

Vlad Toader

Email: va.toader@gmail.com · LinkedIn: [linkedin.com/in/vlad-toader](https://www.linkedin.com/in/vlad-toader)

PROFILE

Software engineer and architect who builds data platforms as software: frameworks, compilers, control planes and orchestrators, designed to stay correct under failure and for other engineers to build on. Core-team engineer on enterprise data platforms, taking lakehouses and batch and streaming systems from architecture to production. Sets the architecture and code design for engineering teams, leads design reviews and raises the bar through code review.

SKILLS

Languages: Python, Java, SQL

Data processing and streaming: Apache Spark (batch, SQL, Structured Streaming), Apache Beam / Dataflow, Kafka, Event Hubs, Pub/Sub, Kinesis, DuckDB, PyArrow, Apache Airflow

Storage and table formats: Apache Iceberg (v2/v3, REST catalogs, Apache Polaris), Parquet, Delta Lake, BigQuery, Cosmos DB, Redis, Azure SQL / SQL Server, PostgreSQL

Platform: Kubernetes (AKS, Karpenter, Spark Operator), Helm, Argo CD, Terraform, Bicep, Azure DevOps, GitHub Actions, GitLab CI

EXPERTISE

Software architecture: frameworks and platforms other teams build on (pipeline blueprints, a shared streaming framework, a lakehouse core library), hexagonal designs where sources, sinks and runtimes are adapters behind contracts, plugin registries and extension points, explicit lifecycle state machines with guarded transitions

Compilers and code generation: compilers from declarative specifications to executable plans (a no-code pipeline compiler targeting Spark; a compiler from HERE NDS map data into tiled binary road topology)

Python: typed codebases (protocols as ports, pydantic models, mypy strict), bounded-memory streaming and concurrent I/O under a shared connection budget, PyArrow and Pylceberg internals, packaging and test infrastructure (pytest, Hypothesis)

Java: production code on the Spark and Beam SDKs (Structured Streaming arbitrary state with timeouts, Beam state and timers, custom coders and composite transforms), bit-level binary decoding and encoding (bit-packed payloads, Protobuf, Avro, CRC checks), dependency injection with Dagger, functional error handling with Vavr routing failures to dead-letter streams, Spring Boot services, Maven multi-module builds

Correctness under failure: idempotent and replayable processing, crash-safe progress tracking without distributed transactions, leases and single-flight coordination, optimistic concurrency (Redis transactions, guarded document updates, Iceberg commits), retry classification and dead-lettering, backpressure and adaptive load control

Testing and delivery: differential tests that compare the system's output with a simple reference implementation, property-based tests, fault injection (forced crashes at every step of a pipeline), end-to-end deployment tests in real cloud accounts; CI/CD and release engineering; design and code reviews

Data systems, to source level: Spark (execution, shuffle and joins, Catalyst and AQE, memory, Structured Streaming state), Iceberg (v1–v3 spec, commit protocol, scan planning, deletion vectors, REST catalogs), Kafka (storage, replication, KRaft, transactions, rebalancing), the Dataflow model (event time, watermarks, exactly-once), database internals (storage engines, MVCC, recovery, query optimisation), distributed systems (replication, partitioning, consensus, schema evolution), Kubernetes internals

WORK EXPERIENCE

DATA PLATFORM ENGINEER | SENIOR DATA ENGINEER – ITINERIS – 01.12.2025 – Current

Via rinf.tech. Itineris builds Microsoft-based software for energy and water utilities.

NEXUS replicates Microsoft Dynamics 365 F&O data into versioned, analytics-ready datasets inside each customer's Azure tenant. Lead engineer and architect of the data platform, from the proof of concept to the production lakehouse; set the architecture and code design for the team and reviewed every change to the core.

- Architected an open-source lakehouse on Apache Iceberg governed by an Apache Polaris REST catalog (per-layer catalogs, OAuth2 credential vending, so no job holds storage keys); Spark runs on AKS through the Spark Operator on Karpenter node pools (spot-first executors, on-demand drivers, scale to zero), with automated compaction, snapshot expiry and orphan cleanup per layer, delivered as Helm charts through Argo CD.
- Designed the Iceberg medallion layers: an append-only v3 Bronze ingesting Synapse Link exports in batched atomic commits, an append-only Silver ledger of every delivered version as the platform's replayable copy of the source, and a current-state layer kept by an incremental MERGE of strictly newer versions, pruned by record-ID ranges so only files holding changed records are rewritten.
- Designed Kimball-style SCD Type 2 history as append-only closed versions beside the current-state table, folded in build order so that an hourly run, a catch-up and a full rebuild produce identical history.
- Made every stage replayable and crash-safe without distributed transactions: each Iceberg commit records in its snapshot summary exactly what it consumed, so a stage always resumes from its own last position, and two alternating Iceberg tags protect the snapshot that history resumes from against expiry at any point of a crash.

