, slows backups and recovery, and leaves outdated components accessible long after their usefulness has expired. Automated cleanup and retention policies are now a standard safeguard against both inefficiency and risk.

Risk:	Best Practice:	Benefit:
Unbounded repository growth increases storage costs and keeps vulnerable components accessible.	✓ Reduce storage cost and security exposure using automated retention policies. ✓ Separate repositories by purpose: long-term releases vs. short-lived builds. ✓ Back up before cleanup; monitor deletion logs.	Lower storage spend, faster backups, reduced attack surface.

Data Point:

Customers implementing cleanup policies reduced blob store growth by 60% year-over-year (Sonatype support data).

9. Optimize Repository Performance and Organization

In many environments, repository design was historically left to evolve organically, with ad hoc structures tolerated as long as builds succeeded. But as teams scale and pipelines multiply, those inefficiencies turn into latency, developer frustration, and operational bottlenecks. Deliberate organization and performance tuning are now expected to keep delivery flowing at enterprise speed.

Risk:	Best Practice:	Benefit:
Poor structure increases latency and developer friction.	✓ Simplify dependency resolution by consolidating endpoints via repository groups. ✓ Avoid excessive proxies per group. Separate snapshots from releases. ✓ For Docker: use subdomain routing, not legacy ports.	Faster builds, simpler CI configuration, and better scaling.

Data Point:

Organizations using recommended grouping and routing practices saw a 3x reduction in average artifact resolution time.

10. Promote Artifacts Instead of Rebuilding

Rebuilding artifacts for every environment was once common practice, accepted as part of the delivery process. But this approach introduces variability, erodes traceability, and undermines confidence that testing matches production. Today, promoting validated artifacts across stages is the standard for ensuring consistency and maintaining a verifiable chain of custody.

Risk:

Rebuilding artifacts in each environment introduces variability and breaks traceability. This undermines testing and auditing, especially when artifacts differ between dev, staging, and production.

Best Practice:

- ✓ Use hosted repositories for each lifecycle stage (e.g., staging, QA, prod).
- ✓ Preserve release integrity by promoting verified artifacts between environments.
- ✓ Avoid rebuilding from source after initial validation, and promote instead.

Benefit:

Ensures the artifact tested is the artifact released, reducing inconsistencies and increasing confidence in every deployment.

Data Point:

Promoting binaries rather than rebuilding them is a foundational DevSecOps pattern used by organizations with mature software delivery pipelines.

11. Enforce Immutable Artifact Versions

Allowing artifacts to be overwritten or re-published under the same version once seemed like a harmless convenience. In practice, it breaks reproducibility, obscures what was actually deployed, and complicates incident response. Immutability ensures teams can reproduce builds, trust tested releases, and quickly determine what reached production.

Risk:

Mutable artifacts (e.g., re-publishing version 1.0.0) break build reproducibility and make it impossible to verify what was actually deployed.

Best Practice:

- ✓ Use Nexus Repository's support for immutable artifacts to prevent tampering and preserve release integrity.
 - ✓ Enforce versioning policies within CI/CD (e.g., Semantic Versioning).
 - ✓ Limit snapshot usage to early-stage development only.
- Use Nexus Repository's artifact history and audit information to track what an artifact is, who published it, and when.

Benefit:

Improves build reproducibility, strengthens trust in releases, and simplifies incident response and compliance reviews.

Data Point:

Immutable artifacts are essential to maintaining SBOM integrity, according to guidance aligned with U.S. Executive Order 14028 and NTIA's SBOM Minimum Elements.

Maturity Benchmarks

Understanding adoption is important, but engineering leaders also need to know whether repository practices are improving delivery. These benchmarks help evaluate how consistently teams can access trusted components, avoid preventable rework, and keep pipelines moving at scale.

