Are you using CodeArtifact? What are alternatives of AWS CodeArtifact ?

Alternatives of AWS CodeArtifact

1. **JFrog Artifactory** – Enterprise-grade artifact manager supporting 25+ formats with advanced proxying and security features.
2. **Sonatype Nexus Repository** – Popular open-source and enterprise solution for managing Java, NPM, Docker, and more.
3. **GitHub Packages** – Native GitHub-integrated registry for storing and publishing Docker, NPM, Maven, and other packages.
4. **GitLab Package Registry** – GitLab's built-in package storage that supports multiple formats with tight CI/CD integration.
5. **Azure Artifacts** – Microsoft's package hosting solution integrated with Azure DevOps for teams using .NET, Java, and Python.
6. **Google Artifact Registry** – GCP's native registry for Docker, Maven, NPM, and Python with IAM-based access control.
7. **Harbor** – Open-source cloud-native registry focused on container images, Helm charts, and vulnerability scanning.

Background Challenge

We had over 10+ dev teams, each working on different microservices. These teams:

- Relied on open-source NPM libraries,
- Built internal proprietary UI components and utility packages,
- And frequently needed to **share code across teams**.

But before CodeArtifact, things were chaotic:

- Some teams were using outdated versions of libraries.
- Sharing internal packages meant copying folders or pushing code to Git repos manually.
- There was no central registry, no control over open-source packages, and no security scanning.

Solution: Standardizing with AWS CodeArtifact

So, we introduced **AWS CodeArtifact** to create a **secure, centralized, and automated NPM package ecosystem**.

🏗️ **Domain & Repository Setup**

We created a domain called globaltech-artifacts, and split it into purpose-specific repositories:

Repository Name	Purpose
globaltech-open-source	Linked to npmjs.com
globaltech-internal-common	For shared internal packages (like @globaltech/ui-kit)
team-a-private	Proprietary packages for Team A
team-b-private	Team B-specific modules

🔁 **Upstream Relationships**

We defined **upstreams** to simplify dependency flow:

- team-a-private → globaltech-internal-common
- globaltech-internal-common → globaltech-open-source

This way:

- A team could pull open-source libraries or shared internal code **without hitting npmjs.com repeatedly**
- Everything was cached and available internally for speed and reliability

---

👨‍💻 **Developer Workflow**

Each team simply updated their .npmrc or used aws codeartifact login to point to their **team-specific repo**.

For example:

bash

```
npm install react
# Fetched from team-a-private → internal-common → open-source → npmjs.com (only once)
```

Developers could:

- Publish proprietary packages with npm publish
- Install shared internal modules or vetted OSS packages with confidence
- Avoid ever dealing with registry setup or authentication manually

---

Security & Governance

We enforced:

- IAM-based access per team (read/write to their repo only)
- Package approval workflows for OSS dependencies
- Audit trails via CloudTrail for who published what, when

The **security team could now review all dependencies**, scan for vulnerabilities, and block suspicious packages **before** they reached production.

Outcomes & Benefits

Category	Before CodeArtifact	After CodeArtifact
OSS Management	Unvetted & inconsistent	Centralized & scanned
Internal Sharing	Manual, error-prone	Seamless via upstreams
Build Speed	Slow (downloads from npm)	Fast (cached in-region)
Reliability	npm outages broke builds	Builds work even offline
Security	No control or scanning	IAM + audits + caching
Auditability	None	Full package/version logs

Impact
In one case, npmjs.com had a temporary outage. While other teams globally were blocked, *GlobalTech's builds ran uninterrupted* — thanks to our internal CodeArtifact cache.

Key Takeaway
*CodeArtifact gave us the reliability of a **private NPM registry**, the **scalability of AWS**, and **the security governance** we needed — all without the pain of managing our own infrastructure.*

What is the primary purpose of AWS CodeArtifact?
A. Hosting machine learning models
B. Storing and managing software packages securely
C. Monitoring CI/CD pipelines
D. Managing EC2 deployments

✅ **Correct Answer: B**

Which of the following is NOT an alternative to AWS CodeArtifact?
A. JFrog Artifactory
B. Sonatype Nexus
C. Amazon S3
D. GitHub Packages

✅ **Correct Answer: C**

What is a key benefit of using upstream repositories in CodeArtifact?
A. To reduce IAM complexity
B. To ensure packages are always downloaded from npmjs.com
C. To create a hierarchical fallback for package resolution
D. To enforce stricter rate limiting

✅ **Correct Answer: C**

What type of access control does CodeArtifact use for security and governance?
A. SSH Keys
B. Basic Auth

C. IAM-based permissions
D. OAuth tokens

✅ **Correct Answer: C**
