



Netflix is powered by Java.

Thousands of Java services, running across hundreds of thousands of JVMs, deliver on-demand and live video to hundreds of millions of members around the world.

As the Java language and platform evolve faster than ever, the cost of adopting a new release or enabling a new feature falls on the people who operate Java in production. Netflix wants to bring that operator's perspective to the Executive Committee, helping make new Java versions and features easier and cheaper to adopt for everyone who builds and runs Java applications.

We have a long history of contributing to the Java ecosystem through build tooling, libraries, security, and performance work, and we adopt new LTS and feature releases early. Through JEPs and projects like Leyden and Generational ZGC, we've worked closely with OpenJDK and the JCP to provide production feedback, identify tooling gaps, and share what we've learned about putting these new platform capabilities into service at scale.

Netflix has benefited enormously from the OpenJDK community, and we'd welcome the chance to give back by helping shape the future of the language and runtime. With your support, we look forward to working with JUGs, JVM vendors, Oracle, and the broader OpenJDK community to help build the best platform for the next generation of Java developers.

JCP Representatives

Paul Baker

Paul Bakker is a member of Netflix's Java Platform team, which develops the frameworks and tooling used to build Java services across Netflix. The team also plays a leading role in evaluating new JDK features and rolling them out across the Netflix ecosystem. Paul is a Java Champion and frequent conference speaker. His "How Netflix Is Using Java" presentations are among the most watched videos about Java. He has co-authored books on the Java

Platform Module System and OSGi and is the creator of the DGS Framework. His work gives him a hands-on perspective on Java platform evolution, framework design, developer productivity, and the adoption of new Java technologies in large-scale production environments.

Danny Thomas

Danny Thomas is a member of Netflix's JVM Ecosystem team, which is responsible for Java build and dependency management, packaging, configuration and adoption of the JVM across the fleet. The team ensures the readiness of the company to adopt the latest Java releases and features. The OSS Nebula plugin ecosystem for Gradle that the team maintains is a significant contributor to the reduction of time to adopt a new Java release from years to months. Danny is a performance engineer and has spent the last 12 years working on developer productivity at Netflix. His decision to adopt generational ZGC by default was covered in the widely read Netflix TechBlog post "Bending pause times to your will with Generational ZGC". Most recently, his "Leave the Class Path in the Rearview Mirror" post advocates for broader adoption of Java Module System and announces tools for both agents and people to more effectively work with the Java language.

Relevant Netflix Java blog posts and talks:

How Netflix Uses Java in 2026

A survey of Netflix's Java platform.

https://www.youtube.com/watch?v=ucJTPda_zx0



Draw the Rest of the Owl: Leyden in Production and the Infrastructure Needed to Get It There

Learn how Netflix used Project Leyden to improve startup time of critical services and the software and SDLC we built to make that happen.

<https://www.youtube.com/watch?v=4kEh8hxAP4U>



Leave the Class Path in the Rearview Mirror

Introducing composable, module system native and agent friendly command line tools for modern Java development,

<https://netflixtechblog.com/leave-the-class-path-in-the-rearview-mirror-67a85b15b6be>



Optimizing Recommendation Systems with JDK's Vector API

Making the largest Netflix services faster and cheaper with new JVM features.

<https://netflixtechblog.com/optimizing-recommendation-systems-with-jdks-vector-api-30d2830401ec>



Java 21 Virtual Threads - Dude, Where's My Lock?

Getting real with virtual threads.

<https://netflixtechblog.com/java-21-virtual-threads-dude-wheres-my-lock-3052540e231d>



Seeing through hardware counters: a journey to threefold performance increase.

Deep debugging of JVM and hardware performance.

<https://netflixtechblog.com/seeing-through-hardware-counters-a-journey-to-threefold-performance-increase-2721924a2822>



Java in Flames

Bringing flame graphs to the JVM.

<https://netflixtechblog.com/java-in-flames-e763b3d32166>



