Matthew Planchard **https://mplanchard.com**
NYC inbox@mplanchard.com

Work History

Spec (2021–Present)

Senior (2021) | Staff (2023) | Principal Engineer (2025)

I was the first application engineer at Spec, leading the buildout of our founding engineering team, the planning and implementation of the core platform, and the establishment of our engineering culture.

During my time at Spec, I proposed, planned, and implemented a variety of organizational and cross-team projects and initiatives. Some of particular note are listed below.

Core Platform *Rust, Networking, Systems, Performance*

- Real-time HTTP proxy providing introspection of requests and responses, with configuration-driven data labeling and persistence, handling over 10k req/second.
- Workflow engine enabling a wide array of actions, from blackholes to honeypots, in response to arbitrary logical combinations of data from past events, current HTTP exchange context, and external sources.
- Average processing time across all customers below 100 μs, with 99.99% uptime and a failure rate of less than one in a billion requests.
- Security audits as part of CI, with new CVEs patched within 24 hours and released within one week.
- Core domain types compiled into WASM and shared with FE code, ensuring single point of truth for critical business logic.

OLTP Data Layer *PostgreSQL, Query Optimization, Partitioning*

- PostgreSQL database with both hash- and time-partitioned tables to enable high insert throughput and effective index sizes.
- Selected, highly optimized queries made available for use in real-time workflows, with associated performance monitoring and alerting.
- Terabytes of ingested data per day for larger customers, managed with custom background jobs to remove old partitions and delete old records.
- Over 10k inserts per second and 50k reads per second, with P99 read latency less than 10 ms for potentially inline queries.
- Experimental framework in Rust for collecting statistics on alternative queries in production environments.
- Custom benchmarking framework in Rust for collecting both timings and system metrics such as CPU, memory, and disk I/O for database operations.

OLAP Pipeline *Change Data Capture, Debezium, Kafka, ClickHouse*

- Transaction-based change data capture using versioned Avro schemas, produced by Debezium connectors into Kafka topics.
- Custom Rust consumers for durable, high-throughput batch processing of topic data into ClickHouse.
- Monitoring, alerting, and troubleshooting on replica slot lag associated with Debezium replication slots.
- Overall pipeline duration of less than one second from PostgreSQL insert to ClickHouse.

Engineering Culture *Philosophy, Quality, Autonomy, Documentation*

- Normative culture and process guides: Engineering Philosophy document, language and framework guidelines, explicit project goals and priorities.
- Ritualized opportunities for knowledge sharing: pair programming, lunch and learns, and formal talks.
- Asynchronous, democratic decision-making through talk pages, PRDs, and technical planning docs.
- Active mentorship and opportunities for new and more junior engineers to own entire features and projects.
- Diátaxis framework to hone documentation focus and organization.

Developer and Production Environments *DevEx, Nix, CI, Docker*

- Monorepo for all engineering projects, with recursive changed-project detection to minimize CI time.
- Extensive CI optimization and parallelization, using standard Unix tools so that CI runs could be replicated and debugged locally.
- Shared, cross-platform development environment in Nix, providing all system and platform dependencies.
- Nix derivations and S3 Nix cache to speed up CI runs by up to 80%.
- Production containers built with Nix for reproducible builds and parity with developer environments.

Bestow (2019–2021) Senior II (2019) | Staff Engineer (2020)

Data-Driven Applications *Rust, Python, WASM, JsonLogic, Process Optimization*

- Encoded insurance application flow and validation in JSON documents using JSONLogic.
- Drove cross-team use of JSON flow, eliminating manual toil necessary to individually add support for new insurance products and align frontend and backend implementations.
- Reduced time-to-market for new insurance products by a factor of three.
- Implemented JSONLogic in Rust with Python and JS bindings to address inconsistencies and bugs in existing implementations.

TypeScript SDK *TypeScript, Business Enablement*

- Utilized data-driven applications to provide an easy-to-use SDK for node and web.
- Unified validation and application logic with Pit of Success and Parse, Don't Validate philosophies.
- Enabled white-labeling efforts by internal teams and B2B sales.

Monolith Rewrite *Python, Rewrite, Strangler Fig*

- Took over an in-progress rewrite of a monolithic Python application that had been contracted out.
- Ensured all original functionality was intact and oversaw the deployment of the rewritten system.
- Worked with Go microservice teams to apply the Strangler Fig pattern, divesting the monolith of various functional components as the microservices came online.

Organization-Wide Observability *New Relic, Monitoring, Observability*

- Led cross-team effort to implement New Relic metrics and request tracing across all microservices.
- Worked with teams to determine appropriate thresholds for alerting, and ensured alerts were appropriately configured.

Previous Experience

Duo/Cisco (2018–2019) **Engineer II**

Worked on native applications and an associated Python server, to provide information to Duo's Zero Trust solution enabling access policies based on device security posture.

Enthought (2017–2018) **Fullstack Web Developer**

Developed IaC deployment pipeline (Docker, Terraform, & Swarm) for both internal and onsite deployments of Enthought's core Python package server, as well as contributing server features.

ihiji (2014–2017) **Intern (2014) | Linux Developer (2015) | Lead (2016)**

Created API server (Python) and monitoring framework (Python/Perl) for IoT devices, utilizing MongoDB, React, and React Native.

Selected Open Source

A more complete listing is available at mplanchard.com/projects.html.

cuid-rs	GitHub	*De facto* standard Rust crate for Collision-resistant Unique IDs (CUIDs)
procfarm	SourceHut	Rust CLI tool for Unix background command parallelization and reporting

Education

M.S. in Biochemistry, University of Southern Mississippi (2014)
Dual B.S. in Biology and Chemistry, University of Southern Mississippi (2012)

Skills

Leadership	technical strategy, alignment, architecture, mentoring, documentation, culture
Performance	profiling, query optimization, benchmarking, observability
Backend	Async Rust, HTTP, Networking (L4 & L7), C++, C, Go
Scripting	Shell, Python, Lisp
Data	PostgreSQL, CDC, ClickHouse, Kafka, Debezium, Redis, MongoDB
Infra	NixOS, Nix, Docker, Nginx, AWS, Kubernetes, Terraform, CI/CD