azul

Cloud Native Compiler Frequently Asked Questions (FAQ)

Q: Who is Azul?

A. Azul, a US private company founded in 2002, is the premier provider of Java™ support. As the only company 100% focused on Java and with the largest Java team outside of Oracle, Azul has the skills and expertise, and globally distributed Support organization to ensure your Java success. Powering some of the largest companies in the world, including:

- 27% of the Fortune 100
- 50% of Forbes Top 10 World's Most Valuable Brands
- 10 out of 10 of the world's Top financial trading companies

Azul has a Java runtime solution to fit your business needs. From high-change velocity SaaS organizations to mission-critical eCommerce applications, leading brands such as Avaya, Bazaarvoice, BMW, Credit Suisse, Deutsche Telekom, LG, Mastercard, Mizuho, Priceline, Salesforce, Software AG, Workday and many more bet their business on Azul every day.

Azul's products and services are used by millions of Java developers, hundreds of millions of devices, and the world's most highly regarded businesses trust Azul to power their applications with exceptional capabilities, performance, security, value, and success. Azul solutions are available for developers, ISVs, enterprises with on-premises and/or cloud deployments, and hardware OEMs building embedded and IoT devices.

Q: What is the Cloud Native Compiler?

A. Cloud Native Compiler is the first offering of the Azul Intelligence Cloud family of products to be made generally available (GA) to Azul customers. The Azul Intelligence Cloud includes a series of cloud-centric analytical and code optimization modules to manage, analyze and scale your Java fleets of cloud instances, servers, desktops and IoT devices.

The Cloud Native Compiler is a scalable Just-In-Time (JIT) compilation server. It enables Java Virtual Machines, Java runtimes allowing Java programs to 'write once, run anywhere', to provide a server-side optimization solution that offloads JIT compilation to separate and dedicated service resources, providing more processing power to JIT compilation while freeing your client JVMs from the burden of doing JIT compilation locally.This allows you to improve Java and JVM-based application performance while right-sizing infrastructures to deliver cost savings, especially when leveraging cloud connectivity at scale.

Q: What is a Just-in-Time (JIT) Compiler?

A. A Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java applications by compiling bytecodes to optimized native machine code at runtime. JIT optimization provides a multitude of benefits, including the ability to use speculative optimizations that lead faster eventual code. However, traditional on-JVM JIT compilers must share the JVM's local CPU resources and compete with the application logic in using that capacity. This presents several challenges:

- The JIT compiler is limited in resources. Resulting optimizations take time to arrive at, and benefits are limited by the practical amount of time that applications can wait for optimization and warm-up to complete.
- The JIT compiler is limited in how aggressively it can afford to optimize code. The resulting optimizations are not as fast as they could be if the optimizer had more resources available

The Cloud Native Compiler provides a server-side solution to overcome these challenges when deployed alongside connected Zulu Prime-based Java Virtual Machines.

Q: When was the Cloud Native Compiler released?

A. Cloud Native Compiler became generally available on October 15th as a customer-managed component in conjunction with Azul Platform Prime.

Q: What is Azul Platform Prime?

A. Azul Platform Prime turbocharges the performance and scalability of your Java applications with a hyper-optimized runtime that maximizes performance while dramatically driving down infrastructure costs.

Based upon Zulu Prime builds of OpenJDK, Prime extends the open-source Java standard, OpenJDK by applying unique innovations to improve the runtime characteristics of Java workloads. This leads to better performance and throughput. Prime boosts the number of transactions from the same hardware and speeds up Java performance even as loads increase. It is proven to reduce capital expenses for servers by as much as 50% and drive continuous value.

Key Azul Platform Prime features include the C4 Garbage Collector, removing pauses including stop-the-world pauses; the **Falcon JIT Compiler,** building on LLVM technology for faster throughput; and ReadyNow! for smoother warm-up. The Falcon JIT compiler replaces OpenJDK's C2 JIT Compiler to run different levels of optimizations, and its upper tier of optimizations produces optimized code that can run significantly faster than code produced by the OpenJDK C2 compiler.

Together, the capabilities combine to ensure your Java applications start fast, run faster, and stay consistently fast while ensuring infrastructure optimization and cost savings of up to 50%.