Practice Area	Average Adoption	Good Adoption	Best-in-Class Adoption
Proxy Usage	<30% of builds use Nexus	31–90% adoption across teams	100% of builds route through Nexus
Repository Group Usage	Inconsistent use of groups	Major repos use groups	All repos grouped with fallback routing
Access Governance	Ad hoc roles and logins	SSO + RBAC partially adopted	Centralized RBAC via SSO; least privilege enforced
Policy Enforcement and Repository Security	Basic access controls; malware alerts inconsistently monitored	Nexus Repository security and Guide policy enforcement on critical repositories	Consistent malware blocking in Nexus Repository and policy enforcement through Guide
Retention Coverage	Minimal cleanup applied	High-churn repos managed	All active repos governed by retention policy
SBOM Adoption	SBOMs created occasionally	SBOMs for critical apps only	SBOMs published for every production release

These benchmarks help engineering leaders prioritize improvements that reduce developer friction, strengthen release confidence, and support consistent delivery across teams.

Executive Operational Adoption Metrics

To keep repository operations aligned with development goals, engineering and platform leaders should track metrics that show whether teams can access trusted components quickly, avoid risky downloads, and recover from disruptions.

Proxy Usage Rate

Measure the percentage of builds and developer environments retrieving dependencies exclusively through Nexus Repository rather than directly from public registries. Low usage indicates that teams may be bypassing the trusted path, creating inconsistent builds and exposing the organization to [shadow downloads](#).

Repository Group Coverage

Assess how consistently teams use repository groups (e.g., maven-public, npm-group) rather than configuring direct access to individual hosted or proxy repositories. This simplifies developer configuration, provides consistent component access, and reduces avoidable resolution issues.

RBAC & SSO Coverage

Track the proportion of users authenticated via SSO and governed by role-based access control. Full adoption simplifies onboarding and offboarding, reduces administrative effort, and gives teams reliable access to the artifacts they need.

Malware Visibility and Prevention Coverage

Track the percentage of proxy repositories protected by Nexus Repository's automated malware blocking and quarantine controls. Confirm that organizational risk thresholds are applied consistently so malicious and suspicious components are stopped before entering development.

Retention Policy Coverage

Identify what portion of hosted and proxy repositories are governed by automated cleanup or retention policies. Broad coverage helps control storage growth, shorten backup and recovery operations, and remove outdated components teams no longer need.

SBOM Adoption Rate

Track how many production releases are accompanied by a [software bill of materials \(SBOM\)](#) stored in Nexus. High coverage reduces last-minute documentation work and helps teams respond faster to customer, regulatory, and incident-response requests.

Together, these metrics help engineering leaders identify delivery friction, improve adoption of the trusted component path, and invest in the repository capabilities that keep teams productive.

Questions to Ask Your Team

- Are all teams, pipelines, and AI agents routing component downloads through Nexus Repository?
- Can developers quickly access the components they need without bypassing repository controls?
- Are Nexus Repository's automated malware blocking controls enabled across all proxy repositories?
- Are our risk thresholds configured to block malicious components and quarantine suspicious ones for review?
- How quickly could we restore Nexus Repository and resume builds after an outage?
- Do we generate and store SBOMs per release?
- Where could Sonatype Guide help developers and AI agents make trusted, policy-aligned component decisions earlier?
- Are our availability, backup, and recovery practices strong enough to keep pipelines moving?

Closing Note

Nexus Repository directly affects how quickly and confidently development teams can ship software. These practices help keep trusted components available, bad components out, and pipelines moving without avoidable disruption or rework.

Backed by the [Nexus One platform](#) and Sonatype intelligence, these capabilities support faster, more predictable software delivery at scale.

Data sourced from the [State of the Software Supply Chain report](#), internal customer benchmarks, and industry reports where noted.



Sonatype is the leader in secure software development built on open source and AI. As the maintainers of Maven Central and creators of Nexus Repository, Sonatype has spent two decades pioneering how the world manages and secures open source software — making Sonatype the trusted authority for modern software supply chains. With unmatched open source visibility and a unified product suite built for modern software development, Sonatype gives enterprises the intelligence and automated governance they need to harness the full potential of open source and AI. Sonatype handles the complexity behind the scenes: guiding component and model selection, blocking harmful malicious code, automating dependency and vulnerability management, and ensuring faster, more reliable builds — so developers spend more time on innovation and less time on remediation and rework. Trusted by more than 15 million developers, Sonatype helps power secure, modern software development at nearly 2,000 global organizations including 70% of the Fortune 100. To learn more about Sonatype, please visit www.sonatype.com