David Barsky

me@davidbarsky.com
davidbarsky.com
github.com/davidbarsky
linkedin.com/in/davidbarsky

Work Experience

Fall 2017–Present **Amazon, Alexa AI**
Software Engineer

- Integrated internal test data generation system into a unified, self-service natural language model building and testing platform.
- Enabled multiple Alexa verticals (e.g., Music, Shopping) to release features independently and concurrently, doubling daily release capacity.
- Drove AWS to sponsor the Rust project's AWS bill for the next three years.
- Reduced execution time of NLU model validation tooling from 45 minutes to 2 minutes.
- Reduced execution time of a test-data generation system from 12 hours to 15 minutes.
- Mentored three interns, who have all returned to Amazon as full-time SDEs.
- Implemented Alexa's natural language model to be hardware-capability aware. This significantly reduced engineering effort needed to support new devices and cut the public Alexa service's memory usage by 30%.

Summer 2016 **Amazon, AWS Payments**
Software Engineer (Intern)

- Worked in Payments organization in Amazon Web Services.
- Architected and developed a distributed, fault-tolerant service for financial data auditing that simplified payments infrastructure and reduced on-call burden.

Education

Fall 2013—2017 **Brandeis University**
Computer Science, B.A. with Honors.
Completed additional coursework in History and Politics

Selected Projects

Tracing (github.com/tokio-rs/tracing)

- Helped launch Tracing, a unified, high-performance instrumentation system for Rust that emits logs, metrics, and traces.
- Designed and wrote a lock-free data store for storing high-cardinality instrumentation data. This store is now the basis of several third-party instrumentation exporters, such as OpenTelemetry, Prometheus, and Amazon's internal metrics format.
- Successfully introduced Tracing into several key AWS services, whose total request volume exceeds two million requests per second.

Tower (github.com/tower-rs/tower)

- Contributed documentation and more ergonomic error handling to Tower, a library for unified client/server middleware.
- Built an HTTP congestion control library that expands/shrinks depending on downstream service health.

Rust Runtime for AWS Lambda (github.com/awslabs/aws-lambda-rust-runtime)

- Launched the Rust Runtime for AWS Lambda.
- Converted project to be asynchronous async/await, enabling concurrent executions and easier in-memory testing of Lambda functions.

Skills

Languages: Rust, Java, Go, Kotlin, Python.
Tooling: AWS (DynamoDB, ECS, Lambda), Linux, Docker, Git, gRPC.