The Cloud Native Compiler specifically leverages the Falcon JIT Compiler feature, running it as a server-side optimization solution and externally to the JVM.

Q: How does the Cloud Native Compiler benefit an Azul Platform Prime workflow?

A. At a high level, Cloud Native Compiler when used with Azul Platform Prime immediately allows users to offload JIT compilation from the JVM to dedicated servers. This frees JVM processes from expensive compute and memory overheads at key times, like warm-up. Cloud Native Compiler also improves performance, ensures consistency of performance, and reduces costs by right-sizing resources.

At a deeper level, using more aggressive optimization levels requires more resources, and when using JVM-local JIT compilers for optimization, resource tradeoffs can often lead to a choice of lowering optimization levels in favor of improved warmup times. Cloud Native Compiler eliminates trade-offs by removing JIT compilation work from individual JVMs and shifting the work of the Falcon JIT compiler to a separate shared service. This shift of work and associated resources allows the Cloud Native Compiler to apply even the most aggressive Falcon JIT optimization levels without disrupting individual JVM behavior.

Q: What do I need to run the Cloud-Native Compiler?

A. You must have access to Zulu Prime builds of OpenJDK (part of the Azul Platform Prime product) and access to a Kubernetes cluster. If you do not have a Kubernetes cluster, the Installing Cloud Native Compiler Documentation will help you set it up. Azul Engineers are available for further guidance.

Cloud Native Compiler is best deployed on elastic (cloud) infrastructures, easy to spin up and easy to take down. It benefits larger Zulu Prime configurations. Anything above 50 server instances could show Cloud Native Compiler benefits for key JIT compilation processes, e.g., warm-up. However, the larger the configurations of JVMs, the greater the likely impact.

Q: How does the Cloud Native Compiler right-size resource? Can it also reduce the footprint of the Zulu Prime JVM?

A. Memory and compute, required by the JIT Compiler in bottleneck scenarios, e.g., warm-up, may no longer be required by the instance running the JVM. Customers often must reserve 2x or more CPU capacity for JVMs than they need just for running their applications to enable JIT compilation during warm-up and de-optimizations. Using Cloud Native Compiler moves this cost to the service, allowing you to serve the same load on a smaller image. This way, you can optimize and right-size your infrastructure for compute and memory requirements while optimizing performance.

In the initial release of Cloud Native Compiler, the JVM switches to running code in interpreted mode when the Cloud Native Compiler service cannot be reached. In upcoming releases, we will have a full fallback to local Falcon JIT Compiler when the service is unresponsive.

Q: Which builds of Zulu Prime Builds of Open JDK are supported by Cloud Native Compiler?

A. 21.09 and above.

Q: What resources are required to run Cloud Native Compiler?

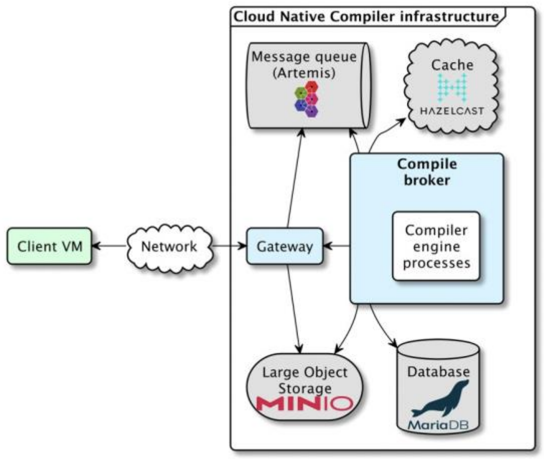
A. The Cloud Native Compiler resources can be dialled up and down as needed. Since the Cloud Native Compiler (CNC) service uses a large amount of resources (recommended 4 CNC vCores for every JVM vCore), it is imperative to correctly configure autoscaling. When no JVMs are being started or restarted, the cloud native compiler can be dialled down to a minimal footprint.

Azul recommends 1 cache for every 15 compile brokers. This will change in future with Cloud Native Compiler updates, in addition to improved autoscaling capabilities. Currently scaling is primarily manual, with some experimental autoscaling options, such as HPA config in the YAMLs.

Remember that the Cloud Native Compiler needs and uses large amounts of capacity for short bursts of time, so consider within an infrastructure of overall optimizing application efficiency, right-sizing resources where and when they are needed.