- Released each build atomically to Synapse serverless SQL, an engine with no Iceberg reader, through views over pinned, tagged Iceberg snapshots switched in one transaction, with retained release tags for point-in-time queries and rollback.
- Built the Cosmos DB control plane: an explicit state machine per ingestion window, lineage across every stage, classification of transient and permanent failures with automatic retry, supersede-never-edit retry chains and operator-only dead-lettering, making every run idempotent and replayable.
- Built a Kubernetes-native orchestrator that chains Spark Operator applications and batch jobs from Helm-rendered manifests, holds a single-flight lease per flow, adopts in-flight workloads after its own crash and enforces a time budget per stage.
- Structured the batch code around ports and adapters: every Spark stage is a typed source → transforms → sink pipeline, and the platform reaches the ingestion side and Kubernetes only through Python protocols; verified by differential tests against a reference implementation, forced-crash scenarios and property-based tests of the state machine.
- Led the move across two generations, from a DuckDB proof of concept to the Spark lakehouse in production. Now designing a metadata-driven catalogue of Gold representations, ordered by dependency, each materialised or kept logical and refreshed incrementally from the ledger.
- Built the proof of concept on a tight budget: a streaming PyArrow/Pylceberg Bronze ingester (parallel range downloads, adaptive concurrency, largest-first scheduling) that handled multi-gigabyte CSVs in constant memory, and a bulk publisher into Azure SQL that sized its loads against the database's log-write telemetry and applied changes with a replay-safe, version-aware MERGE.

Tools/Frameworks: Python, PySpark (Spark 4), Apache Iceberg, Apache Polaris, Pylceberg, DuckDB, PyArrow, Kubernetes, Spark Operator, Karpenter, Helm, Argo CD, pytest, Hypothesis, mypy

Cloud (Azure): AKS, ADLS Gen2, Synapse Link, Synapse Serverless SQL, Cosmos DB, Azure SQL, Key Vault, Entra workload identity, Container Registry, Bicep, Azure DevOps

SENIOR DATA ENGINEER – UNITEDHEALTH GROUP – 01.12.2023 – 30.11.2025

Via Publicis Sapient.

- Built pipelines for UnitedHealth Group's central platform for marketing engagement and feedback data, on Databricks and Apache Spark in a medallion architecture.
- Built ingestion libraries for streaming and batch sources with data-quality checks at each layer, and defined data models with analysts and marketing teams.
- Automated testing, linting and deployment of Databricks assets with GitHub Actions.

Tools/Frameworks: Python, Apache Spark, Delta Lake, Apache Kafka, GitHub Actions

Cloud (Azure): Databricks, Event Hubs, Data Factory, ADLS Gen2

BIG DATA ARCHITECT – BMW – 30.06.2021 – 30.11.2023

Via Accenture. [Project page](#)

Core team of four architects behind the "blueprints" of BMW's Cloud Data Hub, the company-wide data mesh on AWS: frameworks and infrastructure used by 200+ data engineers to build petabyte-scale pipelines. Responsible for the batch blueprints: their architecture, code design and reviews of contributions from across BMW.

- Built the engine behind no-code pipelines: users composed sources, transformations and sinks in a self-service UI, and the blueprints compiled the resulting JSON into Spark jobs on Glue or EMR.
- Designed that compiler as a framework: the specification is validated against typed models and compiled into a dependency graph of operations, each resolved from a plugin registry, with pluggable Parquet and Iceberg readers and writers, per-branch error isolation, checkpointed incremental loads and automatic schema evolution.
- Owned the batch blueprints for every layer of the hub: JDBC ingestion across five database engines with computed partition bounds and look-back deltas, incremental SFTP ingestion, preparation with full and delta loads and deduplication, and EMR infrastructure for teams' own analytics code, with pseudonymisation, lineage, catalogue metadata and per-dataset encryption built into each blueprint.
- Unified the batch blueprints into a single functional library and moved them to Apache Iceberg: MERGE-based upserts, incremental reads between checkpointed snapshots, schema and partition evolution, and in-place and shadow migration of existing tables.
- Shipped each blueprint as code plus Terraform: transient EMR clusters on instance fleets orchestrated by Step Functions, with layered cleanup that guarantees no cluster outlives its job, Glue jobs, monitoring and alerting.
- Ran the blueprints as an internal open-source product: releases every six weeks, contributions from across BMW, CI with security scans (bandit, tfsec) and an end-to-end deployment test in a real integration account.

Tools/Frameworks: Python, PySpark, Apache Iceberg, pydantic, Terraform, GitHub Actions

Cloud (AWS): EMR, Glue (jobs and Data Catalog), Lake Formation, Athena, Step Functions, Lambda, EventBridge, CloudWatch, Kinesis, S3, DynamoDB, KMS

SENIOR BIG DATA ENGINEER – GROUPE RENAULT – 30.09.2020 – 29.06.2021

Pandora, the data-provisioning core of Renault's Industry 4.0 programme: a GCP hub that ingests real-time events from production sites (sensors, RFID, machine controllers, vehicle telematics), models them as digital twins and serves data marts to packaging, quality and vehicle teams.

