Bintray Automation for a Tedious Situation

Why Bintray was Chosen to Scale Thousands of WebJar Files

James Ward
Engineering and Open Source Ambassador
Salesforce.com

Why Bintray Automation was the Best Solution

James Ward's WebJars project is a classic example of how JFrog Bintray can be used to scale up a manual procedure into an automated, on-demand, deployment process that's distributed and available within the community. Realizing that his manual efforts to deploy WebJars during his free time were unrealistic to maintain in the long run, using Bintray's extensive REST API, Ward developed an automated way for contributors to make WebJars available to the community on demand using JCenter, Bintray's public Maven repository

Before
250 Hours of **Manually Scaling** **1000** Classic WebJars
216 Hours of **Manually Scaling** **3600** WebJar Versions
After
On-Demand Deployment and **Automated Scaling**
4,170 Classic WebJars and **8,772** WebJar Versions

JavaScript ecosystem

Historically, the JavaScript ecosystem lacked proper dependency managers, comparable to ones that have existed for years in the Java world. The NPM Registry is generally considered polluted, and incidents like the #npmgate have further diminished the community's trust in the tool and the repository. Many web developers could only dream to have "Maven/Gradle and Maven Central for JS", and WebJars made it possible.

WebJars are CSS and JavaScript libraries, like Bootstrap and JQuery, which are packaged into Java Archive (jar) files and deployed on a public Maven repository. Users can then specify these libraries in their Java package manager, like Maven and Gradle, and consume them as dependencies. This effectively allows web developers to enjoy the advanced build and dependency resolution tools of the Java world in their work.

With over 8 million downloads, using WebJars is now an extremely popular way for web developers to resolve and build their software.

How did WebJars initially get deployed?

Each WebJar had a project containing a POM file, with the Maven build definition, describing all the required metadata and configuration. Running a Maven build on a WebJar pulled the source code for that web library from its source, packaged it into the jar file and deployed it on Maven Central.

Since the number of JavaScript libraries continued to grow, **this initial manual approach for scaling WebJars was just taking up too much time, and it was only going to get worse.** For example, at that time the NPM repository contained close to 300,000 packages, with almost 500 new ones being added daily. This would have translated into many hours of manual work. Adding to this was the licensing challenge that impacted deployment times, because of poor licensing standards that results in many license references not being in a readable form.

Why Bintray Automation was the Best Solution

Automation was the only way to scale the WebJars project, with the following requirements:

- Availability on JCenter (and Maven Central for backwards compatibility)
- On-demand deployment
- No need for an additional repository
- Automatic inclusion of package metadata, such as source code location and license requirements
- Compatibility with NPM and Bower repositories in order to access JavaScript libraries and metadata

JFrog Bintray met all of the WebJar project specifications, including deployment to JCenter (with automatic sync to Maven Central), on-demand availability, compatibility with NPM and Bower repositories, and automated metadata inclusion.

Bintray was the ideal choice as the means for automating the scaling of WebJars. In addition to meeting all of the above prerequisites, Bintray natively supports all major package formats, which allows seamless work with industry-standard development, build, and deployment tools. Moreover, Bintray's API preserves many REST API norms which facilitate learning, use, and the complexities of dealing with licenses.

> *"To do all this I used the Bintray API, which is actually a really fantastic API. It really preserves all of the norms of REST API's. It was really easy to learn and use.*
> *The REST APIs here are really straight forward, just standard REST using the HTTP verbs to do the things you would expect. Super easy!"*
>
> James Ward, Engineering and Open Source Ambassador at Salesforce.com

Bintray's optimal management console allows users to view all deployed packages, with their associated files, versions and status. Additional metadata, such as the source code location and licenses, is also included. Bintray also provides an easy to use interface for users to manage their artifact versions. This is especially useful when a deployment fails and a re-sync is needed.

Bintray exceeded the basic needs of an automated system due to its wide support of package formats, APIs, and management console.

On-Demand Deployments Thrive with Bintray

It now only takes a minute for the automatic scaling deployment process, enabled by Bintray, requiring only the version, group ID and artifact ID to be specified. After which the WebJar is readily available in the library for a Java build.

Within a year, the Bintray solution has successfully resulted in thousands of on-demand deployments that continue to grow.

One year later

1,950	4,650	2,230	4,157
Named Bower Artifacts Deployed On-Demand	Versioned Bower Artifacts Deployed On-Demand	Named NPM Artifacts Deployed On-Demand	Versioned NPM Artifacts Deployed On-Demand

Summary

After volunteering over 460 hours of his time to manually deploy WebJars, James Ward realized that his continued commitment was better spent on developing an on-demand automation process. With JFrog Bintray, the project's requirements were easily met, and Ward was able to scale his WebJars project to allow the community to deploy WebJars on-demand.

Going beyond OSS

This case study shows the value of Bintray APIs to facilitate automation in an OSS project.

Enterprise Bintray users get even more powerful functionalities with the API, including automated handling of access control through access keys and entitlements, detailed stats, detailed logs, dynamic downloads, private repositories, unlimited API queries, and more.

Have an account? **Go Premium**

Don't have an account? **Sign up for a free trial**

Why Bintray was selected as the Best Automation Solution

- **On-demand deployment to JCenter (with automatic sync to Maven Central), NPM/Bower compatibility and automatic metadata**
- **Native support for major package formats and REST API industry standards**
- **Versatile management console**

JFrog Bintray