Q: What logging and monitoring options are available to me?

A. With the first release, you can integrate Cloud Native Compiler into existing Prometheus/Grafana applications. All pods export metrics for Prometheus scraping and include dashboard JSON to add to



Grafana instances. `<cnc-install-dir/grafana/cnc_dashabord.json` is a Grafana configuration file for a dashboard of key Cloud Native Compiler metrics. You can [import the dashboard]() into your existing Grafana installation.

Q: What are the time horizons of new developments?

A. Expect to see significant improvements as outlined above in a few months.

Q: What is the cost of Cloud Native Compiler Support?

A. Support for the Cloud Native Compiler, as of October 15th, is included in the annual cost of Azul Platform Prime. Please see the [Azul Platform Prime pricing]() for more information and support tier options.

Q: What are my alternatives to the Azul Cloud Native Compiler?

A. The most likely alternative is to run JIT compilation, as you are likely already running it, within the JVM.

Ahead of Time [AOT] compilers provide non-JIT compilation alternatives. Cloud Native Compilation is a fundamentally different approach to AOT compilation, and results in dramatic infrastructure savings and footprint reduction. Where pure-AOT solutions focus on statically compiling applications into a small and quick-to-start executable, the resulting code is less optimized, runs slower, and requires more resources (and cost) to carry the same load. order to avoid the burden and local costs of JIT optimization. Such pure-AOT static compilers sacrifice key optimizations capabilities (e.g., speculative optimizations which are key to speed gains in Java). In contrast, Cloud Native Compilation retains the full power of JIT compilation, and "turns it on to the max" by shifting the heavy lifting optimization work away from the running application instances, and onto a shared, elastic, and efficient cloud resource. This allows JVMs to benefit from powerful optimizations and results in faster and more efficient code, which in turn translates to lowered cloud spend as a smaller infrastructure footprint is required to run the same workload. In contrast with AOT static compilers, Cloud Native Compilation lets applications start and get to speed quickly, but without sacrificing their eventual speed and efficiency.

Q: Can I evaluate Cloud Native Compiler?

A. Cloud Native Compiler is free to use with any properly licensed Zulu Prime builds of OpenJDK. If you need Zulu Prime bits, please visit the download page at https://www.azul.com/products/prime/stream-download/. Cloud Native Compiler install components are available on the Azul cdn and on Dockerhub.

Stream Zulu Prime builds are available for development, testing and evaluation. Stable Zulu Prime builds are for production use, super-stable builds that incorporate only CPUs, PSUs, and Azul Platform Prime critical fixes, and do not uptake new features and non-critical enhancements from Stream Builds.

Q: Where can I learn more Cloud Native Compiler?

A. The best place to start is the Documentation, at https://docs.azul.com/cloud_native_compiler/.

Azul can provide product introductions from knowledgeable engineers if appropriate.

Q: Can I report a bug or request an enhancement as part of my Azul support services?

A. Azul offers Platform Prime customers 24x7x365 phone and email support services with the Platinum Support tier, which includes opening bugs or filing enhancement requests with Azul. Depending on the severity of the bug, Azul's world-class Support organization will work to provide application triage, root cause analysis, temporary workarounds, or out-of-cycle patches to address customer's needs.

Q: Are Azul Zulu Prime binaries, (in Azul Platform Prime), "Certified" Java SE compatible & compliant?

A. Yes. Azul is one of three vendors that have licensed from Oracle the OpenJDK Community Technology Compatibility Kit (TCK) for all versions of OpenJDK (e.g., Java 7, 8, 9+). Every binary Azul build for our customers is verified "compatible and compliant" with the Java SE (Standard Edition) specifications using these TCKs. These TCK compliant binaries also carry Intellectual Property (IP) indemnification, which are only granted to binaries that have passed the TCKs. This ensures that migrating from Oracle Java to Azul Platform Prime and other supported Azul OpenJDK binaries is straightforward. Azul Zulu Prime builds are a drop-in JDK and JVM replacement for the Oracle JDK and HotSpot JVM.

For more information on Cloud Native Compiler, email info@azul.com

Contact Azul

385 Moffett Park Drive, Suite 115

Sunnyvale, CA 94089 USA

+1.650.230.6500

www.azul.com