- Built stream ingestion as paired Apache Beam pipelines on Dataflow: a metadata job that decodes heterogeneous payloads (CSV, JSON, XML, Avro, fixed-width) with custom readers, and a mapping job that enriches records against a Redis cache

and publishes them to denormalised, append-only BigQuery tables; delivered new plant and vehicle streams end to end, including their production infrastructure.

- Designed and implemented data replay: a parameterised Dataflow job that re-injects archived or failed data into the pipeline as Pub/Sub messages, with a header flag that lets downstream stages skip what they have already persisted, so failures and broken data marts are re-driven without duplicates across five recovery scenarios.
- Made the Redis enrichment cache recoverable: rebuild from the warehouse, scheduled backup and restore, and optimistic locking (WATCH/MULTI/EXEC) for entities written concurrently, after production incidents exposed the gap.
- Extended the shared Airflow library on Cloud Composer with idempotent Dataflow launch operators and DAG generators, and contributed to the end-to-end test framework that validates pipeline output against expected fixtures.
- Built the operational layer: an error sink into a BigQuery error log, a daily reconciliation of arrived against processed inputs published as custom Cloud Monitoring metrics with alerts, Cloud Functions that turn file drops into DAG triggers, and the platform's monitoring dashboards on GKE.
- Kept legacy Spark jobs on Dataproc running (automated restarts) while migrating them into Beam one job at a time, comparing outputs before each switch; archived and retired the Spanner store after the move to BigQuery.

Tools/Frameworks: Java, Apache Beam, Dagger, Vavr, Python, Apache Airflow, Apache Spark, Redis, Terraform, Maven, GitLab CI

Cloud (GCP): Dataflow, BigQuery, Pub/Sub, Cloud Composer, Cloud Functions, Memorystore, Cloud Storage, Dataproc, Spanner, Cloud SQL, Cloud Monitoring, GKE

SOFTWARE ENGINEER – MSG SYSTEMS – 31.07.2019 – 29.09.2020

Daimler's data platform for highly automated driving in the S-Class and premium Mercedes models: fleet sensor events are processed into road-condition information per road section and published to HERE maps.

- Implemented the road-condition streaming chain in Spark Structured Streaming on Azure HDInsight, consuming vehicle events from Event Hubs over the Kafka protocol: event decoding and keying by map tile (one tile id as the partition key from transport to state to publication), tile-border replication so each tile's state sees its neighbours' events, and the stateful core job that combines events with the road topology using spatio-temporal kernel estimation.
- Migrated legacy Apache Storm topologies to Spark Structured Streaming on a shared application framework: a common job lifecycle, pluggable source and sink interfaces (Kafka/Event Hubs, Data Lake), fail-fast configuration, Kafka-free unit tests and streaming-query listeners that report metrics to Application Insights.
- Built the publishing path to the HERE Open Location Platform (idempotent replacement per tile) and contributed to the Spark batch compiler that derives the road topology from HERE NDS map data.
- Implemented the data-protection stage that anonymises events by thresholding and generalisation (suppressing sparse map tiles, banding values) before they leave the platform.
- Exposed results through Spring Boot services on the Netflix OSS stack (Eureka, Zuul, Hystrix), deployed with Helm to an on-premise Kubernetes cluster and monitored with Prometheus and Grafana.
- Co-developed the Azure DevOps delivery system: shared multi-stage YAML templates covering builds with SonarQube, Terraform, Spark deployment with YARN health checks on HDInsight, system tests, and Helm deployments to Kubernetes.

Tools/Frameworks: Java, Apache Spark (Structured Streaming), Apache Storm, Apache Kafka, Spring Boot, Netflix OSS, Protobuf, Avro, Kubernetes, Helm, Terraform, Prometheus, Grafana, SonarQube, Maven

Cloud (Azure): HDInsight, Event Hubs, Data Lake Storage Gen2, Key Vault, Application Insights, Azure DevOps

EARLIER EXPERIENCE

- **AI RPA research grant – Endava (12.2018 – 07.2019):** robotic process automation with computer vision (OpenCV, Tesseract OCR) and speech recognition; research on LSTM, CNN and R-CNN architectures and NLP (Python, TensorFlow).
- **Java internship – MHP, a Porsche company (07.2017 – 09.2017):** web services for an internal employee platform (Spring Boot, Hibernate, Liquibase, PostgreSQL, Angular 2).
- **Self-employed software engineer (2016 – 2018):** an international ride-sharing and logistics application for rural Germany (Java, .NET Core, Angular, MySQL); research on stochastic processes, Markov chains and machine learning (Python, scikit-learn).

EDUCATION

BACHELOR'S DEGREE – Babeş-Bolyai University, Cluj-Napoca

Field of study: Computer Science (German-language track)

Thesis: Deep Learning and Natural Language Processing for Image Captioning

- Performance stipend (2 years, highest grade); SEEMOUS (South Eastern European Mathematical Olympiad for University Students); National Mathematics Olympiad, bronze medal.

LANGUAGES

Romanian (native) · English (C2) · German (C1, DSD2)